The Magellan Report on Cloud Computing for Science

U.S. Department of Energy
Office of Advanced Scientific Computing Research (ASCR)

December, 2011

CSO 23179

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC

Argonne NATIONAL LABORATORY

The Magellan Report on Cloud Computing for Science

U.S. Department of Energy
Office of Science
Office of Advanced Scientific Computing Research (ASCR)

December, 2011

Magellan Leads

Katherine Yelick, Susan Coghlan, Brent Draney,
Richard Shane Canon

Magellan Staff

Lavanya Ramakrishnan, Adam Scovel, Iwona Sakrejda, Anping Liu,
Scott Campbell, Piotr T. Zbiegiel, Tina Declerck, Paul Rich

Collaborators

Nicholas J. Wright
Richard Bradshaw
Shreyas Cholia
Linda Winkler
John Shalf
Jared Wilkening
Harvey Wasserman
Narayan Desai
Krishna Muriki
Victor Markowitz
Shucai Xiao
Keith Jackson
Nathan M. Mitchell
Jeff Broughton
Michael A. Guantonio
Zacharia Fadikra
Levi J. Lester
Devarshi Ghoshal
Ed Holohann
Elif Dede
Tisha Stacey
Madhusudhan Govindaraju
Gabriel A. West
Daniel Gunter
William E. Allcock
David Skinner
Rollin Thomas
Karan Bhatia
Henrik Nordberg
Wei Lu
Eric R. Pershey
Vitali Morozov
Dennis Gannon
CITRIS/University of California, Berkeley
Greg Bell
Nicholas Dale Trebon
K. John Wu
Brian Tierney
Brian Toonen
Alex Sim
Kalyan Kumaran
Ananth Kalyanraman
Michael Kocher
Doug Olson
Jan Balewski
STAR Collboration
Linda Vu
Yushu Yao
Margie Wylie
John Hules
Jon Bashor

This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02¬06CH11357, funded through the American Recovery and Reinvestment Act of 2009. Resources and research at NERSC at Lawrence Berkeley National Laboratory were funded by the Department of Energy from the American Recovery and Reinvestment Act of 2009 under contract number DE-AC02-05CH11231.

Executive Summary

The goal of Magellan, a project funded through the U.S. Department of Energy (DOE) Office of Advanced Scientific Computing Research (ASCR), was to investigate the potential role of cloud computing in addressing the computing needs for the DOE Office of Science (SC), particularly related to serving the needs of mid-range computing and future data-intensive computing workloads. A set of research questions was formed to probe various aspects of cloud computing from performance, usability, and cost. To address these questions, a distributed testbed infrastructure was deployed at the Argonne Leadership Computing Facility (ALCF) and the National Energy Research Scientific Computing Center (NERSC). The testbed was designed to be flexible and capable enough to explore a variety of computing models and hardware design points in order to understand the impact for various scientific applications. During the project, the testbed also served as a valuable resource to application scientists. Applications from a diverse set of projects such as MG-RAST (a metagenomics analysis server), the Joint Genome Institute, the STAR experiment at the Relativistic Heavy Ion Collider, and the Laser Interferometer Gravitational Wave Observatory (LIGO), were used by the Magellan project for benchmarking within the cloud, but the project teams were also able to accomplish important production science utilizing the Magellan cloud resources.

Cloud computing has garnered significant attention from both industry and research scientists as it has emerged as a potential model to address a broad array of computing needs and requirements such as custom software environments and increased utilization among others. Cloud services, both private and public, have demonstrated the ability to provide a scalable set of services that can be easily and cost-effectively utilized to tackle various enterprise and web workloads. These benefits are a direct result of the definition of cloud computing: on-demand self-service resources that are pooled, can be accessed via a network, and can be elastically adjusted by the user. The pooling of resources across a large user base enables economies of scale, while the ability to easily provision and elastically expand the resources provides flexible capabilities.

Following the Executive Summary we summarize the key findings and recommendations of the project. Greater detail is provided in the body of the report. Here we briefly summarize some of the high-level findings from the study.

- Cloud approaches provide many advantages, including customized environments that enable users to bring their own software stack and try out new computing environments without significant administration overhead, the ability to quickly surge resources to address larger problems, and the advantages that come from increased economies of scale. Virtualization is the primary strategy of providing these capabilities. Our experience working with application scientists using the cloud demonstrated the power of virtualization to enable fully customized environments and flexible resource management, and their potential value to scientists.

- Cloud computing can require significant initial effort and skills in order to port applications to these new models. This is also true for some of the emerging programming models used in cloud computing. Scientists should consider this upfront investment in any economic analysis when deciding whether to move to the cloud.

- Significant gaps and challenges exist in the areas of managing virtual environments, workflows, data, cyber-security, and others. Further research and development is needed to ensure that scientists can

easily and effectively harness the capabilities exposed with these new computing models. This would include tools to simplify using cloud environments, improvements to open-source clouds software stacks, providing base images that help bootstrap users while allowing them flexibility to customize these stacks, investigation of new security techniques and approaches, and enhancements to MapReduce models to better fit scientific data and workflows. In addition, there are opportunities in exploring ways to enable these capabilities in traditional HPC platforms, thus combining the flexibility of cloud models with the performance of HPC systems.

- The key economic benefit of clouds comes from the consolidation of resources across a broad community, which results in higher utilization, economies of scale, and operational efficiency. Existing DOE centers already achieve many of the benefits of cloud computing since these centers consolidate computing across multiple program offices, deploy at large scales, and continuously refine and improve operational efficiency. Cost analysis shows that DOE centers are cost competitive, typically 3–7x less expensive, when compared to commercial cloud providers. Because the commercial sector constantly innovates, DOE labs and centers should continue to benchmark their computing cost against public clouds to ensure they are providing a competitive service.

Cloud computing is ultimately a business model, but cloud models often provide additional capabilities and flexibility that are helpful to certain workloads. DOE labs and centers should consider adopting and integrating these features of cloud computing into their operations in order to support more diverse workloads and further enable scientific discovery, without sacrificing the productivity and effectiveness of computing platforms that have been optimized for science over decades of development and refinement. If cases emerge where this approach is not sufficient to meet the needs of the scientists, a private cloud computing strategy should be considered first, since it can provide many of the benefits of commercial clouds while avoiding many of the open challenges concerning security, data management, and performance of public clouds.

Key Findings

The goal of the Magellan project is to determine the appropriate role of cloud computing in addressing the computing needs of scientists funded by the DOE Office of Science. During the course of the Magellan project, we have evaluated various aspects of cloud computing infrastructure and technologies for use by scientific applications from various domains. Our evaluation methodology covered various dimensions: cloud models such as Infrastructure as a Service (IaaS) and Platform as a Service (PaaS), virtual software stacks, MapReduce and its open source implementation (Hadoop), resource provider and user perspectives. Specifically, Magellan was charged with answering the following research questions:

- Are the open source cloud software stacks ready for DOE HPC science?
- Can DOE cyber security requirements be met within a cloud?
- Are the new cloud programming models useful for scientific computing?
- Can DOE HPC applications run efficiently in the cloud? What applications are suitable for clouds?
- How usable are cloud environments for scientific applications?
- When is it cost effective to run DOE HPC science in a cloud?

We summarize our findings here:

Finding 1. Scientific applications have special requirements that require solutions that are tailored to these needs.

Cloud computing has developed in the context of enterprise and web applications that have vastly different requirements compared to scientific applications. Scientific applications often rely on access to large legacy data sets and pre-tuned application software libraries. These applications today run in HPC centers with low-latency interconnects and rely on parallel file systems. While these applications could benefit from cloud features such as customized environments and rapid elasticity, these need to be in concert with other capabilities that are currently available to them in supercomputing centers. In addition, the cost model for scientific users is based on account allocations rather than a fungible commodity such as dollars; and the business model of scientific processes leads to an open-ended need for resources. These differences in cost and the business model for scientific computing necessitates a different approach compared to the enterprise model that cloud services cater to today. Coupled with the unique software and specialized hardware requirements, this points to the need for clouds designed and operated specifically for scientific users. These requirements could be met at current DOE HPC centers. Private science clouds should be considered only if it is found that requirements cannot be met by the additional services at HPC centers.

Finding 2. Scientific applications with minimal communication and I/O are best suited for clouds.

We have used a range of application benchmarks and micro-benchmarks to understand the performance of scientific applications. Performance of tightly coupled applications running on virtualized clouds using commodity networks can be significantly lower than on clusters optimized for these workloads. This can be true at even mid-range computing scales. For example, we observed slowdowns of about 50x for PARATEC on the Amazon EC2 instances compared to Magellan bare-metal (non-virtualized) at 64 cores and about 7x slower at 1024 cores on Amazon Cluster Compute instances, the specialized high performance computing offering. As a result, current cloud systems are best suited for high-throughput, loosely coupled applications with modest data requirements.

Finding 3. Clouds require significant programming and system administration support.

Effectively utilizing virtualized cloud environments requires at a minimum: basic system administration skills to create and manage images; programming skills to manage jobs and data; and knowledge and understanding of the cloud environments. There are few tools available to scientists to manage and operate these virtualized resources. Thus, clouds can be difficult to use for scientists with little or no system administration and programming skills.

Finding 4. Significant gaps and challenges exist in current open-source virtualized cloud software stacks for production science use.

At the beginning of the Magellan project, both sites encountered major issues with the most popular open-source cloud software stacks. We encountered performance, reliability, and scalability challenges. Open-source cloud software stacks have significantly improved over the course of the project but would still benefit from additional development and testing. These gaps and challenges impacted both usage and administration. In addition, these software stacks have gaps when addressing many of the DOE security, accounting, and allocation policy requirements for scientific environments. Thus, even though software stacks have matured, the gaps and challenges will need to be addressed for production science use.

Finding 5. Clouds expose a different risk model requiring different security practices and policies.

Clouds enable users to upload their own virtual images that can be shared with other users and are used to launch virtual machines. These user-controlled images introduce additional security risks when compared to traditional login/batch-oriented environments, and they require a new set of controls and monitoring methods to secure. This capability, combined with the dynamic nature of the network, exposes a different usage model that comes with a different set of risks. Implementing some simple yet key security practices and policies on private clouds—such as running an intrusion detection system (IDS), capturing critical system logs from the virtual machines, and constant monitoring—can reduce a large number of the risks. However, the differences between the HPC and cloud model require that DOE centers take a close look at their current security practices and policies if providing cloud services.

Finding 6. MapReduce shows promise in addressing scientific needs, but current implementations have gaps and challenges.

Cloud programming models such as MapReduce show promise for addressing the needs of many data-intensive and high-throughput scientific applications. The MapReduce model emphasizes data locality and fault tolerance, which are important in large-scale systems. However, current tools have gaps for scientific applications. The current tools often require significant porting effort, do not provide bindings for popular scientific programming languages, and are not optimized for the structured data formats often used in large-scale simulations and experiments.

Finding 7. Public clouds can be more expensive than in-house large systems.

Many of the cost benefits from clouds result from increased consolidation and higher average utilization. Because existing DOE centers are already consolidated and typically have high average utilization, they are usually cost effective when compared with public clouds. Our analysis shows that DOE centers can range from 2–13x less expensive than typical commercial offerings. These cost factors include only the basic, standard services provided by commercial cloud computing, and do not take into consideration the additional services such as user support and training that are provided at supercomputing centers today. These services are essential for scientific users who deal with complex software stacks and dependencies and require help with optimizing their codes to achieve high performance and scalability.

Finding 8. DOE supercomputing centers already approach energy efficiency levels achieved in commercial cloud centers.

Cloud environments achieve energy efficiency through consolidation of resources and optimized facilities. Commercial cloud data providers emphasize the efficient use of power in their data centers. DOE HPC centers already achieve high levels of consolidation and energy efficiency. For example, DOE centers often operate at utilization levels over 85% and have a Power Usage Effectiveness (PUE) rating in the range of 1.2 to 1.5.

Finding 9. Cloud is a business model and can be applied at DOE supercomputing centers.

Cloud has been used to refer to a number of things in the last few years. The National Institute of Standards and Technology (NIST) definition of cloud computing describes it as *a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction* [63]. Cloud computing introduces a new business model and additional new technologies and features. Users with applications that have more dynamic or interactive needs could benefit from *on-demand, self-service environments* and *rapid elasticity* through the use of virtualization technology, and the MapReduce programming model to manage loosely coupled application runs. Scientific environments at high performance computing (HPC) centers today provide a number of these key features, including resource pooling, broad network access, and measured services based on user allocations. Rapid elasticity and on-demand self-service environments essentially require different resource allocation and scheduling policies that could also be provided through current HPC centers, albeit with an impact on resource utilization.

Recommendations

The Magellan project has evaluated cloud computing models and various associated technologies and explored their potential role in addressing the computing needs of scientists funded by the Department of Energy's Office of Science. Our findings highlight both the potential value and the challenges of exploiting cloud computing for DOE SC applications. Here we summarize our recommendations for DOE SC, DOE resource providers, application scientists, and tool developers. A number of these recommendations do not fit within current scope and budgets, and additional funding beyond currently funded DOE projects might be required.

DOE Office of Science. The findings of the Magellan project demonstrate some potential benefits of cloud computing for meeting the computing needs of the DOE SC scientific community. Cloud features such as customized environments and the MapReduce programming model help in addressing the needs of some current users of DOE resources as well as the needs of some new scientific applications that need to scale up from current environments. The findings also show that existing DOE HPC centers are cost effective compared with commercial providers. Consequently, *DOE should work with the DOE HPC centers and DOE resource providers to ensure that the expanding needs of the scientific community are being met.* If cases emerge where cloud models are deemed necessary in order to fully address these needs, we recommend that DOE should first consider a private cloud computing strategy. This approach can provide many of the benefits of cloud environments while being more cost effective, allowing for more optimized offerings, and better addressing the security, performance, and data management challenges.

DOE Resource Providers. There is a rising class of applications that do not fit the current traditional model of large-scale tightly coupled applications that run in supercomputing centers, yet have increasing needs for computation and storage resources. In addition, various scientific communities need custom software environments or shared virtual clusters to meet specific collaboration or workload requirements. Additionally, clouds provide mechanisms and tools for high-throughput and data-intensive applications that have large-scale resource requirements. The cloud model provides solutions to address some of these requirements, but these can be met by DOE resource providers as well. We make some specific recommendations to DOE resource providers for providing these features. Resource providers include the large ASCR facilities such as the Leadership Computing Facilities and NERSC; institutional resources operated by many of the labs; and the computing resources deployed in conjunction with many large user facilities funded by other science program offices.

1. *DOE resource providers should investigate mechanisms to support more diverse workloads such as high-throughput and data-intensive workloads that increasingly require more compute and storage resources.* Specifically, resource providers should consider more flexible queueing policies for these loosely coupled computational workloads.

2. A number of user communities require a different resource provisioning model where they need exclusive access to a pre-determined amount of resources for a fixed period of time due to increased computational load. Our experience with the Material Genomes project, the Joint Genome Institute, and the analysis of the German *E. coli* strains demonstrates that on-demand resource provisioning can play a valuable

role in addressing the computing requirements of scientific user groups. *We recommend that DOE resource providers investigate mechanisms to provide on-demand resources to user groups.*

3. A number of DOE collaborations rely on complex scientific software pipelines with specific library and version dependencies. Virtual machines are useful for end users who need customizable environments. However, the overheads of virtualization are significant for most tightly coupled scientific applications. These applications could benefit from bare-metal provisioning or other approaches to providing custom environments. *DOE resource providers should consider mechanisms to provide scientists with tailored environments on shared resources.*

4. DOE resource providers are cost and energy efficient. However, the commercial sector is constantly innovating, and it is important that *DOE resource providers should track their cost and energy efficiencies in comparison with the commercial cloud sector.*

5. Private cloud virtualization software stacks have matured over the course of the project. However, there are significant challenges in performance, stability and reliability. *DOE resource providers should work with the developer communities of private cloud software stacks to address these deficiencies before broadly deploying these software stacks for production use.*

6. There are gaps in implementing DOE-specific accounting, allocation and security policies in current cloud software stacks. Cloud software solutions will need customization to handle site-specific polices related to resource allocation, security, accounting, and monitoring. *DOE resource providers should develop or partner with the developer communities of private cloud software stacks to support site-specific customization.*

7. User-created virtual images are powerful. However, there is also a need for a standard set of base images and simple tools to reduce the entry barrier for scientists. Additionally, scientists often require pre-tuned libraries that need expertise from supercomputing center staff. *DOE resource providers should consider providing reference images and tools that simplify using virtualized environments.*

8. The cloud exposes a new usage model that necessitates additional investments in training end-users in the use of these resources and tools. Additionally, the new model necessitates a new user-support and collaboration model where trained personnel can help end-users with the additional programming and system administration burden created by these new technologies. *DOE resource providers should carefully consider user support challenges before broadly supporting these new models.*

Science Groups. Cloud computing promises to be useful to scientific applications due to advantages such as on-demand access to resources and control over the user environment. However, cloud computing also has significant impact on application design and development due to challenges related to performance and reliability, programming model, designing and managing images, distributing the work across compute resources, and managing data. We make specific recommendations to science groups that might want to consider cloud technologies or models for their applications.

1. Infrastructure as a Service provides an easy path for scientific groups to harness cloud resources while leveraging much of their existing application infrastructure. However, virtualized cloud systems provide various options for instance types and storage classes (local vs block store vs object store) that have different performance and associated price points. *Science groups need to carefully benchmark applications with the different options to find the best performance-cost ratio.*

2. Cloud systems provide application developers the ability to completely control their software environments. However, there is currently a limited choice of tools available for workflow and data management in these environments. Scientific groups should consider using standard tools to manage these environments rather than developing custom scripts. *Scientists should work with tool developers to ensure that their requirements and workflows are sufficiently captured and understood.*

3. *Application developers should consider the potential for variability and failures in their design and implementation.* While this is a good practice in general, it is even more critical for applications running in the cloud, since they experience significant failures and performance variations.

4. Cloud technologies allow user groups to manage their own machine images, enabling groups with complex software dependencies to achieve portability. This flexibility comes with the challenge for these groups to ensure they have addressed security concerns. *Science groups should attempt to use standardized secure images to prevent security and other configuration problems with their images. Science groups will also need to have an action plan on how to secure the images and keep them up to date with security patches.*

5. Cloud technologies such as message queues, tabular storage, and blob or object store provide a number of key features that enable applications to scale without the need to use synchronization where it may not be necessary. These technologies fundamentally change the application execution model. *Scientific users should evaluate technologies such as message queues, tabular storage, and object storage during application design phase.*

Tools development and research. A number of site-level and user-side tools to manage scientific environments at HPC and cloud environments have evolved in the last few years. However, there are significant gaps and challenges in this space. We identify some key areas for tool development and research related to adoption of cloud models and technologies to science.

1. Virtual machines or provisioned bare metal hardware are useful to many application groups. However, scientists need to handle the management of these provisioned resources, including the software stack, job and data coordination. *Tool developers should support tools to simplify and smooth workflow and data management on provisioned resources.*

2. Investments are needed in tools that enable automated mechanisms to update images with appropriate patches as they become available and a simple way to organize, share and find these images across user groups and communities. *Tool developers should consider developing services that will enable organization and sharing of images.*

3. Virtualized cloud environments are limited by networking and I/O options available in the virtual machines. Access to high-performance parallel file systems such as GPFS and Lustre, and low-latency, high-bandwidth interconnects such as InfiniBand within a virtual machine would enable more scientific applications to benefit from virtual environments without sacrificing performance or ease of use. *System software developers should explore methods to provide HPC capabilities in virtualized environments.*

4. There is a need for new methods to monitor and secure private clouds. Further research is required in this area. Sites typically rely on OS-level controls to implement many security policies. Most of these controls must be shifted into the hypervisor or alternative approaches must be developed. *Security developers should explore new methods to secure these environments and, ideally, leverage the advanced mechanisms that virtualization provides.*

5. MapReduce can be useful for addressing data-intensive scientific applications, but there is a need for MapReduce implementations that account for characteristics of scientific data and analysis methods. *Computer science researchers and developers should explore modifications or extensions to frameworks like Hadoop that would enable the frameworks to understand and exploit the data models of data formats typically used in scientific domains.*

6. A number of cloud computing concepts and technologies (e.g., MapReduce, schema-less databases) have evolved around the idea of managing "big data" and associated metadata. Cloud technologies address the issues of automatic scaling, fault-tolerance and data locality, all key to success of large-scale systems. *There is a need to investigate the use of cloud technologies and ideas to manage scientific data and metadata.*

Contents

Chapter 1

Overview

Cloud computing has served the needs of enterprise web applications for the last few years. The term "cloud computing" has been used to refer to a number of different concepts (e.g., MapReduce, public clouds, private clouds, etc.), technologies (e.g., virtualization, Apache Hadoop), and service models (e.g., Infrastructure-as-a-Service [IaaS], Platform-as-a-Service [PaaS], Software-as-a-Service[SaaS]). Clouds have been shown to provide a number of key benefits including cost savings, rapid elasticity, ease of use, and reliability. Cloud computing has been particularly successful with customers lacking significant IT infrastructure or customers who have quickly outgrown their existing capacity.

The open-ended nature of scientific exploration and the increasing role of computing in performing science has resulted in a growing need for computing resources. There has been an increasing interest over the last few years in evaluating the use of cloud computing to address these demands. In addition, there are a number of key features of cloud environments that are attractive to some scientific applications. For example, a number of scientific applications have specific software requirements including OS version dependencies, compilers and libraries, and the users require the flexibility associated with custom software environments that virtualized environments can provide. An example of this is the Supernova Factory, which relies on large data volumes for the supernova search and has a code base which consists of a large number of custom modules. The complexity of the pipeline necessitates having specific library and OS versions. Virtualized environments also promise to provide a *portable container* that will enable scientists to share an environment with collaborators. For example, the ATLAS experiment, a particle physics experiment at the Large Hadron Collider at CERN, is investigating the use of virtual machine images for distribution of all required software [10]. Similarly, the MapReduce model holds promise for data-intensive applications. Thus, cloud computing models promise to be an avenue to address new categories of scientific applications, including data-intensive science applications, on-demand/surge computing, and applications that require customized software environments. A number of groups in the scientific community have investigated and tracked how the cloud software and business model might impact the services offered to the scientific community. However, there is a limited understanding of how to operate and use clouds, how to port scientific workflows, and how to determine the cost/benefit trade-offs of clouds, etc. for scientific applications.

The Magellan project was funded by the American Recovery and Reinvestment Act to investigate the applicability of cloud computing for the Department of Energy's Office of Science (DOE SC) applications. Magellan is a joint project at the Argonne Leadership Computing Facility (ALCF) and the National Energy Research Scientific Computing Center (NERSC). Over the last two years we have evaluated various dimensions of clouds—cloud models such as Infrastructure as a Service (IaaS) and Platform as a Service(PaaS), virtual software stacks, MapReduce and its open source implementation (Hadoop). We evaluated these on various criteria including stability, manageability, and security from a resource provider perspective, and performance and usability from an end-user perspective.

Cloud computing has similarities with other distributed computing models such as grid and utility computing. However, the use of virtualization technology, the MapReduce programming model, and tools such as Eucalyptus and Hadoop, require us to study the impact of cloud computing on scientific environments. The Magellan project has focused on understanding the unique requirements of DOE science applications and the role cloud computing can play in scientific communities. However the identified gaps and challenges apply more broadly to scientific applications using cloud environments.

1.1 Magellan Goals

The goal of the Magellan project is to investigate how the cloud computing business model can be used to serve the needs of DOE Office of Science applications. Specifically, Magellan was charged with answering the following research questions:

- Are the open source cloud software stacks ready for DOE HPC science?
- Can DOE cyber security requirements be met within a cloud?
- Are the new cloud programming models useful for scientific computing?
- Can DOE HPC applications run efficiently in the cloud? What applications are suitable for clouds?
- How usable are cloud environments for scientific applications?
- When is it cost effective to run DOE HPC science in a cloud?

In this report, we summarize our findings and recommendations based on the experiences over the course of the project in addressing the above research questions.

1.2 NIST Cloud Definition

The term "cloud computing" has been used to refer to different concepts, models, and services over the last few years. For the rest of this report we use the definition for cloud computing provided by the National Institute of Standards and Technology (NIST), which defines cloud computing as *a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction* [63].

High performance computing (HPC) centers such as those funded by the Department of Energy provide a number of these key features, including resource pooling, broad network access, and measured services based on user allocations. However, there is limited support for rapid elasticity or on-demand self-service in today's HPC centers.

1.3 Impact

The Magellan project has made an impact in several different areas. Magellan resources were available to end users in several different configurations, including virtual machines, Hadoop, and traditional batch queues. Users from all the DOE SC offices have taken advantage of the systems. Some key areas where Magellan has made an impact in the last two years:

- The Magellan project is the first to conduct an exhaustive evaluation of the use of cloud computing for science. This has resulted in a number of publications in leading computer science conferences and workshops.

- A facility problem at the Joint Genome Institute led to a pressing need for backup computing hardware to maintain its production sequencing operations. NERSC partnered with ESnet and was able to provision Magellan hardware in a Hardware as a Service (HaaS) model to help the Institute meet its demands.

- The Argonne Laboratory Computing Resource Center (LCRC), working with Magellan project staff, was able to develop a secure method to extend the production HPC compute cluster into the ALCF Magellan testbed, allowing jobs to utilize the additional resources easily, while accessing the LCRC storage and running within the same computing environment.
- The STAR experiment at the Relativistic Heavy Ion Collider (RHIC) used Magellan for real-time data processing. The goal of this particular analysis is to sift through collisions searching for the "missing spin."
- Biologists used the ALCF Magellan cloud to quickly analyze strains suspected in the *E. coli* outbreak in Europe in summer 2011.
- Magellan was honored with the HPCwire Readers' Choice Award for "Best Use of HPC in the Cloud" in 2010 and 2011.
- Magellan benchmarking work won the "Best Paper Award" at both CloudCom 2010 and DataCloud 2011.
- Magellan played a critical role in demonstrating the capabilities of the 100 Gbps network deployed by the Advanced Networking Initiative (ANI). Magellan provided storage and compute resources that were used by many of the demonstrations performed during SC11. The demonstrations were typically able to achieve rates of 80-95 Gbps.

1.4 Approach

Our approach has been to evaluate various dimensions of cloud computing and associated technologies. We adopted an *application-driven* approach where we assessed the specific needs of scientific communities while making key decisions in the project.

A distributed testbed infrastructure was deployed at the Argonne Leadership Computing Facility (ALCF) and the National Energy Research Scientific Computing Center (NERSC). The testbed consists of IBM iDataPlex servers and a mix of special servers, including Active Storage, Big Memory, and GPU servers connected through an Infiniband fabric. The testbed also has a mix of storage options, including both distributed and global disk storage, archival storage, and two classes of flash storage. The system provides both a high-bandwidth, low-latency quad-data rate InfiniBand network as well as a commodity Gigabit Ethernet network. This configuration is different from a typical cloud infrastructure but is more suitable for the needs of scientific applications. For example, InfiniBand may be unusual in a typical commercial cloud offering; however, it allowed Magellan staff to investigate a range of performance points and measure the impact on application performance.

The software stack on the Magellan testbed was diverse and flexible to allow Magellan staff and users to explore a variety of models with the testbed. Software on the testbed included multiple private cloud software solutions, a traditional batch queue with workload monitoring, and Hadoop. The software stacks were evaluated for usability, ease of administration, ability to enforce DOE security policies, stability, performance, and reliability.

We conducted a thorough benchmarking and workload analysis evaluating various aspects of cloud computing, including the communication protocols and I/O using micro as well as application benchmarks. We also provided a cost analysis to understand the cost effectiveness of clouds. The main goal of the project was to advance the understanding of how private cloud software operates and identify its gaps and limitations. However, for the sake of completeness, we performed a comparison of the performance and cost against commercial services as well.

We evaluated the use of MapReduce and Hadoop for various workloads in order to understand the applicability of the model for scientific pipelines as well as to understand the various performance trade-offs. We worked closely with scientific groups to identify the intricacies of application design and associated challenges while using cloud resources. This close work with our user groups helped us identify some key gaps in current cloud technologies for science.

1.5 Outline of Report

The rest of this report is organized as follows. Chapter 2 describes the cloud service models and discusses application models. Chapter 3 provides a timeline and overview of Magellan project activities. Chapter 4 summarizes the identified requirements or use cases for clouds from discussions with DOE user groups. We describe the Magellan testbed in Chapter 5. We compare and contrast the features and our experiences of various popular virtualized software stack offerings in Chapter 6. We discuss user support issues in Chapter 7 and detail our security analysis in Chapter 8. Our benchmarking efforts are described in Chapter 9. We detail our experiences with the MapReduce programming and specifically the open-source Apache Hadoop implementation in Chapter 10. We present the case studies of some key applications on Magellan and summarize the challenges of using cloud environments in Chapter 11. Our cost analysis is presented in Chapter 12. We present our conclusions in Chapters 13.

Chapter 2

Background

The term "cloud computing" covers a range of delivery and service models. The common characteristic of these service models is an emphasis on *pay-as-you-go* and elasticity, the ability to quickly expand and collapse the utilized service as demand requires. Thus new approaches to distributed computing and data analysis have also emerged in conjunction with the growth of cloud computing. These include models like MapReduce and scalable key-value stores like Big Table [11].

Cloud computing technologies and service models are attractive to scientific computing users due to the ability to get on-demand access to resources to replace or supplement existing systems, as well as the ability to control the software environment. Scientific computing users and resource providers servicing these users are considering the impact of these new models and technologies. In this section, we briefly describe the cloud service models and technologies to provide some foundation for the discussion.

2.1 Service Models

Cloud offerings are typically categorized as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Each of these models can play a role in scientific computing.

The distinction between the service models is based on the layer at which the service is abstracted to the end user (e.g., hardware, system software, etc.). The end user then has complete control over the software stack above the abstracted level. Thus, in IaaS, a virtual machine or hardware is provided to the end user and the user then controls the operating system and the entire software stack. We describe each of these service models and visit existing examples in the commercial cloud space to understand their characteristics.

2.1.1 Infrastructure as a Service

In the Infrastructure as a Service provisioning model, an organization outsources equipment including storage, hardware, servers, and networking components. The service provider owns the equipment and is responsible for housing, running, and maintaining it. In the commercial space, the client typically pays on a per-use basis for use of the equipment.

Amazon Web Services is the most widely used IaaS cloud computing platform today. Amazon provides a number of different levels of computational power for different pricing. The primary methods for data storage in Amazon EC2 are S3 and Elastic Block Storage (EBS). S3 is a highly scalable key-based storage system that transparently handles fault tolerance and data integrity. EBS provides a virtual storage device that can be associated with an elastic computing instance. S3 charges for space used per month, the volume of data transferred, and the number of metadata operations (in allotments of 1000). EBS charges for data stored per month. For both S3 and EBS, there is no charge for data transferred to and from EC2 within a domain (e.g., the U.S. or Europe).

Eucalyptus, OpenStack and Nimbus are open source software stacks that can be used to create a private cloud IaaS service. These software stacks provide an array of services that mimic many of the services provided by Amazon's EC2 including image management, persistent block storage, virtual machine control, etc. The interface for these services is often compatible with Amazon EC2 allowing the same set of tools and methods to be used.

In Magellan, in conjunction with other synergistic activities, we use Amazon EC2 as the commercial cloud platform to understand and compare an existing cloud platform. We use Eucalyptus and OpenStack to set up a private cloud IaaS platform on Magellan hardware for detailed experimentation on providing cloud environments for scientific workloads. The IaaS model enables users to control their own software stack that is useful to scientists that might have complex software stacks.

2.1.2 Platform as a Service

Platform as a Service (PaaS) provides a computing platform as a service, supporting the complete life cycle of building and delivering applications. PaaS often includes facilities for application design, development, deployment and testing, and interfaces to manage security, scalability, storage, state, etc. Windows Azure, Hadoop, and Google App Engine are popular PaaS offerings in the commercial space.

Windows Azure is Microsoft's offering of a cloud services operating system. Azure provides a development, service hosting, and service management environment. Windows Azure provides on-demand compute and storage resources for hosting applications to scale costs. The Windows Azure platform supports two primary virtual machine instance types—the Web role instances and the Worker role instances. It also provides Blobs as a simple way to store data and access it from a virtual machine instance. Queues provide a way for Worker role instances to access the work quantum from the Web role instance. While the Azure platform is primarily designed for web applications, its use for scientific applications is being explored [58, 69].

Hadoop is an open-source software that provides capabilities to harness commodity clusters for distributed processing of large data sets through the MapReduce [13] model. The Hadoop streaming model allows one to create map-and-reduce jobs with any executable or script as the mapper and/or the reducer. This is the most suitable model for scientific applications that have years of code in place capturing complex scientific processes.

The Hadoop File System (HDFS) is the primary storage model used in Hadoop. HDFS is modeled after the Google File system and has several features that are specifically suited to Hadoop/MapReduce. Those features include exposing data locality and data replication. Data locality is a key mechanism that enables Hadoop to achieve good scaling and performance, since Hadoop attempts to locate computation close to the data. This is particularly true in the map phase, which is often the most I/O intensive phase.

Hadoop provides a platform for managing loosely coupled data-intensive applications. In Magellan synergistic activities, we used the Yahoo! M45 Hadoop platform to benchmark BLAST. In addition, Hadoop has been deployed on Magellan hardware to enable our users to experiment with the platform. PaaS provides users wth the building blocks and semantics for handling scalability, fault tolerance, etc. in their applications.

2.1.3 Software as a Service

Software as a Service provides access to an end user for an application or software that has a specific function. Examples in the commercial space include services like SalesForce and Gmail. In our project activities, we use the Windows Azure BLAST service to run BLAST jobs on the Windows Azure platform. Science portals can also be viewed as providing a Software as a Service, since they typically allow remote users to perform analysis or browse data sets through a web interface. This model can be attractive since it allows the user to transfer the responsibility of installing, configuring, and maintaining an application and shields the end-user from the complexity of the underlying software.

2.1.4 Hardware as a Service

Hardware as a Service (HaaS) is also known as "bare-metal provisioning." The main distinction between this model and IaaS is that the user-provided operating system software stack is provisioned onto the raw hardware, allowing the users to provide their own custom hypervisor, or to avoid virtualization completely, along with the performance impact of virtualization of high-performance hardware such as InfiniBand. The other difference between HaaS and the other service models is that the user "leases" the entire resource; it is not shared with other users within a virtual space. With HaaS, the service provider owns the equipment and is responsible for housing, running, and maintaining it. HaaS provides many of the advantages of IaaS and enables greater levels of control on the hardware configuration.

2.2 Deployment Models

According to the NIST definition, clouds can have one of the following deployment models, depending on how the cloud infrastructure is operated: (a) public, (b) private, (c) community, or (d) hybrid.

Public Cloud. Public clouds refer to infrastructure provided to the general public by a large industry selling cloud services. Amazon's cloud offering would fall in this category. These services are on a pay-as-you-go basis and can usually be purchased using a credit card.

Private Cloud. A private cloud infrastructure is operated solely for a particular organization and has specific features that support a specific group of policies. Cloud software stacks such as Eucalyptus, OpenStack, and Nimbus are used to provide virtual machines to the user. In this context, Magellan can be considered a private cloud that provides its services to DOE Office of Science users.

Community Cloud. A community cloud infrastructure is shared by several organizations and serves the needs of a special community that has common goals. FutureGrid [32] can be considered a community cloud.

Hybrid Cloud. Hybrid clouds refer to two or more cloud infrastructures that operate independently but are bound together by technology compliance to enable application portability.

2.3 Other Related Efforts

The Magellan project explored a range of topics that included evaluating current private cloud software and understanding gaps and limitations, application software setup, etc. To the best of our knowledge, there is no prior work that does such an exhaustive study of various aspects of cloud computing for scientific applications.

The FutureGrid project [32] provides a testbed, including a geographically distributed set of heterogeneous computing systems, that includes cloud resources. The aim of the project is to provide a capability that makes it possible for researchers to tackle complex research challenges in computer science, whereas Magellan is more focused on serving the needs of the science.

A number of different groups have conducted feasibility and benchmarking studies of running their scientific applications in the Amazon cloud [67, 40, 15, 52, 53, 51]. Standard benchmarks have also been evaluated on Amazon EC2 [62, 23, 66, 74, 82]. These studies complement our own experiments which show that high-end, tightly coupled applications are impacted by the performance characteristics of current cloud environments.

Chapter 3

Magellan Project Activities

Magellan project activities have extensively covered the spectrum of evaluating various cloud models and technologies. We have evaluated cloud software stacks, conducted evaluation of security practices and mechanisms in current cloud solutions, conducted benchmarking studies on both commercial as well as private cloud platforms, evaluated MapReduce, worked closely together with user groups to understand the challenges and gaps in current cloud software technologies in serving the needs of science, and performed a thorough cost analysis.

Our research has been guided by two basic principles: *application-driven* and *flexibility.* Our efforts are centered around collecting data about user needs and working closely with user groups in evaluation. Rather than rely entirely on micro-benchmarks, application benchmarks and user applications were used in our evaluation of both private and commercial cloud platforms. This application-driven approach helps us understand the suitability of cloud environments for science. In addition, as discussed earlier, there have been a number of different cloud models and technologies researched, including new technologies that were developed during the course of the project. To support the dynamic nature of this quickly evolving ecosystem, a flexible software stack and project activity approach were utilized.

The Magellan project consisted of activities at Argonne and Lawrence Berkeley National Labs. The project activities at both sites were diverse and complementary in order to fully address the research questions. The groups collaborated closely across both sites of Magellan through bi-weekly teleconferences and quarterly face-to-face meetings. Magellan staff from both sites worked closely to run Argonne's MG-RAST, a fully automated service for annotating metagenome samples, and the Joint Genome Institute's MGM pipeline across both sites with fail-over fault tolerance and load balancing. Cyber security experts at both ALCF and NERSC coordinated their research efforts in cloud security. Experiences and assistance were shared as both sites deployed various cloud software stacks. Our experiences from evaluating cloud software was published in ScienceCloud 2011 [72]. In addition, the two sites are working closely with the Advanced Networking Initiative (ANI) research projects to support their cross-site demos planned for late 2011.

Table 3.1 summarizes the timeline of key project activities. As the project began in late 2009, the focus was on gathering user requirements and a conducting a user survey to understand potential applications and user needs. The core systems were deployed, tested and accepted in early 2010. Eucalyptus and Hadoop software stacks were deployed, and users were granted access to the resources shortly after the core systems were deployed. Other efforts in 2010 included early user access, benchmarking, and joint demos (MG-RAST and JGI) performed across the cloud testbed. A Eucalyptus 2.0 upgrade and OpenStack were provided to the users in early 2011. Further benchmarking to understand the impact of I/O and network interconnects were performed in spring and summer of 2011. Finally, support for ANI research projects started in April 2011 and was ongoing at the time of this report.

Table 3.1: Key Magellan Project Activities.

Activity	ALCF	NERSC
Project Start	Sep 2009	
Requirements gathering and NERSC User Survey	Nov 2009 - Feb 2010	
Core System Deployed	Jan 2010 - Feb 2010	Dec 2009 - Jan 2010
Benchmarking of commercial cloud platforms	-	Mar - Apr 2010
Early User Access	Mar 2010 (VM)	Apr 2010 (Cluster), Oct 2010 (VM)
Hadoop testing	Nov 2010	Apr 2010
Hadoop User Access	Dec 2010	May 2010
Baseline benchmarking of existing private cloud platforms	May - June 2010	
Flash storage evaluation	-	June 2010
Joint Demo (MG-RAST)	June 2010	
Hardware as a service/bare metal access	Nov 2010 - Dec 2011	Mar - May 2010
Design and development of Heckle	Mar 2010 - June 2010	-
Eucalyptus testing and evaluation	Feb - Mar 2010	Dec 2009 - Feb 2010 and June 2010
Preparation for joint science demo	Mar 2010 - June 2010	
OSG on Condor-G deployment	May 2010	-
Nimbus Deployments	June 2010	-
SOCC Paper	-	June 2010
GPFS-SNC Evaluation	-	June 2010 - Dec 2010
CloudCom Paper (awarded Best Paper)	-	December 2010
JGI demo at SC	Aug 2010 - Nov 2010	
LISA 2010 Eucalyptus Experience Paper	Nov 2010	-
OpenStack testing	Dec 2010 - Mar 2011	-
Hadoop benchmarking	-	June - Dec 2010
Eucalyptus 2.0 testing	Nov 2010 - Jan 2011	
Eucalyptus 2.0 Deployed	Jan 2011	Feb 2011
Network interconnect and protocol benchmarking	-	Mar 2011 - Sep 2011
ASCAC Presentation	Mar 2011	
User Access OpenStack	Mar 2011	-
MOAB Provisioning	-	May 2011 - Sep 2011
ScienceCloud '11 Joint Paper and Presentation	June 2011	
I/O benchmarking	-	June 2011 - Aug 2011
I/O forwarding work	Mar 2011 - Aug 2011	-
GPU VOCL framework devel & benchmarking	Feb 2011 - Aug 2011	-
ANI research projects	Apr 2011 - Dec 2011	
Magellan project ends	Sep 2011	
ANI 100G active	Oct 2011	
ANI SC demos	Nov 2011	
Virtualization overhead benchmarking	Sep 2011 - Nov 2011	-
Interconnect and virtualization paper at PMBS workshop at SC	-	Nov 2011
I/O benchmarking paper at DataCloud workshop at SC (Best paper)	-	Nov 2011
Magellan ANI ends	Dec 2011	

3.1 Collaborations and Synergistic Activities

Magellan research was conducted as a collaboration across both ALCF and NERSC, as well as leveraging the expertise of other collaborators and projects. At a high-level, Magellan collaboration activities can be classified into three categories:

- **Magellan Core Research.** Magellan staff actively performed the research necessary to answer the research questions outlined in the report with respect to understanding the applicability of cloud computing for scientific applications.

- **Synergistic Research Activities.** Magellan staff participated in a number of key collaborations in research related to cloud computing.

- **Magellan Resource Enabling User Research.** Magellan resources were used extensively by scientific users in their simulations and data analysis. Magellan staff often provided key user support in facilitating this research.

We outline the key Magellan collaborations in this section. Additional collaborations are highlighted throughout the report as well. Specifically the application case studies are highlighted in Chapter 11.

Cloud Benchmarking. The benchmarking of commercial cloud platforms was performed in collaboration with the IT department at Lawrence Berkeley National Laboratory (LBNL), which manages some of the mid- range computing clusters for scientific users; the Advanced Technology Group at NERSC, which studies the requirements of current and emerging NERSC applications to find hardware design choices and programming models; and the Advanced Computing for Science (ACS) Department in the Computational Research Division (CRD) at LBNL, which seeks to create software and tools to enable science on diverse resource platforms. Access to commercial cloud platforms was also possible through collaboration with the IT division and the University of California Center for Information Technology Research in the Interest of Society (CITRIS), which had existing contracts with different providers. This early benchmarking effort resulted in a publication at CloudCom 2010 [46] that was awarded ***Best Paper***.

MapReduce/Hadoop Evaluation. Scientists are struggling with a tsunami of data across many domains. Emerging sensor networks, more capable instruments, and ever increasing simulation scales are generating data at a rate that exceeds our ability to effectively manage, curate, analyze, and share it. This is exacerbated by the limited understanding and expertise on the hardware resources and software infrastructure required for handling these diverse data volumes. A project funded through the Laboratory Directed Research and Development (LDRD) program at Lawrence Berkeley Laboratory is looking at the role that many new, potentially disruptive, technologies can play in accelerating discovery. Magellan resources were used for this evaluation. Additionally, staff worked closely with a summer student on the project evaluating the specific application patterns that might benefit from Hadoop. Magellan staff also worked closely with the Grid Computing Research Laboratory at SUNY, Binghamton in a comparative benchmarking study of MapReduce implementations and an alternate implementation of MapReduce that can work in HPC environments. This collaboration resulted in two publications in Grid 2011 [25, 24].

Collaboration with Joint Genome Institute. The Magellan project leveraged the partnership between the Joint Genome Institute (JGI) and NERSC to benchmark the IMG and MGM pipelines on a variety of platforms. Project personnel were also involved in pilot projects for the Systems Biology Knowledge Base. They provided expertise with technologies such as HBASE which were useful in guiding the deployments on Magellan. Magellan resources were also deployed for use by JGI as a proof-of-concept of Hardware-as-a-Service. JGI also made extensive use of the Hadoop cluster.

Little Magellan. Porting applications to cloud environments can be tedious and time-consuming (details in Chapter 11). To help our users compare the performance of bare-metal and a virtual clusters, we deployed "Little Magellan" – a virtual cluster booted on top of Eucalyptus at NERSC. The virtual cluster consisted of a head node with a public IP and 10 batch nodes. The virtual cluster helped users make an easy transition from the batch queue to the cloud environment. It was configured to use the NERSC LDAP server for ease of access. The software environment was setup for the applications on an EBS volume that was NFS mounted across the cluster. A few of the most active users were enlisted to do testing. When they logged into "Little Magellan" they were presented with an environment that closely resembled makeup of carver.nersc.gov (the login node for the Magellan systems at NERSC). Some of the benchmarking results from the "Little Magellan" virtual cluster is presented in Chapter 9.

Nimbus. Nimbus is an open-source toolkit for cloud computing specifically targeted to supporting the needs of scientific applications. ALCF's Magellan project provided resources and specialized assistance to the Nimbus team to facilitate their plans for research, development, and testing of multiple R&D projects including,

- Exploration of different implementations of storage clouds and how they respond to specific scientific workloads.
- Refining and experimenting with the Nimbus CloudBurst tool developed in the context of the Ocean Observatory Initiative. This included striving to understand how the system scales under realistic scientific workloads and what level of reliability it can provide to the time-varying demands of scientific projects, especially for projects seeking computational resources to compute responses to urgent phenomena such as hurricane relief efforts, earthquakes, or environmental disasters.
- Opening the Nimbus allocation to specific scientific and educational projects as the Nimbus team evaluates usage patterns specific to cloud computing.

SPRUCE. The on-demand nature of computational clouds like Magellan makes them well suited for certain classes of urgent computing (i.e., deadline-constrained applications with insufficient warning for advanced reservations). The Special PRiority and Urgent Computing Environment (SPRUCE) is a framework developed by researchers at Argonne and the University of Chicago that aims to provide these urgent computations with the access to the necessary resources to meet their demands. ALCF's Magellan project provided resources and special assistance to the SPRUCE project as the researchers evaluated the utility gained through the implementation of various urgent computing policies that a computational cloud could enable.

The SPRUCE project has implemented and evaluated four policies on a development cloud. These policies include: (1) an urgent computing session where a portion of the cloud would only be available to urgent computing virtual machines for a short window of time (e.g., 48 hours); (2) pausing non-urgent virtual machines and writing their memory contents to disk, freeing up their memory and cores for urgent virtual machines; (3) preempting non-urgent virtual machines; (4) migrating non-urgent virtual machines to free up resources for urgent virtual machines; and (5) dynamically reducing the amount of physical resources allocated to running non-urgent virtual machines, freeing those resources for urgent VMs. This implementation was used for benchmarking resource allocation delays that is described in Chapter 9.

Transparent Virtualization of Graphics Processing Units Project Magellan project personnel were part of the team of researchers investigating transparent virtualization of Graphics Processing Units (GPGPUs or GPUs) Resources with GPUs have become standard hardware at most HPC centers as accelerator devices for scientific applications. However, current programming models such as CUDA and OpenCL can support GPUs only on the local computing node, where the application execution is tightly coupled to the physical GPU hardware. The goal of the Transparent Virtualization of Graphics Processing Units project, a collaboration between Virginia Tech, the Mathematics and Computer Science Division at Argonne, Accenture Technologies, Shenzhen Institute of Advanced Technology, and the ALCF Magellan staff, was to look at a virtual OpenCL (VOCL) framework that could support the transparent utilization of local or remote GPUs. This framework, based on the OpenCL programming model, exposes physical GPUs as decoupled virtual resources that can be transparently managed independent of the application execution. The performance of VOCL for four real-world applications was evaluated as part of the project, looking at various computation and memory access intensities. The work showed that compute-intensive applications can execute with relatively small amounts of overhead within the VOCL framework.

Virtualization Overhead Benchmarking. The benchmarking of virtualization overheads using both Eucalyptus and OpenStack was performed in collaboration with the Mathematics and Computer Science Division (MCS) at Argonne, which does algorithm development and software design in core areas such as optimization, explores new technologies such as distributed computing and bioinformatics, and performs numerical simulations in challenging areas such as climate modeling, the Advanced Integration Group at ALCF, which designs, develops, benchmarks, and deploys new technology and tools, and the Performance Engineering Group at ALCF, which works to ensure the effective use of applications on ALCF systems and emerging systems. This work is detailed in Chapter 9.

SuperNova Factory. Magellan project personnel were part of the team of researchers from LBNL who received the Best Paper Award at ScienceCloud 2010. The paper describes the feasibility of porting the Nearby Supernova Factory pipeline to the Amazon Web Services environment and offers detailed performance results and lessons learned from various design options.

MOAB Provisioning. We worked closely with Adaptive Computing's MOAB team to test both bare-metal provisioning and virtual machine provisioning through the MOAB batch queue interface at NERSC. Our early evaluation provides an alternate model for providing cloud services to HPC center users allowing them to benefit from customized environments while leveraging many of the services they are already used to such as high bandwidth low latency interconnects, access to high performance file systems, access to archival storage.

Juniper 10GigE. Recent cloud offerings such as Amazon's Cluster Compute instances are based on 10GigE networking infrastructure. The Magellan team at NERSC worked closely with Juniper to evaluate their 10GigE infrastructure on a subset of the Magellan testbed. Detailed benchmarking evaluation using both bare-metal and virtualization was performed and is detailed in Chapter 9.

IBM GPFS-SNC. Hadoop and Hadoop Distributed File System show the importance of data locality in file systems when handling workloads with large data volumes. However HDFS does not have a POSIX interface which is a significant challenge for legacy scientific applications. Alternate storage architecture implementations such as IBM's General Parallel File System - Shared Nothing Cluster (GPFS-SNC), a distributed shared-nothing file system architecture, provides many of the features of HDFS such as data locality and data replication while preserving the POSIX IO interface. The Magellan team at NERSC worked closely with the IBM Almaden research team to install and test an early version of GPFS-SNC on Magellan hardware. Storage architectures such as GPFS-SNC hold promise for scientific applications but a more detailed benchmarking effort will be needed which is outside of the scope of Magellan.

User Education and Support. User education and support have been critical to the success of the project. Both sites were actively involved in providing user education at workshops and through other forums. Additionally, Magellan project personnel engaged heavily with user groups to help them in their evaluation of the cloud infrastructure. In addition, the NERSC project team did an initial requirements gathering survey. At the end of the project, user experiences from both sites were gathered through a survey and case studies, that are described in Chapter 11.

3.2 Advanced Networking Initiative

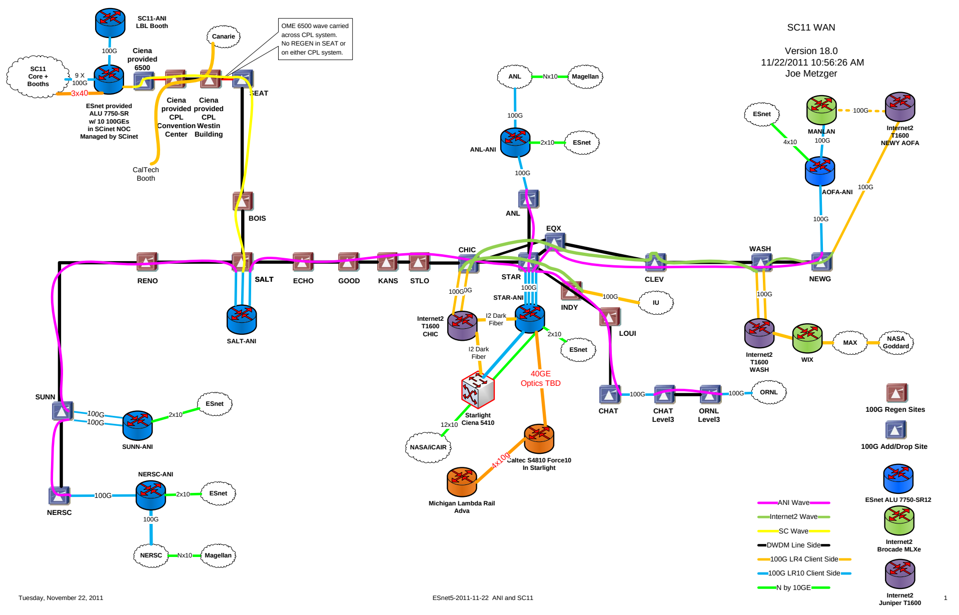


Figure 3.1: Network diagram for the Advanced Networking Initiative during SC11. This includes the Magellan systems at NERSC and ANL(ALCF).

The Advanced Networking Initiative, ANI, is another American Recovery and Reinvestment Act funded ASCR project. The goal of ANI is to help advance the availability of 100Gb networking technology. The project has three major sub-projects: deployment of a 100Gb prototype network, the creation of a network testbed, and the ANI research projects. The latter will attempt to utilize the network and testbed to demonstrate how 100Gb networking technology can help advance scientific projects in areas such as climate research and High-Energy Physics. The Magellan project is supporting ANI by providing critical end points on the network to act as data producers and consumers. ANI research projects such as Climate 100 and OSG used Magellan for demonstrations at SC11 in November 2011. These demonstrations achieved speeds up to 95 Gbps. Please see the ANI Milestone report for additional details about this project and the demonstrations.

3.3 Summary of Project Activities

The remainder of this report is structured around the key project activities and is focused on the research questions outlined in Chapter 1.

- The NERSC survey and requirements gathering from users is summarized in Chapter 4.
- Chapter 5 details the testbed configuration at the two sites. The *flexible* hardware and software stack that is aimed at addressing the suitability of cloud computing to meet the unique needs of science users is highlighted.
- In recent years, a number of private cloud software solutions have emerged. Magellan started with Eucalyptus 1.6.2, but over the course of the project worked with Eucalyptus 2.0, OpenStack, and Nimbus. The features and our experiences with each of these stacks are compared and contrasted in Chapter 6.

- The cloud model fundamentally changes the user support that is necessary and available to end users. A description of the user support model at the two sites is provided in Chapter 7.
- Understanding the security technologies and policies possible in cloud environments is key to evaluating the feasibility of cloud computing for the DOE Office of Science. We outline our efforts in the project and identify the challenges and gaps in current day technologies in Chapter 8.
- Our benchmarking of virtualized cloud environments and workload analysis is detailed in Chapter 9.
- The use of MapReduce for scientific applications and our evaluations is discussed in Chapter 10.
- Case studies of applications are discussed, and the gaps and challenges of using cloud environments from a user perspective are identified in Chapter 11.
- Finally, our cost analysis is discussed in Chapter 12.

Chapter 4

Application Characteristics

A challenge with exploiting cloud computing for science is that the scientific community has needs that can be significantly different from typical enterprise customers. Applications often run in a tightly coupled manner at scales greater than most enterprise applications require. This often leads to bandwidth and latency requirements that are more demanding than most cloud customers. Scientific applications also typically require access to large amounts of data including large volumes of legacy data. This can lead to large startup cost and data storage cost. The goal of Magellan has been to analyze these requirements using an advanced testbed, a flexible software stack, and performing a range of data gathering efforts. In this chapter we summarize some of the key application characteristics that are necessary to understand the feasibility of clouds for these scientific applications. In Section 4.1 we summarize the computational models, and in Section 4.2 we discuss some diverse usage scenarios. We summarize the results of the user survey in Section 4.3 and discuss some scientific use cases in Section 4.4.

4.1 Computational Models

Scientific workloads can be classified into three broad categories based on their resource requirements: large-scale tightly coupled computations, mid-range computing, and high throughput computing. In this section, we provide a high-level classification of workloads in the scientific space, based on their resource requirements and delve into the details of why cloud computing is attractive to these application spaces.

Large-Scale Tightly Coupled. These are complex scientific codes generally running at large-scale supercomputing centers across the nation. Typically, these are MPI codes using a large number of processors (often in the order of thousands). A single job can use thousands to millions of core hours depending on the scale. These jobs are usually run at supercomputing centers through batch queue systems. Users wait in a managed queue to access the resources requested, and their jobs are run when the required resources are available and no other jobs are ahead of them in the priority list. Most supercomputing centers provide archival storage and parallel file system access for the storage and I/O needs of these applications. Applications in this class are expected to take a performance hit when working in virtualized cloud environments [46].

Mid-Range Tightly Coupled. These applications run at a smaller scale than the large-scale jobs. There are a number of codes that need tens to hundreds of processors. Some of these applications run at supercomputing centers and backfill the queues. More commonly, users rely on small compute clusters that are managed by the scientific groups themselves to satisfy these needs. These mid-range applications are good candidates for cloud computing even though they might incur some performance hit.

High Throughput. Some scientific explorations are performed on the desktop or local clusters and have asynchronous, independent computations. Even in the case of large-scale science problems, a number of the data pre- and post-processing steps, such as visualization, are often performed on the scientist's desktop. The increased scale of digital data due to low-cost sensors and other technologies has resulted in the need for these applications to scale [58]. These applications are often penalized by scheduling policies used at supercomputing centers. The requirements of such applications are similar to those of the Internet applications that currently dominate the cloud computing space, but with far greater data storage and throughput requirements. These workloads may also benefit from the MapReduce programming model by simplifying the programming and execution of this class of applications.

4.2 Usage Scenarios

It is critical to understand the needs of scientific applications and users and to analyze these requirements with respect to existing cloud computing platforms and solutions. In addition to the traditional DOE HPC center users, we identified three categories of scientific community users that might benefit from cloud computing resources at DOE HPC centers:

4.2.1 On-Demand Customized Environments

The Infrastructure as a Service (IaaS) facility commonly provided by commercial cloud computing addresses a key shortcoming of large-scale grid and HPC systems, that is, the relative lack of application portability. This issue is considered one of the major challenges of grid systems, since significant effort is required to deploy and maintain software stacks across systems distributed across geographic locales as well as organizational boundaries [30]. A key design goal of these unified software stacks is providing the best software for the widest range of applications. Unfortunately, scientific applications frequently require specific versions of infrastructure libraries; when these libraries aren't available, applications may run poorly or not at all. For example, the Supernova Factory project is building tools to measure the expansion of the universe and Dark Energy. This project has a large number of custom modules [1]. The complexity of the pipeline makes it important to have specific library and OS versions which makes it difficult to take advantage of many large resources due to conflicts or incompatibilities. User-customized operating system images provided by application groups and tuned for a particular application help address this issue.

4.2.2 Virtual Clusters

Some scientific users prefer to run their own private clusters for a number of reasons. They often don't need the concurrency levels achievable at supercomputing centers, but do require guaranteed access to resources for specific periods of time. They also often need a shared environment between collaborators, since setting up the software environment under each user space can be complicated and time consuming. Clouds may be a viable platform to satisfy these needs.

4.2.3 Science Gateways

Users of well-defined computational workflows often prefer to have simple web-based interfaces to their application workflow and data archives. Web interfaces enable easier access to resources by non-experts, and enable wider availability of scientific data for communities of users in a common domain (e.g., virtual organizations). Cloud computing provides a number of technologies that might facilitate such a usage scenario.

4.3 Magellan User Survey

Cloud computing introduces a new usage or business model and additional new technologies that have previously not been applied at a large scale in scientific applications. The virtualization technology that

Table 4.1: Percentage of survey respondents by DOE office

Advanced Scientific Computing Research	17%
Biological and Environmental Research	9%
Basic Energy Sciences	10%
Fusion Energy Sciences	10%
High Energy Physics	20%
Nuclear Physics	13%
Advanced Networking Initiative (ANI) Project	3%
Other	14%

enables the cloud computing business model to succeed for Web 2.0 applications can be used to configure privately owned virtual clusters that science users can manage and control. In addition, cloud computing introduces a myriad of new technologies for resource management at sites and programming models and tools at the user level.

We conducted a survey to understand the requirements and expectations of the user community. We requested NERSC users and other communities in DOE that were interested in cloud computing to detail their application characteristics and expectations from cloud computing. The survey was available through the NERSC Magellan website.

Table 4.1 shows the DOE program offices that the survey respondents belong to. The survey form is included in the appendix. We summarize the details and the results from the survey below.

4.3.1 Cloud Computing Attractive Features

We asked our users to identify the features of cloud computing that were most attractive to them. Users were given several options and were allowed to select more than one category. Figure 4.1 shows the responses selected by our users. Access to additional resources was the most common motivation for most of our users and was selected by 79% of our respondents. A large number of users also wanted the ability to control software environments (59%) and share software or experiment setup with collaborators (52%), which is hard to do in today's supercomputing setup. Additionally, clouds were also attractive due to ease of operation compared to a local cluster (52%) and access to end users of science gateways (48%). A fraction of the users were interested in exploring the use of Hadoop and the MapReduce programming model and the Hadoop File System for their science problems.

In addition, 90% of our survey respondents also mentioned that there were others in their scientific community that were either investigating or interested in investigating cloud computing. A majority of the users also said other collaborators would be able to use their cloud setup.

4.3.2 Application Characteristics

Cloud computing provides a fundamentally different model of resource access than supercomputing centers today. As we explore the applications that might benefit from the cloud environment and consider design decisions, we need to understand the application characteristics better. We asked our users a number of questions about their applications. The users interested in using cloud computing had a varied set of programming models, from traditional supercomputing parallel models, to MapReduce, to custom codes that used one or more of these programming models. Figure 4.2b shows that the application's execution time for a majority of our users was in the order of hours (48%) or minutes (14%). In addition, the memory footprint of a large number of these applications is less than 10 GB, and the bandwidth requirement is largely 1 Gbps today for a single application.

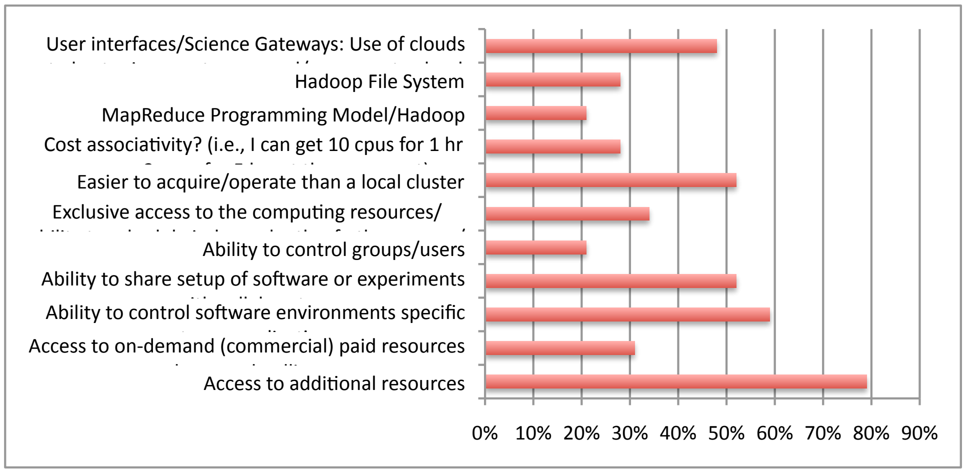
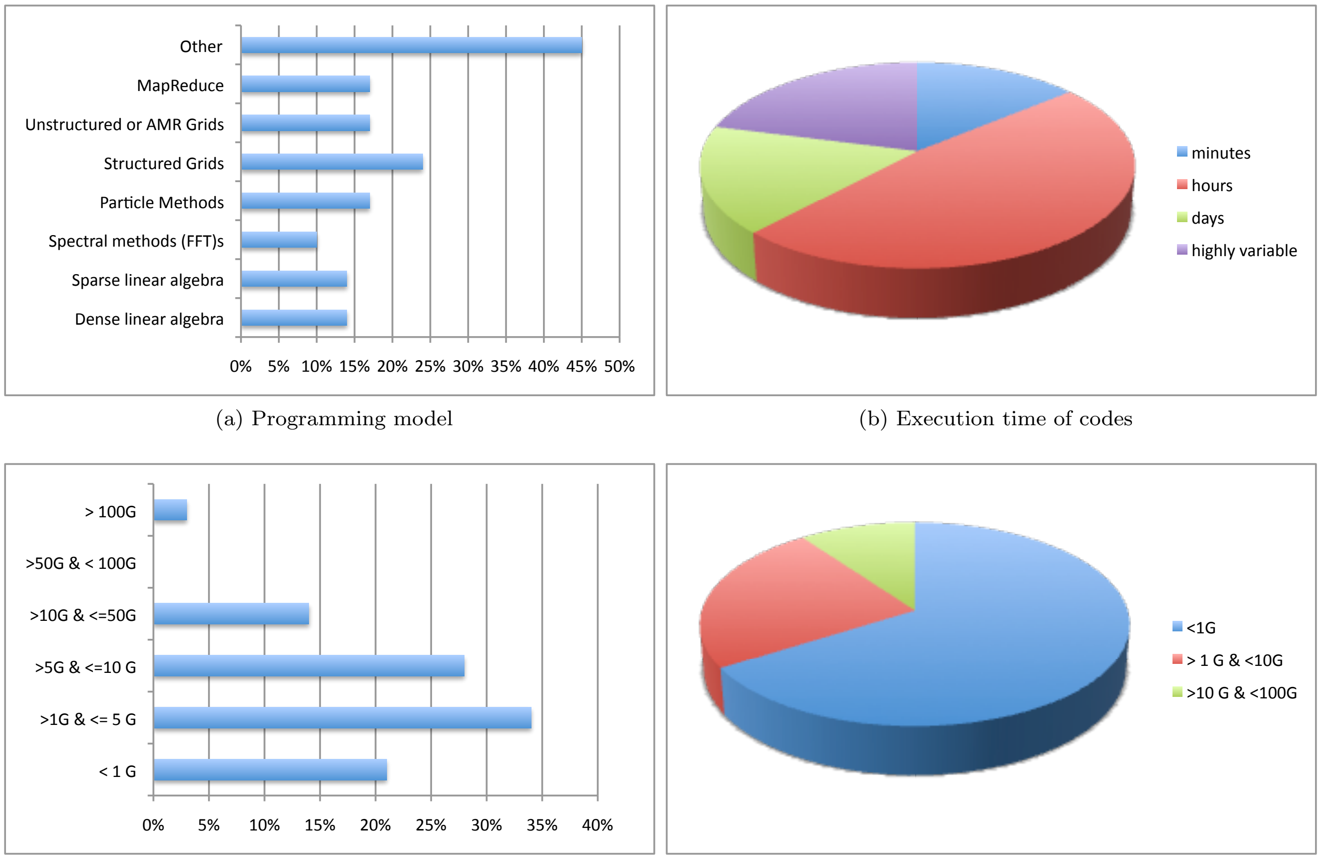


Figure 4.1: Features of cloud computing that are of interest to scientific users.



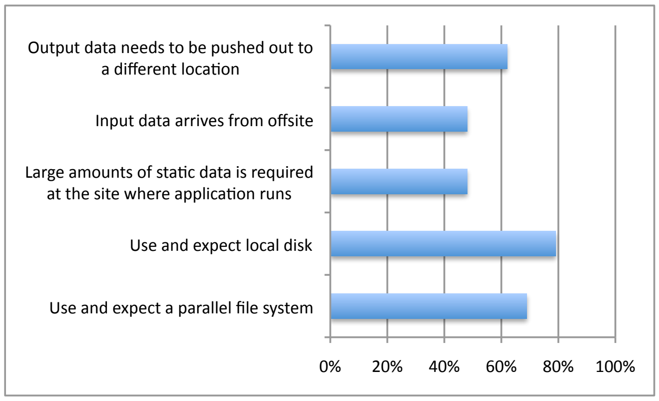
(a) Programming model

(b) Execution time of codes

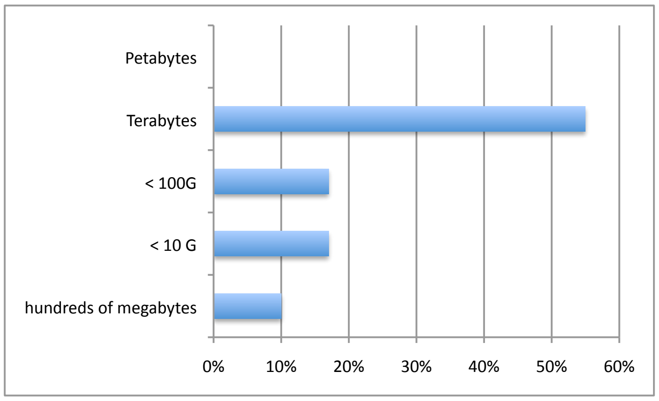
(c) Memory requirements

(d) Bandwidth requirements

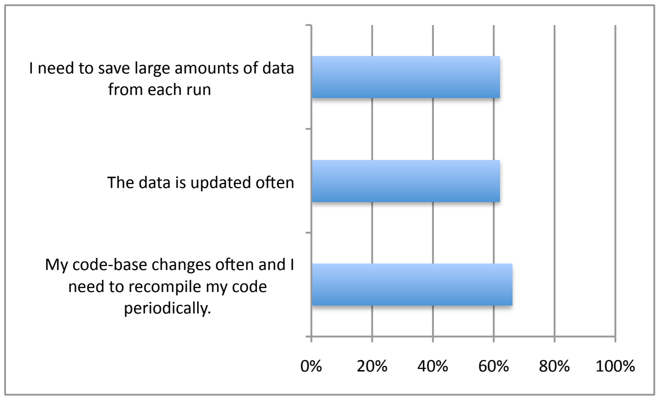
Figure 4.2: Understanding the application characteristics of codes of users who are interested in cloud computing as a platform



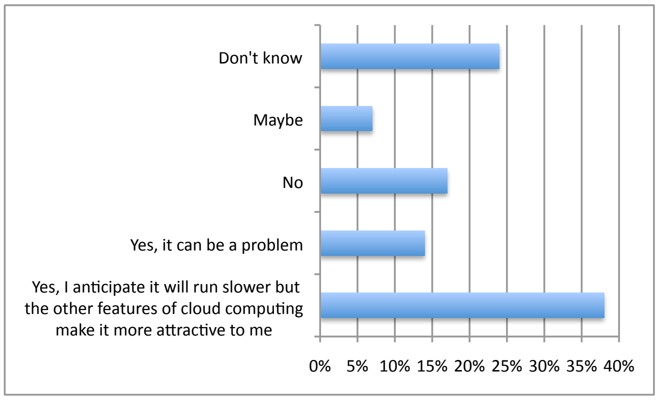
(a) Data characteristics of application



(b) Persistent disk requirements



(c) Characteristics that impact image creation



(d) Performance impact of cloud environments

Figure 4.3: Understanding application characteristics that might impact design decisions in cloud environments

Similarly we polled our users on their data requirements in the cloud. A large number of the scientific applications rely on a parallel file system and expect local disk on the nodes. Applications also have large amounts of data (order of gigabytes or terabytes) that they assume are available at the sites and/or data that arrives from remote sites. While a number of users anticipated the performance of their applications would be adversely impacted by virtualization, many felt that the other advantages of cloud computing still made it an attractive platform for their science. Scientific applications additionally face other challenges when running in cloud environments. Code bases often change and their codes need to be recompiled periodically, requiring new virtual images to be created. Additionally, data is updated often and/or large data from the runs need to be saved since virtual machines do not maintain persistent state.

4.4 Application Use Cases

We followed up the survey with one-on-one discussions with some of the application groups. We summarize the discussions with some of the early cloud adopters and their requirements from Magellan. Many of these users have subsequently run on Magellan resources at both sites and their experiences are described in greater detail in Chapter 11.

4.4.1 Climate 100

Climate scientists increasingly rely on the ability to generate and share large amounts of data in order to better understand the potential impacts of climate change and evaluate the effectiveness of possible mitigation. The goal of the Climate 100 project is to bring together middleware and network researchers to develop the needed tools and techniques for moving unprecedented amounts of data using the Advanced Networking Initiative (ANI) 100 Gbps network. The data for the Climate 100 project consists of on the order of a million files that average about 100 megabytes each. Climate 100 could benefit from cloud environments such as virtual machines and Hadoop to perform large-scale data analysis on the climate data. The volume of data requires coordination of network, compute and disk resources.

4.4.2 Open Science Grid/STAR

The STAR nuclear physics experiment studies fundamental properties of nuclear matter from the data collected at Brookhaven National Laboratory's Relativistic Heavy Ion Collider. The STAR experiment has access to a number of different resource sites that are used for regularly processing experiment data.

STAR experiments are embarrassingly parallel applications (i.e., non-MPI codes) where each job fits in one processor or core. The STAR suite of applications use cases consists of various analysis and simulation programs. The Monte Carlo cases in STAR are well suited to run in cloud environments due to the minimal requirements for data transfer and I/O. The ability to control the software stack in a virtual machine is very attractive to the community due to the complexity of the software stack. The ability to grow in scale during burst periods using existing virtual machine images can greatly enhance scientific productivity and time to solution. The group has previously demonstrated the use of Amazon EC2 resources to process large amounts of data in time for the Quark Matter physics conference. Different groups in the STAR community are interested in cloud computing and virtual machines. The community continues to investigate use of virtual machine images as a way of packaging and distributing software for future experiments.

4.4.3 Supernova Factory

The Supernova Factory project is building tools to measure the expansion of the universe and dark energy. The experiment has a large number of simulations that require custom environments where the end results also need to be shared with other collaborators, and hence there is a need to co-allocate compute and network resources and move the data to storage resources. The Supernova Factory relies on large data volumes for the supernova search, and the code base consists of a large number of custom modules. The complexity of the pipeline makes it necessary to have specific library and OS versions and ends up being a barrier to making use of other large resources. The Supernova Factory project finds cloud computing attractive due to the ability to control software environments and the ability to manage and control user accounts and groups for access to the software. Initial experiments conducted by the group in collaboration with Magellan project personnel on Amazon EC2 show that the cloud is a feasible platform for this application. There is also interest in using Hadoop to coordinate and manage the loosely coupled jobs.

4.4.4 ATLAS

The ATLAS project is investigating the use of cloud platforms to support analysis jobs. The ATLAS project has hundreds of jobs that operate on terabytes of data and can greatly benefit from timely access to cloud resources. The cloud environment also promises to be an effective platform for transitioning scientific codes from testing on the desktop to large-scale cloud resources. The group is investigating the use of virtual machine images for distribution of all required software [10]. This would enable sites to boot the virtual machines at different sites with minimal or no work involved with software management.

4.4.5 Integrated Microbial Genomes (IMG) Pipeline

The Integrated Microbial Genomes (IMG) pipeline at the DOE Joint Genome Institute (JGI) provides analysis of microbial community metagenomes in the integrated context of all public reference isolate microbial genomes. IMG has workloads that need to run periodically every few weeks to months for content maintenance [8]. Timeliness of completion of workloads is critical for the community, and the tremendous growth of these data sets makes access to large number of resources critical. The computational stage consists of functional annotation of individual genes, identification of pair-wise genes, and identification of chromosomal clusters. The most computationally intensive step is performing a BLAST analysis against a reference database. Subsequent steps characterize the genes based on the alignments reported by BLAST. The BLAST output alone is typically over a terabyte. Consequently the analysis of the output to find the top matches and identify taxons can be time consuming and must be done in parallel. There is interest in using technologies such as Hadoop to ease management of these loosely coupled application runs.

Currently the output of "all vs. all" pairwise gene sequence comparisons is stored in compressed files. However, modifying individual entries and querying the data is not easy in this format. The group is interested in exploring the use of HBase for managing data, which will allow the users to update individual rows and perform simple queries.

4.5 Summary

The results of the detailed survey have helped us in understanding the science requirements for cloud environments and have influenced the direction of research in the project. We summarize the requirements gathered from the user survey and corresponding project activities:

- The user requirements for cloud computing are diverse, ranging from access to custom environments to the MapReduce programming model. These diverse requirements guided our *flexible* software stack. Users of Magellan had access to (a) traditional batch queue access with the additional capability of custom software environments through xCAT; (b) customized virtual machines through Eucalyptus or OpenStack front ends, enabling users to port between commercial providers and the private cloud; (c) a Hadoop installation that allowed users to access the MapReduce programming model, the Hadoop Distributed File System and other job management features such as fault tolerance. More details of the software stack are presented in Chapter 5, our Hadoop evaluation is presented in Chapter 10, and our user experiences are summarized in Chapter 11.

- It is important to understand whether commercial cloud platforms such as Amazon EC2 and private cloud software such as Eucalyptus and Hadoop met the needs of the science. We identified the existing gaps, which are summarized in the position paper *Defining Future Platform Requirements for e-Science Clouds*.

- In addition to understanding the gaps, we undertook a benchmarking effort to understand the computational scalability of commercial cloud platforms. These results are presented in Section 9.
- Our users were also interested in a private cloud that would enable them to get the benefits of cloud environments in conjunction with other facilities provided at supercomputing centers. For example, HPC systems typically provide high-performance parallel file systems that enable parallel coordinated writes to a shared file system with high bandwidth and capacity. HPC centers also typically provide an archival storage system to archive critical output and results.
- Cloud computing addresses the portability of the software stack, a known issue with current grid systems. In current supercomputing centers, the sites manage the operating system and common middleware that is needed across multiple groups. Users compile and install their applications on specific systems (often with help from site personnel for optimizations necessary for particular hardware). As we move to the cloud computing model, sites need to be able to provide tooling and support for managing a diverse set of kernels and operating systems that might be required by specific groups. The clear separation of responsibilities for software upgrades and operating system patches no longer exists, and sites will need mechanisms to bridge the gap between supporting user-supported images and site security policies. A cloud system where users have complete control of their images, has certain implications on site security policies. Our security efforts are guided by these custom environments and are discussed in Chapter 8.

Chapter 5

Magellan Testbed

As part of the Magellan project, a dedicated, distributed testbed was deployed at Argonne and NERSC. The two sites architected, procured, and deployed their testbed components separately, although the resources were chosen to complement each other. Deploying a testbed (versus acquiring services on existing commercial cloud systems) provided the flexibility necessary to address the Magellan research questions. Specifically our hardware and software were configured to cater to scientific application needs, which are different from the typical workloads that run on commercial cloud systems. For example, the ability to adjust aspects of the system software and hardware allowed the Magellan team to explore how these design points impact application performance and usability. In addition, the diverse user requirements for cloud computing, ranging from access to custom environments to the MapReduce programming model, led to a requirement for a dynamic and reconfigurable software stack. Users had access to customized virtual machines through OpenStack (at Argonne only) and Eucalyptus (at both sites), along with a Hadoop installation that allowed users to evaluate the MapReduce programming model and the Hadoop Distributed File System. Both OpenStack and Eucalyptus provide an application programming interface (API) that is compatible with the Amazon EC2 API, enabling users to port between commercial providers and the private cloud. Access to a traditional batch cluster environment was also used at NERSC to establish baseline performance and to collect data on workload characteristics for typical mid-range science applications that were considered suitable for cloud computing. The hardware and software deployed within the testbed are described in the following section.

5.1 Hardware

The Magellan testbed hardware was architected to facilitate exploring a variety of usage models and understanding the impact of various design choices. As a result, the testbed incorporated a diverse collection of hardware resources, including compute nodes, large-memory nodes, GPU servers, and various storage technologies. In the fall of 2011, Magellan will be connected to the 100 Gb network planned for deployment by the DOE-SC-funded Advanced Networking Initiative (ANI). A portion of the Magellan resources will remain available after the cloud research is completed in order to support the ANI research projects.

Both Argonne and NERSC deployed compute clusters based on IBM's iDataplex solution. This solution is targeted towards large-scale deployments and emphasizes energy efficiency, density, and serviceability. Configurations for the iDataplex systems are similar at both sites. Each compute node has dual 2.66 GHz Intel Quad-core Nehalem processors, with 24 GB of memory, a local SATA drive, 40 Gb Infiniband (4X QDR), and 1 Gb Ethernet with IPMI. The system provides a high-performance InfiniBand network, which is often used in HPC-oriented clusters but is not yet common in mainstream commercial cloud systems. Since the network has such a large influence on the performance of many HPC and mid-range applications, the ability to explore the range of networking options from native InfiniBand to virtualized Ethernet was an important design goal. The testbed was architected for flexibility and to support research, and the hardware was chosen to be similar to the high-end hardware in HPC clusters, thus catering to scientific applications.

Argonne. The Magellan testbed at Argonne includes computational, storage, and networking infrastructure. There is a total of 504 iDataplex nodes as described above. In addition to the core compute cloud, Argonne's Magellan has three types of hardware that one might expect to see within a typical HPC cluster: Active Storage servers, Big Memory servers, and GPU servers, all connected with QDR InfiniBand provided by two large, 648-port Mellanox switches.

There are 200 Active Storage servers, each with dual Intel Nehalem quad-core processors, 24 GB of memory, 8x500 GB SATA drives, 4x50 GB SSD, and a QDR InfiniBand adapter. The SSD was added to facilitate exploration of performance improvements for both multi-tiered storage architectures and Hadoop. These servers are also being used to support the ANI research projects.

Accelerators and GPUs are becoming increasingly important in the HPC space. Virtualization of heterogeneous resources such as these is still a significant challenge. To support research in this area, Argonne's testbed was outfitted with 133 GPU servers, each with dual 6 GB NVIDIA Fermi GPU, dual 8-core AMD Opteron processors, 24 GB memory, 2x500 GB local disks, and a QDR InfiniBand adapter. Virtualization work on this hardware was outside of the scope of the Magellan project proper; see [48, 86] for details.

Several projects were interested in exploring the use of machines with large amounts of memory. To support them, 15 Big Memory servers were added to the Argonne testbed. Each Big Memory server was configured with 1 TB of memory, along with 4 Intel Nehalem quad-core processors, 2x500 GB local disks, and a QDR InfiniBand adapter. These were heavily used by the genome sequencing projects, allowing them to load their full databases into memory.

Argonne also expanded the global archival and backup tape storage system shared between a number of divisions at Argonne. Tape movers, drives, and tapes were added to the system to increase the tape storage to support the Magellan projects.

Finally, there is 160 terabytes (TB) of global storage. In total, the system has over 150 TF of peak floating point performance with 8,240 cores, 42 TB of memory, 1.4 PB of storage space, and a single 10-gigabit (Gb) external network connection. A summary of the configuration for each node is shown in Table 5.1.

Table 5.1: Node configuration on the Argonne Magellan cluster. There are 852 nodes in the cluster.

Feature	Description
Processor	Dual 2.66 GHz Intel Quad-core Nehalem (704 nodes), Quad Intel Nehalem Quad-core Nehalem (15 nodes), Dual 2 GHz 8-core AMD Opteron (133 GPU nodes)
GPUs	NVidia Fermi 2070 6 GB cards (133 nodes, 2 per node)
Memory	24 GB (837 nodes), 1 TB (15 nodes)
Local Disk	1 TB SATA (504 nodes), 8x500 GB SATA and 4x50 GB SSD (200 nodes), 2x500 GB SATA (148 nodes)
High Performance Network	40 Gb InfiniBand (4X QDR)
Ethernet Network	On-board 1 Gb
Management	IPMI

NERSC. The NERSC Magellan testbed also provided a combination of computing, storage, and networking resources. There are 720 iDataplex nodes as described earlier; 360 of them have local 1 TB drives. In addition, 20x400GB SSDs from Virident were acquired to explore the impact and performance of the SSD technology. The total system has over 60 TF of peak floating point performance. A summary of the configuration for each node is shown in Table 5.2.

The InfiniBand fabric was built using InfiniBand switches from Voltaire. Since the system is too large to fit within a single switch chassis, multiple switches are used, and they are connected together via 12x

Figure 5.1: Photos of the Magellan System at Argonne. The top image shows the front of the compute cloud system as it is being installed. The bottom image shows the second IB rack after deployment of phase 2 hardware.

QDR links (120 Gb/s) configured as a fully connected mesh topology. This topology is less expensive than a traditional, full fat tree network yet still provides a relatively high bisection bandwidth.

In cooperation with vendors, a set of 66 nodes has been equipped in a dual port (IB, 10 GigE) cards and connected through a 10 GigE interface to compare and explore the RoCE protocol performance.

A high-performance scalable storage system was also deployed. The storage is configured to run IBM's high-performance parallel file system, GPFS. The storage hardware consists of four scalable storage units, each with 300 1 TB SATA drives. Each unit is capable of delivering over 5 GB/s of peak bandwidth. There is approximately 240 TB of RAID storage in each scalable unit. The total system provides almost 1 PB of total storage. The existing NERSC SAN was augmented to accommodate the additional storage. Nine Magellan I/O nodes are connected to the SAN and dedicated to providing GPFS services. The file system is mounted across the cluster and service nodes. These file systems are also shared across the center. The service nodes include the network nodes that will be used for testing of the Advanced Networking Initiative (ANI).

NERSC also expanded the archival storage system. A Sun (now Oracle) SL8500 library was installed and integrated with three existing SL8500 libraries. Each SL8500 is capable of holding 10,000 tapes. The library also included 35 T10K-C tape drives. The T10K-C can store 1 TB of uncompressed data on a single tape. Consequently, a single SL8500 library can potentially store up to 10 PB of uncompressed data. The hardware is managed by the High-Performance Scalable Storage (HPSS) software [42].

Table 5.2: Node configuration on the NERSC Magellan cluster. There are 720 nodes in the cluster.

Feature	Description
Processor	Dual 2.66 GHz Intel Quad-core Nehalem
Memory	24 GB (48 GB in 160 nodes)
Local Disk	1 TB SATA (in 360 nodes)
High Performance Network	40 Gb InfiniBand (4X QDR)
Ethernet Network	On-board 1 Gb
Management	IPMI

5.2 Software

Flexibility was a major objective in selecting the components of the software stack at the two sites. Since Magellan was to be used to explore a variety of cloud service models and programming models, it was critical that the software stack enable the system to be quickly reconfigured. Both sites used configuration tools often seen in the HPC computing world to provide flexible management of the software stacks. These tools allowed the sites to provision nodes in the cluster from traditional batch environments to Amazon EC2-like models to Hadoop-based MapReduce clusters with the goal of exploring how the software can be used to dynamically re-provision hardware between the different environments based on demand, energy consumption, utilization, as well as other factors.

Both sites deployed Hadoop as well. Hadoop has been attracting the attention of multiple scientific communities. From the user survey and discussions with users, Hadoop was cited as an area of high interest.

Argonne. The Argonne Magellan testbed uses the Argonne-developed tools bcfg2 [18, 17] and Heckle to provide advanced, bare-metal provisioning ("Hardware as a Service" or HaaS) and configuration management. Heckle allows users to dynamically provision different OS images across raw hardware quickly and easily, providing all the benefits of custom operating systems without the overheads and challenges of virtualizing specialized resources such as accelerators or high-performance networks.

Bcfg2 is a system management tool that provides facilities to build sophisticated configurations across large numbers of compute nodes. In the context of providing environments for HPC users on cloud resources,

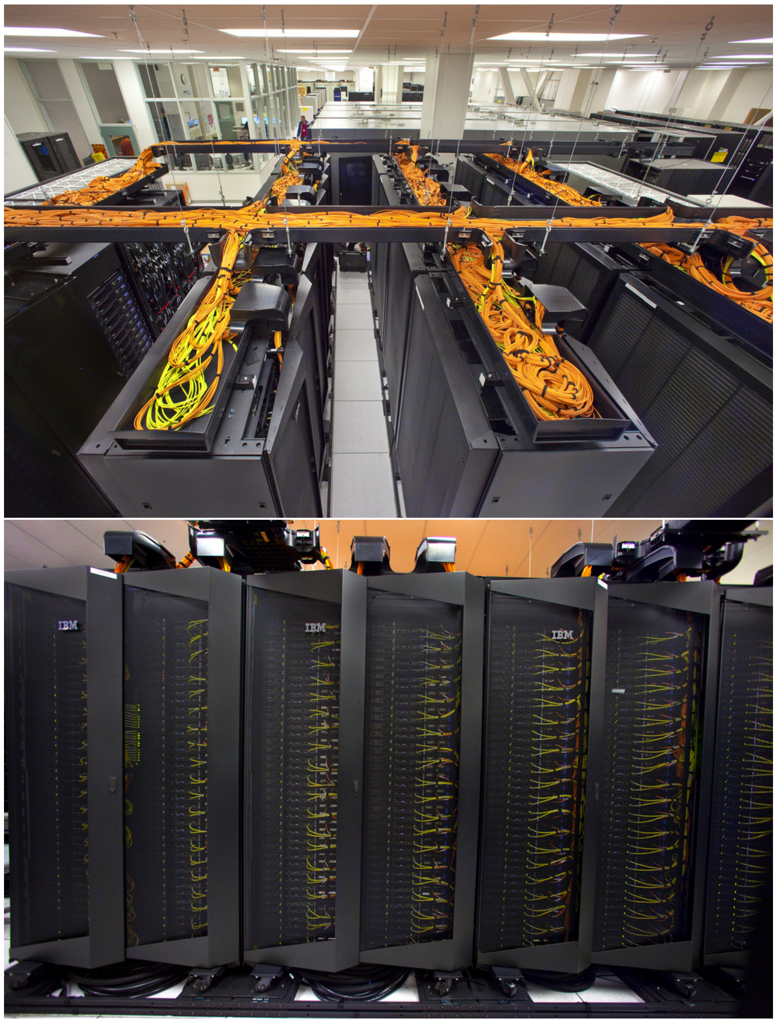


Figure 5.2: Photos of the Magellan System at NERSC. The top image shows the overhead cabling system which is suspended from the ceiling to seismically isolate the racks from each other. The lower image shows three racks of compute nodes. The seismic isolation platforms are visible below the racks.

it fills the critical role of building customized per-user configurations in a flexible and inexpensive fashion. Heckle is a low-level node provisioning tool developed by Argonne as part of the Magellan project. It enables non-virtualized computational resources to be built on demand with user-provided configuration specifications. For example, users can request 10 nodes running a RedHat image; much as a cloud does, these nodes are built automatically with the user's image and the user receiving root access. Upon job completion, the nodes are deactivated, in preparation for the next user and associated software build. Both Bcfg2 and Heckle are open-source, developed at Argonne, and form the basis for system management efforts on Magellan and elsewhere (used to manage several diverse testbed systems at Argonne, as well as used by many institutions around the world, from commercial companies such as LinkedIn to other national laboratories such as Oak Ridge National Laboratory). IBM's xCAT cluster management software [85] was used to provide access to the IPMI management interface to automatically power on and off nodes.

The core compute servers had numerous cloud software stacks installed in various stages of availability to the users during the project. These included Eucalyptus 1.6 and 2.0, OpenNebula, OpenStack, Ubuntu Enterprise Cloud (UEC), and Nimbus. By the end of the project, these had converged to a large OpenStack public cloud, a small Nimbus cloud, and a small development OpenStack cloud. The public clouds were open to all users for science, development, and testing. Deploying many different cloud software stacks allowed Argonne to explore the state of open-source cloud software, and to evaluate status for usability, deployment, and support models. A portion of the Active Storage servers were configured as a persistent Hadoop cluster. A 66-node Hadoop cluster was provided to the users at Argonne. Although there was not heavy usage, it did allow the users to explore the Hadoop model for their applications. The remainder were configured as HaaS and were part of the raw provisioning pool, along with the GPU and Big Memory servers.

NERSC. NERSC is using xCAT and Adaptive Computing's Moab Adaptive Computing Suite to provision and manage the Magellan testbed. xCAT can provision nodes using a variety of methods, including diskless, disk-full, and hybrid. NERSC also utilizes the IPMI management interface to automatically power on and off nodes. In addition to providing command-line tools, xCAT also provides an API that can be used by external resource managers such as Moab from the Adaptive Computing Suite by Adaptive Computing.

The Adaptive Computing Suite product represents Adaptive Computing's foray into the cloud computing arena. It supplies a set of tools that can be used to manage private clouds by providing an array of capabilities including integration with xCAT, green computing capabilities, and support for virtual private clusters.

NERSC explored some of the aspects such as integration with xCAT that enables Moab to automatically provision nodes based on user job requirements. Moab can do bare metal provisioning with xCAT or use xCAT's hypervisor extensions to provision virtual machines. Both types of provisioning have been exercised; however, bare-metal proved to be more production-ready. Moab can also interact with xCAT to power off nodes when they are idle. While NERSC traditionally runs with very high utilization, this capability is still interesting to explore in the cloud computing context.

Virtual Private Clusters (VPCs) offer another service model for offering compute resources. A VPC can include a suite of services and compute elements to create an entire cluster. For example, a VPC might include a head node running a private job scheduler, a database server, and a scalable number of compute nodes. Eventually, VPCs could be integrated with network services to create a virtual private network connection to other resources.

NERSC deployed Eucalyptus 1.6 at first on half a rack (40 nodes) and upgraded to 2.0 as soon as the new version became available. Later on, due to popular demand from a small but committed user base, the cluster got expanded to include 80 nodes. The cluster size remained small enough to avoid exposing major scalability problems. Taking into account the steep learning curve and the novelty of the cloud technology, a decision was made to remain with the Eucalyptus software stack and allow users to fully explore its capabilities. We have had multiple development resources, including another Eucalyptus 2.0 cluster equipped in a 10 GigE network, to compare the impact of networking on the performance of virtual clusters.

Since core user-side tools do not support parallel computing, NERSC provided users with a set of scripts bundled into *VirtualCluster* module. This module gave users a simple interface to build a virtual cluster with a head node equipped with a Torque batch system.

NERSC deployed Hadoop from Cloudera in a configuration that varied between 80 and 200 nodes to explore scalability and various storage media (local drives, SSDs). In addition, other tools from the Hadoop ecosystem such as Pig (a platform for analyzing large data sets), Chukwa (data collection system for monitoring large distributed systems), HBase (scalable, distributed database that supports structured data storage for large tables) and Hive (data warehouse infrastructure that provides data summarization and ad hoc querying) were made available to the users.

Chapter 6

Virtualized Software Stacks

Virtualized cloud environments have become nearly synonymous with clouds. While the NIST definition of clouds does not explicitly mention virtualization, infrastructure-as-a-service (IaaS) implementations typically rely on virtualization to enable users to customize the environment, protect users from each other, and enable multi-tenancy on resources to improve resource utilization. Amazon, one of the most established providers of IaaS, has developed a set of web-based APIs for controlling virtualized resources. The popularity and flexibility of these APIs has led to several open-source implementations of software to operate private clouds. At the start of the project, Eucalyptus was the most established and sophisticated of the various options. Since then, other offerings such as OpenStack, have appeared and gained traction. In the Magellan project, we evaluated various software stacks including Eucalyptus, OpenStack and Nimbus. In this section, we detail our experiences with these software packages.

6.1 Eucalyptus

The Eucalyptus project provides an open source platform for conducting cloud computing research. It is API-compatible with Amazon's EC2. Eucalyptus supports nearly all the core features of EC2, including creating virtual instances, elastic block storage (EBS), S3, and elastic IP addresses. Since the software is open-source, it is possible to make modifications and add hooks and callouts to gather data. This makes it a useful tool for exploring virtualized cloud environments. Eucalyptus provides a convenient platform for creating and managing a private cloud platform for multiple users. However, Eucalyptus, like much of the emerging cloud computing software, is still evolving and has design principles that may not be compatible with supercomputing center policies. In addition, bugs of various flavors were encountered.

Architecture. The Eucalyptus architecture has a number of services including Cloud Controller (CLC), Cluster Controller (CC), Walrus, Storage Controller (SC) and Node Controller (NC) (Figure 6.1). The Node Controller runs on the compute nodes whereas the other services might run on one or more nodes. In addition, Eucalyptus permits a cloud system to include a number of cluster controllers. Our deployment was limited to a single cluster controller that exposes a number of scaling problems. As shown in Figure 6.1, the Eucalyptus network model forwards all traffic from the virtual machines running in each cluster through the cluster controller. This setup allows the system to implement security groups more easily; however, it also creates a potential network bottleneck for the running virtual machines. Even a moderate amount of network traffic spread across many virtual machines can saturate the cluster controller's network bandwidth. Testing of the virtual machine capacity revealed that Eucalyptus had a limit to the number of simultaneously running virtual machines, between 750 and 800 instances. The limit is related to the message size limit in the communication protocol and enforces a hard limit on the number of simultaneously running virtual machines. Also, many of the cluster controller operations appeared to be serialized, resulting in certain operations taking longer to complete when additional node controllers were added. When the cluster was

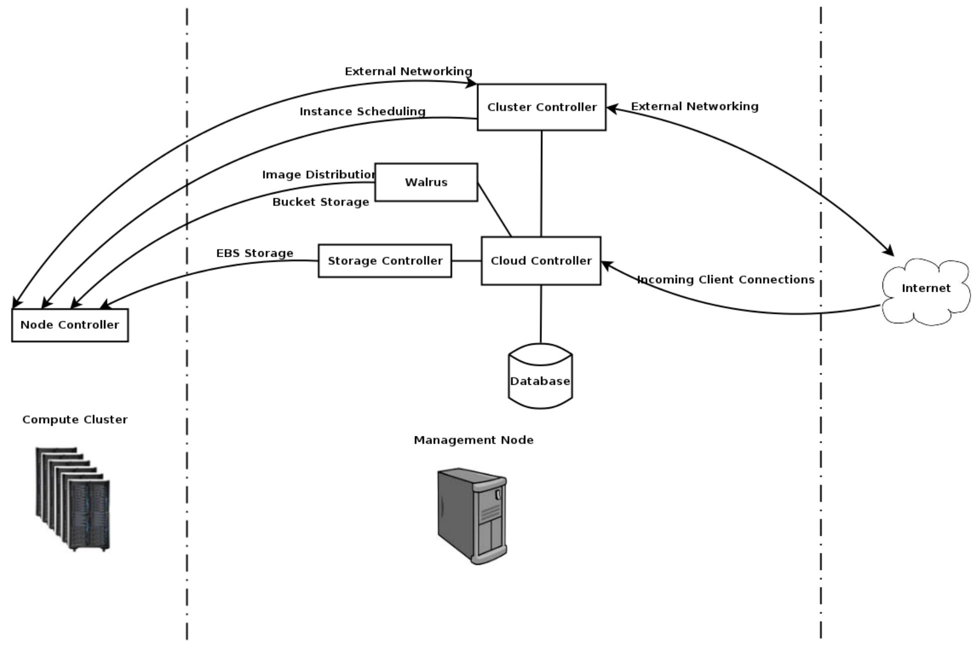


Figure 6.1: Architecture diagram for Eucalyptus cloud software stack.

expanded to more than 200 node controllers, there was a noticeable deterioration in performance for certain operations. For instance, terminating large numbers of virtual machines instances caused delays for other operations such as creating a new virtual machine instance.

Image Management. In today's environments, sites manage the operating system and some base software packages, and scientists manage the specific application software and data. This division clearly demarcates the responsibility of system and application software maintenance between the site system administrators and the users. In the virtual machine model, users also need to have an understanding of the OS administration. Support lists at both sites received a large quantity of e-mails that pertained to basic system administration questions. It was clear that this additional requirement for the users would impact the user support model for a site. Users need to know how to create, upload, and register images, and have system administration skills for the OS in the virtual machine. This is a problem for IaaS in general, not just Eucalyptus.

For Eucalyptus, although users manage their virtual machine images, the kernel and ramdisk are registered and managed by the system administrators, allowing sites to control kernel versions. While this can improve security and stability it can also create additional overhead for managing the cloud system if a large number of kernels and OS versions are required.

Infrastructure. Eucalyptus uses a number of system services such as DHCP. It is difficult to get Eucalyptus to co-exist with other services running on the system, since Eucalyptus and other similar cloud software expect to have complete control over system software in the cluster.

Allocation, Accounting and Scheduling Policies. The open source version of Eucalyptus does not provide an allocation and accounting model. Thus, there is no simple mechanism to ensure fairness or enforce any allocation policies among a group of users who might be competing for resources. Batch queue systems such as Moab, PBS, etc. which are typically used in HPC centers provide customizable policy mechanisms and scheduling algorithms to limit and prioritize access to resources. Another challenge for Eucalyptus and other similar cloud stacks is the inability to queue requests to run in the future if resources aren't available.

While both private and public clouds may strive to be "unlimited", in practice this is impractical. Having a mechanism to queue requests is useful for many scientific application use cases.

Logging and Monitoring. Eucalyptus has verbose logging that could be useful for tracking events occurring on the systems. However, monitoring capabilities to understand the behavior of the virtual machines, detect failures, and rectify failure events at the service level are limited. Early versions of Eucalyptus (i.e. 1.6.2) also had limited ability to recover gracefully from system-level failures. For example, restarting the Cloud Controller would typically result in the loss of IP address assignments for running instances and require all running instances to be terminated in order to fully recover.

Portability. Eucalyptus provides an Amazon-compatible user API to manage virtual machines. This enables easy portability of tools and applications between the public and private cloud. However, moving images between the two systems still requires a fair amount of IT expertise and can be time consuming and tedious.

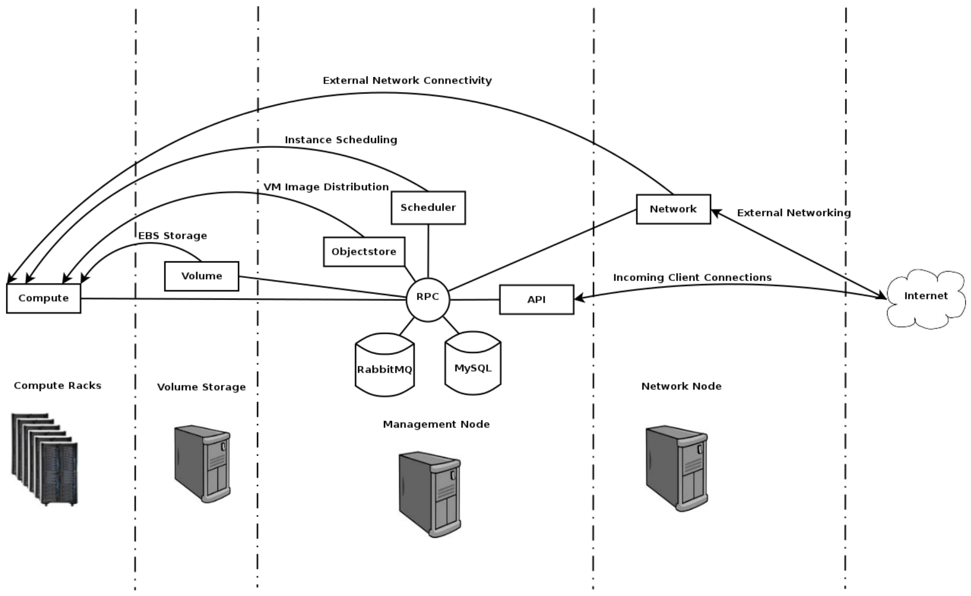


Figure 6.2: Architecture diagram for OpenStack cloud software stack.

6.2 OpenStack Nova

OpenStack is a fully open-sourced joint project between the National Aeronautics and Space Administration (NASA) and Rackspace that implements compute and storage components. OpenStack offers greater flexibility for a heterogeneous physical infrastructure, and provides extensive support for customization and configuration. While OpenStack is relatively new, a number of organizations and vendors have adopted OpenStack as their preferred cloud management technology, and the size and activity of their user community has grown. In less than a year, the number of dedicated developers on the project has expanded more than tenfold, and they remain accessible to the user community via various methods.

A number of different versions of OpenStack were tested during the Magellan project. Like Eucalyptus, OpenStack is still very much in a development stage, and successfully installing and running this software in our environment required a significant amount of modification to both the code and the underlying system and network architecture. The effort expended yielded significant results, including instance uptimes of greater than three months that were not possible with earlier versions. Many of our users commented favorably on the stability and performance of the system. Newer releases show even more promise, as developers continue to add features and improve stability.

Infrastructure. OpenStack's management infrastructure (shown in Figure 6.2) is comprised of a number of individual services which communicate with each other via asynchronous message passing and a MySQL database. The API service handles remote client interaction and the serving of instance metadata to hypervisor nodes. The object store and volume services manage the storage and access of virtual machine images and EBS volumes, respectively, and the scheduler handles the allocation of compute resources to spread load evenly across the hypervisor cluster. The networking service is responsible for all networking tasks, both local and externally-routable, and the compute service runs on the hypervisors themselves, performing the local management of individual KVM instances.

User Management. All users on an OpenStack cluster belong to one or more projects, and are assigned roles both globally and within their projects. These roles govern their level of access, with the union of global and project roles determining the precise privileges. Each user/project combination requires its own set of client credentials e.g., a user belonging to three projects will require three different sets of credentials. Each project is given its own local subnet and has project-wide instance firewall rules and access privileges to published images and EBS volumes. The Nova management tools allow administrators to apply quotas to user projects, limiting the number of instances a user may have allocated simultaneously, or the number or total size of volumes they may create. User interaction with the OpenStack API is accomplished via the eucatools client in the same manner as Eucalyptus; as such it is API-compatible with both Eucalyptus and Amazon EC2.

Image and Volume Management. User virtual machine images are managed by the object store service, which is an Amazon S3-like bucket storage application dedicated solely to storing and serving images. Image access privileges are project-specific, meaning that should a user's roles within a project allow it, they have access to use and modify all of the project's private images. Unlike Eucalyptus, users are also allowed to register kernels and ramdisk images; making customization of existing images and addition of new ones far easier for the user and reducing the support burden on the administrators. EBS volumes are handled by the volume service, which creates, stores, exports and deletes iSCSI volumes. All volume privileges are bound to the user.

Machine Virtualization and Scheduling. The OpenStack compute service runs on each hypervisor node, and is responsible for the creation and destruction of virtual machine instances and all associated hypervisor-local resources, such as project-LAN networking and virtual disks. The underlying virtualization technology in use at Argonne is KVM, though others are supported as well, such as Xen, UML, and the relatively new LXC. The hypervisor for a group of instances is controlled by the Scheduling service, which spreads the compute load evenly across the hypervisor cluster. The versions of OpenStack used by the Magellan project did not support live migration, and compute resources are acquired by the first user to request them. This has the effect of spreading even small single-core instances across the hypervisor cluster and subsequently blocking a number of large 8-core jobs from running.

Accounting. All information pertinent to instances launched on an OpenStack cluster is recorded in the OpenStack database. The information in the database is available to administrators. For example, the database includes the date and time of instance creation and destruction, the user and project responsible for the instance, the current state of the instance, the image from which the instance was created.

Network. All networking for OpenStack instances is managed by a Network service. Both project-LAN and externally-routable IPs are assigned to instances via this service. Each project-LAN is given its own VLAN in order to separate the broadcast domains of projects from each other. DHCP and an intricately designed NATing chain is used to accomplish this task. Externally-routable addresses are broadcast to a border router by the networking service, and are then 'attached' to the requested virtual machine via additional network address translation rules.

As all networking is handled by one service, this introduces a performance bottleneck for communication-intensive applications. Furthermore, the complexity of the networking setup is quite large, and requires a significant amount of effort to understand. Some inflexibility with regard to support of physical network architecture were encountered in the design as well.

Scalability. Overall, OpenStack scales well, and performance on the 420 node cloud at ALCF was reliable up to per-project instance counts of more than 200. Instance counts of 300 or greater have been successfully accomplished, though at that scale the single API node becomes overloaded and instance launch times increase dramatically.

6.3 Nimbus

Nimbus is an open-source IaaS cloud-computing framework, created by a team of developers at Argonne National Labs and the University of Chicago. It has been successfully deployed on a number of sites around the country, one of which is an 82 node cloud running on Magellan project hardware at Argonne.

Internals. Nimbus supports the creation of virtual machines using both Xen and KVM, the latter being used on the Magellan cloud. The core services include the Workspace Service, which handles all virtual machine management and user interaction via grid-credential or Amazon EC2 API. The configuration clients run on individual virtual machines, and control setup of services, user and virtual cluster credentials, and image metadata acquisition. The *cloudinit.d* script is one of these clients, so user images can easily be modified for compatibility with Amazon EC2, similar to Eucalyptus and OpenStack.

Running Nimbus on Magellan. The two biggest challenges encountered running Nimbus on Magellan hardware involved issues with networking and software dependencies. The network issues revolved primarily around the security measures present on site. Difficulties with ingress firewall settings caused user ssh sessions to timeout prematurely, and complicated the application of public IP addresses to virtual machines; this prompted one user to eschew the use of externally-routable IPs altogether. Taken together, these issues made management of the cloud difficult; and even simple administrative tasks required exorbitant effort.

In order to alleviate the aforementioned networking difficulties, an image propagation protocol called LANTorrent was developed. LANTorrent is a multicast system that allows researchers to quickly distribute VM images across the cloud at one time; tests on Magellan resulted in 1000 simultaneous boots. LANTorrent forms a store and forward chain of recipients that allows the image to be sent at a rate much faster than can be accomplished with http or ssh transfers. This significantly lessened the time a user's cluster required to boot.

6.4 Discussion

In the previous section, we detailed current cloud offerings. Here we examine some of the gaps and challenges in using existing cloud offerings directly for scientific computing.

Resources. Cloud computing carries with it the assumption of an unlimited supply of resources available on demand. While this was true in early days of cloud computing when demand for resources was still ramping up, more recently users have noticed that their requests have not been satisfied on providers such as Amazon EC2 due to insufficient capacity. This situation is similar to current day supercomputing and grid resources that are often over-subscribed and have long wait queues. Thus for the user, scientific cloud computing as an unlimited supply of cycles tends to be less promising. There is a need for differentiated levels of service similar to Amazon's current offerings but with advanced resource request interfaces with specific QoS guarantees to avoid users needing to periodically query to see if resources have become available.

Portability. The vision of grid computing has been to enable users to use resources from a diverse set of sites by leveraging a common set of job submission and data management interfaces across all sites. However, experience revealed that there were challenges due to differing software stacks, software compatibility issues, and so on. In this regard, virtualization facilitates software portability. Open source tools such as Eucalyptus and OpenStack enable transition from local sites to Amazon EC2, but site-specific capabilities and cloud interfaces in general are diverse, often making it difficult to easily span multiple sites. In addition, cloud-software stack's networking models and security constraints make the cost of data movement to and especially from the cloud relatively expensive, discouraging data portability. These costs are also an issue in situations where scientists would like to perform post-processing on their end-desktop or local clusters, or would like to share their output data with other colleagues. More detailed cost discussions are presented in Chapter 12.

Performance. Traditional synchronous applications which rely on MPI perform poorly on virtual machines and have a huge performance overhead, while application codes with minimal or no synchronization, modest I/O requirements, with large messages or very little communication tend to perform well in cloud virtual machines. Traditional cloud computing platforms are designed for applications where there is little or no communication between individual nodes and where applications are less affected by failures of individual nodes. This assumption breaks down for large-scale synchronous applications (details in Chapter 9).

Additionally, in the cloud model time-to-solution is a more critical metric than individual node performance. Scalability is achieved by throwing more nodes at the problem. However, in order for this approach to be successful for e-Science, it is important to understand the setup and configuration costs associated with porting a scientific application to the cloud - this must be factored into the overall time-to-solution metrics in resource selection and management decisions.

Data Management. As discussed earlier, scientific applications have a number of data and storage needs. Synchronized applications need access to parallel file systems. There is also a need for long-term data storage. None of the current cloud storage methods resemble the high-performance parallel file systems that HPC applications typical require and currently have accessible to them at HPC centers. General storage solutions offered by cloud-software stacks include EBS volumes and bucket store, which at the time of this writing are inadequate for scientific needs, on both performance and convenience.

6.5 Summary

The open-source cloud software stacks have significantly improved over the duration of the project. However, these software stacks do not address many of the DOE security, accounting and allocation policy requirements for scientific workloads. Thus, they could still benefit from additional development and testing of system and user tools.

Chapter 7

User Support

Many of the aspects of cloud computing that make it so powerful also introduce new complexities and challenges for both users and user support staff. Cloud computing provides users the flexibility to customize their software stack, but it comes with the additional burden of managing the stack. Commercial cloud providers have a limited user support model and typically additional support comes at an extra cost. This chapter describes the user support model that was used for the Magellan project, including some of the challenges that emerged during the course of the project. We discuss the key aspects of cloud computing architecture that have bearing on user support. We discuss several examples of usage patterns of users and how these were addressed. Finally, we summarize the overall assessment of the user support process for mid-range computing users on cloud platforms.

7.1 Comparison of User Support Models

HPC centers provide a well-curated environment for robust, high-performance computing, which they make accessible to non-expert users through a variety of activities. In these environments, substantial effort is put into helping users to be productive and successful on the hosted platform. These efforts take a number of forms, from building a tuned software environment that is optimized for HPC workloads, to user education, and application porting and optimization. These efforts are important to the success of current and new users in HPC facilities, as many computational scientists are not necessarily deeply knowledgeable in terms of the details of modern computing hardware and software architecture.

HPC centers typically provide a single system software stack, paired with purpose built hardware, and a set of policies for user access and prioritization. Users rely on a relatively fixed set of interfaces for interaction with the resource manager, file system, and other facility services. Many HPC use cases are well covered within this scope; for example, this environment is adapted for MPI applications that perform I/O to a parallel file system. Other use cases such as high-throughput computing and data-intensive computing, may not be so well supported at HPC centers. For example, computer scientists developing low level runtime software for HPC applications have a particularly difficult time performing this work at production computing centers. Also, deploying Hadoop on demand for computations, could be performed within the framework of a traditional HPC system, albeit with significant effort and in a less optimized fashion.

Cloud systems provide Application Programming Interfaces (API) for low level resource provisioning. These APIs enable users to provision new virtual machines, storage, and network resources. These resources are configured by the user and can be built into complex networks including dozens, hundreds, or potentially even thousands of VMs with distinct software configurations, security policies, and service architectures. The flexibility of the capabilities provided by cloud APIs is substantial, allowing users to manage clusters built out of virtual machines hosted inside of a cloud. This power comes at some cost in terms of support. The cloud model offers a large amount of flexibility, making it difficult and expensive to provide support to cloud users. The opportunities for errors or mistakes greatly increase once a user begins to modify virtual machine configurations. User expertise plays a role here, as it does on HPC systems; some users are comfortable building new configurations and designing system infrastructure while many others are not.

This difference in capabilities demonstrates an important set of trade-offs between cloud and HPC system support models; specific purpose built APIs (like those provided by HPC centers) can provide deep support for a relatively fixed set of services, while general purpose APIs (like those provided by IaaS cloud systems) can only provide high level support efficiently. The crux of the user support issue on cloud systems is to determine the level of support for various activities ranging from complete to partial support.

7.2 Magellan User Support Model and Experience

Both Magellan sites leveraged their extensive experience supporting diverse user communities to support the Magellan user groups. In addition, the Argonne Magellan user support model benefited from extensive experience supporting users of experimental hardware testbeds. These testbeds are similar to clouds in that users often need administrative access to resources. This experience enabled us to understand the tradeoff between depth of support and range of user activities required for private cloud environments. Both sites also heavily leveraged existing systems to manage mailing lists, track tickets, and manage user accounts. Our user support model consisted of five major areas: monitoring system services, direct support of the high level tools and APIs provided by the system, construction and support of a set of baseline VM configurations, training sessions, and building a community support base for more complex problems. We will discuss each of these areas.

Monitoring. Monitoring is a critical component for providing any user service; even if the services are being offered as a test bed. While cloud APIs provide access to more generic functionality than the traditional HPC system software stack, basic testing is quite straightforward. An initial step to monitor system operational status was to extend the existing monitoring infrastructure to cover the cloud systems. At ALCF, an existing test harness was extended in order to run system correctness tests on Magellan easily and often. The test harness ran a variety of tests, from allocating VM instances to running performance benchmarks. These tests confirmed that the system was performing reliably and consistently over time. A similar approach was used at NERSC, where tests were run on a routine basis to ensure that instances could be spawned, networking could be configured, and the storage system was functioning correctly. These tests enabled the centers to proactively address the most routine problems.

Cloud setup support. The second component of the support model was helping users make use of the basic cloud services provided by Magellan. These activities included generating users credentials for access to the cloud services and providing documentation to address routine questions about how to setup simple VM instance deployments, storage, networks, and security policies. This approach provided enough help for users to get started, and is analogous in an HPC setting to ensuring that users could login to a system and submit jobs.

Image support. Our initial assessment suggested that supporting more complex user activities would be prohibitively expensive, so we opted to take a different approach. Instead of direct support, we provided a number of pre-configured baseline configurations that users could build on, without needing to start from scratch. We provided a number of starting VM instance configurations that were verified to work, as well as several recipes for building common network configurations and storage setups. This approach was relatively effective; users could easily build on the basic images and documented examples. NERSC also provided tools to automate building virtual clusters with pre-configured services such as a batch system and an NFS file system.

Documentation and Tutorials. Both sites also provided tutorials and online documentation to introduce users to cloud models and address the most common questions and issues. Tutorials were organized for both

Hadoop and Eucalyptus users. These sessions primarily provided users with an overview of the basic steps to using these technologies and provided an opportunity to discuss some of the details of how to port applications to these environments. For example, ALCF held a "Welcome to Magellan Day" workshop that ran a full day and included 4 hours of intensive hands-on training in using Eucalyptus, building and managing images, and debugging images. This workshop demonstrated the difficulties in training scientific users to become, in essence, system administrators.

Community Support Model. For advanced user activities, we relied on a community support model. In order for community support to work, it needs a critical mass of experienced users willing to share their expertise. Building a community of experienced users requires significant outreach and training, as well as an extended period of time to develop this community support base. In addition, it needs to be clear when issues are system problems versus a user issue. Early in the project, the cloud software stacks were sufficiently immature that this process was difficult. Aside from these issues, the community support model worked reasonably well. Successful groups often had a primary user who was either already experienced or gained the expertise to build the infrastructure for the rest of the group to use. While this approach is not always feasible, it worked particular well in many cases. For example, one Argonne scientific team was able to build a complete cluster inside of the ALCF Magellan public cloud space, including the Sun Grid Enging (SGE) batch queueing system and storage, which also had secure access to kerberized NFS resources outside the system. They were able to route jobs into the system for execution. The team was able to successfully achieve this complicated configuration because one member of their team had extensive system administration expertise.

7.3 Discussion

Many of the problems users faced on both the virtual and hardware-as-a-service clusters involved Linux administrative tasks. Most HPC users have little systems administration experience, so even relatively minor issues frequently required support from Magellan staff. Often, this could be accomplished by sharing advice or links to online documentation via email, but at other times it required launching rescue images to diagnose and repair a damaged system. Installing software and debugging system errors were often necessary to kick-start a user's efforts. This process was complicated further when the users chose Linux distributions outside of the baseline images which the Magellan staff had pre-configured.

Magellan staff also handle challenges related to access to commercial software or other common tools or infrastructure. For example, some users were accustomed to using proprietary software which could not always be bundled into a virtual machine without violating license agreements. If a user had the appropriate license, they had the freedom to install and customized the images to include the software, but this often led to issues with supporting modified images.

Security related challenges were also commonplace. Many of these issues revolved around user credentials. We discuss additional security challenges in Chapter 8. On the virtual cluster, there were typically problems with the users' client-side tools installation or SSL libraries. For example, only certain versions of the client-side ec2 tools work with Eucalyptus. At ALCF, almost all of the users had some initial difficulty using the cryptographic tokens which were required to access the bare-metal provisioning service. While these were often easily resolved, they constituted a significant portion of the total support requests for this service. Another common issue was with externally-routable IPs and firewall settings on the instances, which by default block all incoming connections. Again these were easily resolved, but increased the support load.

The final challenges that users often encountered were related to managing data and workflows (discussed in Chapter 11). The solution to these challenges are often application specific requiring a detailed understanding of the data and workflow. Furthermore, to fully leverage the advantages of the cloud model, such as elasticity, often requires the user to fundamentally rethink the execution model. All of this makes it difficult to offer a cost-effective support model.

7.4 Summary

User support models for technical computing users remains one of the open issues for cloud systems. Technical computing users have grown accustomed to a high level of support from HPC center personnel. In this setting, purpose built systems are constructed for a specific use case of large scale parallel applications. As a result, many users have not developed the skills that are needed to effectively work in the more flexible environment that clouds provide. Several possible solutions are discussed below.

Users could be educated to become comfortable designing and deploying infrastructure. This education effort would be time-consuming and is unlikely to be sufficient, leaving some technical computing users unable to use large scale cloud computing resources. This approach would also require that scientists develop skills which are not directly related to their area of study, most likely reducing their scientific output.

Another potential solution is for an analogous HPC-tuned environment to be made available on a private or public cloud. This solution is the most similar to current production computing centers, and would be likely comparable in cost to current HPC center support activities. In fact, this approach was explored at NERSC using a virtual version of an existing cluster ("Little Magellan" described in Chapter 3) which was used by users to easily measure the overhead of running in a virtualized cloud environment. ALCF also worked with the Argonne Laboratory Computing Resource Center (LCRC), which provides the large laboratory mid-range computing cluster, to provide a virtual version of the existing Fusion cluster within the Magellan public OpenStack cloud. In addition, the ALCF/LCRC team was able to set up the Fusion resource manager so that jobs could expand into the Magellan public cloud in a secure fashion with a virtual Fusion environment, providing dynamic expansion into the cloud when the Fusion resources were fully used.

Another approach would involve building user communities around common domains and applications. In this model, sophisticated users would provide VM images which are optimized for a particular science domain and are used by a large number of users. Many of these communities would mirror virtual organizations that are already in place. This model would work well for large, established organizations, but could present challenges for smaller communities composed primarily of novice users.

Chapter 8

Security

Security personnel at both Argonne and NERSC worked alongside the systems engineers, support staff, users, and others during this project. The goals for the security work on Magellan included assessing the system and designing security controls to protect the cloud testbed. In addition, the security teams were expected to view the testbed through the lens of DOE security requirements and report on whether a cloud system can meet those requirements.

In this section, we discuss security work done on the Magellan system as well as assessments of how DOE standards apply to the cloud testbed. It is important to keep in mind that these discussions are based on the Magellan cloud testbed and they would not necessarily extend to public clouds systems such as Amazon EC2. Assessment of Amazon EC2 and other public cloud offerings is outside the scope of this project.

The two cloud testbeds had similar architectures but each, of course, had features unique to each site. Argonne utilized both Eucalyptus and OpenStack cloud stacks on their system during the project. NERSC used Eucalyptus for their testbed. The basic configuration of both Eucalyptus and OpenStack, included a server that provided user-accessible web services, a server that handled network connectivity with the virtualized systems, a server that handled storage, and the compute nodes that ran the virtualized system images. Each of these services can be installed on a single system or distributed. In the case of Eucalyptus, the three core services were installed on a single system. ALCF also experimented with distributing the services over a number of systems with OpenStack. Both systems provided a similar architecture and common aggregation points for user interaction and network activity generated by the virtualized systems.

8.1 Experiences on Deployed Security

There are limited service provider security tools available today. Thus, we faced a number of challenges in identifying how to perform the usual work of observation and analysis. As with any new resource, we compared what (if anything) differentiated cloud computing from any other large multiuser system. In particular, we evaluated how the cloud model is different from the shell access/job submission model that we currently provide across our production systems? From this, we were able to define problems we had to solve and identify how to reconfigure the basic system configuration or build tools for solving them. Most of this work is an early prototype, just the initial steps to a true production quality set of tools and procedures which would work in conjunction with the more traditional set of tools and procedures used in HPC. Our work could be thought of as a special case of the more general problem of HPC security—i.e., continuum of risk space from nominal user interaction to user shell, root shell, and finally whole system image.

The set of defenses we implemented can be roughly broken out into two groups—static and dynamic. Examples of static defenses are account management, port scanning and firewalls. Dynamic defenses interact with the cloud instance, interpreting actions and recording behaviors based on well defined local security policy.

Firewalls are a natural tool to protect sensitive infrastructure components from intrusion from unwanted address space. One of the first things done was to isolate the cloud, cluster, and node controller instances from the address space used by the virtual instances to protect infrastructure from everything except a small address space. It is also necessary to prevent spoofing of infrastructure address space from the set of user initiated images, since the VM instances can create arbitrarily addressed network traffic.

Another simple test is to scan public facing address space for unexpected ssh ports as well as exceptionally susceptible accounts (such as password-less root accounts). Mandating a full system scan before releasing a VM for public access was discussed and a method was proposed, but this method was not implemented on the Magellan deployment.

Taking a closer look at user activities—both in terms of site security policy as well as machine readable logging—was another step in making the cloud systems more equivalent to our current batch systems. Enforcing local computer security policy on dynamic user-run virtual machine systems (such as Eucalyptus or EC2) is not something that traditional intrusion detection systems excel at, since they tend to be designed with fairly rigid notions of the mappings between systems, users, and addresses. To resolve this problem, we used data from two locations. First we used the EC2 branch of the Python Boto package (via euca2ools) to query the Cloud Controller for information about registered instances, IP addresses, users, and instance groups firewall rules. This provided a clean interface and data source for actively querying the Cloud Controller. The second source takes advantage of the Eucalyptus network architecture since all ingress/egress as well as intra-instance network traffic passes through a choke point on the cloud controller. The data for both cases was then processed by an instance of the Bro intrusion detection system. As already suggested, this provides two principal benefits. The first is the ability to express local site security policy by configuring a set of scripts we created for this purpose. The data from Bro helps track successful and failed logins, VM instantiation, inter-cluster scan detection, and rules about system firewall configurations. Once the local policy is defined, activities which violate that policy can be quickly identified and acted on. In addition, all activity was logged in a well defined format.

8.2 Challenges Meeting Assessment and Authorization Standards

While the technical aspects of computer security tend to gather the most attention, significant work is required to ensure that any large scale technology can be placed in the context of the current language found in Security Controls and Security Assessment and Authorization (A&A). Since any detailed interpretation of these documents is far outside the scope of this project, we will only discuss a few key points. Current best documentation involving cloud resources and security guidance is the FedRAMP [27] document, which covers three main areas:

- *List of baseline security controls for low and moderate impact cloud systems.* NIST Special Publication 800-53R3 provides the foundation for the development of these security controls.

- *Processes in which authorized cloud computing systems will be monitored continuously.* This draft defines continuous monitoring deliverables, reporting frequency, and responsibility for cloud service provider compliance with the Federal Information Security Management Act.

- *Proposed operational approaches for assessment and authorization for cloud computing systems that reflect on all aspects of an authorization, including sponsorship, leveraging, maintenance, and continuous monitoring, a joint authorization process, and roles and responsibilities for federal agencies and cloud service providers.* This is detailed under the risk management framework in NIST Special Publication 800-37R1.

Since a traditional HPC site will have covered a reasonable number of these controls in their regular A&A work, we will focus on a small number of unique issues presented by providers of cloud computing infrastructure.

The set of controls set out in the FedRAMP document are designated for low and medium baselines. For the current control baseline, we have a low impact cloud system. Since the focus of the FedRAMP document was on consumers of cloud resources rather than producers, we needed to digest the contents of the document from an alternate perspective. As well, since institutional responses to individual control entities are driven by local requirements, needs, and philosophy, any sort of detailed response is well outside the scope of this document. However, at a minimum, the following steps should be taken.

1. A "user" should be a uniquely identifiable entity (such as a user account) where a complete Access Control, Audit and Accountability process can take place, and who then is granted the right to launch a VM within their own system context.
2. For pre-vetted instances (where the user is not granted root access), site security controls must be applied for sections Access Controls (AC), Audit and Accountability (AU), and Identification and Authentication (IA) from the NIST 800-53 publication.
3. For non-vetted images, or where the users will be granted root access, active scanning and analysis is required.

In the same way that the technical aspects of computer security are evolving for cloud computing providers, identifying how this new technology will fit into the current sets of security controls remains an area for exploration.

8.3 Recommended Further Work

There are a number of areas that we feel should be addressed to advance the state of security on a cloud-based system like Magellan. While many tools and techniques used with typical large multiuser systems can be applied to cloud-based systems, these tools and techniques would evolve more rapidly through tighter integration into the cloud-stack software.

Further Security Control Automation. Some work was done during the project to integrate the cloud software with automated security systems. Additional work is required in this area. With the dynamic nature of cloud systems, the security controls must evolve to be similarly dynamic and flexible. The integration of security policy tools directly into the cloud control software would be ideal, as it would allow enforcement of a site's security policies while the user worked instead of being applied after the fact by systems that monitor for changes. This would also enhance the responsiveness of the system. Security setting modifications could be requested by the user, checked against a site's policies, and implemented or denied, all in a single integrated operation.

Enhanced Forensic Capabilities. Due to the ephemeral nature of cloud resources, performing forensic analysis on these virtualized systems is fraught with many challenges, both technical and legal. When there is a need to analyze a physical system involved in a security incident or crime, the system is usually taken into evidence. Its storage devices are copied and verified, and those files are then analyzed. The practice of computer forensics is well established, and evidence produced through this analysis is accepted in courts worldwide. Virtualized systems provide a new twist for computer investigators. Should the need arise to collect evidence from a virtualized system, the methods and practices are not as clear-cut. Can the investigator capture an image of the virtual disk associated with a given resource, or must they analyze the physical hardware that underlies the infrastructure? Depending on the size of the cloud itself and the underlying software, the answer to this question could have widespread effects on the cloud system. If a portion of the physical system must be removed for processing, the resources of the cloud could be diminished significantly. On the other hand, if an investigator captures and analyzes only the virtual disk associated with a given virtual system, will that satisfy the court? Or will the opposing attorneys be able to raise enough doubt about the evidence in the judge or jury that it may be thrown out?

Aside from legal questions, a survey of existing forensic tools for virtual systems should be made, and any gaps identified and plans made to address them. There may be some benefit to building forensic capture and analysis capabilities directly into the cloud software. This could speed response time during an incident and also provide vendors in the computer forensic space a way to integrate their tools with the cloud software.

Improved Logging and Auditing Capabilities. A definite challenge the teams experienced while running the cloud testbeds was the lack of sufficiently detailed logs that could be used for auditing and reporting purposes. Calculating individual and project usage was difficult and required development of specialized tools to gather data from disparate sources and attempt to combine it to generate usage reports. This disjointed logging can also hamper security analysis and audits by needlessly complicating the required effort to review a system's usage. It is our recommendation that some effort be made to consolidate and focus the log records of cloud systems to ensure they provide the necessary detail needed for not only troubleshooting but also regular auditing activities.

Improved User Authentication and Role-Based Access Control. Depending on the software used to implement a cloud system, there are varying degrees of role-based access control available. Eucalyptus had very limited options for user roles and resource allocation. OpenStack options for role-based access controls were more advanced, and there was even basic resource management available to prevent a small number of users from utilizing too many resources. In order to meet DOE requirements, these features would need to be greatly expanded and enhanced beyond their current state. The ability to define roles and even delegate role assignment to project leads could allow more flexibility for the system users while making sure that security policies were being met. By formalizing roles and building them into the cloud system, auditing users and access would be greatly simplified. The addition of a "billing system" that tracked usage of resources among different users and projects accurately and automatically could allow for more advanced scheduling of resources beyond the current simplistic model, where resources are either available immediately or not at all.

User authentication also needs improvement. The current model of authenticating users via certificate is usable, but is inflexible, for instance, it does not provide a way to incorporate two-factor authentication. There are workarounds that can address this issue for systems that might require it today, but more integrated options should be explored in this space. Perhaps storing the cloud authentication certificates on smart cards could be a possible solution. The software would need to be modified to make use of alternative authentication methods, and testing would show which methods are most effective and flexible for end users.

Assessment of Public Cloud Infrastructures. As we have mentioned earlier, our work centered on evaluating operating a cloud computing infrastructure for scientific workloads. Our assumptions and conclusions should be considered in this context. While we did discuss public cloud infrastructures and how they compared to the Magellan cloud testbed, serious evaluation of those systems was not undertaken. Even though assessment of public cloud infrastructures was not in the scope of this project, we believe they should be evaluated and compared to privately run cloud systems to better understand the costs and challenges associated with those types of systems.

8.4 Summary

Securing a cloud system has many parallels to securing large multi-user clusters. Of course there are some unique challenges, but our experiences suggest that these could be met if significant effort were spent on the system design, enhancing the existing tools, and building some key capabilities that do not exist today. Our work indicates that the risks of allowing users full reign over their virtual machines is not as onerous a problem as it first appears. Much of the risk involves user error, which might be mitigated by introducing a high-quality user interface that simplifies management tasks. Overall, the cloud systems we worked with did not reveal any security hurdles that were considered intractable.

Chapter 9

Benchmarking and Workload Analysis

Scientific applications can be broadly classified as (a) *tightly coupled computations,* or (b) *asynchronous or loosely coupled computations.* Tightly coupled applications are typically MPI codes that range from mid-range to large-scale. Mid-range applications that need tens to hundreds of processors often run on local clusters, while the large-scale codes are run at supercomputing centers. There is an interest in understanding if these mid-range applications could run in cloud environments, eliminating the need for various users to manage their own clusters. Within the mid-range space, many of the scientific applications that run on the desktop or local clusters have asynchronous or loosely coupled computations. In recent years, the increased scale of digital data due to low-cost sensors and other technologies [58] has resulted in the need for these applications to scale to larger environments such as cloud environments. The requirements of these high-throughput applications are similar to those of the internet applications that currently dominate the cloud computing space, but with far greater data storage and throughput requirements. It is important to understand the behavior of each of these classes of applications in cloud environments.

We have performed a number of benchmarking experiments to understand the performance of applications in virtualized cloud environments as compared with traditional HPC systems. To understand which applications might work well in cloud environments, we conducted an exhaustive benchmarking study and workload analysis. Specifically,

- We evaluated the impact of virtualization and interconnect performance to understand the nature of clouds required for scientific applications. This three-phase benchmarking effort considered public clouds, private clouds, and a variety of different interconnects including InfiniBand, 10 GigE, and 1 GigE (Section 9.1).

- We conducted benchmarking tests to understand the virtualization overheads in the OpenStack virtual environments (Section 9.2).

- We conducted an I/O benchmarking study to understand the range of I/O performance that applications can anticipate in virtualized cloud environments (Section 9.3).

- Solid-state storage (SSS) is poised as a disruptive technology alongside clouds to handle large data volumes. We did a preliminary flash storage evaluation to understand the performance of different products available at the time of evaluation (Section 9.4)

- We worked closely with various Magellan user groups to understand the performance impact on their applications of virtualized cloud environments (Section 9.5).

- Finally, we performed some workload analysis to understand the characteristics of scientific applications and correlate it with our benchmarking data (Section 9.6).

9.1 Understanding the Impact of Virtualization and Interconnect Performance

The benchmarking was conducted in three phases. In Phase 1, we compared the performance of a commercial cloud platform (Amazon EC2) with NERSC systems (Carver and Franklin) and a mid-range computing system (Lawrencium) at LBNL. The Carver and Magellan systems are identical in hardware configuration, and the traditional cluster configuration for Magellan uses the same software configuration as Carver. The typical problem configurations for the NERSC-6 benchmarks (described below) are defined for much larger systems. We constructed reduced problem size configurations to target the requirements of mid-range workloads. In Phase 2, we repeated the experiments of the reduced problem size configurations on Magellan hardware to further understand and characterize the virtualization impact. Finally, in Phase 3, we studied the effect of scaling and interconnects on the Magellan testbed and Amazon.

9.1.1 Applications Used in Study

DOE supercomputer centers typically serve a diverse user community. In NERSC's case, the community contains over 4,000 users working on more than 400 distinct projects and using some 600 codes that serve the diverse science needs of the DOE Office of Science research community. Abstracting the salient performance-critical features of such a workload is a challenging task. However, significant workload characterization efforts have resulted in a set of full application benchmarks that span a range of science domains, algorithmic methods, and concurrencies, as well as machine-based characteristics that influence performance such as message size, memory access pattern, and working set sizes. These applications form the basis for the Sustained System Performance (SSP) metric, which represents the effectiveness of a system for delivered performance on applications better than peak FLOP rates [55].

As well as being representative of the DOE Office of Science workload, some of these applications have also been used by other federal agencies, as they represent significant parts of their workload. MILC and PARATEC were used by the NSF, CAM by NCAR, and GAMESS by the NSF and DoD HPCMO. The representation of the methods embodied in our benchmark suite goes well beyond the particular codes employed. For example, PARATEC is representative of methods that constitute one of the largest consumers of supercomputing cycles in computer centers around the world [64]. Therefore, although our benchmark suite was developed with the NERSC workload in mind, we are confident that it is broadly representative of the workloads of many supercomputing centers today. More details about these applications and their computational characteristics in the NERSC-6 benchmark suite can be found in Ref. [3].

The typical problem configurations for these benchmarks are defined for much larger "capability" systems, so we had to construct reduced size problem configurations to target the requirements of mid-range workloads that are the primary subject of this study. For example, many of the input configurations were constructed for a system acquisition (begun during 2008) that resulted in a 1 petaflop peak resource that contains over 150,000 cores—considerably larger than is easily feasible to run in today's commercial cloud infrastructures. Thus, problem sets were modified to use smaller grids and concomitant concurrencies along with shorter iteration spaces and/or shorter simulation durations. We also modified the problem configurations to eliminate any significant I/O in order to focus on the computational and network characteristics. For the scaling studies described in Section 9.1.5, we used the larger problem sets from MILC and PARATEC as described below in the application description.

Next we describe each of the applications that make up our benchmarking suite, describe the parameters they were run with, and comment upon their computation and communication characteristics.

CAM. The Community Atmosphere Model (CAM) is the atmospheric component of the Community Climate System Model (CCSM) developed at NCAR and elsewhere for the weather and climate research communities [7, 6]. In this work we use CAM v3.1 with a finite volume (FV) dynamical core and a "D" grid (about 0.5 degree resolution). In this case we used 120 MPI tasks and ran for three days simulated time (144 timesteps).

CAM uses a formalism effectively containing two different, two-dimensional domain decompositions, one for the dynamics that is decomposed over latitude and vertical level, and the other for remapping that is decomposed over longitude and latitude. Optimized transposes move data from the program structures between these decompositions. CAM is characterized by relatively low computational intensity that stresses on-node/processor data movement, and relatively long MPI messages that stress interconnect point-to-point bandwidth.

GAMESS. The GAMESS (General Atomic and Molecular Electronic Structure System) code from the Gordon research group at the Department of Energy's Ames Lab at Iowa State University contains various important tools for *ab initio* quantum chemistry calculations. The benchmark used here calculates the B3LYP DFT energy and gradient for a 43-atom molecule and runs on 64 cores. GAMESS is the only benchmark for which no problem size scaling was performed.

GAMESS uses an SPMD approach but includes its own underlying communication library, called the Distributed Data Interface (DDI), to present the abstraction of a global shared memory with one-side data transfers even on systems with physically distributed memory. On the cluster systems included here, GAMESS was run using socket communication. On the Cray XT4, an MPI implementation of DDI is used in which only one-half of the processors allocated compute while the other half are essentially data movers. GAMESS is characterized by considerable stride-1 memory access—which stresses memory bandwidth—and interconnect collective performance.

GTC. GTC is a fully self-consistent, gyrokinetic 3-D particle-in-cell (PIC) code with a non-spectral Poisson solver [57]. It uses a grid that follows the field lines as they twist around a toroidal geometry representing a magnetically confined toroidal fusion plasma. The version of GTC used here uses a fixed, 1-D domain decomposition with 64 domains and 64 MPI tasks. The benchmark runs are for 250 timesteps using 10 particles per grid cell (2 million grid points, 20 million particles). Communications at this concurrency are dominated by nearest neighbor exchanges that are bandwidth-bound. The most computationally intensive parts of GTC involve gather/deposition of charge on the grid and particle "push" steps. The charge deposition utilizes indirect addressing and therefore stresses random access to memory.

IMPACT-T. IMPACT-T (Integrated Map and Particle Accelerator Tracking–Time) is an object-oriented Fortran90 code from a suite of computational tools for the prediction and performance enhancement of accelerators. It includes the arbitrary overlap of fields from a comprehensive set of beamline elements, and uses a parallel, relativistic PIC method with a spectral integrated Green function solver. A two-dimensional domain decomposition in the y-z directions is used along with a dynamic load balancing scheme based on domain. Hockneys FFT algorithm is used to solve Poissons equation with open boundary conditions. The problems chosen here are scaled down quite considerably from the official NERSC benchmarks, in terms of number of particles and grid size (4X), 2-D processor configuration (64 cores instead of 256 and 1,024), and number of time steps run (100 instead of 2,000). IMPACT-T performance is typically sensitive to memory bandwidth and MPI collective performance. (Note that although both GTC and IMPACT-T are PIC codes, their performance characteristics are quite different.)

MAESTRO. MAESTRO is used for simulating astrophysical flows such as those leading up to ignition in Type Ia supernovae. Its integration scheme is embedded in an adaptive mesh refinement algorithm based on a hierarchical system of rectangular non-overlapping grid patches at multiple levels with different resolutions. However, in the benchmarking version, the grid does not adapt and, instead, a multigrid solver is used. Parallelization is via a 3-D domain decomposition in which data and work are apportioned using a coarse-grained distribution strategy to balance the load and minimize communication costs. The MAESTRO communication topology pattern is quite unusual and tends to stress simple topology interconnects. With a very low computational intensity, the code stresses memory performance, especially latency; its implicit solver technology stresses global communications; and its message passing utilizes a wide range of message sizes from short to relatively moderate. The problem used was the NERSC-6 medium case ($512\times512\times1024$

grid) on 256 cores, but for only 3 timesteps. This problem is more typically benchmarked on 512 cores for 10 timesteps.

MILC. This code represents lattice gauge computation that is used to study quantum chromodynamics (QCD), the theory of the subatomic strong interactions responsible for binding quarks into protons and neutrons and holding them together in the nucleus. QCD discretizes space and evaluates field variables on sites and links of a regular hypercube lattice in four-dimensional space time. It involves integrating an equation of motion for hundreds or thousands of time steps that requires inverting a large, sparse matrix at each integration step. The sparse, nearly singular matrix problem is solved using a conjugate gradient (CG) method, and many CG iterations are required for convergence. Within a processor, the four-dimensional nature of the problem requires gathers from widely separated locations in memory. The inversion by CG requires repeated three-dimensional complex matrix-vector multiplications, which reduces to a dot product of three pairs of three-dimensional complex vectors. Each dot product consists of five multiply-add operations and one multiply operation. The parallel programming model for MILC is a 4-D domain decomposition in which each task exchanges data with its eight nearest neighbors as well as participating in the all-reduce calls with very small payload as part of the CG algorithm. MILC is extremely dependent on memory bandwidth and prefetching and exhibits a high computational intensity.

Two different MILC configurations were used in the benchmarking efforts. In one case (Section 9.1.3) we used a $32\times32\times16\times18$ global lattice on 64 cores with two quark flavors, four trajectories, and eight steps per trajectory. In contrast, MILC benchmarking for NERSC procurements have used up to 8,192 cores on a $64^3\times144$ grid with 15 steps per trajectory. For the scaling study (Section 9.1.5) we used a $64\times32\times32\times72$ global lattice with two quark flavors, four trajectories, and 15 steps per trajectory.

PARATEC. PARATEC (PARAllel Total Energy Code) performs *ab initio* density functional theory quantum-mechanical total energy calculations using pseudopotentials, a plane wave basis set, and an all-band (unconstrained) conjugate gradient (CG) approach. Part of the calculation is carried out in Fourier space; custom parallel three-dimensional FFTs are used to transform the wave functions between real and Fourier space.

PARATEC uses MPI and parallelizes over grid points, thereby achieving a fine-grain level of parallelism. The real-space data layout of wave functions is on a standard Cartesian grid. In general, the speed of the FFT dominates the runtime, since it stresses global communications bandwidth, though mostly point-to-point, using relatively short messages. Optimized system libraries (such Intel MKL or AMD ACML) are used for both BLAS3 and 1-D FFT; this results in high cache reuse and a high percentage of per-processor peak performance.

The benchmark used here is based on the NERSC-5 input that does not allow any aggregation of the transpose data. The input contains 250 silicon atoms in a diamond lattice configuration and runs for six conjugate gradient iterations. For the scaling study (Section 9.1.5), the input is based on the NERSC-6 input. The input contains 686 silicon atoms in a diamond lattice configuration and runs for 20 conjugate gradient iterations. In contrast, a real science run might use 60 or more iterations.

HPCC. In addition to the application benchmarks discussed above, we also ran the High Performance Computing Challenge (HPCC) benchmark suite [41]. HPCC consists of seven synthetic benchmarks: three targeted and four complex. The targeted synthetics are DGEMM, STREAM, and measures of network latency and bandwidth. These are microkernels which quantify basic system parameters that separately characterize computation and communication performance. The complex synthetics are HPL, FFTE, PTRANS, and RandomAccess. These combine computation and communication and can be thought of as very simple proxy applications. Taken together, these benchmarks allow for the measurement of a variety of lower-level factors that are important for performance, which is why we chose to use them for this work.

9.1.2 Machines

All results were obtained during normal, multi-user, production periods on all machines unless specified otherwise.

Magellan. All the experiments described in this section were performed on the Magellan system at NERSC. Chapter 5 described the testbed in detail; here we summarize the key characteristics relevant to understanding the benchmarking results. Magellan is a 720-node IBM iDataPlex cluster. Each node has two quad-core Intel Nehalem processors running at 2.67 Ghz, 24 GB of RAM, and two network connections: a single Quad Data Rate (QDR) InfiniBand network connection and a GiB Ethernet connector. The InfiniBand network is locally a fat tree with a global 2-D mesh. Our 10G network is based on the Juniper Qfabric Switching System with four Q/F nodes, two Q/F Interconnects, and Junos 11.3.

Some early experiments were performed on Carver, a machine identical in hardware to the Magellan cloud testbed at NERSC. Carver is a 400-node IBM iDataPlex cluster also located at NERSC, and the nodes have an identical configuration to Magellan. All of the codes compiled on Carver used version 10.0 of the Portland Group suite and version 1.4.1 of OpenMPI.

The virtual machine environment on Magellan at NERSC is based on Eucalyptus 1.6 (studies in Section 9.1.4) and 2.0 (studies in Section 9.1.5), an open source software platform that allows organizations to leverage existing Linux-based infrastructure to create private clouds. Our Eucalyptus installation uses Kernel-based Virtual Machines(KVM) as a hypervisor. We modified Eucalyptus to use Virtio for disk access. We use the KVM option for the emulated e1000 nic for the network. All codes were compiled with the Intel compiler version 11.1 and used version 1.4.2 of OpenMPI.

All tests were performed on an instance type configured with 8 CPUs/20 G memory/20 G disk. The guest OS is CentOS release 5.5 (Linux kernel 2.6.28-11). We set up a virtual cluster where the head node mounts a block store volume that has all the application binaries and data. The block store volume is mounted on the other virtual machines via NFS. All the virtual machine communication traffic goes over the Ethernet network.

Amazon EC2. Amazon EC2 is a virtual computing environment that provides a web services API for launching and managing virtual machine instances. Amazon provides a number of different instance types that have varying performance characteristics. CPU capacity is defined in terms of an abstract Amazon EC2 Compute Unit. One EC2 Compute Unit is approximately equivalent to a 1.0–1.2 GHz 2007 Opteron or 2007 Xeon processor. For our tests we used the m1.large instance type, which has four EC2 Compute Units, two virtual cores with two EC2 Compute Units each, and 7.5 GB of memory. The nodes are connected with Gigabit Ethernet. To ensure consistency, the binaries compiled on Lawrencium were used on EC2. All of the nodes are located in the US East region in the same availability zone. Amazon offers no guarantees on proximity of nodes allocated together, and there is significant variability in latency between nodes.

In addition to the variability in network latency, we also observed variability in the underlying hardware on which the virtual machines are running. By examining /proc/cpuinfo, we are able to identify the actual CPU type of the non-virtualized hardware. In our test runs, we identified three different CPUs: the Intel Xeon E5430 2.66 GHz quad-core processor, the AMD Opteron 270 2.0 GHz dual-core processor, and the AMD Opteron 2218 HE 2.6 GHz dual-core processor. Amazon does not provide a mechanism to control which underlying hardware virtual machines are instantiated on. Thus, we almost always end up with a virtual cluster running on a heterogeneous set of processors. This heterogeneity also meant we were unable to use any of the processor-specific compiler options.

During the course of the project Amazon started providing access to Cluster Compute (CC) instances in addition to the other types of instances previously available. These new CC instances are significantly different from the other types in terms of performance characteristics. This is the first instance type which guarantees the hardware architecture of the nodes and reduces the chance for any performance variation due to varying hardware types. Amazon has defined the specification on these CC instances to have dual socket Intel Nehelam CPUs at 2.97 GHz and 23 GB of memory per node. The nodes are connected by

Table 9.1: HPCC Performance

Machine	DGEMM	STREAM	Latency	Bandwidth	RandRing Lat.	RandRing BW	HPL	FFTE	PTRANS	RandAccess
	Gflops	GB/s	μs	GB/s	μs	GB/s	Tflops	Gflops	GB/s	GUP/s
Carver	10.2	4.4	2.1	3.4	4.7	0.30	0.56	21.99	9.35	0.044
Franklin	8.4	2.30	7.8	1.6	19.6	0.19	0.47	14.24	2.63	0.061
Lawrencium	9.6	0.70	4.1	1.2	153.3	0.12	0.46	9.12	1.34	0.013
EC2	4.6	1.7	145	0.06	2065.2	0.01	0.07	1.09	0.29	0.004

a 10G Ethernet network, allowing applications on the nodes to communicate at high speed. Amazon also guarantees that the hardware under CC instances is not shared with any other Amazon EC2 instances, and at any given time each node will be running only one virtual instance (single occupancy). Another new feature with CC instances is a logical entity called Placement Group which helps in launching multiple CC instances into a cluster of instances with low communication latency by trying to place them as close as possible within the network. Cluster Compute instances are currently available only in US East region; all instances are 64 bit and use a Linux operating system.

The Amazon environment provides a set of virtual machines with no shared resources between them. Almost all HPC applications assume the presence of a shared parallel file system between compute nodes, and a head node that can submit MPI jobs to all of the worker nodes. Thus we replicated a typical HPC cluster environment in the cloud by creating virtual clusters [29, 52]. We used a series of Python scripts to configure a file server, a head node, and a series of worker nodes. The head node could submit MPI jobs to all of the worker nodes, and the file server provided a shared file system between the nodes.

To implement the shared file system, we attached an Amazon Elastic Block Store (ENS) [19] device to the file-server virtual machine. EBS provides a block level storage volume to EC2 instances that persists independently from the instance lifetimes. On top of the EBS volume, we built a standard Linux ext3 file system, which was then exported via NFS to all of the virtual cluster nodes.

Franklin is a 9660-node Cray XT4 supercomputer located at NERSC. Each XT4 compute node contains a single quad-core 2.3 GHz AMD Opteron “Budapest” processor, which is tightly integrated to the XT4 interconnect via a Cray SeaStar-2 ASIC through a 6.4 GB/s bidirectional HyperTransport interface. All the SeaStar routing chips are interconnected in a 3D torus topology, where each node has a direct link to its six nearest neighbors. Each node has 8 GB of RAM (2 GB per core). Codes were compiled with the Pathscale (MAESTRO) and Portland Group version 9.0.4 (all others) compilers.

Lawrencium is a 198-node (1584-core) Linux cluster operated by the Information Technology Division at LBNL. Each compute node is a Dell Poweredge 1950 server equipped with two Intel Xeon quad-core 64 bit, 2.66 GHz “Harpertown” processors, connected to a Dual Data Rate (DDR) InfiniBand network configured as a fat tree with a 3:1 blocking factor. Each node contains 16 GB of RAM (2 GB per core). Codes were compiled using Intel 10.0.018 and Open MPI 1.3.3.

9.1.3 Evaluation of Performance of Commercial Cloud Platform

The results from this study were presented at CloudCom 2010 [46]. The paper was awarded Best Paper at the conference. We summarize the key results from the paper here. For more details please refer to the paper.

The goal of this phase of benchmarking was to study the performance of real scientific workloads on EC2. We use real scientific applications, as described above, that are representative of the whole NERSC workload. We also instrument the runs using the Integrated Performance Monitoring (IPM) [77] framework. This allows us to determine, in a non-perturbative manner, the amount of time an application spends computing and communicating using MPI. This information provides insight into which aspects of the underlying architecture are affecting performance the most.

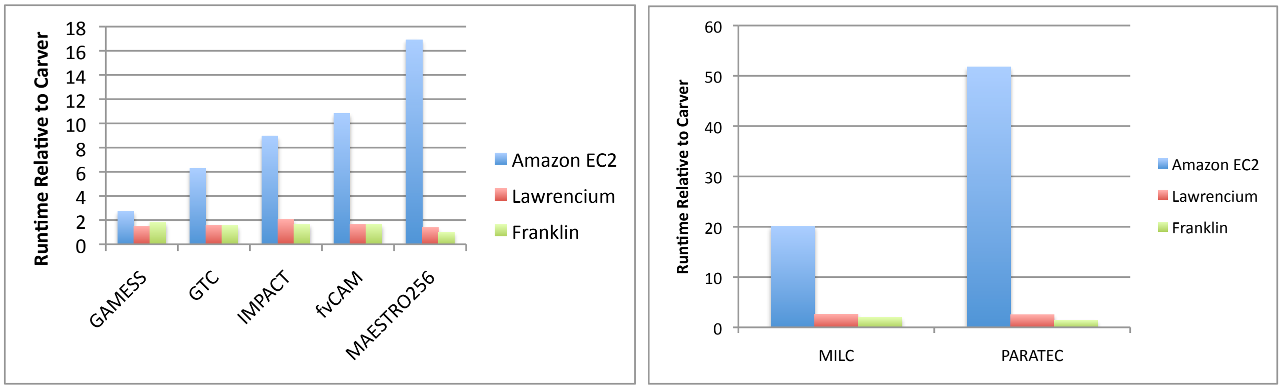


Figure 9.1: Runtime of each application on EC2, Lawrencium and Franklin relative to Carver.

HPCC Results

The results of running HPCC v.1.4.0 on 64 cores of the four machines in our study are shown in Table 9.1. The DGEMM results are, as one would expect, based on the properties of the CPUs. The STREAM results show that EC2 is significantly faster for this benchmark than Lawrencium. We believe this is because of the particular processor distribution we received for our EC2 nodes for this test. We had 26 AMD Opteron 270s, 16 AMD Opteron 2218 HEs, and 14 Intel Xeon E5430s, of which this measurement represents an average. The AMD Opteron-based systems are known to have better memory performance than the Intel Harpertown-based systems used in Lawrencium. Both EC2 and Lawrencium are significantly slower than the Nehalem-based Carver system, however.

The network latency and bandwidth results clearly show the difference between the interconnects on the tested systems. For display we have chosen both the average ping-pong latency and bandwidth, and the randomly-ordered ring latency and bandwidth. The ping-pong results show the latency and the bandwidth with no self-induced contention, while the randomly ordered ring tests show the performance degradation with self-contention. The un-contended latency and bandwidth measurements of the EC2 Gigabit Ethernet interconnect are more than 20 times worse than the slowest other machine. Both EC2 and Lawrencium suffer a significant performance degradation when self-contention is introduced. The EC2 latency is 13× worse than Lawrencium, and more than 400× slower than a modern system like Carver. The bandwidth numbers show similar trends: EC2 is 12× slower than Lawrencium, and 30× slower than Carver.

We now turn our attention to the complex synthetics. The performance of these is sensitive to characteristics of both the processor and the network, and their performance gives us some insight into how real applications may perform on EC2.

HPL is the high-performance version of the widely reported Linpack benchmark, which is used to determine the TOP500 list. It solves a dense linear system of equations, and its performance depends upon DGEMM and the network bandwidth and latency. On a typical high performance computing system today, roughly 90% of the time is spent in DGEMM, and the results for the three HPC systems illustrate this clearly. However, for EC2, the less capable network clearly inhibits overall HPL performance, by a factor of six or more. The FFTE benchmark measures the floating point rate of execution of a double precision complex one-dimensional discrete Fourier transform, and the PTRANS benchmark measures the time to transpose a large matrix. Both of these benchmarks' performance depends upon the memory and network bandwidth and therefore show similar trends. EC2 is approximately 20 times slower than Carver and four times slower than Lawrencium in both cases. The RandomAccess benchmark measures the rate of random updates of memory, and its performance depends on memory and network latency. In this case, EC2 is approximately 10× slower than Carver and 3× slower than Lawrencium.

Overall, the results of the HPCC runs indicate that the lower performing network interconnect in EC2

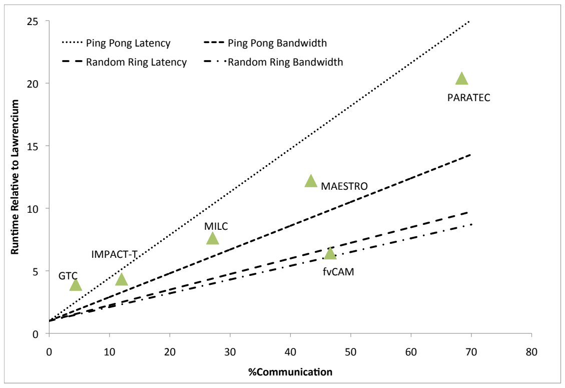


Figure 9.2: Correlation between the runtime of each application on EC2 and the amount of time an application spends communicating.

has a significant impact upon the performance of even very simple application proxies. This is illustrated clearly by the HPL results, which are significantly worse than would be expected from simply looking at the DGEMM performance.

Application Performance

Figure 9.1 shows the relative runtime of each of our test applications relative to Carver, which is the newest and therefore fastest machine in this study. For these applications, at these concurrencies, Franklin and Lawrencium are between 1.4× and 2.6× slower than Carver. For EC2, the range of performance observed is significantly greater. In the best case, GAMESS, EC2 is only 2.7× slower than Carver. For the worst case, PARATEC, EC2 is more than 50× slower than Carver. This large spread of performance simply reflects the different demands each application places upon the network, as in the case of the compact applications that were described in the previous section. Qualitatively we can understand the differences in terms of the performance characteristics of each of the applications described earlier. PARATEC shows the worst performance on EC2, 52× slower than Carver. It performs 3-D FFTs, and the global (i.e., all-to-all) data transposes within these FFT operations can incur a large communications overhead. MILC (20×) and MAESTRO (17×) also stress global communication, but to a lesser extent than PARATEC. CAM (11×), IMPACT (9×) and GTC (6×) are all characterized by large point-to-point communications, which do not induce quite as much contention as global communication, hence their performance is not impacted quite as much. GAMESS (2.7×), for this benchmark problem, places relatively little demand upon the network, and therefore is hardly slowed down at all on EC2.

Qualitatively, it seems that those applications that perform the most collective communication with the most messages are those that perform the worst on EC2. To gain a more quantitative understanding, we performed a more detailed analysis, which is described in the next section.

Performance Analysis Using IPM

To understand more deeply the reasons for the poor performance of EC2 in comparison to the other platforms, we performed additional experiments using the Integrated Performance Monitoring (IPM) framework [77, 84]. IPM is a profiling tool that uses the MPI profiling interface to measure the time taken by an application in MPI on a task-by-task basis. This allows us to examine the relative amounts of time taken by an application

Table 9.2: HPCC Performance

Machine	DGEMM Gflops	STREAM GB/s	Latency μs	Bandwidth GB/s	RandRing Lat. μs	RandRing BW GB/s	HPL Tflops	FFTE Gflops	PTRANS GB/s	RandAccess GUP/s
Native	10.20	3.78	1.93	3.45	5.26	0.324	0.57	28.57	9.25	0.0420
TCP o IB	10.21	3.75	23.69	1.41	50.66	0.236	0.55	26.97	7.81	0.0139
TCP o Eth	10.21	3.75	21.59	0.61	53.72	0.009	0.37	2.41	0.59	0.0139
VM	6.96	2.64	80.04	0.38	217.638	0.011	0.24	1.60	0.63	0.0095

for computing and communicating, as well as the types of MPI calls made. These measurements will enable us to determine which particular aspects of the EC2 hardware configuration are most inhibiting performance on an application-by-application basis in a quantitative manner. Previous measurements with IPM have shown that it has extremely low overhead [84], less than 2%, giving us confidence that by instrumenting the applications with IPM we are not altering their runtime characteristics.

One of the simplest metrics available from IPM is the percentage of the runtime that the application spends communicating (time in the MPI library, to be precise). Figure 9.2 shows the relative runtime on EC2 compared to Lawrencium plotted against the percentage communication for each application as measured on Lawrencium. The overall trend is clear: the greater the fraction of its runtime an application spends communicating, the worse the performance is on EC2.

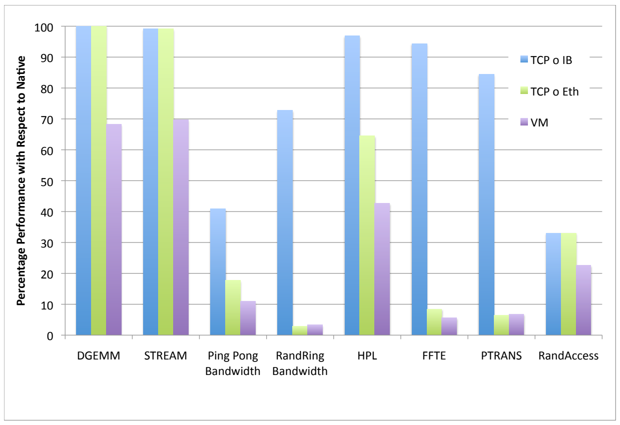
The principal exception to this trend is fvCAM, where the performance on EC2 is much faster than would be expected from the simple considerations described above. To understand why this is, we analyzed the communication characteristics of each of the applications to determine if fvCAM was an anomalous case. To determine these characteristics we classified the MPI calls of the applications into four categories: small and large messages (latency vs bandwidth limited) and point-to-point vs collective. (Note that for the purposes of this work we classified all messages < 4 KB to be latency bound. The overall conclusions shown here contain no significant dependence on this choice.) From this analysis it is clear why fvCAM behaves anomalously: it is the only one of the applications that performs most of its communication via large messages, both point-to-point and collectives. To understand why this causes the performance on EC2 to be faster, consider the HPCC challenge results shown in table 9.1. Compared to Lawrencium, the EC2 ping-pong latency is 35$\times$ worse, whereas the ping-pong bandwidth is 20$\times$ worse and the random ring latency and bandwidth are 13 and 12$\times$ worse. (Note that to a reasonable approximation, the performance of point-to-point messages will follow the trends of the ping-pong measurements, and the performance of collectives will follow those of the random-ring measurements.) Therefore, any application that spends a significant amount of its communication performing large messages via point-to-point messages in the latency limit, which in fact is all of the applications here except fvCAM, will be slower, relatively speaking, than one which operates in the bandwidth limit or primarily performs collectives. The slowdown expected for an application that only performs *one* of these four potential modes of communication is also shown in Fig. 9.2. As is clear from the figure, the faster performance for fvCAM is quite reasonable given its communication pattern.

Thus, using quantitative analysis based upon instrumenting our applications with IPM, we are able to explain the performance impacts of different application when running on EC2.

9.1.4 Understanding Virtualization Impact on Magellan Testbed

The results reported in the previous section highlight the overhead from virtualization on Amazon EC2. In this next study we attempted to understand the impact of virtualization on performance using the Magellan testbed. In addition, we collected data that enables us to understand the impact of the layers in the hierarchy of protocols. Although performance is highly dependent on the type of workload, we see that the performance decrease from virtualization is largely due to overheads in the communication, i.e., the use of the TCP over Ethernet as the communication mechanism between virtual machines. In this study we attempted to study that impact for a range of applications from the NERSC-6 benchmark suite with reduced problem sizes. In the next section, we study the effect of the communication protocol, especially in correlation to scaling effect.

Table 9.2 shows the values for the HPCC benchmarks on our four configurations. Figure 9.3 shows the



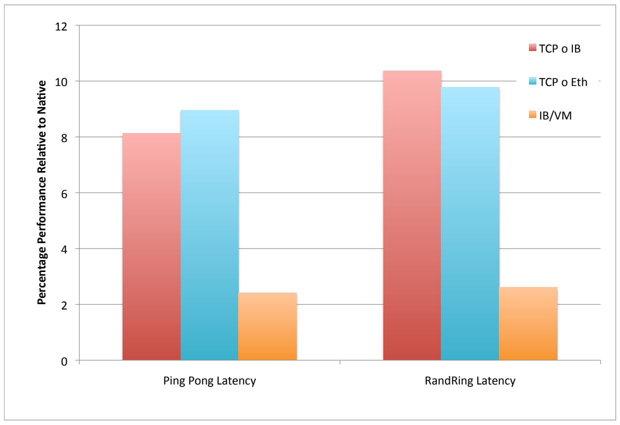


Figure 9.3: Performance of the HPCC challenge benchmarks relative to the native hardware configuration for each of the cases examined. (a) Shows the performance for all of the components of the benchmark except the latency measurements which are shown in (b) for clarity. Note the very different vertical scale on (b).

performance of each of the benchmarks relative to that obtained on the native host using the InfiniBand interconnect. First we consider Figure 9.3a. The first two measurements, DGEMM and STREAM, are measures of the floating point performance and memory bandwidth, respectively. As they do not depend on the network, they are in this case a probe to allow us to determine the impact of virtualization on compute performance. In both cases, the performance inside the virtual machine is about 30% slower than running on the native hardware, which is in line with previous measurements. [46] There seems to be two reasons: the overhead in translating the memory address instructions, and the KVM virtual environment not supporting all the instructions on the CPU such as SSE, etc. Subsequent to this study, a later version of KVM was installed on the Magellan testbed that supports the advanced CPU instructions in the virtual machine. The CPU overhead was observed to be negligible after this upgrade.

Next we consider the network bandwidth measurements, of which we made two kinds, ping-pong and random ring. Ping-pong is a measure of point-to-point bandwidth, whereas random ring consists of each task simultaneously sending to a randomly selected partner, with the intent to induce network contention. The ping-pong results show a decrease in performance relative to the native measurement for each of the cases examined. The biggest drop relative to the native performance comes from using TCP over IB, with additional decreases in performance observed for TCP over Ethernet and virtualization. The random ring bandwidth clearly shows the lack of capability of the Ethernet connection to cope with significant amounts of network contention. In this case, in contrast to the ping-pong case, the virtualization does not significantly further decrease the performance. The network latency measurements, both ping-pong and random ring, (Figure 9.3b) show qualitatively different trends to the bandwidths, as well as a significantly larger overall performance decrease, even at the TCP over InfiniBand level. The principal increase in latency occurs when switching to TCP over IB, then a further increase occurs from within the virtual machine. The TCP over IB and TCP over Ethernet latencies are approximately the same, which indicates the latency is primarily a function of the transport mechanism, as one would expect. The performance of the HPL benchmark is primarily a function of the DGEMM performance and the ping-pong bandwidth. The exact ratio depends upon the balance point of the machine, thus as the network capabilities become relatively worse, the HPL performance decreases. Both FFT and PTRANS are sensitive to random ring bandwidth and mirror the trends observed there. Random access is sensitive to network latency primarily and in a similar manner mimics the trends observed previously in the network latency measurements.

The performance of our application benchmarks relative to the native using the various machine config-

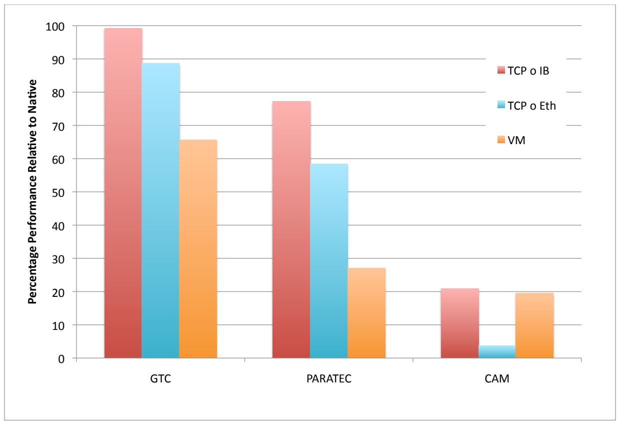


Figure 9.4: Relative performance of each of the cases examined for the three applications: GTC, PARATEC and CAM.

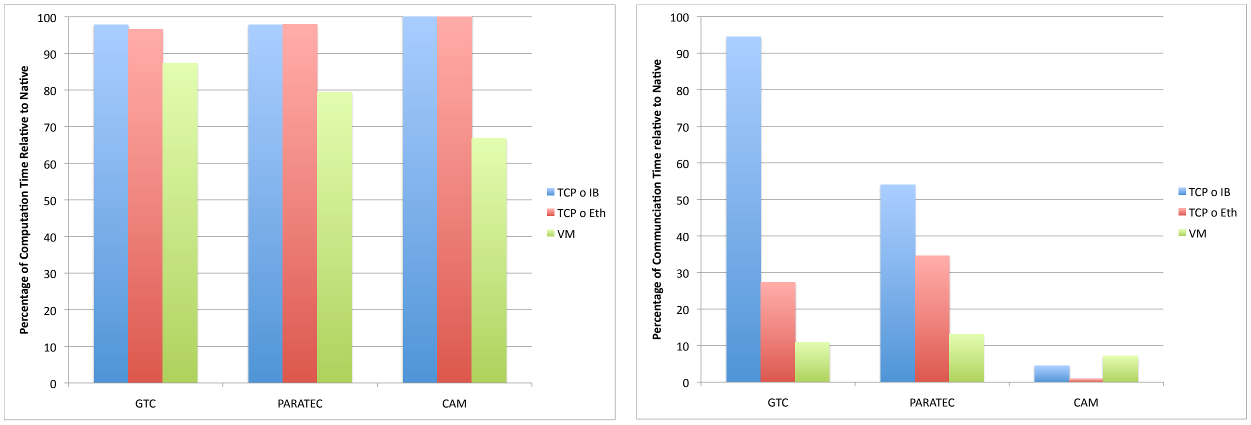


Figure 9.5: Performance ratio for the compute (a) and communication (b) parts of the runtime for each of the three applications GTC, PARATEC and CAM for each of the cases under consideration.

urations are shown in Figure 9.4. For all the codes, except CAM using TCP over Ethernet, the performance shows the expected trend, with each successive configuration showing decreased performance. To understand the reasons for the differing performance impact on each of the three codes, we analyzed them using the Integrated Performance Monitoring framework [77, 84].

The principal performance measurement made by IPM is the percentage of the runtime spent communicating (the time in the MPI library), which also allows us to infer the time spent computing. The results of this analysis are shown in Figure 9.5 which shows the compute time ratios for the various configurations. As expected from the HPCC results, only the VM configuration shows significant performance degradation for the compute portion of the runtime. Why the different applications show different slowdowns is unclear at this point, but we suspect it is related to the mix of instructions that they perform and the different slowdowns each experiences in the VM. The communication ratios are shown in Figure 9.5b. In this case the differences between the different applications and between the various configurations are much more pronounced. As well as providing overall information about the amount of time spent in the MPI library, IPM also reports the type of MPI routine called, the size of message sent, and the destination of the message. Using this information, we can understand why the different applications show different slowdowns. In the case of GTC, most of the communication is bandwidth limited, and to nearest neighbors, hence the slowdown shows a similar pattern to that observed for the ping-pong bandwidth measurements. For PARATEC, most of communication is for messages that are in the latency limit, although not all, so simple mapping onto the HPCC communication patterns is not possible. For CAM, the communication pattern is closest to the random ring bandwidth measurement. Also for CAM, the master processor reads the input file and broadcasts its content using MPI, then writes the intermediate output files; the time in I/O shows up as MPI time. In section 9.3, we present some detailed I/O benchmarking that quantifies the overhead in I/O in virtual environments.

9.1.5 Scaling Study and Interconnect Performance

This work was accepted for publication in the Performance Modeling, Benchmarking and Simulation of High Performance Computing Systems Workshop (PMBS11) held in conjunction with SC11, Seattle, November 2011 [70]

Our benchmarks and other studies [62, 23, 66, 74, 46] highlight the performance impact in virtual environments. These studies show that tightly coupled applications perform poorly in virtualized environments such as Amazon EC2. However, there is limited understanding about the type of cloud environments (e.g., interconnects, machine configuration, etc.) that might benefit science applications. HPC resource providers are interested in understanding the performance trade-offs of using cloud platform solutions such as virtualization in supercomputing centers.

Today's cloud environments are typically based on commodity hardware and use TCP over Ethernet to connect the nodes. However, compute resources found at HPC centers typically have much higher performing interconnects, such as InfiniBand, that provide higher bandwidth and lower latency. Thus the network performance of current cloud environments is quite different from that at HPC centers. In this study we use the Magellan cloud testbed to compare and understand the performance impact of the fabrics (InfiniBand, 10G) and protocols (InfiniBand, TCP, virtualization) between an HPC resource and a typical cloud resource. Specifically, we consider the following: native performance using InfiniBand, TCP over InfiniBand, TCP over 10G Ethernet, TCP over 10G Ethernet with virtualization, and TCP over 1G Ethernet. Additionally, we compare the performance of Amazon Cluster Compute instance types, the specialized HPC offering over 10G Ethernet.

In this study, we address the question of what virtualized cloud environments must provide in order to be beneficial for HPC applications. We use standard benchmarks such as the HPC Challenge [41] to understand the overheads. We also select a subset of benchmarks from the NERSC-6 benchmark suite based on our earlier work to capture the application behavior in virtualized environments.

Method

Users typically use high performance supercomputing centers for their medium to large-sized runs. High performance supercomputing centers provide high performance networking and storage infrastructure, system administration, and user support, in addition to the compute servers, which are beneficial to the users. However, in these environments users have less flexibility and no explicit control of the software stack.

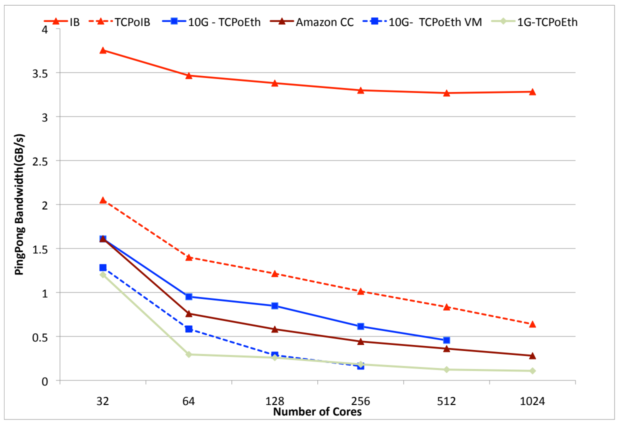
The recent emergence of cloud computing as a potential platform for scientific computing has resulted in the need to revisit scientific computing environments and consider the performance that is possible in these environments. Previous work has shown that virtualized cloud environments impact the performance of tightly coupled applications. However, studies conducted on Amazon EC2 provide limited understanding of the causes of the performance decrease due to the black box nature of these cloud services. Thus, we use the Magellan testbed to understand the impact of performance on applications with different networking protocols and fabrics.

InfiniBand interconnects are common in supercomputing centers due to their performance and scalability. The InfiniBand transport protocol provides a richer set of capabilities than TCP by leveraging the

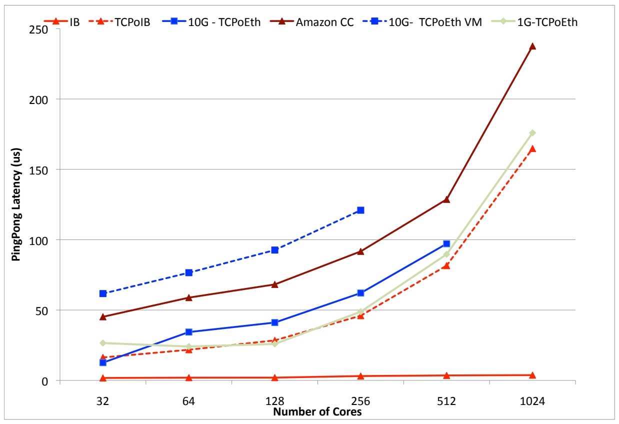
Table 9.3: HPCC performance. All bandwidths are in GB/s and latencies in μs.

Cores	Fabric/Protocol	PingPongLat	NatOrLat	RandOrLat	PingPongBW	NatOrBW	RandOrBW
32	IB	1.72856	1.50204	2.79134	3.7545	1.05444	0.409803
32	TCPoIB	16.3001	18.4059	32.6339	2.0509	0.82211	0.312221
32	10G - TCPoEth	12.6114	13.6137	40.0417	1.60887	0.419289	0.0808118
32	Amazon CC	45.1947	39.0053	76.4316	1.6116	0.276396	0.0812501
32	10G- TCPoEth VM	61.6953	61.4166	222.948	1.28227	0.112029	0.0143262
32	1G-TCPoEth	26.5507	65.2075	84.1106	1.20135	0.0450691	0.00994543
256	IB	3.08863	4.00543	14.5425	3.29896	1.04822	0.284198
256	TCPoIB	45.994	39.196	66.4258	1.01308	0.614246	0.221942
256	10G - TCPoEth	62.0875	43.1061	86.4562	0.613839	0.272497	0.0613771
256	Amazon CC	91.6839	87.2135	228.95	0.442522	0.226378	0.0585243
256	10G- TCPoEth VM	120.933	120.306	434.962	0.162396	0.0806004	0.0123568
256	1G-TCPoEth	48.8237	38.5046	67.7118	0.182378	0.0512721	0.00251061

underlying link layer protocol and remote direct memory access (RDMA) features. We consider this as a baseline for performance benchmarking and quantify the loss in performance from each different configuration. Specifically we compare the following configurations: (a) Infiniband (RDMA) over Infiniband, (b) TCP over Infiniband, (c) TCP over 10G Ethernet, (d) Amazon Cluster Compute (CC), (e) TCP over 10G Ethernet in virtual machines, and (f) TCP over 1G Ethernet. We use standard benchmarks such as HPC Challenge (HPCC) [41] and application benchmarks to understand how communication protocols impact performance and how they compare with the performance in virtualized environments. This approach enables us to determine the performance impact of the different protocols separately from the overhead from virtualization.



(a) Ping-Pong Bandwidth

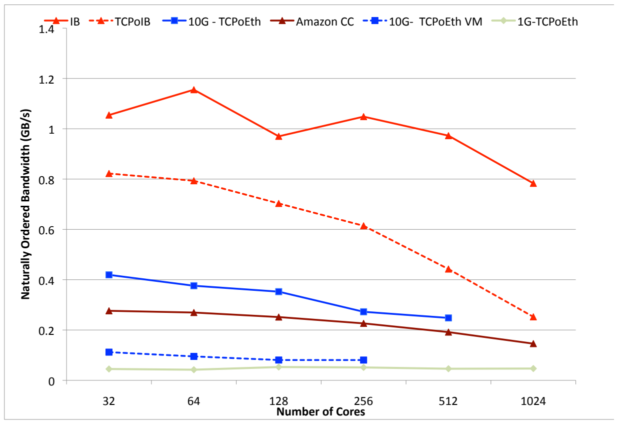


(b) Ping-Pong Latency

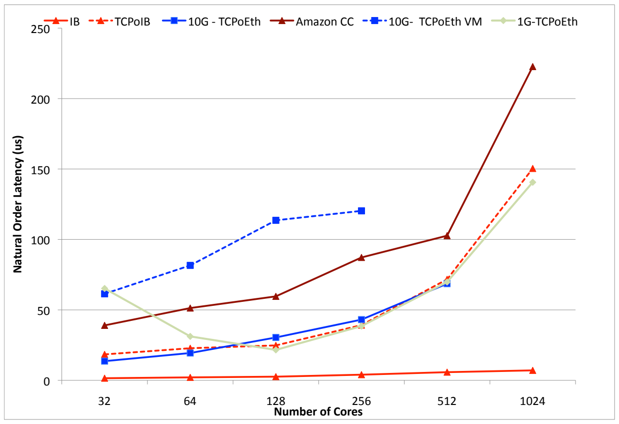
Figure 9.6: Measurements of ping-pong (a) bandwidth and (b) latency as a function of core count for various interconnect technologies and protocols.

HPCC

Table 9.3 shows the latency and bandwidth for different interconnects and protocols at concurrencies 32 and 256. The table shows the results of three types of network measurements: ping-pong, random ring, and natural ring. Ping-pong is a measure of point-to-point bandwidth; random ring consists of each task simultaneously sending to a randomly selected partner; and natural ring sends messages to another partner in its natural order. The sequence ping-pong, natural ring to random ring represents an increase in the amount

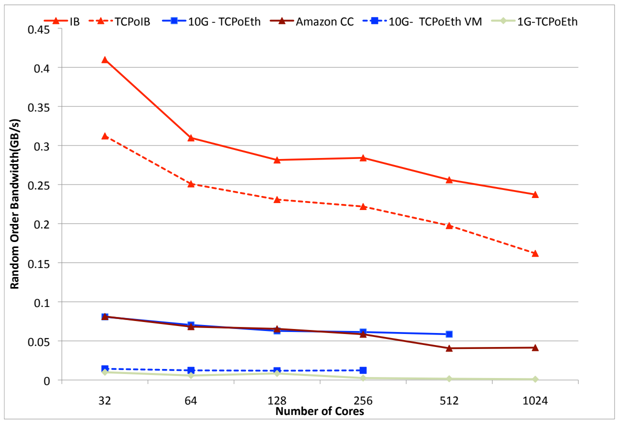


(a) Naturally Ordered Ring Bandwidth

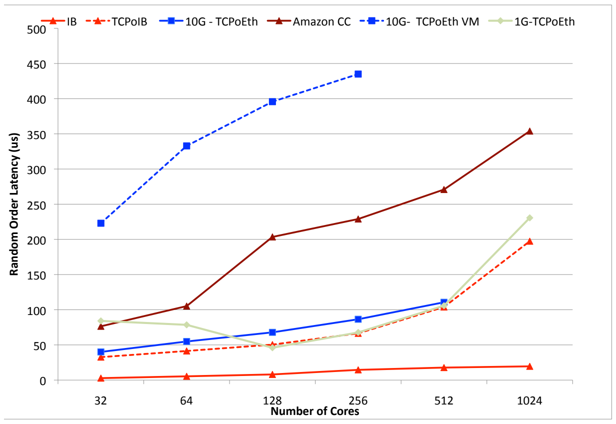


(b) Naturally Ordered Ring Latency

Figure 9.7: Measurements of naturally ordered ring (a) bandwidth and (b) latency as a function of core count for various interconnect technologies and protocols.



(a) Random Ring Bandwidth



(b) Random Ring Latency

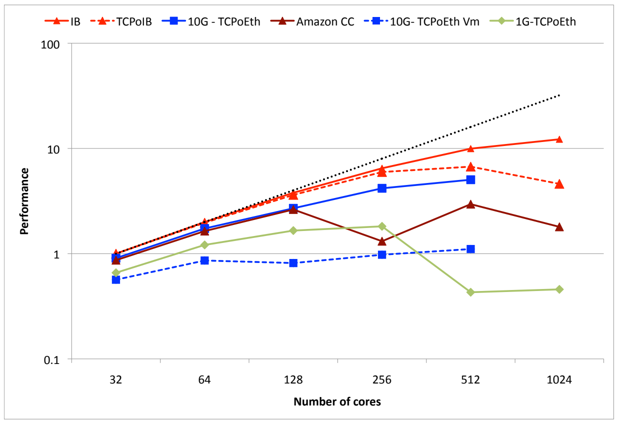
Figure 9.8: Measurements of random ring (a) bandwidth and (b) latency as a function of core count for various interconnect technologies and protocols.

of network contention and thus allows an understanding to be gained of how a particular interconnect will behave under increasing load, as well as if particular interconnects cope worse under load than others. Figures 9.6, 9.7 and 9.8 show the results graphically across all the core counts measured.

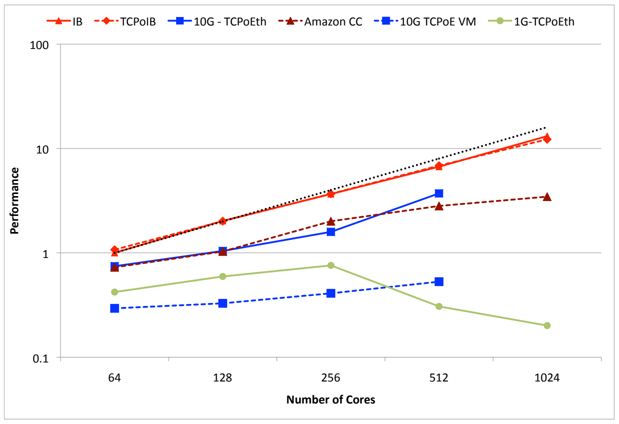
The ping-pong latency results at 32 cores shows that InfiniBand (IB) has the lowest latency. The latency of 10G is about 6× that of IB. At higher concurrency (256), the 10G TCP over Ethernet (TCPoEth) latency increases significantly, showing almost a 5× increase from 32 cores. InfiniBand continues to show low latency at this concurrency, while all the others show ∼ 2 to 2.5× increase in latency. The Amazon CC and the 10G TCPoEth VM latency plots show a similar trend to the 10G TCPoEth but show about 4 to 5× increase in latency, presumably from the virtualization overhead. The principal increase in latency occurs when switching to TCP over IB; then a further (much smaller) increase occurs with the different configurations. This indicates the latency is primarily a function of the transport mechanism, as one would expect. The random ring latency shows a 6× increase in latency at the VM level from the 10G TCPoEth native at 32 cores, showing the impact of contention at the virtualization layer.

The ping-pong BW of the TCPoIB connection is slightly better than 10G TCPoEth at 32 cores but almost 2× better at 256 cores. There is minimal impact from the virtualization on the Bandwidth at 32 cores but significant impact at 256 cores. The random ring bandwidth clearly shows the lack of capability of the Ethernet connection to cope with significant amounts of network contention.

Generally speaking, the results show that as the contention is increased, the performance of all the interconnects decreases. It is also clear that the latencies are affected by contention by a greater amount than the bandwidths; and that as the core counts increase, the decrease in the performance of the latencies is greater than that of the bandwidths. Thus not only do the more capable interconnects have measurably better performance at low core counts, their performance advantage increases as the core count and/or the amount of contention increases.



(a) PARATEC



(b) MILC

Figure 9.9: Performance of (a) PARATEC and (b) MILC plotted on a log-log scale as a function of core count using several different interconnect technologies and/or protocols. The dotted line represents ideal scaling based upon the InfiniBand performance at the lowest core count.

PARATEC and MILC

Figure 9.9 shows the performance of PARATEC and MILC for each of the different interconnect technologies and protocols. The performance for each is shown relative to the InfiniBand performance at the lowest core count. This allows us to show both the relative performance differences between the technologies as well as how they affect parallel scaling. Figure 9.9a shows that for PARATEC, the InfiniBand results are the fastest at all core counts. The change to protocol from InfiniBand to TCP, represented by the TCPoIB line, only affects performance at the higher core counts, 512 and 1024, where it makes the performance 1.5× and 2.5× slower than the native IB, respectively. Presumably this is because at these concurrencies the extra overhead required for TCP as compared to native InfiniBand communication starts to have a noticeable impact, since, at these concurrencies, a larger number of smaller messages are being sent as compared to those at lower concurrencies. The 10G TCPoEth performance is within 10% of Infiniband at 32 cores but drops to about 2× slower at 512 cores. As expected, 1G TCPoEth shows worse performance than 10G. At 32 cores it is 1.5× slower than IB and at 256 cores it is about 3.5× slower. At 512 cores the runtime increases to more than 1.5× the 32 core value, indicating that the interconnect simply cannot keep up with the demands placed upon it by the application at this core count. The 10G TCPoEth VM results are by far the poorest. They show that the overhead of virtualization, at least as configured on the Magallen cluster, is significant. At 32 cores, the performance is 1.75× worse than IB. As the core count increases, the performance does not increase, and the 10G VM line is not parallel to the 10G one, which indicates that the performance degradation due to virtualization is increasing as a function of core count. The Amazon CC results mirror the 10G TCPoEth results at low core counts, which is a little surprising as the Amazon cluster also has virtualization. However, the virtualization technologies underlying Amazon are different from that on the Magellan cluster. Additionally, Amazon likely has a more highly tuned VM environment than what is available on the Magellan cluster through vanilla installation of open-source cloud software. At 256 cores and above, the performance of Amazon CC starts decreasing, as compared to 10G TCPoEth, possibly because the performance of the 10G switch on the Magellan cluster may be higher than that on the Amazon cluster, or possibly due to virtualization overheads at higher core counts, similar to the 10G VMs on Magellan.

Figure 9.9b shows that for MILC, InfiniBand is also the fastest at all core counts. In this case the TCPoIB results are indistinguishable from the native IB. Also, the performance at the highest core count, 1024, is still extremely good, indicating that we are still in the region where the application is scaling well, in contrast to the PARATEC case. The 10G TCPoEth is minimally 35% slower at all core counts. The performance of 1G TCPoEth is about 2.5× slower than InfiniBand at 64 cores and is about 4.8× slower at 256 cores. Again, above 256 cores the interconnect simply cannot keep up. The 10G TCPoEth VM results are by far the poorest. At 32 cores, the performance is 3.4× worse than IB; at 256 cores, it is almost 9× worse. The Amazon CC numbers almost exactly mirror the 10G TCPoEth, in contrast to the PARATEC case, due to the better scaling behavior of MILC at these core counts.

In both cases the performance difference between the interconnect technologies is smallest at low core counts, which makes sense, as that is the point at which the applications are communicating the least. As the concurrency increases, the differences between the performance for each interconnect type become greater because the applications are communicating more frequently, sending more data and thus stressing the interconnect more. This is identical to the trends we observed for the HPCC data, and is even more true for PARATEC than for MILC at these core counts.

Generally speaking, HPC applications are run at the highest possible concurrency at which they run efficiently, in order to minimize the wall clock time to solution. These results demonstrate the productivity advantages that a scientist can gain by using a computing resource that has a high performance interconnect. For example, PARATEC using 256 cores is 1.5× faster using InfiniBand than 10GE. Thus a cluster that was 1.5× smaller in terms of number of nodes could be purchased to achieve the same throughput. Note that for 1G TCPoEth, the ratio is 3.5×.

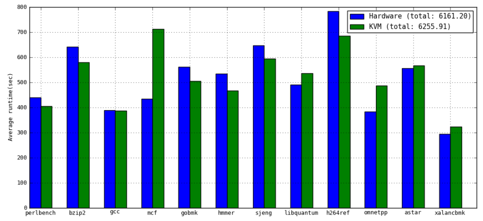
We also note that the basic performance trends can be understood with reference to the HPCC data. Again, the greater the congestion and the core count, the worse the low performing interconnects are. Thus we see that the interconnects performance has significant impact on overall performance for scientific applications, even at mid-range scales.

9.2 OpenStack Benchmarking

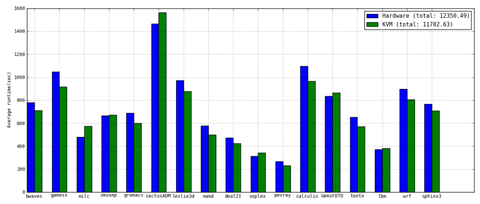
All of the tests in this section were run on the ALCF OpenStack virtual machines or virtual clusters, and a corresponding number of IBM x3650 servers were used for the bare-metal (or 'raw') comparison. This allowed us to determine the precise overhead cost of running in an open-stack virtualized environment. All software between the virtual and raw hardware clusters was identical, and MPICH2 v1.4 was used by the applications.

9.2.1 SPEC CPU

SPEC-CPU2006 is an intensive benchmarking suite made available by the Standard Performance Evaluation Corporation [78]. It is designed to stress the CPU, memory, and compiler of a given system for the purpose of acquiring general performance numbers. A total of 29 benchmarks were run in two passes on both the virtual machine and on raw hardware; the first pass used integer operations and the second pass used floating point operations. Only the time-to-solution is reported below.



(a) Integer



(b) Floating point

Figure 9.10: SPEC CPU benchmark results.

At first glance, the results in Figure 9.10 seem to show greater performance for the virtualized machine; the most notable exception being MCF. This particular benchmark is noted as having been modified for increased memory cache performance. Two other benchmarks that show better performance on the raw hardware are MILC and CactusADM; these are also known as particularly memory-cache friendly applications. This leads us to believe that KVM hides the performance penalty associated with cache misses. As such, CPU cycles spent idle while an application replenishes its L2 cache are only added to the total runtime on hardware. Further investigation of the impact of the memory cache was out of the scope of this project. Other studies have shown the effects of memory performance in VMs[43]

9.2.2 MILC

As an example of a primarily compute-bound scientific application, the MIMD Lattice Computation[61] application was chosen as the second benchmark. MIMD Lattice Computation is a commonly used application for parallel simulation of four-dimensional lattice gauge theory. Two passes were run on each cluster, using one core per node and 8 cores per node, respectively.

The results in Figure 9.11 show an almost six-fold increase in runtime for the one-core-per-node run, which is almost certainly a result of poor MPI performance due to high latency (quantified below in the Phloem SQMR discussion). As for the 256-core run, while the time-to-solution improves for the raw hardware cluster, it increases by a factor of five for the virtual cluster. Again, this is a result of poor MPI performance due to high latency, an exponential increase in the number of core-to-core messages being passed, and contention among processes for network resources. This is further exacerbated by the OpenStack network model which requires a single network service to mediate all internal traffic.

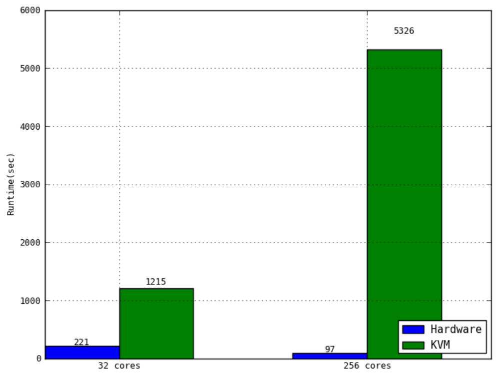


Figure 9.11: MILC benchmark results on OpenStack run on 32 nodes (1 and 8 cores per node).

9.2.3 DNS

DNS3D, a computational fluid dynamics application, was chosen as our second real-world MPI application for testing purposes. It was considered both because of its nature as a primarily network-bound application and its widespread use among many scientific disciplines.

Ten trials were run using 4-cores-per-node, 8-node clusters on both virtual machines and raw hardware. The average time-to-solution for the virtual and raw hardware clusters was 2.721 and 0.214 seconds respectively, giving a factor of ten performance penalty for running in a virtual environment. Like the previous test, this is again most likely a result of high network latency and contention for network bandwidth.

9.2.4 Phloem

For a general set of MPI benchmarks, the Phloem toolkit was chosen [68]. This suite of applications was designed for diagnosis of performance problems in an MPI cluster, and it was run with the expectation of determining which aspects of virtual networking in particular are detrimental to the performance of parallel applications. The first application shown is SQMR, a general MPI point-to-point message bandwidth benchmark, run on a 16-rank 2-node cluster. The results shown below are averages of ten 1000-round runs.

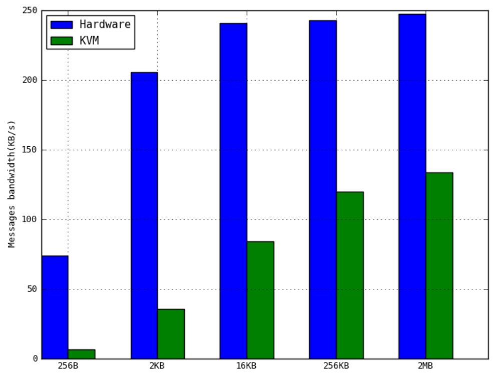


Figure 9.12: Phloem SQMR point-to-point average throughput (10K rounds).

As message size increases, overhead related to latency and connection instantiation become less significant. The results shown in Figure 9.12 demonstrate that while the raw hardware cluster has neared its practical maximum bandwidth at 16 KB, the virtual cluster is still 50 kbps below its average at 2 MB. Of particular note, variability in the results on the virtual cluster was incredibly high; with a 2 MB message size, a maximum reported bandwidth value of 181.74 kbps and a minimum of 32.24 kbps were reported.

The difference between clusters in zero-length message bandwidth was also of concern. While an average of 329,779 messages passed between ranks every second on the raw hardware cluster, the virtual cluster could only manage an average of 21,793 per second. This is a strong indictment of the virtual network model as it pertains to MPI workload performance.

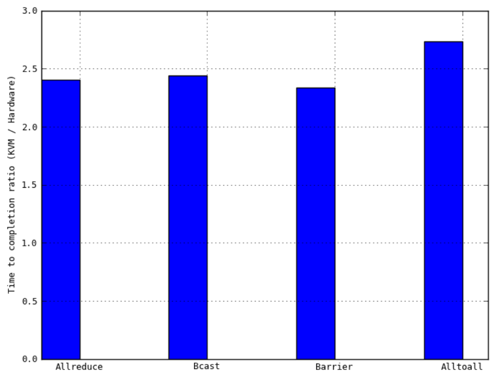


Figure 9.13: Phloem: Selected MPI method tests, average performance ratio (16 ranks, 8 ranks per node).

Figure 9.13 shows the performance penalties associated with individual one-to-many and many-to-many MPI calls, as measured by the Phloem mpiBench_Bcast utility, run on a 16-rank 2-node cluster to maintain relevance with regard to the above SQMR tests. These are the averages of the per-method virtual cluster over raw hardware cluster time-to-solution ratios, chosen because they remain remarkably consistent across varying message sizes. This demonstrates that, in aggregation, the high point-to-point latency illustrated by the SQMR tests will result in a consistently reproducible multiplier for general parallel coding methods.

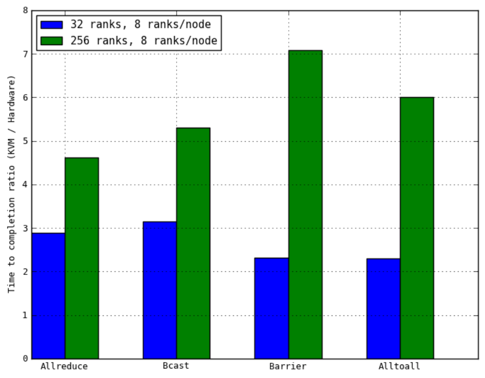


Figure 9.14: Phloem: Selected MPI method tests, average performance ratio across 1000 trials.

Figure 9.14 shows the scaling characteristics of the virtual cluster as compared to raw hardware, again charted as a ratio of virtual cluster time-to-completion over raw hardware time-to-solution. Both sets of trials used 8 ranks per node, with one using a total of 32 ranks, and another using 256. The most notable performance degradation relative to hardware occurs on an MPI barrier call, which falls off by an extra factor of five.

9.3 I/O Benchmarking

This work appeared in the DataCloud Workshop held in conjunction with SC11, Seattle, November 2011 [34] and received the Best Paper award at the workshop

Increasingly scientists are generating and analyzing large data sets to derive scientific insights. Cloud computing technologies have largely evolved to process and store large data volumes of web and log data. Our earlier results have shown that the communication-intensive applications do poorly in these environments. However, there is limited understanding of the I/O performance in virtualized cloud environments. Understanding the I/O performance is critical to understanding the performance of scientific applications in these environments.

I/O is commonly used in scientific applications for storing output from simulations for later analysis; for implementing algorithms that process more data than can fit in system memory by paging data to and from disk; and for checkpointing to save the state of application in case of system failure. HPC systems are typically equipped with a parallel file system such as Lustre or GPFS that can stripe data across large numbers of spinning disks connected to a number of I/O servers to provide scalable bandwidth and capacity. These file systems also allow multiple clients to concurrently write to a common file while preserving consistency. On systems such as those at NERSC, and most DOE HPC centers, often there are two file systems available: local and global. Local file systems accessible on a single platform typically provide the best performance, whereas global file systems simplify data sharing between platforms. These file systems are tuned to deliver the high performance that is required by these scientific applications. Thus it is critical to understand the I/O performance that can be achieved on cloud platforms in order to understand the performance impact on scientific applications that are considering these platforms.

In this study, we evaluate a public cloud platform and the Eucalyptus cloud platform available on the Magellan testbed. We benchmark three instance types: small, large, and Cluster Compute, the specialized HPC offering. The Magellan virtual machine testbed runs the Eucalyptus 2.0 cloud software stack on top of KVM and uses Virtio for disk access.

We used IOR [44] benchmarks and a custom timed benchmark for analyzing the I/O performance on clouds. We compared the performance of different instance types, both local and block store and different availability regions on Amazon to understand the spectrum of I/O performance.

9.3.1 Method

We evaluate the performance of I/O that impacts the overall performance of the applications running in virtual machines on the cloud. Previous work has shown that virtualized cloud environments impact the performance of tightly coupled applications. However, studies conducted on Amazon EC2 provide limited understanding of the causes of the performance decrease, due to the black box nature of these cloud services. The performance impact has been suspected to be from I/O and networking aspects of these virtualized resources. We measure the I/O performance on a range of Amazon resources and the Magellan testbed to understand the impact of various storage options on the performance of applications.

We measure I/O performance using standard and custom benchmarks on the cloud platforms mentioned above over different dimensions. We use the IOR (Interleaved or Random) benchmark to compare the I/O performance across all platforms. In addition, we developed a timed I/O benchmark that records the I/O performance over a period of time at predetermined intervals to assess variability. We also measure the performance of various storage options on virtual machines, and on Amazon record the performance across two availability regions.

Workloads. We executed two sets of benchmarks to analyze and gather statistics of I/O performance on different virtualized cloud environments: (a) IOR and (b) timed benchmark. IOR (Interleaved or Random) is a popular benchmarking tool for understanding the I/O performance of high-performance parallel file systems on HPC systems. IOR can create a variety of I/O patterns using several I/O interfaces (POSIX, MPI-IO, HDF5) with a variable number of clients. IOR provides the capability to set both block and transfer sizes of the files to be read and written. We experimented with the block size and transfer size, and all results reported were executed for a block size of 100G and a transfer size of 2M, which gives the best results among all other transfer sizes. The I/O type can also be set through a flag to ensure that the I/O operations are all direct, instead of the default buffered I/O. The timed benchmark is a custom benchmark designed to measure the I/O performance of a file system over a period of time. The two major factors controlling the timed benchmark, apart from the block and transfer size, are the total time duration and the frequency interval at which the performance is measured. The timed benchmark has been tested to give equivalent results to that of IOR, with very little overhead. This benchmark helps in understanding the variability of I/O performance on different instance types over time.

Machine Description. We perform our experiments on Magellan and Amazon instances. All codes were compiled with GCC version 4.1.2 and used version 1.4.2 of OpenMPI. All tests were performed on instance type c1.xlarge that is configured with 8 CPUs/20G memory/20G disk. The guest OS is CentOS release 5.5 (Linux kernel 2.6.28-11). For the MPI-IO tests, we created a virtual cluster where the head node mounts a block store volume that has all the application binaries and data. The block store volume is mounted on the other virtual machines via NFS. All the virtual machine communication traffic goes over the Ethernet network.

Apart from the VM setup, the tests were also executed to benchmark against a high-speed NERSC file system, Global Scratch. This file system uses IBM's GPFS and has a peak performance of approximately 15 GB/sec. This result was used to understand the potential IO performance impact of switching virtualized environments.

The tests on Amazon's EC2 cloud were performed on three instance types: m1.small (small instance), c1.xlarge (high-CPU extra large instance), and cc1.4xlarge (Cluster Compute quadruple extra large instance). These types were carefully selected in order to cover most of the varied resource combination provided by Amazon EC2. The guest OS for the small and xlarge instances was Fedora 8 (Linux kernel 2.6.18-xenU-ec2-v1.0), 32- and 64-bit respectively; and for the Cluster Compute (CC) instance it was CentOS release 5.4. The configuration of Amazon instances is summarized in Table 9.4. The connectivity of the CC instances was over a 10 Gigabit Ethernet network to provide low latency and high bandwidth to the instances within a cluster. The table also shows the advertised I/O performance expected for each of these instances.

Table 9.4: Architecture of Amazon EC2 Instance Types. Source: http://aws.amazon.com/ec2/instance-types/

Inst-type	API name	CPU Family	Expected I/O Performance	EC2 Compute Units	Memory (GB)	Local Storage (GB)
small	m1.small	Intel Xeon E5430	Moderate	1	1.7	160
xlarge	c1.xlarge	Intel Xeon E5410	High	20	7	1690
cc	cc1.4xlarge	2 x Intel Xeon X5570	Very High	33	23	1690

In all cases we ran the benchmarks three times and report the best result. In most cases the three measurements were in close agreement with each other. In other cases, we study the variability aspects in greater detail through our timed benchmark.

Evaluation Criteria. In our evaluation we try to cover the breadth of parameters that can affect the I/O performance on virtualized environments.

Direct and Buffered I/O. We measure both buffered I/O and direct I/O to understand the effects of buffering caches. We identify how the I/O operations (both read and write) perform based on whether the I/O is buffered or direct. Buffered I/O uses buffer caches to prefetch and store the data in anticipation of the application asking for it, resulting in performance gains. Direct I/O does not use caches and hence the memory is freed as soon as the I/O is complete. Thus, direct I/O reduces the CPU overhead associated with I/O by eliminating the data copy between the kernel buffer and the user buffer.

Virtual Machine Instance Types. The commercial cloud vendors provide instances with different machine configurations. As shown in Table 9.4, the I/O performance advertised by Amazon on each of these instances is also different. This makes it important to analyze the I/O performance to determine the cost-benefit model enabling applications to make the appropriate choice based on the I/O performance they might desire. We measure the performance on three different instance types.

Storage Devices. Virtual machines available on Amazon and through private cloud software solutions such as Eucalyptus provide multiple storage options. Virtual machines have access to non-persistent local disk on the instance. In addition, an instance might mount a block level storage volume that persists independently from the life of an instance. Amazon Elastic Block Store is designed to be highly available, highly reliable storage volume suitable for applications requiring a database, file system, or access to raw block level storage. Additionally, Amazon S3 is an Internet simple storage service that offers a highly scalable, reliable, and low-latency data storage infrastructure through a web service interface. We benchmark the I/O performance on local ephemeral devices and EBS volumes. We do not consider Amazon S3 in our benchmarking since it has limited applicability to our applications. Thus, our benchmarking effort analyzes the effects of shared vs dedicated and local vs networked disk storage in virtual machines.

Availability Regions. Amazon provides the ability to place instances in multiple locations or regions that are geographically dispersed. In the US, Amazon provides services in two regions—US East (Virginia) and US West (Northern California)—with slightly different price points. Cluster compute instances are only available currently in the US East region.

Shared File System. MPI is used extensively in high-performance computations where multiple processes write onto a shared file system. An important analysis would be to measure the performance of an application running multiple instances and sharing a file system. Scientific applications that require a shared file system often use an EBS volume shared across instances through NFS.

Time of Run. Cloud virtual instances often share the underlying resources with other virtual instances, thus resulting in variable performance. Hence, it becomes important to analyze the I/O patterns over a period of time. We performed a study over a few days and over a large number of instances to understand this variability.

We used the selected benchmarks and configuration parameters and conducted multiple tests to capture the variation and trends of I/O performance on the Magellan testbed and the Amazon cloud infrastructure. Specifically, we compared the effects of buffer caching and IOR results across all platforms, and we evaluated the effects of MPI-IO in these virtualized environments. We also conducted large-scale tests and 24 runs to understand the variability experienced on these platforms.

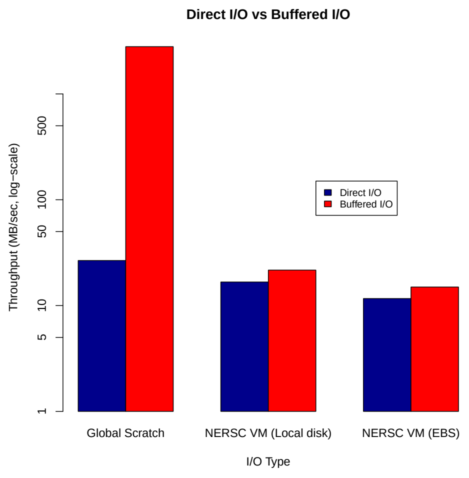


Figure 9.15: IOR Results: Comparison of Direct vs Buffered I/O on NERSC systems

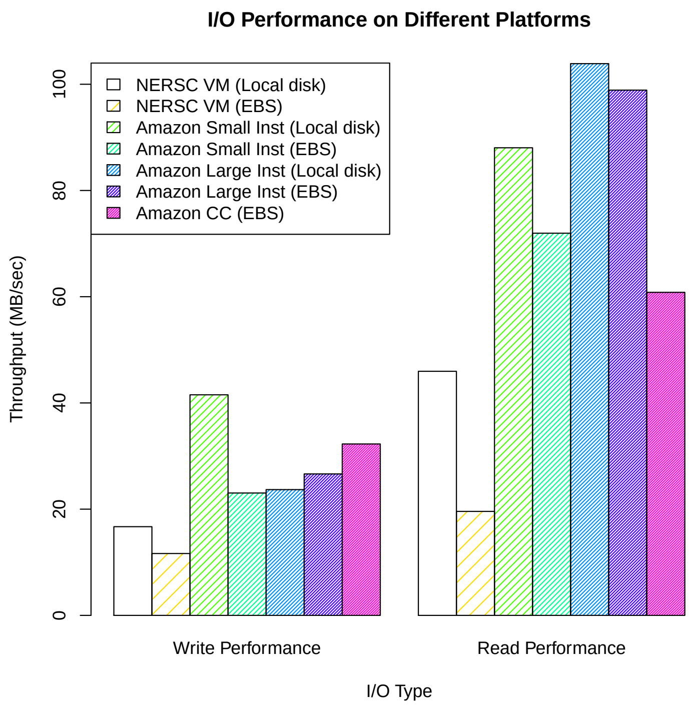
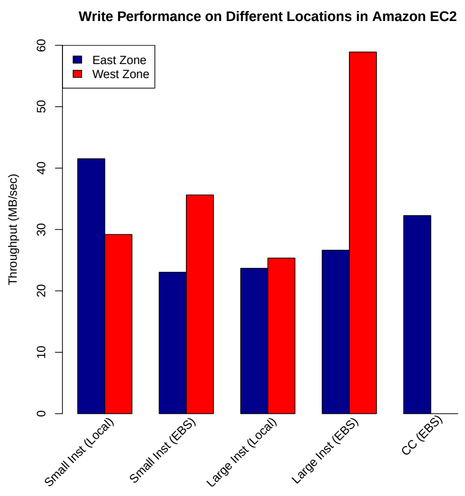
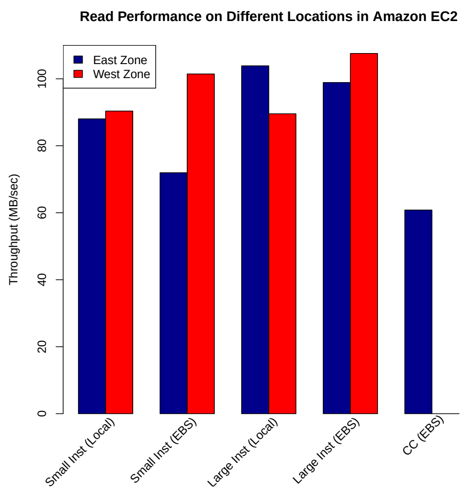


Figure 9.16: IOR Results: I/O Performance on All Platforms



(a) Amazon Write Performance



(b) Amazon Read Performance

Figure 9.17: IOR Results: Comparison of Amazon platforms

9.3.2 IOR Results

In this section we report our results from the IOR benchmark. For all tests, a block size of 100G and a transfer size of 2M were used.

Buffer Caching

Figure 9.15 shows the performance of direct I/O and buffered I/O on different configurations on the Magellan testbed. Buffer caching does not seem to be enabled on Amazon instances. As expected, the global scratch file system at NERSC shows significant performance improvement with buffer caching enabled. High performance file systems such as GPFS are optimized to provide high performance using buffer caching, etc. On the NERSC VMs, there was a very small increase in performance. Thus buffer caching seems to have minimal or no impact on virtualized resources. The remaining results in our evaluation only consider direct I/O.

Comparison Across All Platforms

Figure 9.16 shows a comparison of I/O performance on all the virtualized cloud environments. The summarized performance results show the impact of network and disk bandwidth and latencies on the I/O performance. The I/O performance on Amazon instances is better than on the VMs at NERSC. These differences could be from the differences in underlying hardware as well as limited performance tuning done on the NERSC virtual machine systems. Note that this performance is significantly lower than what is achievable on NERSC scratch file systems (>2500 MB/s).

The local disk on the instance performs better than the EBS volumes on the VMs at NERSC. On Amazon, the I/O performance on the local ephemeral disks on the small instances is better than writes to the EBS volumes over the network. This may be due to the low network bandwidth for small instances. For the extra-large instances, the EBS volumes show better performance than that of the local disks. This may be attributed to the fact that there is higher network bandwidth associated with these instance types, possibly resulting in better I/O operations on the networked EBS volumes. However, the difference is minimal. Clearly, EBS performance is better with Cluster Compute instances than the other two instance types (for EBS volumes), this is likely due to the use of a 10 Gigabit Ethernet network. The read operations show similar trends except for the Cluster Compute instances, which are significantly lower than expected. I/O

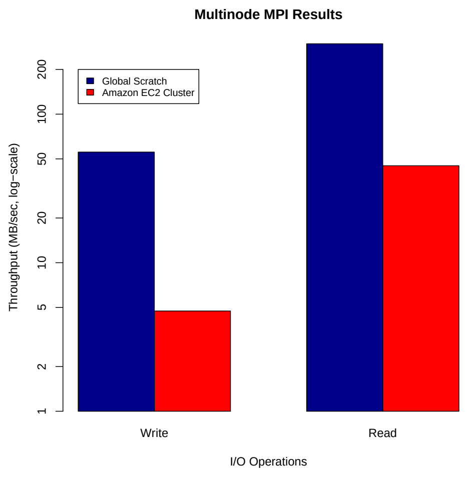


Figure 9.18: Multinode MPI Shared File System Results on NERSC Global Scratch and Amazon Cluster Compute Instances

device virtualization requires a virtual machine monitor (VMM) or a privileged VM for every I/O operation, and hence there is an additional level of abstraction which degrades the performance in VMs. We also observed a fair amount of variability in the VM performance that we try to understand better through the timed benchmark.

Effect of Region

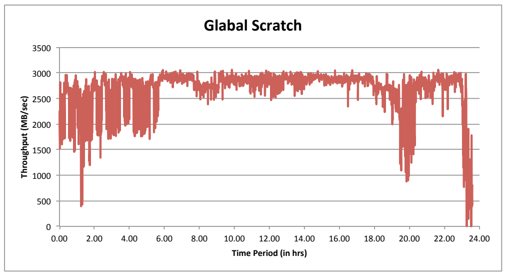
As described in the previous sections, Amazon's instances are located in various places and are divided into regions. The graphs in Figure 9.17a and Figure 9.17b show performance variations due to the location of instances in different regions. In general, the instances in the west zone showed slightly higher performance than the east zone. Note that cluster compute instances are not available in the west zone and also do not have any local disk space. In two cases, the east zone showed better performance than west zone: in writes in small instance (local) and reads in large instance (local disk). This can be attributed largely to the variability we see in performance in Amazon instances that we discuss in the next section.

Multi-Node

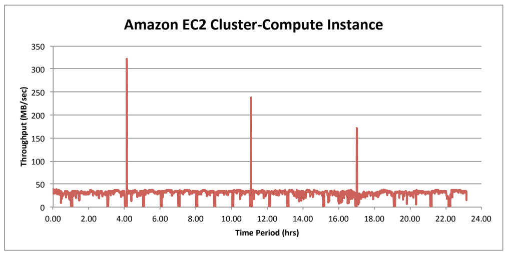
Because a large number of HPC applications use MPI, performance analysis on virtualized cloud environments when a shared file system is used by multiple processes is important to measure. Figure 9.18 depicts the performance comparison between the NERSC global scratch file system and a shared EBS volume mounted with NFS across ten nodes on an Amazon EC2 cluster compute instance type. If multiple instances executing MPI tasks share a single EBS volume to perform I/O, there is high resource contention over limited network, which degrades the overall performance. On the other hand, the NERSC global scratch is a parallel file system that is designed and optimized for such concurrent access. Hence, the nodes in an HPC cluster over a high-speed shared file system perform better than the instances on the cloud, as shown in Figure 9.18.

9.3.3 Performance Variability

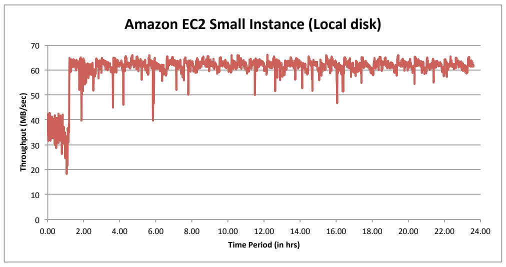
It is critical to understand the I/O performance variability over time and instances. Thus we used our timed benchmark to measure the performance on instances over a 24-hour time period and over a number of instances over one-hour periods.



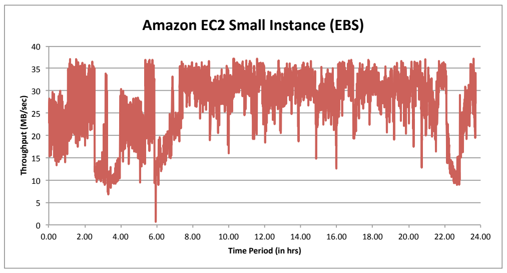
(a) Global Scratch (Buffer caching enabled)



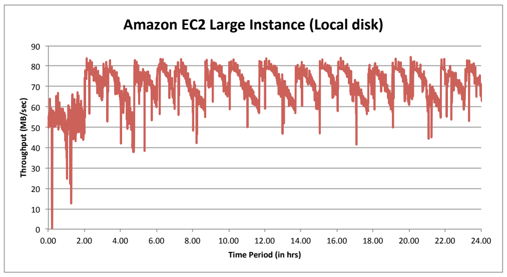
(b) Amazon EC2 Cluster-Compute Instance



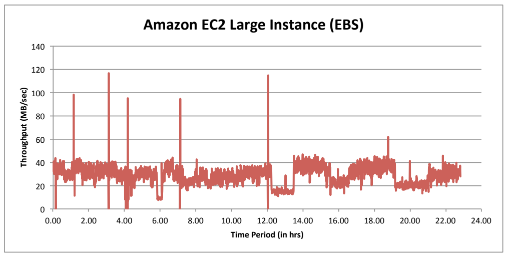
(c) Amazon EC2 Small Instance (Local disk)



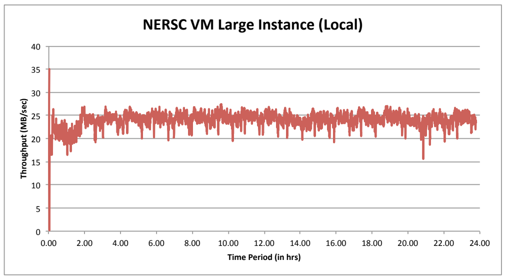
(d) Amazon EC2 Small Instance (EBS)



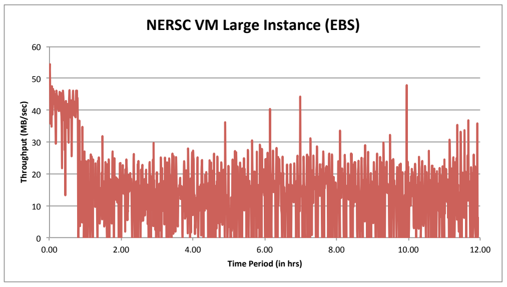
(e) Amazon EC2 Large Instance (Local disk)



(f) Amazon EC2 Large Instance (EBS)



(g) NERSC VM (Local disk)



(h) NERSC VM (EBS)

Figure 9.19: Timed Benchmark Results.

24-Hour Results

The 24-hour cycle tests enable us to study potential changes in performance from typical usage patterns such as peak vs non-peak usage of cloud services. The results in Figure 9.19 shows the timed benchmark results on Amazon and NERSC systems. Figure 9.19a shows I/O performance on the global scratch file system at NERSC. The benchmark was run for a complete 24-hour duration starting at 12 noon PDT. Since the shared file system is being accessed by many users during the daytime, the performance degrades at both ends of the spectrum. After the peak processing hours, there is significant improvement in I/O performance due to limited contention for the shared resources. The I/O performance graphs for the timed benchmark on various virtual environments show more periodic results with occasional spikes. The reason for the occasional spikes is not known at the current time and will be investigated in future work. Figure 9.19g, showing a 24-hour run of the test starting at 12 midnight PDT, suggests a more consistent I/O performance on local disk of a VM instance. The graphs showing the results for EC2 instances clearly signify that the I/O performance doesn't follow a specific pattern over time, but most of results are either consistent or periodic over time. Figure 9.19c and Figure 9.19e show the I/O performance on local disks for different instances. The occasional drop in performance may be due to the sharing of underlying resources. On the other hand, Figure 9.19d, Figure 9.19f, and Figure 9.19b show the performance graphs on EBS volumes over the small, large, and Cluster Compute instances respectively. Since EBS volumes are attached to every instance separately over the network, there is no sharing of disks or volumes; but interestingly there are certain spikes in the graphs which may be attributed to the network traffic.

Large-Scale Tests

These tests are run to gauge the variability of performance for multiple instances over a certain time period. These tests are run on 25 Amazon EC2 small instances on local disk in each region (US East and US West) in the same time period but on different days. All the tests were executed on a peak usage period of cloud resources on both the east and the west zones. We selected the time period between 11:00 AM and 2:00 PM PDT or 2:00 PM to 5:00 PM EDT. Timed benchmark was used to capture readings at regular intervals for one hour to understand the variability among instances. The histogram plots and the kernel density plots for the tests are shown in Figure 9.20. The histogram plots show the frequency distribution of throughput performance, and the density plots using the Gaussian kernel show the corresponding probability density function for the throughput variance.

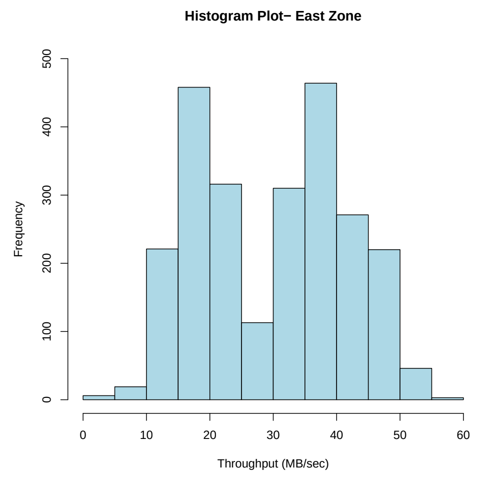
These large-scale tests reveal important characteristics of I/O performance on the US East and US West regions. Mean performance in the east zone has a throughput of 30 MB/sec. The area of the graph in Figure 9.20c with the throughput variance between 10 MB/sec–20 MB/sec and 35 MB/sec–45 MB/sec fall under less than one standard deviation from the mean. The peak performance in the east zone is 60 MB/sec. The performance in the west zone, shown in Figure 9.20d, varies significantly, with the peak performance going up to as high as 88 MB/sec. The mean performance in the west zone is between 10 MB/sec–20 MB/sec, which shows a high standard deviation in the west zone.

Discussion

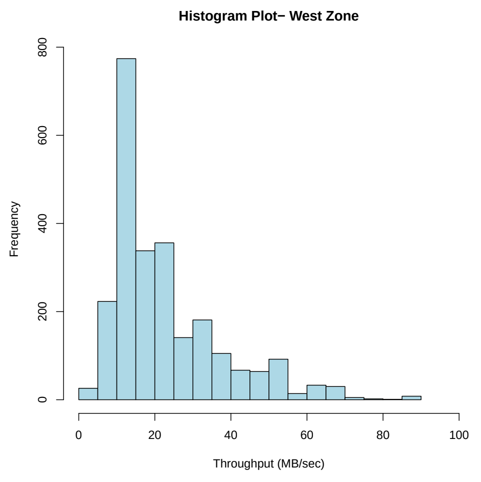
The performance on virtual hosts tends to show a fair amount of variability due to the contention and sharing of underlying resources.

Buffer Caching. Based on our testing, buffer caching does not seem to be enabled on virtualized resources. Scientific applications running in HPC centers are able to use high performance file systems such as GPFS that show significantly higher peak performance than what is seen in today's virtual environments. This can have a significant impact on overall application performance for I/O intensive applications.

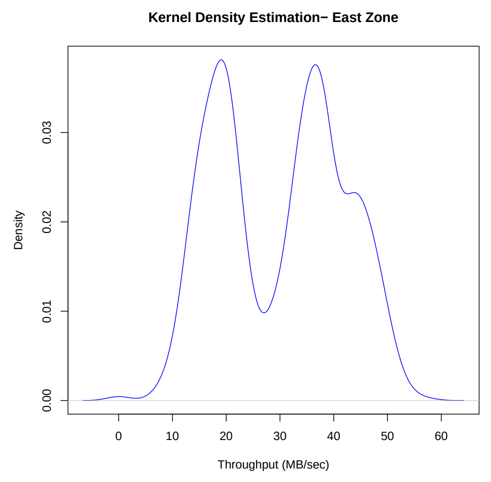
Storage Options on VM. A virtual machine has ephemeral local disk and has the option to mount an elastic block storage volume. Typically, the performance of the local disk tends to be slightly higher than



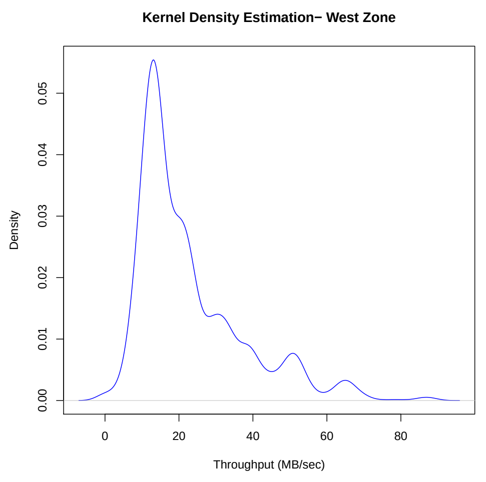
(a) Large Scale Test on Amazon US-East Region



(b) Large Scale Test on Amazon US-West Region



(c) Large Scale Test on Amazon US-West Region



(d) Large Scale Test on Amazon US-West Region

Figure 9.20: Large Scale Test Results Histogram and Kernel Density Plots.

the EBS corresponding volumes. This is especially true in the Amazon small instances that are bandwidth limited. At least during the duration of our tests, we noticed that the larger instance types showed lesser advantage with the local disks, possibly due to the increased network bandwidth available in the large instances. However, users need to do a performance-cost analysis of local disk vs EBS for their particular application [47].

Instance Type. The anticipated I/O performance on the instance type is expected to get better with the larger better instances. However, in our limited testing, we encountered situations where we were able to get better I/O performance on the small instance local disk than the large instance. Our tests also show that the small instances do tend to show a fair amount of variability, and hence more extensive testing might be needed to capture these behaviors over time. The EBS performance appeared to improve with more capable instance types, possibly due to the better network available to the larger and/or the CC instances.

Availability Regions. We observed that the west zone performance was better than the performance of the east zone. The west zone VMs on Amazon have slightly higher price points, possibly resulting in better performance. However, our large-scale tests also show that the west zone has a higher standard deviation than the east zone.

9.4 Flash Benchmarking

Solid-state storage (SSS) is poised as a disruptive technology. This impact would likely affect both the cloud computing and scientific computing spaces. For these reasons, flash storage evaluation was included in the Magellan project. Magellan at NERSC first evaluated several products before deploying a larger storage system based on that evaluation. The technologies that were evaluated include three PCIe connected solutions and two SATA connected solutions. Table 9.5 summarizes the products that were evaluated.

Table 9.5: Summary of flash-based storage products that were evaluated.

Manufacturer	Product	Capacity
PCIe Attached Devices - All use SLC Flash		
Texas Memory Systems	RamSAN 20	450 GB
FusionIO	ioDrive Duo	320 GB
Virident	tachIOn	400 GB
SATA Attached Devices - Both use MLC Flash		
Intel	X25-M	160 GB
OCZ	Colossus	250 GB

NAND flash devices have dramatically different performance characteristics compared with traditional disk systems. NAND typically delivers much higher random read rates. However, NAND chips must be erased prior to writing new data. This erase cycle can be extremely slow. For example, NAND may require several milliseconds to erase a block, yet can perform a read of block in several microseconds [79]. To help mask this impact, many high-end devices utilize background grooming cycles to maintain a pool of erased blocks available for new writes.

Early benchmarking efforts focused on measuring the bandwidth the devices can deliver. Plots in Figure 9.21 show the performance of the devices across a range of blocks sizes for both sequential read and write operations. As expected the PCIe attached devices outperformed the SATA attached devices. More interestingly, the solid-state devices are still sensitive to the block size of the IO operation. This is likely due to both overheads in the kernel IO stack, as well as additional transaction overheads in the devices. In general, the card devices provide more balanced performance for writes versus reads compared to the MLC based

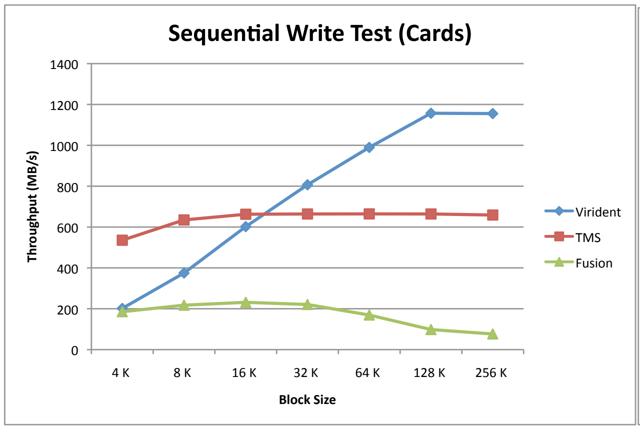
SATA devices. This is likely due to both the better performance characteristics of the SLC Flash devices used in the PCI devices, as well as more sophisticated flash translation logic coupled.

Additional benchmarking focused on identifying how the devices behaved under sustained heavy random-write load to address the impact of any grooming cycles. For some devices this impact can be quite high. For applications with very high-burst I/O followed by several minutes of idle I/O, this grooming cycle could be hidden from the application. However, few applications are likely to be so well behaved, and mixed workloads further complicate the picture.

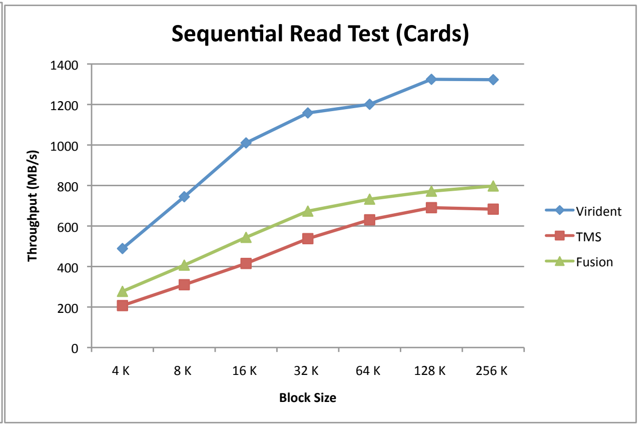
For the degradation experiments, a large block (128 KB) is randomly re-written over a fraction of the device. The measured bandwidth as a function of time is shown in Figure 9.22a. Typically within the first 15 minutes of the experiment, we see quite a bit of noise, which is most likely due to the flash controller switching algorithms as the device transitions from having spare blocks into the process of actually utilizing these blocks. Within about 30 minutes, all of the devices have reached a steady state often with a drastic decline in random-write bandwidth. We also compared the steady state bandwidth as a fraction of peak for each device achieved as a function of fullness which is shown in Figure 9.22b. For the SATA drives the performance degradation is significant, although it shows no variation with time or fullness, typically 5-10% of the peak is observed right from the beginning of the experiment. For the PCI cards, the performance degradation is significant. In this benchmark the Virident tacIOn card is the best performing. It shows the lowest deviation from peak with 30-70% fullness, and is equal to the TMS Ramsan at 90% fullness. The FusionIO ioDrive card performs almost identically to the TMS Ramsam one for 30% and 50% but for 70% and 90% fullness is significantly worse, where it only achieves 15% of its peak bandwidth with 90% fullness.

Additional studies were underway at the conclusion of the project. These efforts focused on understanding application performance on the devices. Applications included databases, out-of-core applications, and applications that perform checkpoint operations. In general, these studies demonstrated that while the underlying flash devices are capable of delivering high bandwidth and IO operations-per-second, the impact to the applications can often be modest. While one expects the superior performance of flash, particularly for random read operations, to lead to improved application performance, experience shows that this is not always the case. Often code paths in the kernel or applications have not been sufficiently optimized to take advantage of the performance the new devices can deliver. In some cases, this is due to decades worth of optimization to deal with the distinct performance characteristics of disk. Swap algorithms and I/O schedulers will often buffer and flush data to minimize the number of writes. However, these optimizations can increase the latency and prevent the flash devices from maintaining a sustained stream of I/O operations. Additionally, many of the devices benefit from a high number of threads issuing write operations. This prevents the buffer queues from being exhausted too quickly. For disk systems, this can have an adverse effect of creating extra head movement, which slows down throughput. The difference in the performance characteristics between disk and solid state storage make it clear that applications need to be evaluated to see what near-term benefit SSS can provide. Eventually, improvements in the algorithms in the kernel, middleware, and applications may be required to realize the full potential. Despite these results, there are many cases were flash is being used effectively. Most of these involve read-intensive random IO. A common example is to accelerate databases that are primarily performing lookup operations.

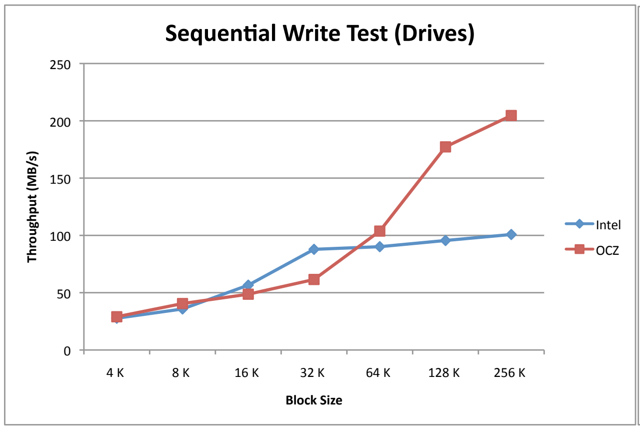
The results of our benchmarking efforts of flash devices illustrate the importance of the interface used to access the storage (PCE versus SATA) and the flash translation logic used in the devices. The study also found that applications may not always realize all of the benefits of the devices due to other limitations and bottlenecks. Future changes in the technologies used in solid-state storage devices will likely effect some of these characteristics. For example, next-generation technologies like Phase Change Memory and Memristor are less sensitive to wearing and do not exhibit the same erasure overheads of NAND flash. However, even with this potential improvements, changes are required in the IO software stack and other areas in order to achieve the full potential of these devices.



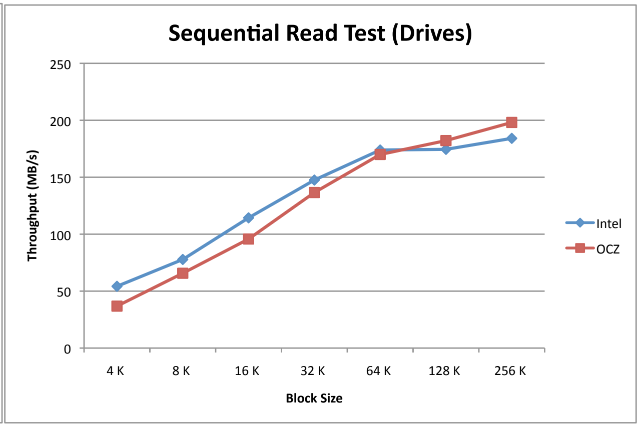
(a) Write Test for PCIe attached Flash Storage



(b) Read Test for PCIe attached Flash Storage

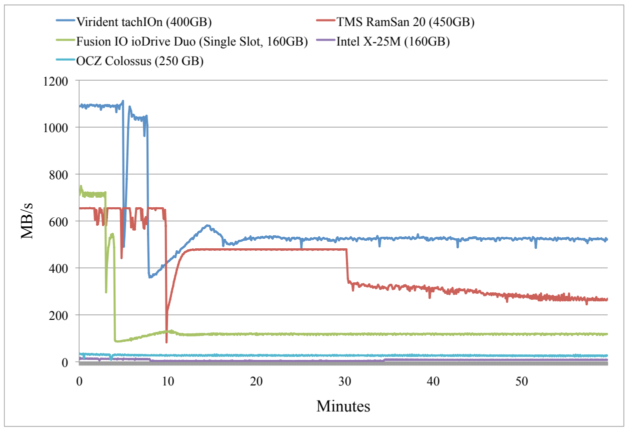


(c) Write Test for SSDs

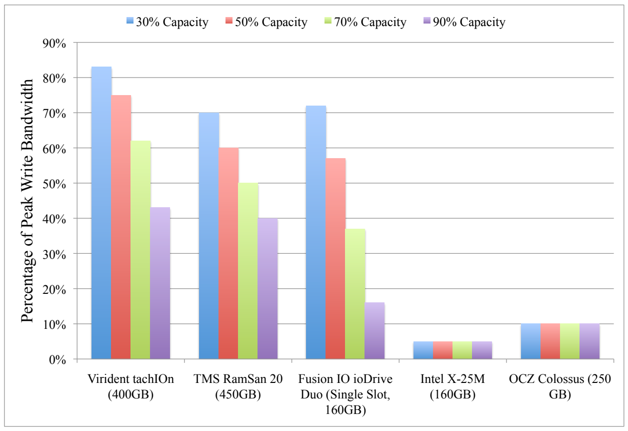


(d) Read Test for SSDs

Figure 9.21: Performance of Sequential Read and Write I/O Benchmarks performed on PCIe attached flash storage devices and SSDs. The plots are for 16 concurrent threads issuing IO with varying block size.



(a) Transient Random-Write Bandwidth Degradation (90% Capacity)



(b) Steady-State Random-Write Bandwidth Degradation

Figure 9.22: Graphs illustrating the degradation in the IO bandwidth to various flash devices under a sustained random write workload. The graph on the left shows the transient behavior while the graph on the right compares the steady-state performance of the devices while varying the utilized capacity.

9.5 Applications

We worked closely with our users to help them port their applications to cloud environments, to compare application performance with existing environments (where applicable), and to compare and contrast performance with other applications. In this section, we outline select benchmarking results from our applications.

9.5.1 SPRUCE

As mentioned in Chapter 3, the Magellan staff collaborated with the SPRUCE team to understand the implications of using cloud resources for urgent computing. In this context, some benchmarking was performed to measure the allocation delay (i.e., the amount of time between a request for some number of instances and the time when all requested instances are available) as the size of the request increases. These benchmarking experiments were conducted on three separate cloud software stacks on the ALCF Magellan hardware: Eucalyptus 1.6.2, Eucalyptus 2.0 and OpenStack. The results of these benchmarks revealed some unexpected performance behaviors. First, Eucalyptus version 1.6.2 offered very poor performance. The allocation delay linearly increased as the size of the request increased. This was unexpected given that in the benchmark experiments, the image being created was pre-cached across all the nodes of the cloud, so one would expect that the allocation delays would have been much more stable, given that the vast majority of the work is being done by the nodes and not the centralized cloud components. Also, as the number of requested instances increased, the stability of the cloud decreased. Instances were more likely to fail to reach a running state and the cloud also required a resting period in between trials in order to recover. For example, for the 128-instance trials, the cloud needed to rest for 150 minutes in between trials, or else all of the instances would fail to start. In Eucalyptus version 2.0, the performance and stability issues appeared to be resolved. The allocation delays were much flatter (as one would expect), and the resting periods were no longer required. OpenStack, in comparison, offered shorter allocation delays, the result of an allocation process which utilized copy-on-write and sparse files for the disk images. As the images were pre-cached on the nodes, this resulted in significantly shorter allocation delays, particularly for smaller request sizes. However, the plot of the allocation delay was again not as flat as might be expected (see Figure 9.23).

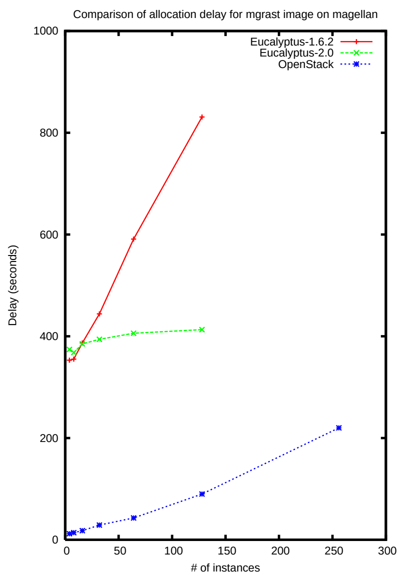
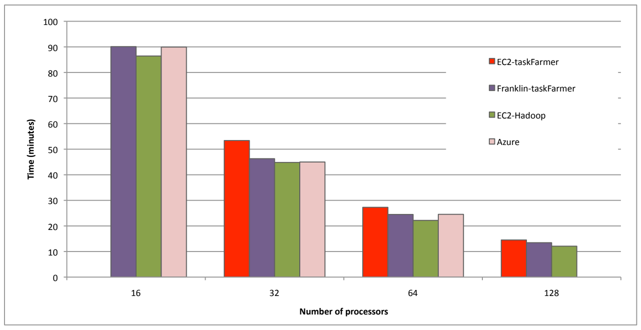
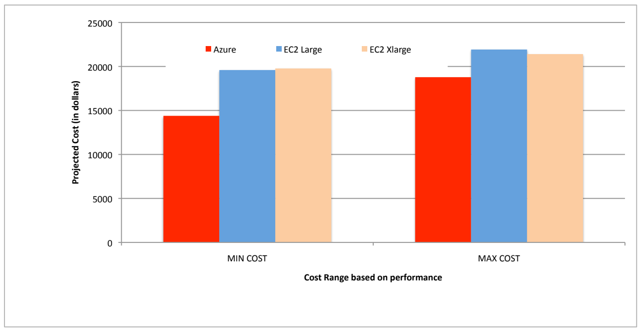


Figure 9.23: The allocation delays for the MGRAST image as the number of requested instances increased on the ALCF Magellan cloud. The delays are shown for the Eucalyptus (versions 1.6.2 and 2.0) and OpenStack provisioning. The MGRAST image was pre-cached across all nodes.



(a) Performance comparison



(b) Projected Cost for 12.5 million gene sequences

Figure 9.24: Performance and cost analysis of running BLAST on a number of cloud platforms

9.5.2 BLAST

The Joint Genome Institute's Integrated Microbial Genomes (IMG) pipeline is a motivating example of a workload with growing needs for computing cycles due to the growth of sequence data expected from next-generation sequencers. One of the most frequently used algorithms in IMG, the Basic Local Alignment Search Tool (BLAST), finds regions of local similarity between sequences. We benchmarked the BLAST algorithm on HPC systems (Franklin at NERSC) and cloud platforms including Amazon EC2, Yahoo! M45 and Windows Azure. We managed the execution of loosely coupled BLAST runs using a custom developed task farmer and Hadoop.

For this evaluation, we ran 2500 sequences against a reference database of about 3 GB, and compared the performance data. Figure 9.24 shows the performance of the task farmer on Franklin and Amazon EC2 and the performance of Hadoop on EC2 and the BLAST service on Windows Azure. For this workload, the performance of cloud platforms compares favorably with traditional HPC platforms.

On Amazon EC2, the database and the input data sets were located on elastic block store (a high-speed, efficient, and reliable data store provided by Amazon for use with one node) that is served out to the other worker nodes through NFS.

We were provided friendly access to the Windows-based BLAST service deployed on Microsoft's cloud solution, Windows Azure. The experiments were run on Azure's large instance, which has four cores, a speed of 1.6 GHz, and 7 GB of memory. This instance type is between the large and xLarge instances we used on Amazon EC2. On Azure, the total time for data loading into the virtual machines from the storage blob and back to the storage blob is also included in the timing. Additionally the data sets were partitioned equally across the nodes, and we see the effects of bunching. Thus the model here is slightly different from our setup on Hadoop but gives us an idea of the relative performance.

We ran the same experiment on Yahoo! M45, a research Hadoop cluster. The performance is comparable to other platforms when running a small number of sequences. The load on the system affects execution time, inducing large amount of variability, but performance seems to be reasonable on the whole. However, when running 250 concurrent maps (10 genes per map, 2500 genes total), there is a heavy drop in performance—the overall search completes in 3 hours and 39 minutes. This seems to be the result of thrashing, since two map tasks, each requiring about 3.5 GB, are run per node with 6 GB memory.

We extrapolated from the running times the cost of commercial platforms to run a single experiment of 12.5 million sequences (run every 3 to 4 months). The cost of a single experiment varies from about $14K to $22K based on the type of machine and the performance achieved on these platforms (Figure 9.24).

There is a reasonable amount of effort required to learn these environments and to get them set up for a particular application. Early results of this work were presented at the System Biology Cloud Workshop at SC09. A manuscript is in preparation with additional experiments that explore different data storage and communication options.

9.5.3 STAR

The STAR experiment data analysis was run across different Magellan systems. More details of the application are presented in our case studies in Chapter 11. The STAR team has also experimented with other cloud systems including Amazon. Table 9.6 shows the performance and cost across different Amazon instances. The team ran on two different types of Amazon instances: (a) Standard Large Instance (m1.large) with 7.5 GB of memory and 850 GB of local instance storage, and (b) High-Memory Quadruple Extra Large Instance (x2.4xlarge) with 68.4 GB of memory and 1690 GB of local instance storage. The m1.large instance is one of the standard instance types, whereas x2.4xlarge is a high-memory instance. Table 9.6 demonstrates the importance of selecting the right instance type for a particular application. Both performance and cost are better for the high-memory instance for the STAR application. The difference in performance between the Amazon high memory instances and Magellan is due to the differences in underlying hardware and memory allocated to the VMs.

9.5.4 VASP

VASP (Vienna Ab-initio Simulation Package) is a package for performing ab initio quantum-mechanical molecular dynamics (MD) using pseudopotentials and a plane wave basis set. This package is available on all major NERSC systems and has a broad user base (close to 200 licensed users at this point). In order to facilitate a comparison between VASP run on bare metal and on a virtual cluster, the user made use of "Little Magellan"—a virtual cluster booted on top of Eucalyptus (details in Chapter 3).

The user ran first an 8-core VASP job (one batch node), and timing was comparable with similar runs on the batch queue system. After that, a set of identical 16-core runs was set up on both the batch queue system and "Little Magellan." The output was checked for accuracy, and the results were exactly the same (as expected) to the last digit of reported precision. However, with two nodes only, the timing on the virtual

Table 9.6: Comparison of STAR performance and cost across different Amazon instance types and Magellan

Instance type	m1.large	x2.4xlarge	Magellan
cores	2	8	8
EC2 units/core	2	3.25	-
EC2 units/instance	4	26	-
Cost (\$)/instance hour	\$0.38	\$2.28	-
Cost (\$)/hour/EC2 unit	\$0.10	\$0.09	-
Wall hours/job	2.119	1.242	1.269
CPU hours/job	1.500	1.240	1.263
Cost (\$)/job	0.403	0.354	-

cluster was 2 to 3 times slower than on bare metal. Due to the performance hit and the need to move data in and out of the virtual cluster, which did not have access to the NERSC Global File System, the user did not consider this a resource environment worth pursuing at the current time.

9.5.5 EnergyPlus

Another type of application that was tested in the "Little Magellan" environment was a code called EnergyPlus (E+) that does building simulations, such as estimating the energy needs of a building before it is built. The user had been tasked with improving the performance, and used OpenMP with great results when comparing serial with 1, 2, 4 and 8 OpenMP threads. Code ran in 10 different configurations (each of them 1, 2, 4 and 8 threads) on both virtual cluster and bare metal. Results for 1, 2 and 4 threads were identical. The results for 8 threads were 1015% slower.

9.5.6 LIGO

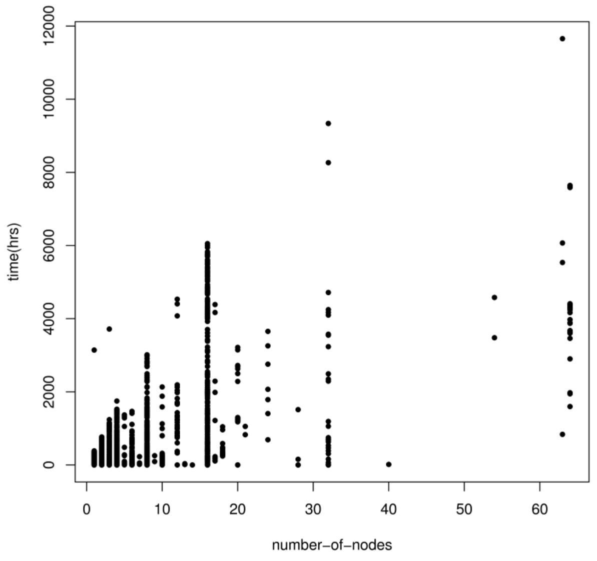
LIGO was another application that used the Magellan testbed through the "Little Magellan" setup. The application is described in greater detail in Section 11.2.3. LIGO users stated that they saw no difference between running through the serial queue on Magellan and on the virtual machines. The nature of the LIGO workload, i.e., minimal data and limited communication, makes it a suitable candidate to use virtualized cloud resources. There was negligible difference in CPU performance between bare metal and virtualized resources for this application.

9.6 Workload Analysis

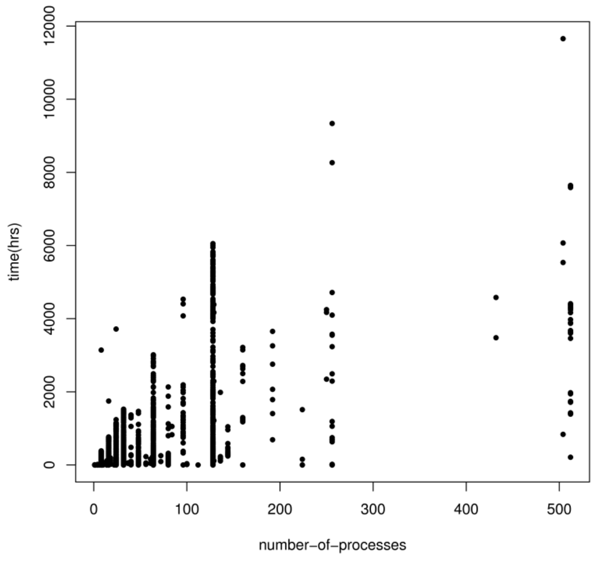
Our benchmarking results show that applications with minimal communication and I/O do well in cloud environments. In order to further understand the range of HPC applications that might benefit from cloud environments, we analyzed the workflows of researchers who were using the Magellan batch queue system at NERSC.

Workload profile information was collected using the Integrated Performance Monitoring (IPM) analysis infrastructure. IPM is a low-overhead profiling infrastructure that provide a high-level report on the execution of a job. IPM reports hardware counters data, MPI function timings, and memory usage. It provides a low-overhead means to generate scaling studies or performance data. IPM is available to users on NERSC systems using modules.

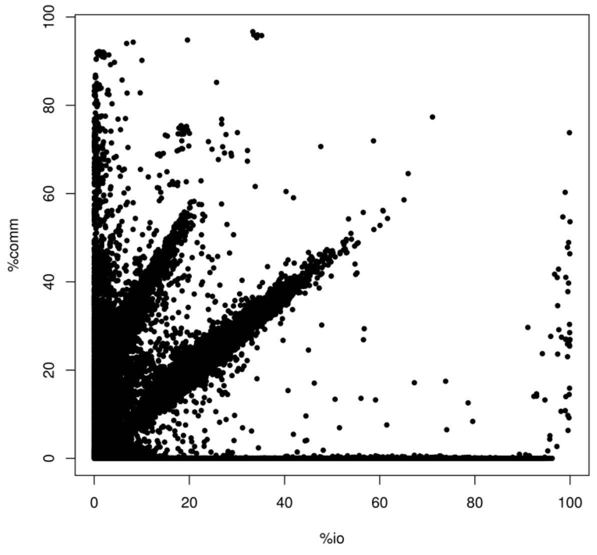
On the Magellan systems, we force IPM profiling information to be collected for every job. The user gets normal output at the end of the job, and a log file is written to a shared space. To aid with the analysis, we built a generic infrastructure using MongoDB, also a cloud technology. MongoDB is a scalable, open source, document-oriented data store that supports dynamic schemas. An existing parsing script in IPM was modified to output a JSON-style document that was then used to populate MongoDB.



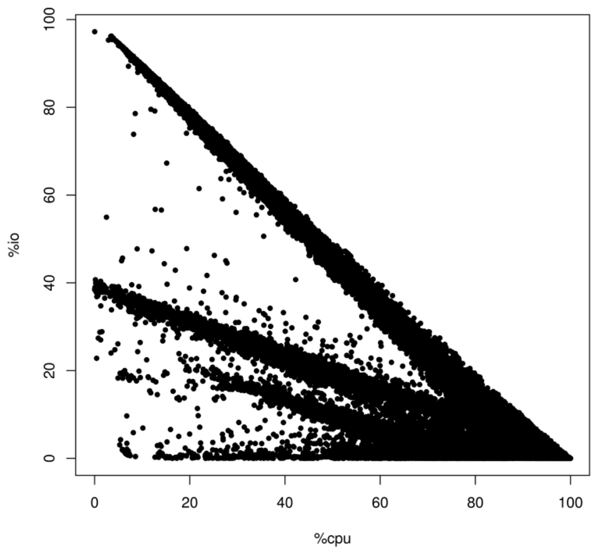
(a) Hosts



(b) Tasks



(c) IO Comm



(d) CPU IO

Figure 9.25: IPM Analysis.

Figure 9.25 shows some preliminary analysis of the data collected on the NERSC Magellan system over a period of seven months, for over 220K runs of several classes of HPC applications. The Magellan batch queue was available to NERSC users; however, it is not clear how the workload on Magellan compares with the general workload on NERSC systems. Figure 9.25a shows the scatter plot of number of nodes used by a job vs duration of the job. Most of the jobs in the period that we profiled used fewer than 256 cores. The total wall clock time used by the job on all processors did tend to vary from a few hundred to a few thousand hours. Figure 9.25b shows a similar trend, but the X-axis shows the number of processes instead of tasks.

Figure 9.25c shows the scatter plot of the jobs percentage of I/O vs percentage of communication. The jobs closer to the origin of the graph (i.e., less communication and less I/O) are likely to do well in cloud environments. The graph also seems to indicate that there are clear clusters of I/O vs communication. Figure 9.25d shows the percentage of CPUs vs percentage of I/O. A number of these applications tend to be CPU intensive, thus clustering in the $> 50\%$ part of the X-axis. Applications that are closer to the X-axis are likely to do better in virtual environments. Additional analysis will need to classify jobs by user and application to understand these patterns better and also understand the volume of I/O and communication.

The IPM data is a rich source of workload data that helps us understand the types of applications that run well on systems such as Magellan.

9.7 Discussion

Virtualized cloud computing environments promise to be useful for scientific applications that need customized software environments. Virtualization is known to have a significant performance impact for scientific applications. We analyzed the performance of a number of different interconnect technologies and I/O performance to understand the performance trade-offs in this space and gain an understanding of what a private cloud configured for scientific computing should look like.

9.7.1 Interconnect

Our benchmarking approach enabled us to understand the impact of the layers in the hierarchy of protocols. Although performance is highly dependent on the type of workload, we see that the performance decrease from virtualization is largely due to overheads in the communication, i.e., the reliance of the virtual machines on TCP over Ethernet as the communication mechanism.

Our results show that while the difference between the interconnects is small at lower core counts, the impact is significant even at the mid-range size of problems of 256 cores to 1024 cores. While the bandwidth is slightly impacted at higher concurrency, the latency takes a much higher hit. Scientific applications tend to run at higher concurrencies to achieve the best time to solution. Thus to serve the needs of these applications, we need good interconnects.

In addition to the network fabric and the protocol, the virtualization software stack imposes an additional overhead to the performance of the applications. This overhead also is impacted by the communication pattern of the application and the concurrency of the run.

The higher bound on the performance on virtual machines today is what is achievable with TCP over Ethernet clusters. Our experiments show that the availability of InfiniBand interconnects on virtual machines would boost the performance of scientific applications by reducing the communication overheads. Scientific applications would be able to leverage the benefits of virtualization and get a significant increase in performance if future environments had InfiniBand available for the communication network.

Our performance evaluation illustrates the importance of a high-bandwidth, low-latency network. However, combining virtual machines and an InfiniBand (IB) network presents several challenges. Possible methods for extending the InfiniBand network inside a virtual machine include routing IP over InfiniBand from the Hypervisor to the virtual machine; bridging Ethernet over InfiniBand through the Linux bridge to the virtual machine; and full virtualization of the NIC. We will briefly discuss these options, including the benefits and trade-offs, and the current issues.

Routing IP packets from the virtual machine (VM) to the hypervisor (HV) and then using IP over InfiniBand is the most readily available approach today. In this approach, the VM communicates through the HV using a virtual network interface. Then standard IP routing is used to route the packets over the InfiniBand interface. This approach provides the lowest performance of the various options. Since the packet must traverse multiple TCP stacks (in the VM and through the HV), the overhead is high. Also there are numerous integration challenges in integrating this with cloud management software like Eucalyptus and OpenStack.

Utilizing Ethernet over InfiniBand (EoIB) is another approach. In EoIB, first proposed by Mellanox, the Ethernet frames are encapsulated in an InfiniBand packet. The InfiniBand stack presents a virtual Ethernet NIC. This virtual NIC would be presented on the HV and then bridged to the VM. This approach has several advantages over the other approaches. VLANs should be supported in this model, which would provide some support for isolating networks. Bridging of the virtual Ethernet interface is similar to how Ethernet NICS are commonly bridged for a virtual machine. This means that packages like Eucalyptus could more easily support it. Since the communication would be TCP-based, some performance would be lost compared with remote memory access offered by native InfiniBand. However, the packets would only go through a single TCP stack (on the VM) on each side. So the performance should exceed the IP over InfiniBand method. While this approach looks promising, the current EoIB implementation does not support Linux bridging. Another disadvantage is that only Mellanox currently plans to support EoIB, and it requires a special bridge device connected to the InfiniBand network. However, the devices are relatively low cost.

Full virtualization of the InfiniBand adapter is just starting to emerge. This method would provide the best performance, since it would support full RDMA. However, it would be difficult to isolate networks and protect the network from actions by the VM. Furthermore, images would have to include and configure the InfiniBand stack. This would dramatically complicate the process of creating and maintaining images. Finally, only the latest adapters from Mellanox are expected to support full virtualization. The version of adapters in Magellan will likely not support this method. We are exploring options for evaluating virtualization of the NIC and looking at alternative methods.

High-performance networks like InfiniBand can dramatically impact the performance, but integrating them into virtual machine-based systems requires further research and development. Alternatively, it may be more attractive to look at how some of the benefits of virtualization can be provided to the users without using true virtualization. For example, our early experiments with MOAB/xCAT show that dynamic imaging of nodes or software overlay methods allow the user to tailor the environment for their application without incurring the cost of virtualization.

9.7.2 I/O on Virtual Machines

The I/O performance results clearly highlight that I/O can be one of the causes of performance bottlenecks on virtualized cloud environments. Performance in VMs is lower than on physical machines, which may be attributed to an additional level of abstraction between the VM and the hardware. Also, it is evident from the results that local disks perform better than block store volumes. Block store volumes can be mounted on multiple instances and used as a shared file system to run MPI programs, but the performance degrades significantly. Thus these may not be suitable for all the MPI applications that compose the HPC workload. Our large-scale tests also capture the variability associated with virtual machine performance on Amazon.

HPC systems typically provide a high-performance scratch file system that is used for reading initial conditions, saving simulation output, or writing checkpoint files. This capability is typically provided by parallel file systems such as GPFS, Lustre, or Panasas. Providing this capability for virtual machine-based systems presents several challenges including security, scalability, and stability.

The security challenges focus around the level of control a user would have within the virtual machine. Most parallel file systems use a host-based trust model, i.e., a server trusts that a client will enforce access rules and permissions. Since the user would likely have root privileges on the VM, they would be capable of viewing and modifying other users' data on the file system.

Accessing shared file systems from user controlled virtual machines can also create scalability and stability challenges. Since each node could potentially run many virtual machine instances, the file system could see a multiplier of file system clients. If each node was running eight small instances (one on each core), then the file system would have to deal with eight times more clients. Furthermore, virtual machines are often terminated instead of performing a clean shutdown. This could lead to clients frequently joining and leaving the file system cluster. File systems like GPFS and Lustre employ timeout and heartbeat methods that assume a relatively stable client pool. Clients randomly disappearing could lead to long hangs while outstanding locks held by terminated instances were timed out.

There are several potential methods to address these issues. One would be to project only the portions of the file system that the user owns into the virtual machine. This could be done using protocols like NFS and re-exporting the parallel file system. This option was explored by ALCF Magellan staff, but ran into challenges with Linux kernel and I/O libraries. Another approach would be to forward the file I/O operations to a proxy server that would have the file systems mounted. The operations would then be performed on the proxy server as the user who owned the virtual machine instance. The standard file system client would enforce the access controls.

There are existing file system modules that use the Linux FUSE file system interface to forward I/O over connections like SSH. The performance over SSH would be poor, but would at least provide access to data. This method was used by the ALCF/LCRC project that built an extension of the Argonne LCRC Fusion within the OpenStack cloud and worked reasonably well. Alternatively, the I/O Forwarding Scalability Layer (IOFSL) project [2] is developing a high-performance I/O forwarding layer that could potentially help. While IOFSL is focused on developing an I/O forwarding system to support ultra-scale HPC systems, the same mechanisms can be used for virtual machines. These solutions would also help mitigate the scaling and stability challenges, since they would reduce the number of real clients handled by the file system and would act as a firewall between the virtual machines and the file system.

9.8 Summary

Applications with minimal communication and I/O perform well in cloud environments. Our evaluation shows that there is a virtualization overhead that is likely to get better with ongoing research. But the primary performance impact for HPC applications comes from the absence of high-bandwidth, low-latency interconnects in current virtual machines. Thus a majority of current DOE HPC applications will not run efficiently in today's cloud environments. Our results show that the difference between interconnects is pronounced even at mid-range concurrencies. Similarly there is an I/O performance issue on virtual machines, and the absence of high-performance file systems further impacts the productivity of running scientific applications within the cloud environment.

Chapter 10

MapReduce Programming Model

The MapReduce programming [13] was developed by Google to address its large scale data processing requirements. Just reading terabytes of data can be overwhelming. Thus, the MapReduce programming model is based on the idea of achieving linear scalability by using clusters of standard commodity computers. Nutch, an open source search project, that was facing similar scalability challenges implemented the ideas described in the MapReduce [13] and the Google File System papers [33]. The larger applicability of using MapReduce and the distributed file systems for other projects was recognized and Hadoop was split off as a separate project from Nutch in 2006.

MapReduce and Hadoop have gained significant traction in the last few years as a means of processing large volumes of data. The MapReduce programming model consists of two functions familiar to functional programming, *map* and *reduce*, that are each applied in parallel to a data block. The inherent support for parallelism built into the programming model enables horizontal scaling and fault-tolerance. The open-source implementation Hadoop is now widely used by global internet companies such as Facebook, LinkedIn, Netflix, etc. Teams have developed components on top of Hadoop resulting in a rich ecosystem for web applications and data-intensive computing.

A number of scientific applications have characteristics in common with MapReduce/Hadoop jobs. A class of scientific applications which employ a high degree of parallelism or need to operate on large volumes of data might benefit from MapReduce and Hadoop. However, it is not apparent that the MapReduce and Hadoop ecosystem is sufficiently flexible to be used for scientific applications without significant development overheads.

Recently, there have been a number of implementations of MapReduce and similar data processing tools [21, 25, 36, 45, 59, 73, 87]. Apache Hadoop was the most popular implementation of MapReduce at the start of the Magellan project and it continuous to gain traction in various communities. It has evolved rapidly as the leading platform and has spawned an entire ecosystem of supporting products. Thus we use Hadoop, as a representative MapReduce implementation for core Magellan efforts in this area. We have also collaborated with other groups to compare different MapReduce implementations (Section 10.6.2). Additionally, Magellan staff were also involved with an alternative implementation of MapReduce (described in Section 10.6.3).

In Sections 10.1, 10.2 and 10.3 we provide an overview of MapReduce, Hadoop and Hadoop Ecosystem. In Section 10.4 we describe our experiences with porting existing applications to Hadoop using the Streaming model. In Section 10.5 we describe our benchmarking effort with Hadoop and Section 10.6 describes other collaborative efforts. We present a discussion of the use of Hadoop for scientific applications in Section 10.7. Additional case studies using Hadoop are described in Chapter 11.

Table 10.1: Comparison of HDFS with a parallel filesystems (GPFS and Lustre) .

	HDFS	GPFS and Lustre
Storage Location	Compute Node	Servers
Access Model	Custom (except with Fuse)	POSIX
Typical Replication	3	1
Typical Stripe Size	64 MB	1 MB
Concurrent Writes	No	Yes
Performance Scales with	No. of Compute Nodes	No. of Servers
Scale of Largest Systems	O(10k) Nodes	O(100) Servers
User/Kernel Space	User	Kernel

10.1 MapReduce

MapReduce is a programming model that enables users to achieve large-scale parallelism when processing and generating large data sets. The MapReduce model has been used to process large amounts of index, log and user behavior data in global internet companies including Google, Yahoo!, Facebook and Twitter. The MapReduce model was designed to be implemented and deployed on very large clusters of commodity machines. It was originally developed, and is still heavily used for massive data analysis tasks. MapReduce jobs were proposed to run on the proprietary Google File System (GFS) distributed file system [33]. GFS supports several features that are specifically suited to MapReduce workloads, such as data locality and data replication.

The basic steps in a MapReduce programming model consists of a *map* function that transforms the input records into intermediate output data, grouped by a user-specified key. The *reduce* function processes intermediate data to generate a final result. Both the map and reduce functions consume and generate key/value pairs. The reduce phase gets a sorted list of key, value pairs from the MapReduce framework. In the MapReduce model all parallel instances of mappers or reducers are the same program; any differentiation in their behavior must be based solely on the inputs they operate on.

The canonical basic example of MapReduce is a word-count algorithm where the map emits the word itself as the key and the number 1 as the value; and the reduce sums the values for each key, thus counting the total number of occurrences for each word.

10.2 Hadoop

Hadoop is an open-source distributed computing platform that implements the MapReduce model. Hadoop consists of two core components: the job management framework that handles the map and reduce tasks and the Hadoop Distributed File System (HDFS) [76]. Hadoop's job management framework is highly reliable and available, using techniques such as replication and automated restart of failed tasks. The framework has optimization for heterogeneous environments and workloads, *e.g.*, speculative (redundant) execution that reduces delays due to stragglers.

HDFS is a highly scalable, fault-tolerant file system modeled after the Google File System. The data locality features of HDFS are used by the Hadoop scheduler to schedule the I/O intensive *map* computations closer to the data. The scalability and reliability characteristics of Hadoop suggest that it could be used as an engine for running scientific ensembles. However, the target application for Hadoop is very loosely coupled analysis of huge data sets. Table 10.1 shows a comparison of HDFS with parallel file systems such as GPFS and Lustre. HDFS fundamentally differs from parallel file systems due to the underlying storage model. HDFS relies on local storage on each node while parallel file systems are typically served from a set of dedicated I/O servers. Most parallel file systems use the POSIX interface that enables applications to be portable across different file systems on various HPC systems. HDFS on the other hand has its own interface requiring applications to be rewritten to leverage HDFS features. FUSE [31] provides a POSIX interface on top of HDFS but with a performance penalty.

10.3 Hadoop Ecosystem

Hadoop is used by a number of global Internet companies and an entire open-source ecosystem of supporting software components has evolved around the core MapReduce and HDFS framework. The native Hadoop program has a Java API allowing map and reduce functions to be written as Java programs. Hadoop Pipes and Hadoop Streaming, add-on-components to Hadoop, enable applications written in C++ and other languages to be plugged in as maps and reduces. Dumbo is a python library for streaming. Additionally, higher-level languages such as Pig, a data-flow language and Jaql, a JSON (Javascript Object Notation) based semi-structured query processing language have evolved.

The Hadoop ecosystem is rapidly growing and we mention some popular components here. HBase and Cassandra provides various database implementations on top of Hadoop. Hive is a data warehouse infrastructure that provides data summarization and ad hoc querying.

Mahout is a machine learning library and Giraph provides large-scale graph processing on Hadoop. Oozie is a workflow tool that allows chaining of multiple MapReduce stages. There are a number of other building-blocks such as ZooKeeper, a high-performance coordination service that provides semantics for concurrent access. Chukwa and Scribe are data collection and general log aggregation frameworks. Hadoop is also available as a service on Amazon's EC2 and is called Elastic MapReduce.

The evaluation of the MapReduce programming model as part of the Magellan project focuses on the suitability of the core components of Hadoop, i.e., MapReduce framework and HDFS. However, in the context of collaboration with various scientific groups, we used a number of other components from the Hadoop Ecosystem including Pig, HBase and Elastic MapReduce.

10.4 Hadoop Streaming Experiences

The Hadoop streaming model allows one to create map-and-reduce jobs with any executable or script as the mapper and/or the reducer. This is the most suitable model for scientific applications that have years of code in place capturing complex scientific processes and hence was the focus of our work with scientific applications.

10.4.1 Hadoop Templates

The Hadoop streaming model enables applications to easily plug-in existing applications. However Hadoop has a number of parameters that need to be configured which can be tedious when starting with new applications. Magellan staff developed a simple template where an application can specify a set of parameters and the application can easily be run through the streaming model. This greatly simplifies things and enables users to plug-in applications into Hadoop with minimal work. The user can specify the paths to the binaries/scripts for the mappers and reducers and the input and output directories. The number of mappers in Hadoop is typically controlled by the size of the input blocks. For scientific applications, we designed the template to have one map per input file matching with the model of many current high-throughput and data-intensive applications. The user can control how many reduce tasks they would desire for their application. In addition, we allow scientists to use Hadoop as a job management framework while using high performance file systems such as GPFS that are already available at HPC centers and may already store legacy data sets. The parameters are shown below.

```
my_mapper=[path to binary/script for the mapper]
my_reducer=[path to binary/script for the reducer]
input_dir=[input directory]
output_dir=[output directory]
num_reduce_tasks=[specify number of reduce tasks]
file_system=[Options right now are hdfs and gpfs]
```

10.4.2 Application Examples

Magellan staff worked with some key scientific groups to port their applications to the streaming model. We describe some of these in this section. Additional case studies are presented in Chapter 11.

BLAST. JGI's IMG pipeline has been tested in the Hadoop framework to manage a set of parallel BLAST computations. The performance across an HPC machine, virtual machines, and a Hadoop cluster was found to be comparable (within 10%), making this a feasible application for cloud environments. The experience with running the IMG pipeline on virtual machines is described in greater detail in Chapter 11. Here we describe the challenges with running this in the Hadoop framework. BLAST takes a multi-line sequence input for each run. By default, MapReduce assumes that inputs are line-oriented and each line can be processed independently. One could define and write Java classes to handle other input formats. However there are difficulties with managing API versions and jar files that can make the process cumbersome. We experimented with a wrapper script that formatted the input to be line oriented when passed to the Hadoop framework and the input was reformatted in the streaming mapper script to be compatible with application needs. This highlights the need for more custom plugins in Hadoop for science applications.

Tropical storm detection code. Another application that we plugged into Hadoop was TSTORMS. It is a single-threaded analysis program to find tropical storms in climate data. TSTORMS input is a NetCDF file that has a binary format. While Hadoop and HDFS can handle binary files, current versions of Hadoop streaming can't handle binary input and output. NetCDF libraries provide utilities to manage an ASCII version of NetCDF files and to convert between the ASCII and binary versions. While not the most efficient way, we explored pre-converting to an ASCII format and loading the converted files into HDFS. The streaming mapper function then converts it back to a binary format before running the application. In addition, TSTORMS does not take inputs on standard input and thus the file needs to be streamed to disk and the input file path is then passed to the binary. The application enjoyed the benefits of scalability and fault-tolerance in the Hadoop framework but it highlights the difficulties of working with scientific data formats in this environment.

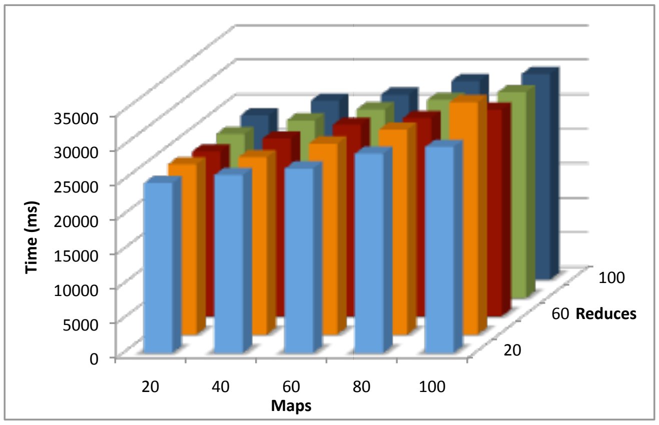
Atmospheric River Detection. The atmospheric river detection code, similar to TSTORMS works on NetCDF input files. The challenges presented for TSTORMS also holds for this application. In addition, this application takes a parameter that defines the day to look at in the NetCDF file. Thus, each map should be differentiated based on a combination of file and a parameter. This can't be handled easily in the current Hadoop implementation where a mapper is differentiated by the file or data it operates on. Additionally, we experimented with a wrapper script, that sequentially process all days in a file. Additional parallelism could be achieved for such applications if Hadoop would allow maps to be differentiated by not just a file but by additional parameters.

10.5 Benchmarking

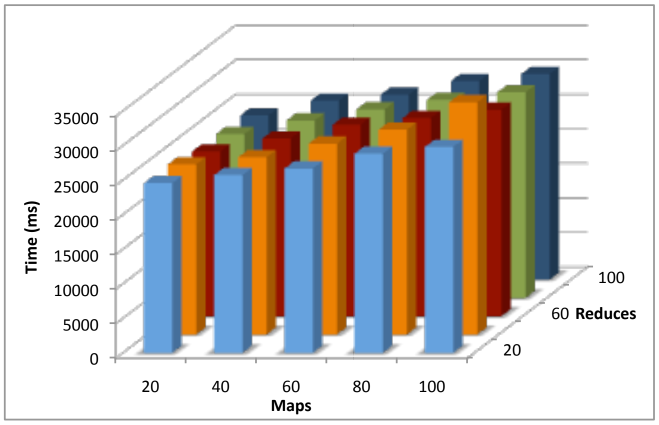
In addition to understanding the programming effort required for applications to use the Hadoop framework, we conducted some benchmarking to understand the effects of various Hadoop parameters, underlying file systems, network interface etc.

10.5.1 Standard Hadoop Benchmarks

First, we used the standard Hadoop benchmark tests on the NERSC Magellan Hadoop cluster (described in Chapter 5). The benchmarking data enables us to understand the effect of various parameters and system configurations.



(a) 100 lines



(b) 1000 lines

Figure 10.1: Hadoop MRBench.

MRBench evaluates the performance of MapReduce systems while varying key parameters such as data size and the number of Map/Reduce tasks. We varied the number of lines of data written from 100 to 1000 and varied the number of maps and reduces. Figure 10.1 shows the time with varying maps and reduces for a) 100 and b) 1000 lines. As the number of maps and reduces increases the time increases; however the difference is less than 10 seconds. The number of lines written at the orders of magnitude we measure shows no perceptible effect. MRBench can be provided with custom mapper and reducer implementations to measure specific system or application behavior. This could be used to develop benchmarks that emphasize the nature of scientific workloads.

TestDFSIO measures the I/O performance of HDFS. Figure 10.2 shows the throughput for small and large file sizes with varying concurrent writers/files. For small file sizes, the throughput remains fairly constant with varying number of concurrent writers. However, the throughput decreases rapidly as the number of concurrent files/writers increases. This seems to be dependent on the HDFS block size and will require tuning and additional benchmarking to understand the configuration necessary for scientific workloads at a specific site.

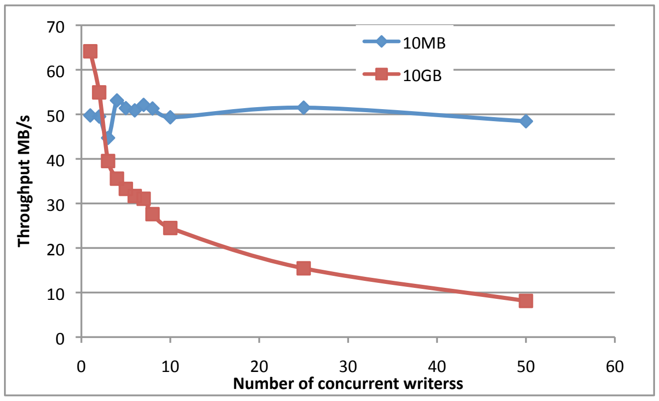


Figure 10.2: HDFS throughput for different file sizes with varying number of concurrent writers.

TeraSort is a standard map/reduce application for Hadoop that was used in the terabyte sort competition. TeraGen generates the data, and TeraSort then samples the input data and uses map/reduce to sort the data in total order. Figures 10.3 and 10.4 show the comparison of using HDFS and GPFS as the underlying file system for TeraGen with varying number of maps. HDFS and GPFS have been designed for largely different usage scenarios and the goal of our comparison is not a quantitative performance comparison of the two system. Each of these file systems has its own strengths for certain workloads. Our goal here is to understand if scientific applications can benefit from Hadoop's job management framework while using POSIX compliant file systems available in HPC centers.

Figure 10.3 shows the time for TeraGen to generate 1 TB of data in Hadoop on both file systems. We see that the performance of GPFS shows a slight decrease in performance as the number of concurrent maps is increased. On the other hand, HDFS's performance significantly improves as number of maps increases as HDFS is able to leverage the additional bandwidth available from disks in every compute node. Figure 10.4 shows the effective bandwidth for both systems and we can see that HDFS's effective BW is steadily increasing. Our GPFS system is used in production use and the variability seen here is from other production workloads using the system. Hadoop and HDFS have been designed to handle high-levels of parallelism for data-parallel applications. These results show that for small to medium scale parallelism, applications can use Hadoop with GPFS without any loss in performance.

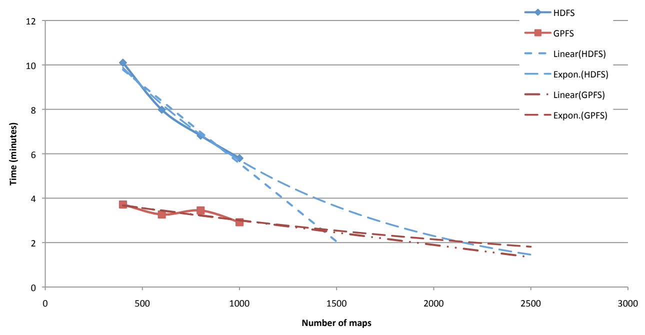


Figure 10.3: HDFS and GPFS Comparison (Time)

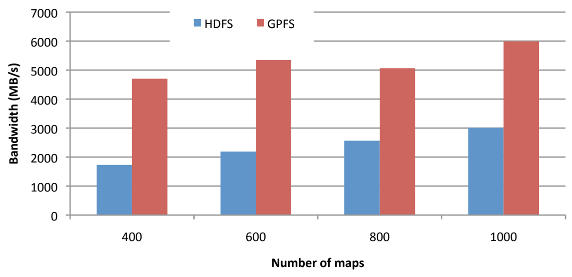
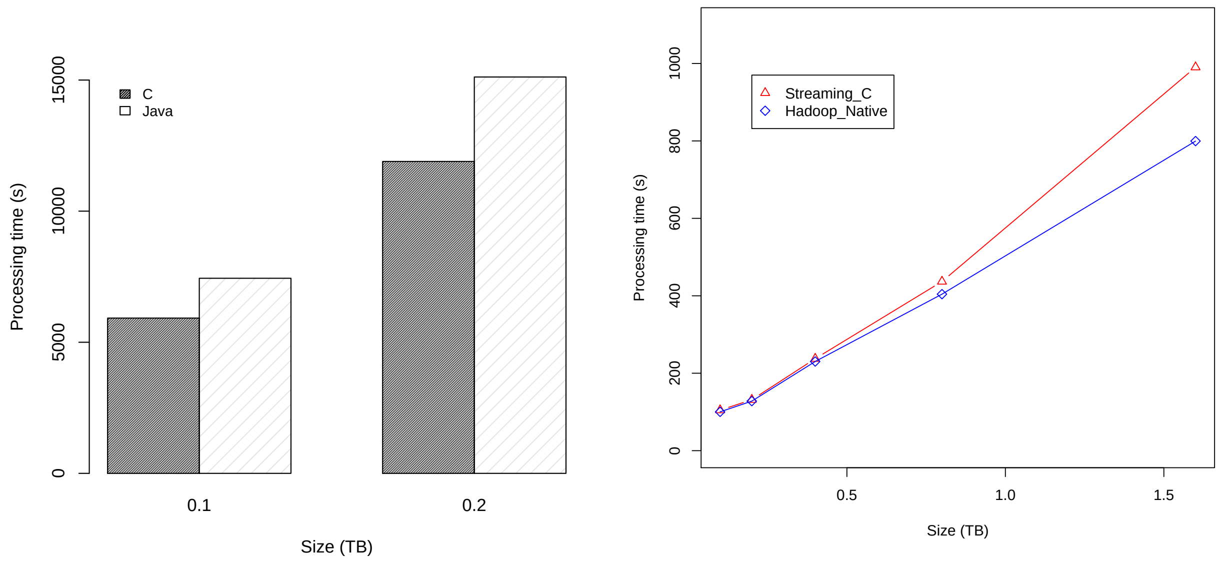


Figure 10.4: HDFS and GPFS Comparison (Bandwidth)

10.5.2 Data Intensive Benchmarks

The following work was funded through a project evaluating Hadoop specifically for data intensive scientific applications. Magellan staff supervised the student who conducted these experiments and all experiments were run on Magellan Hadoop testbed. A paper describing the results in detail is in preparation. Here we *summarize some key results that complement the Magellan evaluation*

Scientists are struggling with a deluge of data in various disciplines. Emerging sensor networks, more capable instruments, and ever increasing simulation scales are generating data at a rate that exceeds our ability to effectively manage, curate, analyze, and share it. Data-intensive Computing is expected to revolutionize the entire hardware and software stack. Hadoop provides a way for "Big data" to be seamlessly processed through the MapReduce model. The inherent parallelization, synchronization and fault-tolerance make it ideal for highly-parallel data intensive applications. Thus it is important to evaluate Hadoop specifically for data-intensive workloads to understand the various trade-offs. In this study, we specifically evaluate Hadoop for various data operations and understand the impact of the file system, network and programming model on performance.



(a) Streaming Language Differences

(b) Streaming Comparison

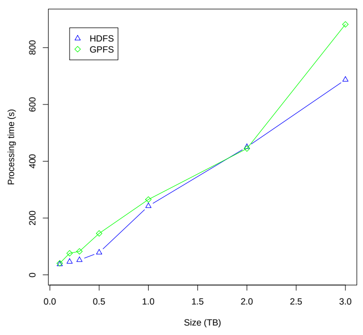
Figure 10.5: Streaming.

Experiment Setup

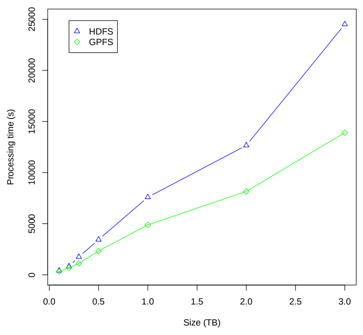
All experiments were conducted on the Magellan NERSC Hadoop testbed described in Chapter 5. The three common data operations in scientific applications were identified as:

- **Filter.** A filter operation is when data is analyzed and the output result is a subset of the entire data set. Scientific applications that involve searching for a certain pattern e.g., finding a tropical storm in a region would fit this kind of data operation. The volume of input data processed is significantly larger than the volume of output data.

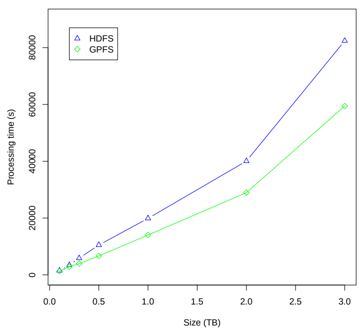
- **Reorder.** A reorder operation is when the input is reordered in some way resulting in an output dataset. Sorting the input data is an example of this kind of operation. Reorder results in an output dataset that is identical to the input data size.



(a) Filter

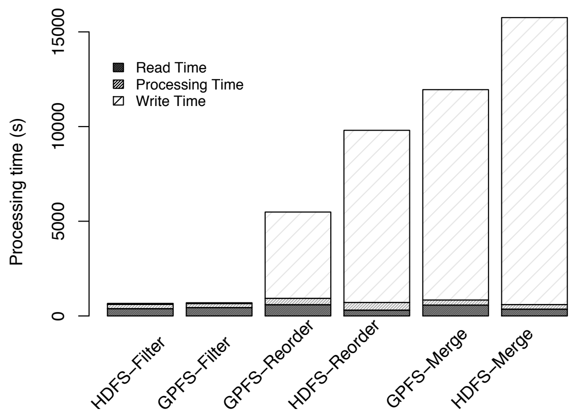


(b) Reorder



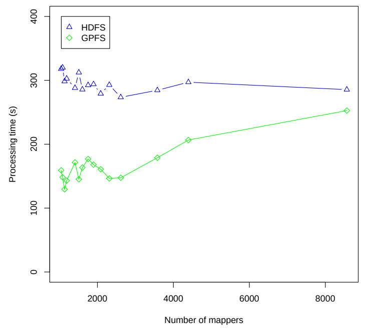
(c) Merge

Figure 10.6: File System Comparison with varying file sizes.



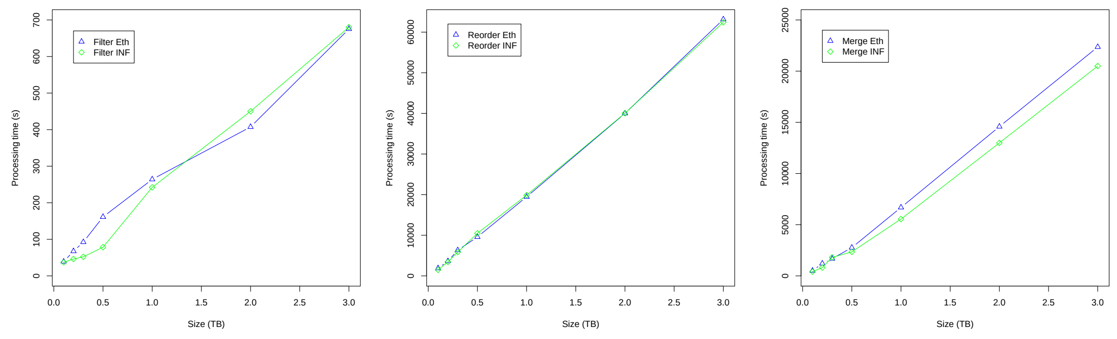
(a)

Figure 10.7: File System Comparison



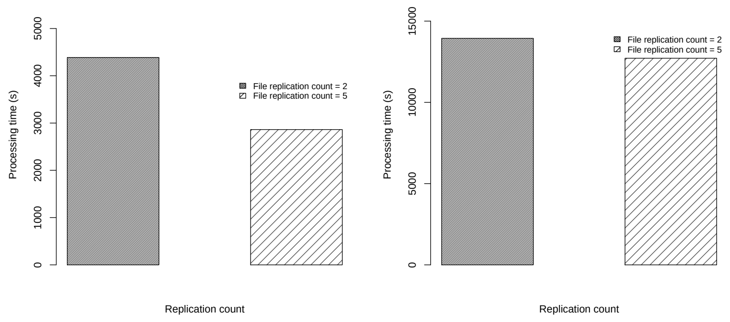
(a)

Figure 10.8: Effect of changing the number of maps for a Filter operation.



(a) Filter (b) Reorder (c) Merge

Figure 10.9: Effects of network with varying file size. *Eth corresponds to a 1 Gb Ethernet network.* Inf corresponds to an QDR 4x Infiniband network.



(a) Filter (b) Reorder

Figure 10.10: Effects of replication.

- **Merge.** Some scientific application results in a merge of two or more data sources. For example, observation data might need to be merged with simulation data. Such a merge operation results in an output data volume that is significantly larger than the input data size.

We developed a synthetic benchmark suite to represent each of these data operations. We used the Wikipedia data that is 6TB for our analysis. We use a subset of this data for some experiments as noted below. For a filter operation, we constructed an index map of all the titles of the wikipedia pages and their corresponding files. For the reorder operation, we perform a data conversion where we convert <timestamp> and </timestamp> to <date+time> to </date+time>. For the merge operation, we create an index of all lines of the wikipedia data.

Result

We summarize the results from our experiments understanding the effects of streaming, file systems, network and replication.

Streaming. Existing scientific applications can benefit from the MapReduce framework using the streaming model. Thus for our first experiment, we constructed the filter operation in both C and Java (using the Hadoop APIs) and compared the two to understand the overheads of streaming. Comparing the overheads of streaming are tricky since there is some differences in timing induced from just language choices. Hadoop streaming also does not support Java programs since Java programs can directly use the Hadoop API. Figure 10.5a shows the comparison of the timings of both programs running on a single node. We see that the C implementation is more efficient and also the performance improves as the data size increases. Figure 10.5b shows the comparison of the streaming C version with the native Hadoop version for varying file sizes. We notice that the performance through Hadoop is similar for smaller file sizes. This actually indicates that the streaming has additional overhead since the C version is more efficient as such. As the file size increases we see that the overhead for streaming increases.

Effect of file system. Figure 10.6 shows the comparison of performance of the data operations on both GPFS and HDFS. For the filter operation, there is negligible difference between HDFS and GPFS performance untill 2TB. However at 3TB HDFS performs significantly better than GPFS. For the reorder and merge, GPFS seems to achieve better performance than HDFS and the gap increases with increasing file sizes.

Figure 10.7 shows the comparison of HDFS and GPFS for all three data operations for a 2TB data set. The figure shows the split of the processing time. The difference between HDFS and GPFS is negligible for the filter operation. We notice that for the reorder and merge, GPFS seems to achieve better performance than HDFS overall at this data size. However, HDFS performs better than GPFS at reads whereas GPFS performs better than HDFS for the write part of the application at the given concurrency.

Our earlier results compared the performance for TeraGen on HDFS and GPFS with varying number of maps. TeraGen is a write-intensive operation. Figure 10.8 shows the effect of varying number of mappers on processing time for the filter operation on both HDFS and GPFS. We see that with increasing maps the performance of GPFS goes down significantly. The clearly noticeable performance variations are likely due to artifacts of Hadoop's scheduling.

Thus, HDFS seems to achieve better read performance than GPFS and better write performance at higher concurrencies.

Effect of network. Scientific applications in HPC centers traditionally use high performance, low-latency networks. However Hadoop has traditionally been run on commodity clusters based on Ethernet networks. The shuffle phase between the map and reduce phase is considered to be the most network intensive operation since all the keys are sorted and data belonging to a single key is sent to the same reducer resulting in large data movement across the network. Figure 10.9 shows the comparison of the network on various data operations with varying file sizes. We observe that filter and reorder are not affected as much by the changes in the network. The merge operation shows that the application performs better on the Infiniband network at larger file sizes. This is likely due to the growth in the data in the merge compared with the reorder or filter.

Replication. Replication in Hadoop is used for fault-tolerance as well as increasing the effect of data locality of the input files. Figure 10.10 shows the effects of varying replication on the filter and reorder data operations with 2TB data set. We see that for the read-intensive filter operation increasing the replication factor significantly impacts performance. While the reorder operation benefits from the replication, the effects are minimal since the operation is dominated by the write costs.

10.5.3 Summary

Our benchmarking for data-intensive applications in Hadoop reveals a number of key factors. The performance an application can achieve in Hadoop is largely workload dependent. Smaller concurrencies applications can benefit from the job management framework in Hadoop without being constrained by the non-POSIX compliant HDFS. The network interface seems to have minimal impact on the performance though it is likely that the Shuffle algorithm needs to be modified to fully exploit the high-speed network. Mellanox recently announced an unstructured data accelerator (UDA) software plugin that will allow Hadoop frameworks to leverage RDMA (Remote Direct Memory Access) [60]. Similarly we see that some applications can greatly benefit from the data locality and replication aspects of Hadoop.

10.6 Other Related Efforts

Magellan staff collaborated with other groups and Magellan resources were used in a number of other efforts evaluating MapReduce and Hadoop for scientific applications. We summarize some of these efforts in this section.

10.6.1 Hadoop for Scientific Ensembles

This work was performed by a graduate summer intern funded through the Diversity Program at Lawrence Berkeley National Lab. The student was co-supervised by Magellan staff. More details are available in the published paper [14].

Scientific ensembles have many characteristics in common with MapReduce workloads, as they both employ a high degree of parallelism that must be run in coordination to arrive at a meaningful result. However these scientific workloads have key distinguishing characteristics: multiple files per task and specific parameters per task. Thus, we must investigate further to see whether MapReduce and its open-source implementations are a convenient programming model to manage loosely coupled asynchronous scientific ensembles.

In this study, we select a set of common scientific ensemble patterns and implement the patterns with Hadoop, the open-source MapReduce implementation.

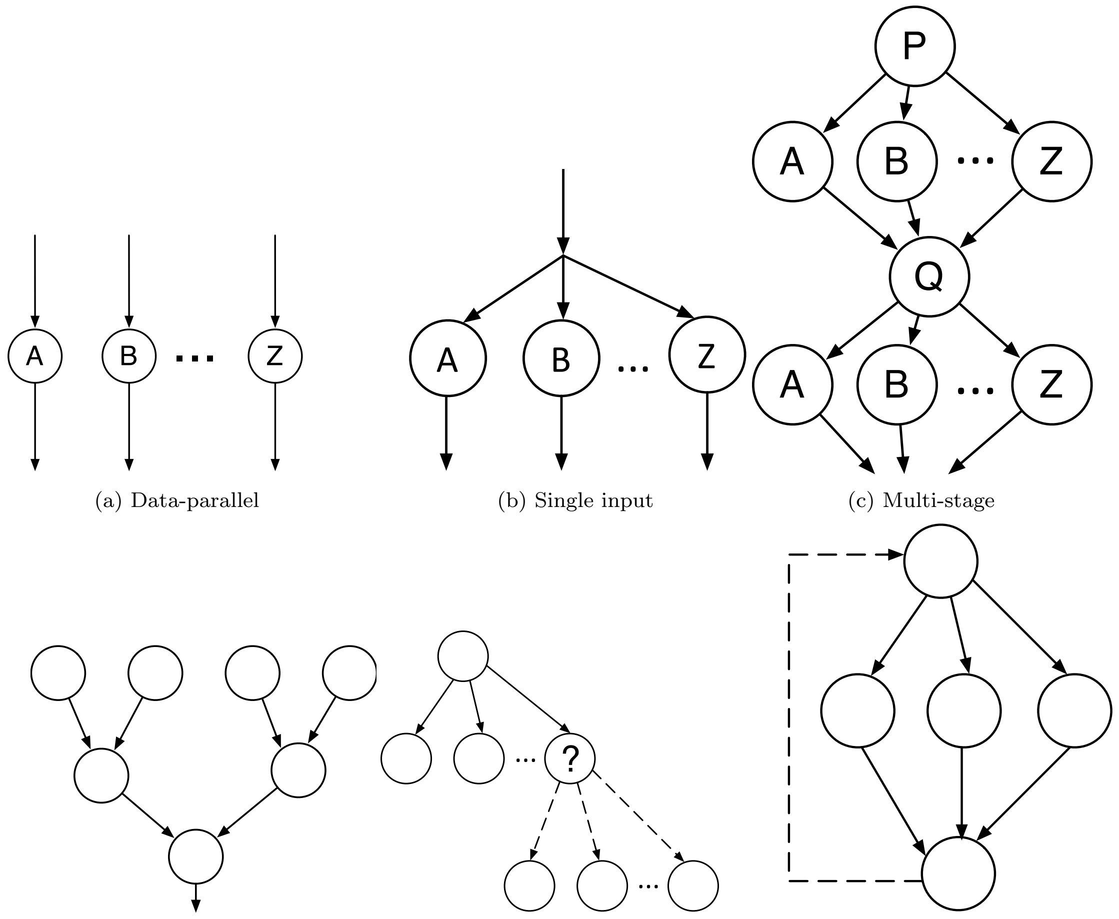
Hadoop Jobs and Scientific Ensembles

Scientific ensembles and Hadoop jobs have a number of similarities in their characteristics and requirements:

- Hadoop jobs consist of two primary phases – *map* and *reduce*. The jobs consist of a large number of maps that performs data transformation and one or more reduces that performs a combine operation to produce the final result. Scientific ensembles might have many execution phases but they can be roughly categorized as *data setup or problem decomposition, data transformation and data aggregation.* The problem decomposition is implicit in vanilla Hadoop jobs where the input is divided into blocks for the workers to operate on. The data transformation and data aggregation phases are similar to the map and reduce phases of a Hadoop job.

- Both scientific Hadoop jobs and scientific ensembles have a similar structure – large number of parallel tasks in the "map" phase and a smaller number of tasks in the "reduce" phase.
- Both scientific ensembles and Hadoop jobs exhibit data-flow parallelism i.e., they operate on either different data sets or different parameters on the same data sets.
- Scientific ensembles and Hadoop jobs both require the ability to scale up easily as more data or additional parameters need to be analyzed.
- The large-scale parallelism makes fault-tolerance and data proximity critical for both applications.

However there are a number of differences between Hadoop workloads and typical scientific ensembles. First, Hadoop jobs consists of a map phase followed by a reduce phase whereas scientific ensembles might have a diverse set of tasks followed by a variable number of stages with additional tasks. Second, Hadoop assumes that each mapper works on a subset of the data and there is no affinity between map tasks and data blocks. In contrast, scientific applications often operate on files rather than blocks of data. Scientific applications may also require diverse codes to run as map tasks and/or may take additional parameters that might change the processing in the map tasks appropriately.



(a) Data-parallel (b) Single input (c) Multi-stage

(d) Inverse tree (e) Dynamic (f) Iteration

Figure 10.11: Workflow Patterns

Workload Patterns Analysis

A schematic of each pattern is shown in Figure 10.11. We consider a data-parallel ensemble pattern that is closest to the default MapReduce pattern with just a map phase and zero reduces(Figure 10.11a). The single input (Figure 10.11b) ensemble is representative of a scientific ensemble where a single input file might trigger multiple simulation or analysis in parallel with different parameters.The multi-stage pattern (Figure 10.11c) has three phases in stage one (a problem decomposition, computation or data transformation and a data aggregation phase) and the final stage triggers additional computation and data aggregation steps. The inverse tree pattern (Figure 10.11d) is selected to represent a mult-stage data aggregation. The dynamic (Figure 10.11e) and iteration patterns (Figure 10.11f) capture the run-time variability that might result in change in the workflow structure. These workflow patterns also exhibit diverse characteristics in terms of number of stages, number of initial inputs, number of outputs and hierarchy depth.

Detailed results are available in the paper [14]. We summarize the implementation challenges using a three-point difficulty scale. We rate the Hadoop Implementation and Data Management categories as `easy`, `moderately difficult`, or `very difficult`. Similarly, we rate the MapReduce Realization and Performance/Reliability categories as `minimal`, `moderate`, or `significant` impact. While our criteria for comparison is largely qualitative, we summarize the suitability of Apache Hadoop/MapReduce by assigning scores of 0, 0.5, 1 (higher score indicating more difficult or more impact) to each of the categories for better understanding of the relative difficulty of each of the categories and patterns. We intentionally use a discrete scoring scheme rather than continuous scoring since our intention is to understand the difficulty of managing scientific ensembles in Hadoop. A quantitative performance analysis is outside the scope of this discussion.

We could implement all patterns within Apache Hadoop but some of the patterns required significant programming or wrappers for manipulation. These manual mechanisms can be difficult for application scientists and also tend to be error-prone. The dynamic and iteration patterns present the most challenges for implementation in Hadoop. The inverse tree present some challenges while the data-parallel and single input are the easiest to implement with Apache Hadoop.

Our analysis shows that it was possible to implement the general scientific ensembles patterns in Apache Hadoop with varying degrees of difficulty. However, in the case of some patterns we required a significant amount of custom code making it difficult for scientists to use Hadoop without significant programming expertise. Thus, there are a number of gaps and challenges when supporting scientific ensembles through current MapReduce implementations. Our analysis shows that there are opportunities to generalize some of the features required by applications into the programming model and execution framework. We summarize them here:

- Our initial analysis shows the formalizations of these workflow patterns in the MapReduce programming model. There is further research needed into programming model abstractions or extensions to MapReduce that can effectively support these workflow models while providing the ease of programming and scaling that MapReduce provides. Support for task diversity, parameters, multiple inputs will need to be considered.
- Data locality is increasingly becoming important for scientific applications as data volumes increase due to advances in computing hardware and sensor technologies. However new advances are needed to handle data locality of multiple files and patterns such as dynamic and iterative.
- There is additional support needed in execution frameworks that can handle dynamic tasks and iterations.
- Additionally, MapReduce implementations that are conducive to handling scientific codes and languages will be needed in the near future.

10.6.2 Comparison of MapReduce Implementations

This work was conducted at SUNY Binghampton and used Magellan resources for experiences. In addition Magellan staff participated in the study. The paper was published in Grid 2011 [24]

In this study, we compare and study the performance of Hadoop[4], Twister[22] and LEMO-MR [26] in various real-world application usage scenarios, including data-intensive, iterative, CPU-intensive, and memory-intensive loads. We not only compare the chosen frameworks in several real-world application scenarios, but also in real-world cluster scenarios, including fault-prone clusters/applications, physically heterogeneous clusters and load-imbalanced clusters. This performance study provides insight into the relative strengths and weaknesses of different implementations under different usage scenarios. We link the observed behavior of the frameworks to the design and implementation choices of the particular MapReduce platform being tested. Our test framework will be made available to the community for researchers to compare frameworks for their custom usage scenarios and data sizes of interest.

The study identified key applications and cluster categories pertinent to MapReduce performance, i.e., data-intensive, CPU intensive, memory intensive, cluster heterogeneity, load-balancing, iterative applications, fault-tolerance tests. The paper [24] discusses the results in greater detail, here we summarize the key points.

- Twister is efficient for jobs requiring significant *map* side data processing and very limited *reduce* side data processing, such as CPU and memory-intensive jobs. For our test infrastructure, Twister was better than Apache Hadoop when data transfer between the Mapper and Reducer was less than 93 MB, because it uses memory based transfers as opposed to the file based transfers used by Hadoop. The 93 MB threshold can be increased slightly by allocating more memory to the worker processes, but is still limited by the maximum memory footprint of a Java process on a given node.

- Hadoop is designed to make use of the maximum disk space available to each node and hence is known to scale seamlessly for very large data sizes. LEMO-MR uses a hybrid approach starting with a memory-based transfer scheme between Map and Reduce, then progressively changing to the Hadoop model for larger data sizes.

- Twister and LEMO-MR are optimized for CPU-intensive Map and Reduce jobs compared to Hadoop, which spends extra CPU cycles on high overhead prone fault tolerance related tasks. For the matrix multiplication experiment, the CPU utilization for Twister and LEMO-MR was 93% while it was 89% for Hadoop.

- As the processing power is increased for a given application, Twister's performance improvement is best, closely followed by LEMO-MR. Hadoop has the highest overhead per node, and as such, when nodes were increased from 8 to 64 nodes, Twister's speedup improvement was by a factor of 7.5, LEMO-MR's was 7, and Hadoop's was by a factor of 4.

- LEMO-MR has the least memory footprint allowing for 520MB of data to be processed per node, closely followed by Twister which allowed 500 MB of data to be loaded into memory for processing, per node. Hadoop has the highest memory footprint and it allowed only 100 MB of data to be processed per node, before throwing an *out of memory* error.

- The uniformity of input chunk to each node renders Hadoop unable to make efficient use of a heterogeneous cluster. It is unable to determine what "slow" means for tasks mapped to nodes with different memory and processor configurations, and as a result load transfer operations occur excessively and inefficiently.

- In a homogeneous cluster, Hadoop's load management significantly outperforms both LEMO-MR and Twister. When random nodes experience stress, induced by our tests, Hadoop benefits from its speculative execution model which duplicates tasks and is not affected by the slow nodes.

- For applications requiring multiple iterations, Twister proved to be best. Apart from its ease in running iterative applications, compared to Hadoop and LEMO-MR, Twister also offers significant performance advantages to such applications. This is illustrated with Twister being from 2 up to 5 times faster than Hadoop and LEMO-MR.

This benchmarking work highlights the strengths and gaps in each of the frameworks. Additional application benchmarking will be needed to further understand and improve these frameworks for scientific use. Additionally maintaining a benchmark suite aimed at scientific application usage and tracking the performance data for new MapReduce implementations as they become available is important.

10.6.3 MARIANE

MARIANE, an alternative MapReduce implementation was developed at SUNY Binghampton and used Magellan resources for experiences. In addition Magellan staff participated in the study. The paper was published in Grid 2011 [25]

The MapReduce model in its most popular form, Hadoop [4], uses the Hadoop Distributed File System (HDFS) [76] to serve as the Input/Output manager and fault-tolerance support system for the framework. The use of Hadoop, and of the HDFS is however not directly compatible with HPC environments. This is because Hadoop implicitly assumes dedicated resources with non-shared disks attached. Hadoop has successfully worked at large scale for a myriad of applications , however it is not suitable for scientific (legacy) applications that rely on a POSIX compliant file systems in the grid/cloud/HPC setting. HDFS is not POSIX compliant. In this study, we investigate the use of Global Parallel File System (GPFS), Network File System (NFS) for large scale data support in a MapReduce context through our implementation, MARIANE.

For this purpose we picked three application groups, two of them, UrlRank and “Distributed Grep”, from the Hadoop repository, and a third of our own design: an XML parsing of arrays of double, tested here under induced node failures, for fault-tolerance testing purposes. The first two are provided with the Hadoop application package. Even as we limit our evaluation to NFS and GPFS, our proposed design is compatible with a wide set of parallel and shared-block file systems. We present the diverse design implications for a successful MapReduce framework in HPC contexts, and collected performance data from the evaluation of this approach to MapReduce along side Apache Hadoop at the NERSC Magellan cluster.

Design considerations in MARIANE

The design of MARIANE rests upon of three principal modules, representing the tenets of the MapReduce model. Input/Output management and distribution rests within the `Splitter`. Concurrency management in the role of `TaskController`, while fault-tolerance dwells with the `FaultTracker`. **Figure 10.12** shows the design used for MARIANE.

Input Management

While Hadoop and most MapReduce applications apportion the input amongst participating nodes, then transfer each chunks to their destinations, MARIANE relies on the inherent shared file system it sits on top for this feat. The framework leverages the data visibility offered by shared-disk file systems to cluster nodes. This feature exonerates MARIANE from operating data transfers, as such a task is built-in and optimized within the underlying distributed file system. Input management and split distribution are thus not performed on top of an existing file system (FS), but rather with the complicity of the latter. This absolves the application from the responsibility of low-level file management and with it, from the overhead of efficiently communicating with the FS through additional system layers. Furthermore, MARIANE in doing so, benefits not only from file system and data transfer optimizations provided by evolving shared-disk file system technology, but can solely focus on *mapping* and *reducing*, rather than data management at a lower level.

Input Distribution

Input distribution is directly operated through the shared file system. As the input is deposited by the user, the FS is optimized to perform caching and pre-fetching to make the data visible to all nodes on-

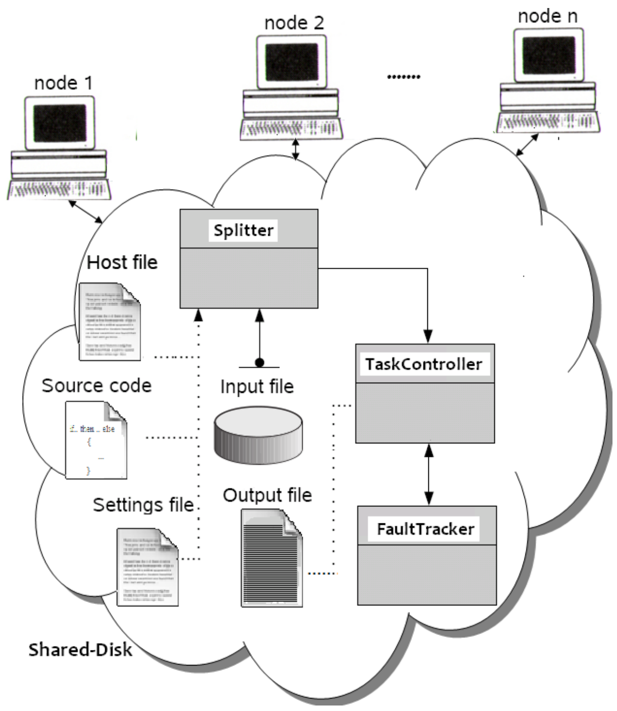


Figure 10.12: Architecture used for MARIANE

demand. This frees the MapReduce framework from accounting, and transferring input to the diverse nodes. Another benefit of shared-disk file systems with MapReduce, one which became apparent as the application was implemented, is the following: current MapReduce implementations, because of their tightly coupled input storage model to their framework require cluster re-configuration upon cluster size increase and decrease. This does not allow for an elastic cluster approach such as displayed by the Amazon EC2 cloud computing framework[20], or Microsoft's Azure [83]. Although cloud computing is conceptually separate from MapReduce, we believe that the adoption of some of its features, more specifically "elasticity", can positively benefit the turn around time of MapReduce applications. With the input medium isolated from the processing nodes, as MARIANE features, more nodes can be instantaneously added onto the cluster without incurring additional costs for data redistribution or cluster re-balancing. Such operations can be very time consuming, and only allow for the job to be started at their completion, when all data settles on the cluster nodes involved [4]. With MARIANE, input storage is independent from the diverse processing nodes. Separating the I/O structure from the nodes allows for a swift reconfiguration and a faster application turnaround time. In Hadoop's case, removing a node holding a crucial input chunk means finding a node holding a duplicate of the chunk held by the exiting node and copying it to the arriving node, or just re-balancing the cluster, as to redistribute the data evenly across all nodes. Such an operation with large scale input datasets can be time consuming. Instead, according to the number of participating workers in the cluster, nodes can be assigned file markers as to what part of the input to process. Should a node drop or be replaced, the arriving machine simply inherits its file markers, in the form of simple programming variables. This mechanism, as our performance evaluation will show, also makes for an efficient and light-weight fault-tolerant framework.

Task Tracker and Task Control

The task tracker, also known as "master" makes the *map* and *reduce* code written by the user, available to all participating nodes through the shared file system. This results on the application level to a one time instruction dispatch, rather than *map* and *reduce* instructions streaming to as many participating nodes as there are in the cluster. Upon launch, the nodes designated as mappers also subsequently use the *map* function, while those designated as reducers, use the *reduce* function. The task tracker monitors task progress from the cluster nodes, and records broken pipes and non-responsive nodes as failed. A completion list of the different sub-tasks performed by the nodes is kept in the master's data structure. Upon failure, the completion list is communicated to the `FaultTracker`. Slow nodes are similarly accounted for, and their work is re-assigned to completed and available machines. In a redundant situation caused by two nodes running similar jobs, in the case perhaps of a slow node's job being rescheduled, the system registers whichever sub-job completes first. This particular scheme is akin to Hadoop's *straggler* suppressing mechanism, and serves as a load balancing maneuver.

Fault Tolerance

While Hadoop uses task and input chunk replication to fault-tolerance ends, we opted for a node specific fault-tolerance mechanism, rather than an input specific one. With this approach, node failure does not impact data availability, and new nodes can be assigned failed work with no need for expensive data relocation. Upon failures, we elected for an exception handler to notify the master before terminating, or in the case of sudden death of one of the workers, the rupture of a communication pipe. Furthermore, the master receives the completion status of the *map* and *reduce* functions from all its workers. Should a worker fail, the master receives notification of the event through a return code, or a broken pipe signal upon sudden death of the worker. The master then updates its node availability and job completion data structures to indicate that a job was not completed, and that a node has failed. We later evaluate this low overhead fault-tolerant component along with Hadoop's data replication and node heartbeat detection capability, and assess job completion times in the face of node failures.

The Fault-Tracker consults the task completion table provided by the master node, and reassigns failed tasks to completed nodes. Unlike Hadoop, the reassignment does not include locating relevant input chunks to the task and copying them, if those chunks are not local to the rescuing node. The task reassignment procedure rather provides file markers to the rescuing node so it can process from the input, the section assigned to the failed node. As the input is visible to all nodes, this is done without the need to transfer huge data amounts, should massive failures occur. The `FaultTracker`'s operation is recursive; should a rescuing node fail, the rescuer is itself added to the completion table and the module runs until all tasks are completed, or until all existing nodes die or fail. In the interim, dead nodes are pinged as a way to account for their possible return and inscribe them as available again. Hadoop uses task replication regardless of failure occurrence. It also does not keep track of nodes that might have resurfaced after suddenly failing. MARIANE for its part only uses task replication in the case of slow performing nodes, when the sloth is detected, and not before. To this effect, whichever version of the replicated sub-job completes first is accepted.

Summary

MARIANE is a MapReduce implementation adapted and designed to work with existing and popular distributed and parallel file systems. Detailed performance results are available in the paper [25]. With this framework, we eliminate the need for an additional file system and along with it, the involvement of MapReduce in expensive file system maintenance operations. By doing so, we show a significant increase in performance and decrease in application overhead along with a dramatic speed up in fault tolerance response as we compare our results to Apache Hadoop.

10.7 Discussion

We have evaluated MapReduce, specifically Hadoop for different set of applications as well as under different system configurations. In this section we summarize our findings.

10.7.1 Deployment Challenges

Hadoop 0.20 runs all jobs as user *hadoop*. This results in a situation where users may not be able to access non-HDFS files generated by the job. Thus, the permissions need to be fixed after the application completes, and the world-readable files make it hard to ensure data privacy for the end users. A recently released version fixes this model using Kerberos but this version was not tested in this project.

Additionally, the Hadoop configuration has a number of site-specific and job-specific parameters that are hard to tune to achieve optimal performance.

10.7.2 Programming in Hadoop

Apache Hadoop and a number of the other open source MapReduce implementations are written in Java. Scientific codes are often written in Fortran, C, C++ or use languages such as python for analysis. The Hadoop streaming model allows one to create map-and-reduce jobs with any executable or script as the mapper and/or the reducer. This is the most suitable model for scientific applications that have years of code in place capturing complex scientific processes. For an application to be able to use this model, it needs to read input through *stdin* and pass output through *stdout*. Thus, legacy applications are limited to using the streaming model that may not harness the full benefits of the MapReduce framework.

10.7.3 File System.

The Hadoop Distributed File System (HDFS) does not have a POSIX compliant interface severely restricting the adoptability for legacy applications. Some projects have implemented native Hadoop based applications that take advantage of the full capabilities of the MapReduce model by using HDFS. For example, the authors of CloudBurst have implemented short read mapping for genomic sequence data in Hadoop [75]. However, rewriting applications to use a new file system interface is unlikely to be practical for most application groups, especially those with large legacy code bases that have undergone decades of validation and verification.

HDFS's data locality features can be useful to applications that need to process large volumes of data. However, Hadoop considers only the data locality for a single file and does not handle applications that might have multiple input sets. In addition, in legacy Hadoop applications, each map task operates on a single independent data piece, and thus only data locality of the single file is considered.

10.7.4 Data Formats.

Apache Hadoop considers inputs as blocks of data where each map task gets a block of data. Scientific applications often work with files where the logical division of work is per file. Apache Hadoop has internal support to handle text data. New file formats require additional java programming to define the format, appropriate split for a single map task, reader that loads the data and converts it to a key value pair.

10.7.5 Diverse Tasks.

Traditionally, all mapper and reducer tasks are considered identical in function roughly working on equal sized workloads. Implementing different mapper and reducer requires logic in the tasks that differentiate the functionality since there is no easy way to specify it in the higher-level programming model. In addition, differences in inputs or algorithms can cause worker processing times to vary widely. This could result in timeouts and restarted tasks due to the speculative execution in Hadoop. If there is a large difference in processing time between tasks, this will cause load imbalance. This may dramatically impact the horizontal scalability.

10.8 Summary

The explosion of sensor data in the last few years has resulted in a class of scientific applications that are loosely coupled and data intensive. These applications are scaling up from desktops and departmental clusters and now require access to larger resources. These are typically high-throughput, serial applications that do not fit into the scheduling policies of many HPC centers. They also could benefit greatly from the features of the MapReduce programming model.

In our early studies, Hadoop, the open source implementation of MapReduce and HDFS, the associated distributed file system, have proven to be promising for some scientific applications. The built-in replication and fault tolerance in Hadoop is advantageous for managing this class of workloads. In Magellan, we have also experimented with running Hadoop through a batch queue system. This approach can enable users to reap the benefits of Hadoop while running within the scheduling policies geared towards large parallel jobs.

However programming scientific applications in Hadoop presents a number of gaps and challenges. Thus, there is a need for MapReduce implementations that specifically consider characteristics of scientific applications. For example, there is a need for MapReduce frameworks that are not closely tied to HDFS and are available to use with other POSIX file systems. In addition, MapReduce implementations that account for scientific data access patterns (such as considering data locality of multiple input files) are desired. Our early evaluation of alternate MapReduce implementations (through collaborations) shows promise for addressing the needs of high-throughput and data-intensive scientific applications.

Chapter 11

Application Experiences

A diverse set of scientific applications have used the Magellan cloud testbed, resulting in significant scientific discoveries while helping to evaluate the use of cloud computing for science. Early adopters of cloud computing have typically been scientific applications that are largely data parallel, since they are good candidates for cloud computing. These applications are primarily throughput-oriented (i.e., there is no tight coupling between tasks); the data requirements can be large but are well constrained; and some of these applications have complex software pipelines, and thus can benefit from customized environments. Cloud computing systems typically provide greater flexibility to customize the environment when compared with traditional supercomputers and shared clusters, so these applications are particularly well suited to clouds.

We outline select scientific case studies that have used Magellan successfully over the course of the project. The diversity of Magellan testbed setups enabled scientific users to explore a variety of cloud computing technologies and services, including bare-metal provisioning, virtual machines (VMs), and Hadoop. The Magellan staff worked closely with the users to reduce the learning curve associated with these new technologies. In chapter 9 we outlined some of the performance studies from our applications. In this chapter, we focus on the advantages of using cloud environments, and we identify gaps and challenges in current cloud solutions.

We discuss case studies in three areas of cloud usage: (a) bare metal provisioning (Section 11.1), (b) virtualized resources (Section 11.2), and c) Hadoop (Section 11.3). We discuss the suitability of cloud models for various applications and how design decisions were implemented in response to cloud characteristics. In section 11.4 we discuss gaps and challenges in using current cloud solutions for scientific applications, including feedback that we collected from the user community through a final survey.

11.1 Bare-Metal Provisioning Case Studies

Many scientific groups need dedicated access to computing resources for a specific period of time, often requiring custom software environments. Virtual environments have been popularized by cloud computing technologies as a way to address these needs. However, virtual environments do not always work for scientific applications due to performance considerations (Chapter 9) or the difficulties with creating and maintaining images (Section 11.4). In this section, we highlight a set of diverse use cases that illustrate alternate provisioning models for applications that need on-demand access to resources.

11.1.1 JGI Hardware Provisioning

Hardware as a Service (HaaS) offers one potential model for supporting the DOE's scientific computing needs. This model was explored early with Magellan when a facility issue at the Joint Genome Institute (JGI) led to their need for rapid access to additional resources. NERSC responded to this by allocating 120 nodes of Magellan to JGI.

Using ESnet's OSCARS [37] and Science Data Network, nine 1 Gb Layer 2 circuits were provisioned between NERSC's Oakland Scientific Facility (OSF) and the JGI Walnut Creek facility. In essence, the JGI internal network was extended 20 miles to the OSF. These two locations are adjacent on the ESnet Bay Area MAN (Metropolitan Area Network), and therefore there was ample bandwidth with relatively low latency. Dedicated switches at OSF were connected to the Layer 2 connection, and the allocated compute nodes were connected to this network. The InfiniBand network was disconnected on the allocated nodes to maintain isolation. OS provisioning was handled using JGI's existing management infrastructure, and the allocated nodes were booted over the Layer 2 network. As a result, the nodes had immediate access to all the critical data, databases, and account management services at JGI. This included accessing NFS file servers located in Walnut Creek. Within days, JGI users were running jobs on the allocated hardware. The hardware and network proved extremely stable. No changes were required on the part of the users—they simply submitted jobs to the existing Sun GridEngine scheduler as usual, and the jobs were transparently routed to the nodes in Magellan. This effort not only demonstrated the validity of hardware as a service, it enabled JGI to continue operations through a potential crisis with no impact on production sequencing operations.

While this demonstration was a success, the experience revealed several areas for improvement. Ultimately, true hardware as a service requires a nearly automated, on-demand ability to provision hardware and create the necessary network connections to remote resources. For example, the requirement to disconnect the InfiniBand network should be handled by software. This could potentially be addressed by disabling ports on the switch or in the subnet manager. Alternatively, virtualization could be used to create virtual machines that would aid in the separation of resources. Another improvement would be to automate the provisioning and configuration of the network. ESnet's OSCARS provides much of this capability, but it would need to be integrated with a resources manager like Moab or GridEngine. Eventually, one could envision a model where a system administrator at a DOE site would identify the need for more resources, and with a few simple commands could request resources from a cloud resource pool. The cloud scheduler would allocate the resources and communicate with OSCARS to provision the network link between the two distant sites. Finally, the nodes would be provisioned, either using a user-provided image or by directly booting from management services at the home site.

11.1.2 Accelerating Proton Computed Tomography Project

Proton computed tomography (pCT) reconstructs the internal structure of a target object irradiated by a proton beam that is rotated around the object. It is similar in concept to standard medical tomography, which uses X-rays. Proton based reconstructions are desirable in proton-mediated cancer therapy as proton therapy requires a lower radiation dose than X-ray tomography; also, proton-based image reconstructions will more accurately depict the target, thereby producing a better treatment plan. pCT is more challenging to solve than X-ray CT because the charged protons interact with the target and do not travel is straight lines. Individual protons pass through two sensor plane pairs before and behind the target object, recording the position of each passage. These positions determine the proton entry and exit trajectories. Individual protons are absorbed in a calorimeter behind the target object, recording the final energy of the proton, which is correlated to the thickness and density of the object through which it passes. Increased computing power is needed to solve for the reconstruction under the constraints of approximate, non-linear proton paths. Current techniques formulate the problem as the optimization of an over-determined sparse linear system of equations. The Accelerating Proton Computed Tomography Project, a collaboration between Northern Illinois University (NIU), Argonne National Laboratory, NIPTRC (Norther Illinois Proton Treatment and Research Center), Loma Linda University Medical Center (LLUMC), University of Chicago, and California State University at San Bernadino, were able to utilize the bare-metal provisioning on ALCF's Magellan GPU cloud to solve a system of 130 million equations with two million unknowns in under 34 minutes on just 30 GPU nodes. Production level problems will require solving systems with two billion equations and 30 million unknowns in under 10 minutes to significantly improve the efficacy of the treatment. The current code runs on the Magellan GPUs and has been used to reconstruct subsets of the entire volume. It has scaled to 120 GPUs and uses MPI and CUDA for parallelization.

11.1.3 Large and Complex Scientific Data Visualization Project (LCSDV)

vl3 is a parallel volume rendering framework that scales from a laptop to Blue Gene/P supercomputers, and also runs on GPU-based clusters. The framework supports extreme-scale datasets and the rendered images can be viewed on a laptop as well as large tiled walls. vl3 incorporates collaborative tools that enable sharing and interacting with visualizations remotely. The vl3 framework is extensible and allows for the development of custom renderers and data filters, enabling domain-specific visualizations. To date applications include radiology, micro tomography, cardiac electrical activity simulation data, earthquake simulations, climate simulations and astrophysics simulations. To parallelize the volume rendering process, vl3 divides the volume data into sub volumes to be rendered in parallel on the nodes. The rendering is built using OpenGL shading language to enable portability on CPUs and GPUs. The rendered images are then composited in parallel into a final image. vl3 has demonstrated scalable performance for 6400x6400x6400 time-varying volume data of ENZO and has scaled to 16K GPU cores. vl3 can render a 4Kx4Kx4K volume dataset at 2 frames per second (fps) and stream the rendered images over 10Gbps networks to remote displays. This enables scientists at their home institutions to interact with large-scale simulation data at supercomputing facilities. As part of the LCSDV project, an Argonne National Laboratory project, a number of images were rendered on the ALCF's Magellan GPU cloud using bare-metal provisioning. Two of the images rendered are shown in Figure 11.1 and Figure 11.2.

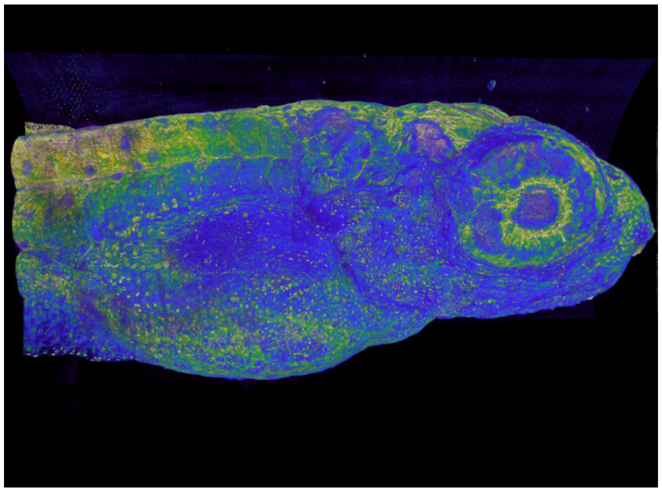


Figure 11.1: A vl3 volume rendering of Zebra Fish data collected at Argonne National Laboratorys Advanced Photon Source.

11.1.4 Materials Project

The goal of the Materials Project, a collaboration between LBNL and MIT, is to accelerate materials discovery through advanced scientific computing and innovative design tools. The aim of the Materials Project is compute the properties of every known inorganic material from first principles methods and to store these computed properties in a database, sharing the data with the scientific community and potentially changing how material science is approached. A user of the database can then apply statistical learning methods to predict new materials and accelerate the rate of the materials design process.

To perform these quantum mechanical calculations, the project uses the ubiquitous periodic pseudopotential planewave code, VASP (Vienna Ab-initio Simulation Package). The calculations are driven by a lightweight workflow engine, which is inspired by a distributed computing model. The workflow to perform these high-throughput first principles calculations involves several steps, including picking a candidate crystal from a database, pre-processing, submitting an MPI job, real-time analysis to detect and potentially fix

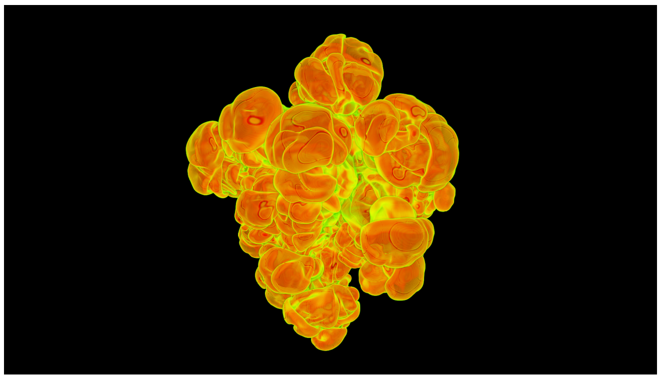


Figure 11.2: An AMR volume rendering of FLASH Astrophysics simulation using vl3.

VASP errors, and updating the database.

The workflow manager was run for three weeks within an allocated sub-queue on Magellan consisting of 80 nodes. During this time the group ran approximately 20,000 VASP calculations and successfully determined the relaxed ground state energy for 9000 inorganic crystal structures. The number of cores for each VASP MPI calculations was chosen to be consistent with the number of cores on a Magellan node. The project found 8 cores to be a good match for addressing the wide distribution of computational demands (e.g., number of electrons, k-point mesh) inherent to high-throughput computing.

There were several challenges running high-throughput quantum mechanical MPI calculations on Magellan. For example, the workflow manager running on the login node would often get killed due to resource limits. Another problem was the robustness of the workflow manager. This was related to updating the state in the MasterQueue database that would corrupt the database. The virtual cluster on Magellan was an essential component to compute in an atomic manner and as a result, dramatically simplified the workflow. This allowed us to avoid bundling several VASP calculations into a single Portable Batch System (PBS) script. This bundling process generates another layer of complexity whose sole purpose is to work within the constraints of the queue policies of supercomputing centers, which limit the number of jobs per user.

Magellan was an excellent solution for high-throughput computing for the Materials Project. The virtual cluster that NERSC staff provided allowed the users to leverage the computing resources in a flexible way, as per the needs of the application, enabling the group to focus on the challenging materials science problems of data storage and data validation.

The Materials Project workflow is an example of a class of next-generation projects that require access to a large number of resources for a specific period of time for high-throughput computing. There is limited or no support available for such workloads at DOE centers today: job queue policies limit the number of jobs allowed in the queue; there are no tools to enable such high-throughput jobs to seamlessly access a large number of resources; and machine firewall policies often restrict or constrain processes such as workflow managers and connections to external databases. Cloud technologies provide a way to enable customized environments for specific time periods while maintaining isolation to eliminate any impact to other services at the center.

11.1.5 *E. coli*

During the weekend of June 3–5, 2011, hundreds of scientists worldwide were involved in a spontaneous and decentralized effort to analyze two strains of *E. coli* implicated in an outbreak of food poisoning in Germany. Both strains had been sequenced only hours before and released over the internet. An Argonne and Virginia tech team worked throughout the night and through the weekend to annotate the genomes and to compare them to known *E. coli* strains. During the roughly 48-hour period, many *E. coli* strains were annotated using Argonne's RAST annotation system, which used the Magellan testbed at ALCF to expand its compute capacity. Around the world, scientists were submitting additional *E. coli* genomes to the RAST servers to be annotated as word got out that DNA sequences were available. Additional backend computing instances were made available by the ALCF Magellan staff to keep up with demand for increased annotation. Such on-demand changes to handle the load and provide the fast turnaround were made possible by the Magellan cloud tools. This project was unique in that it utilized both bare-metal provisioning to leverage native hardware performance (provided by the Argonne bcfg2 and Heckle tools), as well as ALCF's OpenStack cloud for populating the databases.

Once the genomes were annotated, a comprehensive genome content analysis was done that required building molecular phylogenies (evolutionary trees) for each of the proteins in these new strains. These datasets enabled a detailed comparison to the nearly two hundred strains of *E. coli* that are in the public databases. This comparative analysis was quickly published on the internet and made available to the community. Overall, work that normally would have required many months was completed in less than three days by leveraging the on-demand computing capability of the ALCF Magellan cloud. This effort demonstrated the potential value in using cloud computing resources to quickly expand the capacity of computational servers to respond to urgent increases in demand.

11.2 Virtual Machine Case Studies

In our second set of case studies, we outline some of applications that used the Eucalyptus and OpenStack clouds at both sites to harness a set of virtual machines (VMs) for their application needs. A number of applications from diverse domains have used Magellan's cloud computing infrastructure. A number of these applications tend to be data-parallel, since these are the most suitable to run in cloud environments. We outline the suitability, gaps, and challenges of virtual cloud environments for these select applications.

11.2.1 STAR

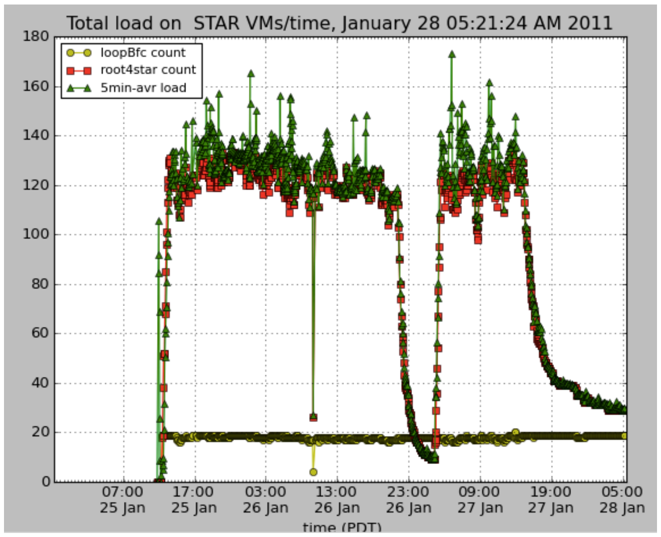


Figure 11.3: A plot of the status of processing near-real-time data from the STAR experiment using Magellan at NERSC over a three-day period. The yellow circles show the number of instances running; the green triangles reflect the total load average across the instances; and the red squares plot the number of running processing tasks. (Image courtesy of Jan Balewski, STAR Collaboration).

STAR is a nuclear physics experiment that studies fundamental properties of nuclear matter from the data collected at Brookhaven National Laboratory's Relativistic Heavy Ion Collider. Previously, STAR has demonstrated the use of cloud resources both on Amazon and at other local sites [28, 56]. STAR used Magellan resources to process near-real-time data from Brookhaven for the 2011 run data. The need for on-demand access to resources to process real-time data with a complex software stack makes it useful to consider clouds as a platform for this application.

The STAR pipeline transferred the input files from Brookhaven to a NERSC scratch directory. The files were then transferred using *scp* to the VMs; and after the processing was done, the output files were moved back to the NERSC scratch directory. The output files were finally moved to Brookhaven, where the produced files were catalogued and stored in mass storage. STAR calibration database snapshots were generated every two hours. Those snapshots were used to update database information in each of the running VMs once per day. Update times were randomized between VMs to optimize the snapshot availability.

The STAR software stack was initially deployed on NERSC/Magellan. After two months of successful running, the software stack was expanded to a coherent cluster of over 100 VMs from three resource pools including NERSC Eucalyptus (up to 60 eight-core VMs), ALCF Nimbus cloud (up to 60 eight-core VMs), and ALCF OpenStack cloud (up to 30 eight-core VMs). Figure 11.3 shows the virtual machines used and corresponding aggregated load for a single run over three days.

In summary, over a period of four months, about 18,000 files were processed, 70 TB of input data was moved to NERSC from Brookhaven, and about 40 TB of output data was moved back to Brookhaven. The number of simultaneous jobs over the period of time varied between 160 and 750. A total of about 25K CPU days or 70 CPU years was used for the processing.

The STAR data processing is a good example of applications that can leverage a cloud computing testbed such as Magellan. The software can be easily packaged as a virtual machine image, and the jobs within the STAR analysis also require little communication, thus having minimal impact from virtualization overheads. Using cloud resources enabled the data processing to proceed during the five months of experiments and to finish at nearly the same time.

There were a number of lessons learned. The goal was to build a VM image that mirrored the STAR reconstruction and analysis workflow from an existing system at NERSC. Building such an image from scratch required extensive system administration skills and took weeks of effort. Additionally, several days effort was required when the image was ported to the ALCF Nimbus and OpenStack clouds. The image creator needs to be careful not to compromise the security of the VMs by leaving personal information like passwords or usernames in the images. Additionally, the image must be as complete as possible while limiting its size, since it resides in memory. Using cloud resources was not a turnkey operation and required significant design and development efforts. The team took advantage of on-demand resources with custom scripts to automate and prioritize the workflow.

11.2.2 Genome Sequencing of Soil Samples

Magellan resources at both ALCF and NERSC were used to perform genome sequencing of soil samples pulled from two plots at the Rothamsted Research Center in the UK. Rothamsted is one of the oldest soil research centers and has a controlled environment, off limits to farming and any other disturbances by humans, reaching back 350 years. The specific aim of this project was to understand the impact of long-term plant influence (rhizosphere) on microbial community composition and function. Two distinct fields were selected to understand the differences in microbial populations associated with different land management practices.

The demonstration used Argonne's MG-RAST metagenomics analysis software to gain a preliminary overview of the microbial populations of these two soil types (grassland and bare-fallow). MG-RAST is a large-scale metagenomics data analysis service. Metagenomics use shotgun sequencing methods to gather a random sampling of short sequences of DNA from community members in the source environment. MG-RAST provides a series of tools for analyzing the community structure (which organisms are present in an environment) and the proteins present in an environment, which describe discrete functional potentials. MG-RAST also provides unique statistical tools including novel sequence quality assessment and sample

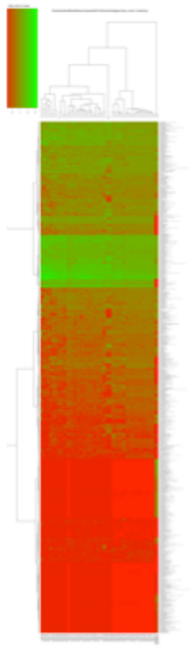


Figure 11.4: The heatmap indicates the relative abundance (red = low, green = high) of functional roles (rows; functional roles were assessed with subsystems) present in each of the deep soil samples (columns; grassland = second to last column, bare fallow = last column) and 61 soil related metagenomic samples previously analyzed with MG-RAST. Samples were clustered with each other (column dendrogram); the functional roles within the samples were also clustered (row dendrogram). Preliminary analysis indicates that the two deep soil samples exhibit a high degree of correlation with each other that is not shared with the 61 other samples. Data were preprocessed to remove sampling based artifacts.

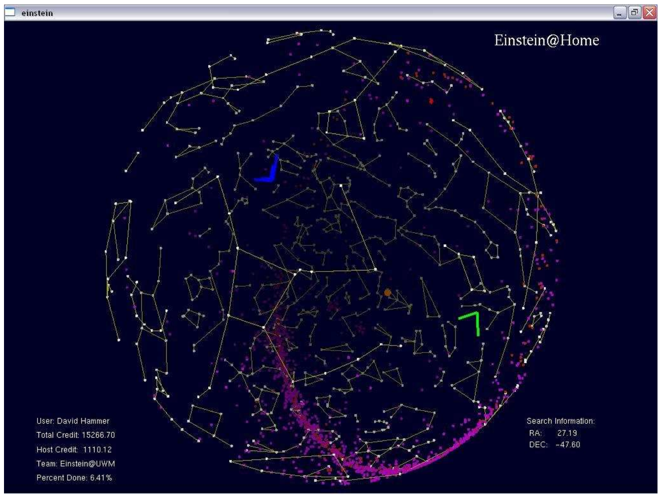
comparison tools. MG-RAST is unique in its high-throughput analysis strategies, which have improved its computational capacity by factor of 200 in the last two years. It uses a distributed load-balancing scheme to spread the computationally expensive sequence similarity comparison across a large number of distributed computation resources, including the Magellan systems. MG-RAST has been used to analyze 3 terabases of metagenomic sequence data in the last six months.

The goal of the demonstration was to utilize the testbeds at both sites, and to explore potential failover techniques within the cloud. The demonstration used 150 nodes from ALCF's Magellan to perform the primary computations over the course of a week, with NERSC's Magellan acting as a failover cloud. Half a dozen machines on ALCF's Magellan were intentionally failed. Upon detecting the failures, the software automatically started replacement machines on the NERSC Magellan, allowing the computation to continue with only a slight interruption.

The same virtual machine image was used on both the ALCF and NERSC Magellan clouds. However, this was not a simple port and required some changes to the image. The instance uses all eight cores on each cloud node and about 40% of the available memory. The demonstration was a single run within the Deep Soil project and represents only $\tilde{1}$/30th of the work to be performed.

Use of the Magellan cloud allowed the project to dynamically utilize resources as appropriate, while the project continued to work on algorithmic improvements. This demonstration showed the feasibility of running a workflow across both the cloud sites, using one site as a failover resource. The project had many challenges similar to STAR discussed above in terms of image management and application design and development.

11.2.3 LIGO



User: David Hammer
Total Credit: 15266.70
Host Credit: 1110.12
Team: Einstein@UWM
Percent Done: 6.41%

Search Information:
RA: 27.19
DEC: -47.60

Figure 11.5: A screenshot of the Einstein@Home screensaver Showing the celestial sphere (sky) with the constellations. Superimposed on this are the gravity wave observatories (the three colored "angle images") and the search marker (a bulls-eye). Recent binary radio pulsar search work units also have a gray disc, representing the radio receiver dish upon which the search is focused for that unit. (Image courtesy of Einstein@Home.)

The Laser Interferometer Gravitational Wave Observatory (LIGO) collaboration is a dynamic group of more than 800 scientists worldwide focused on the search for gravitational waves from the most violent events in the universe. The collaboration developed the Einstein@Home program that uses donated idle time on private personal computers to search for gravitational waves from spinning neutron stars (also called pulsars) using data from the LIGO gravitational wave detector. Later, the LIGO researchers ported this method to the Open Science Grid (OSG). An important aspect of porting Einstein@Home to run on OSG was strengthening of fault-tolerance and implementation of automatic recovery from errors. Fixing problems manually at scale simply isn't practical, so Einstein@OSG eventually automated the process.

LIGO used the Magellan serial queue and the Eucalyptus setup at NERSC to understand the advantages of running in cloud environments. A virtual cluster was set up in the NERSC Eucalyptus deployment for LIGO to access the virtual resources. The virtual cluster was set up to serve an OSG gatekeeper on a head virtual machine to provide the user a seamless transition between the traditional batch queue submission and the virtual cluster.

Our work with LIGO shows how center staff might set up virtual clusters for use by certain groups. To keep the image size small, the gatekeeper stack (approximately 600 MB) was installed on a block store volume. In order to work around the virtual nature of the head node, it was always booted with a private IP; and a reserved public IP, which is static, was assigned after the instance was booted. The gsiftp server had to be configured to handle the private/public duality of the IP addressing. No other changes were required to the LIGO software stack. LIGO was able to switch from running through the NERSC Globus gatekeeper to running through the virtual gatekeeper without major difficulties, and ran successfully.

11.2.4 ATLAS

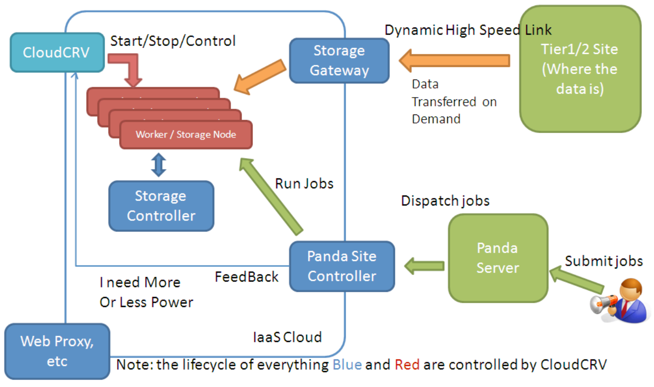


Figure 11.6: ATLAS CloudCRV job flow.

ATLAS is a particle physics experiment at the Large Hadron Collider (LHC) at CERN. The ATLAS detector is searching for new discoveries in the head-on collisions of protons of extraordinarily high energy. ATLAS used Magellan resources at NERSC to build a highly scalable compute cluster on the Eucalyptus cloud. The configuration of the cluster is similar to a normal ATLAS compute cluster: *Panda* is used as the primary job distributer, where a user can submit jobs into a queue. A storage gateway is installed in the cluster to move data into and out of the cluster. A Hadoop File System (HDFS) is installed on the worker nodes so that each worker node also acts as a storage node.

The key difference between the Magellan cluster and a conventional ATLAS cluster is that it fully utilizes the agility provided by the cloud. The cluster can automatically adjust the number of workers according to the number of jobs waiting in the queue. When there are many jobs waiting in the queue, new worker nodes are automatically added to the cluster. When the jobs are finished, idle workers are terminated automatically to save cost.

To address many of the challenges in working with virtual clusters that we discussed in earlier case studies, the ATLAS project developed and deployed a tool called CloudCRV. CloudCRV can automatically deploy and configure individual VMs to work as a cluster. It can also scale the size of the cluster automatically according to demand. As shown in Figure 11.6, the lifetime of all core services and the worker nodes are controlled by CloudCRV.

11.2.5 Integrated Metagenome Pipeline

The Integrated Microbial Genomes (IMG) system hosted at the DOE Joint Genome Institute (JGI) supports analysis of microbial community metagenomes in the integrated context of all public reference isolate microbial genomes. The content maintenance cycle for data involves running BLAST for identifying pairwise gene similarities between new metagenome and reference genomes, where the reference genome baseline is updated with new (approximately 500) genomes every four months. This processing takes about three weeks on a Linux cluster with 256 cores. Since the size of the databases is growing, it is important that the processing can still be accomplished in a timely manner. The primary computation in the IMG pipeline is BLAST, a data parallel application that does not require communication between tasks and thus has similarities with traditional cloud applications. The need for on-demand access to resources makes clouds an attractive platform for this workload.

The pipeline is primarily written in Perl, but it includes components written in Java, as well as compiled components written in C and C++. The pipeline also uses several reference collections (typically called databases), including one for RNA alignment and a periodically updated reference database for BLAST. The pipeline and databases are currently around 16 GB in size. This does not include Bio-perl, BLAST, and other utilities. The pipeline was run across both Magellan sites through Eucalyptus. A simple task farmer framework was used to distribute the workload across both sites. As virtual machines came up, a client would query the main server for work and run the computation.

11.2.6 Fusion

Fusion is a traditional 320-node high-performance compute cluster operated by the Laboratory Computing Resource Center at Argonne. In order to expand the capabilities of the cluster, a project was launched to prototype an application that could offload traditional HPC style jobs to Magellan resources at the ALCF in a transparent fashion. HPC batch jobs are often pre-coupled with a lot of expectations regarding file system locations, amount and type of resources, software availability, etc. To seamlessly offload these types of jobs on the ALCF Magellan cloud, a compute node image was created that establishes a VPN connection back to the Fusion HPC cluster on launch, and appears to the users as one of the default batch computation systems. Upon submission of a job, the application requests a number of instances from the cloud on the fly, passes the user's job to them for execution, and terminates them when the work is completed. The end result was a success; a number of test serial jobs were offloaded to the Magellan cloud at ALCF. This case study demonstrates a scenario where serial jobs could be redirected to cloud resources, enabling the HPC resources to cater to the larger parallel jobs, therefore making better use of specialized hardware.

11.2.7 RAST

SEED is an open-source comparative-analysis genomics effort. The compute-intensive protein-base searches were offloaded from their traditional cluster onto the Magellan resources at ALCF during periods of heavy demand. Incoming raw genome sequences were processed with Glimmer to identify individual genes. These were subsequently annotated with RAST. The subsequent protein-base sequences were then processed via BLAST on a persistent 32-node Magellan virtual cluster against the NCBI and Swiss-Prot protein-sequence databases, which were shared among the two clusters via NFS. The results were then fed into the SEED database, enabling users to query similarities among genomes. This was a heavily compute-bound application, and performance was deemed to be very good on virtual resources.

RAST had a number of similar challenges in managing virtual machine images and tying to existing infrastructure such as NFS and databases to make this load balancing transparent.

11.2.8 QIIME

QIIME (Quantitative Insights Into Microbial Ecology) is a toolset for analyzing high throughput 16S amplicon datasets. 16S rRNA is a highly conserved structure in microbial genomes that can be sequenced with the aid of universal primers. These regions provide a moderate resolution genomic method to discern the different operational taxonomic units (OTUs) present in an environment. QIIME calculates diversity metrics for multiplexed 16S amplicon sequence data sets. It operates in both serial and parallel modes, and has a native mode for cloud style systems. Researchers at the University of Colorado Boulder used ALCF's Magellan to test n3phele, a tool for managing bioinformatic analysis in the cloud, and demonstrated:

- Integration of the n3phele platform hosted on Google Apps Engine with the Magellan resources at ALCF using the published APIs.
- Upload and installation of the QIIME 16s metagenomic analysis software into the ALCF Magellan environment.
- Running of metagenomic analysis workloads using Magellan resources through the n3phele browser based desktop environment.

Key to the success of this research investigation was the secure and open access to ALCF's Magellan cloud through the exposed web service APIs and the ability for workloads operating within the Magellan environment to access resources on the public internet using protocols such as HTTP, HTTPS, FTP and SSH. Support provided by the ALCF Magellan staff was integral in accomplishing this work.

11.2.9 Climate

Climate scientists are better able to understand global climate change and evaluate the effectiveness of possible mitigations by generating and sharing increasingly large amounts of data. The Climate 100 data consists of on the order of a million files that average a few hundred megabytes each. Climate simulations running on large-scale supercomputers are used to generate these data sets. However, the analysis of these simulations can be performed on a variety of resources and is well suited for cloud resources. Climate 100 simulation analysis has been run on virtual machines on Magellan [65]. The results from this endeavor demonstrated that virtualized environments had portability benefits, and the performance made it a viable option for such large-scale data analysis. The loosely coupled analysis runs are well suited for the diverse cloud environments. The Climate 100 analysis is also an ANI project and used Magellan resources for the ANI demos.

11.3 Hadoop

A number of applications from diverse communities have experimented with Hadoop at NERSC, including climate analysis, bioinformatics applications, etc. Bioinformatics and biomedical algorithms have large numbers of high-throughput jobs with minimal or no coordination required between individual jobs, making them suitable for MapReduce/Hadoop. In this section we outline three case studies that show some of the uses of Hadoop on Magellan.

11.3.1 BioPig

BioPig is a framework that works on top of Apache Pig to enable biologists to create simple scripts that can perform sophisticated analysis of genomic data. Pig is a framework and language that layers on top of the Apache Hadoop MapReduce framework. The Pig framework allows a user to express an operation such as filter that is then converted into a set of Map and Reduce steps that are executed in Hadoop. This allows users to more easily exploit the capabilities of Hadoop without learning the details of how to program for it. BioPig extends Pig using custom defined functions (CDF). The extensions provide common operations that are performed on genomic data, such as converting sequencer reads to a k-mer space.

One of the goals of using BioPig is to enable analysis that is currently difficult to tackle using current tools. For example, one goal was to find new cellulases — enzymes that can break down cellulose — in a very large (497 GB) metagenomic data set extracted from a cow's rumen. Researchers were also interested in using the framework to enable more routine tasks such as k-mer histogram generation. Attempts had already been made to use serial and MPI programming techniques. With serial programming, the applications ran out of memory, except on a 1 TB RAM machine. The MPI program worked well, but required specific knowledge about MPI and was relatively hard to debug. The promise with BioPig is that it could achieve some of the scaling benefits of MPI without requiring special training.

The challenges in using Hadoop were in debugging. The Hadoop job tracker website shows the progress of jobs graphically and numerically and allows the user to drill down to the individual log files for jobs. A BioPig user who writes their own custom defined functions will need to use these tools for debugging. Another challenge that was encountered was that even though the framework is designed to scale to very large datasets, users still need to be careful with how they use it. For example, in one case runs were creating an index structure that ended up filling the entire Hadoop file system (HDFS) space. The runs were adjusted to work around this by applying an encoding method to the indexed data. Steps were required to control the amount of RAM used by the mappers and reducers. This amounts to setting the right maximum for the Java virtual machine's heap size. Thus significant tuning of parameters was required to effectively run these applications in Hadoop.

Over the course of the tests, there was significant use of Magellan Hadoop for this effort. Magellan was used both for the testing and development as well as to perform several runs of the pipeline against the cow rumen data set.

11.3.2 Bioinformatics and Biomedical Algorithms

A group in Washington State University is currently developing MapReduce algorithms and implementations for some of the emerging data-intensive problems in bioinformatics and computational biology. The group started using the NERSC Magellan Hadoop cluster in November 2010. The scientific and computational problems addressed (in either completed or ongoing efforts) are as follows:

Identifying peptides from mass spectrometry data obtained from environmental microbial communities. The group developed a simple MapReduce implementation called MR-MSPolygraph for parallelizing a novel serial hybrid search method developed by researchers at Pacific Northwest National Laboratory. The data intensive nature of the problem coupled with built-in features of Hadoop such as load balancing makes Hadoop an appealing platform for this application. Due to the inherent data parallel nature of the underlying approach, the input could be divided into smaller chunks and efficiently parallelized. However, the main challenges involved managing the workload imbalances across chunks (e.g., the same chunk size may lead to a variable workload), and implementing the necessary file I/O operations during processing of each chunk. These challenges were overcome by conducting some parametric studies (e.g., finding the empirically optimal work granularity to ensure load balance), and by resorting to GPFS file reads during processing.

Experimental results showed that the implementation scales linearly, giving a peak speedup of 398x on 400 map tasks. More importantly, 64,000 experimental spectra were analyzed in six hours on 400 cores, reducing the time to solution drastically (a serial run would have needed more than 2,000 CPU hours). An applications note on this research was accepted in Bioinformatics [50].

Protein sequence clustering. The goal of this project is to identify clusters within graphs built out of known or predicted protein sequences from public metagenomic repositories. Among other applications, clustering functions can be useful in identifying protein families represented in environmental communities, but the volume of sequence information and the complexity of the clustering process necessitate a high degree of parallelism. The group is developing a new MapReduce algorithm for clustering massively large graphs and have been testing it on the Magellan Hadoop cloud.

Sequence indexing. In another ongoing effort, a new MapReduce algorithm for building the suffix array data structure for DNA and protein sequences is being developed. Once fully developed, this tool can help index public sequence repositories for efficiently supporting pattern matching and sequence searching operations.

11.3.3 Numerical Linear Algebra

The QR factorization is an important decomposition in the areas of scientific computing and numerical linear algebra. In particular, tall and skinny (TS) matrices, where the number of rows is much larger than the number of columns, have several applications. Communication-avoiding algorithms for QR decompositions of TS matrices have been developed[16] that lend themselves well to the MapReduce programming model. Previously, Constantine and Gleich demonstrated how to effectively use MapReduce to perform TSQR[12]. One component that is missing from MapReduce TSQR is how to effectively (i.e., with speed, accuracy, and a small amount of memory) generate Q explicitly. For a class project of the parallel computing course at the University of California, Berkeley (CS 267) taught by James Demmel and Katherine Yelick, a student project's goal was to look at iterative refinement as a method for generating Q. Another goal was to compare MapReduce implementations of TSQR to a Cholesky QR implementation.

Apache Hadoop was used as the MapReduce architecture, and experiments were conducted on Magellan at NERSC. Tools such as Dumbo were used for the Python wrappers around the Hadoop environment and for deploying jobs on Magellan. The map and reduce tasks were implemented using Python 2.6, and Hadoop streaming was used to plug the Python tasks in Hadoop.

11.4 Discussion

Cloud computing promises to be useful to scientific applications due to advantages such as on-demand access to resources and control over the user environment. However, numerous challenges exist in terms of composing the application programming model, designing and constructing images, distributing the work across compute resources, and managing data. In this section, we summarize the challenges that we observed as we worked with our users. We also collected feedback from our users through a survey at the end of the project. We summarize the responses from our users in this section.

11.4.1 Setup and Maintenance Costs

For use of both virtual machines and MapReduce/Hadoop models, our users invested a fair bit of time and effort in getting the setup right. Our bare-metal provisioning was better for users who needed urgent access to resources (e.g., *E. coli*) and definitely resulted in shorter setup times. Many of the challenges with setup on virtual machines related to image, data, and workflow management. Hadoop streaming results in significantly less porting effort, but to get the full benefits of Hadoop, applications need to be rewritten in the MapReduce framework using standard Hadoop Java APIs. In addition to initial setup, there are some long term maintenance costs that are also involved. For example, science users need to keep images up to date with patches and worry about software compatibility with upgrades, etc. Similarly, Hadoop users need to keep up with API changes in Hadoop, etc. Maintenance and setup costs could be reduced by science-specific tools, but many of the setup and maintenance costs are similar to managing scientific codes and data in large-scale environments such as HPC.

11.4.2 Customizable Environments

Virtualization environments give cloud users flexibility in controlling and managing their own software stacks. But the added flexibility comes with additional responsibilities on the end-user. In current supercomputing centers, the sites manage the operating system and common middleware that is needed across multiple groups. Users compile and install their applications on specific systems (often with help from site personnel for optimizations necessary for particular hardware). In cloud systems, the end-user is responsible for managing all aspects of the operating system and other software that might be required by the application.

The user has to identify, manage, operate, and maintain the operating system and dependent libraries in addition to their specific application software. This requires end-users to have system administration skills themselves, or additional support. Thus centers will need to provide tools and support for managing a diverse set of kernels and operating systems that might be required by specific groups. The clear separation of responsibilities for software upgrades and operating system patches no longer exists, and sites will need mechanisms to bridge the gap between supporting user-supported images and site security policies.

Tools exist to bundle a running operating system and upload it to the cloud system. However, some customization is typically required. Users need to have an understanding of standard Linux system administration, including managing *ssh daemons, ntp*, etc. Furthermore, debugging and testing can be tedious, since it often requires repacking and booting instances to verify the correct behavior. This also requires experimentation to determine what applications and data are best to include in the image or handle through some other mechanism. The simplicity of bundling everything in the image needs to be balanced with the need to make dynamic changes to applications and data. This process is complex and often requires users to carefully analyze what software pieces will be required for their application, including libraries, utilities, and supporting datasets. If the application or supporting datasets are extremely large or change quickly, then the user stores the data outside of the image due to limits on image size and its impact on virtual machine boot-up times and memory that is available to the application at run-time. All our scientific users who had to create images ranked it as *medium* or *hard*.

In order to help user groups try out cloud computing without the setup costs, we set up a virtual cluster with a PBS queue that users could submit to. This largely made the virtual environment look similar to the HPC environments. This environment allowed users to try out the virtual environment and compare it to bare-metal performance. The success of this approach can be seen from one of the users who remarked, “Everything works beautifully on the cluster! I deployed our software on the gatekeeper and ran a job through PBS. I didn’t hit a single problem. From my perspective it makes no difference.”

11.4.3 On-Demand Bare-Metal Provisioning

As illustrated through the use cases presented in Section 11.1, on-demand bare-metal provisioning through serial queues, reservations, and HaaS can meet many of the needs of scientific users seeking the extra features of cloud environments such as custom environments. Bare-metal provisioning provides a number of advantages, including access to high-performance parallel file systems, better performance for applications that cannot tolerate the virtualization overhead, and access to specialized hardware for which virtualization does not currently exist (e.g. GPUs).

11.4.4 User Support

It is important to understand that a large number of the users of Magellan had already used cloud resources, had significant experience with virtual environments and system administration, or had computer scientists or IT personnel on the project helping. In spite of previous experience, these users faced numerous challenges. Scientific users with less experience in cloud tools require additional user support and training to help them port and manage application in these environments; 96% of the users who responded to the survey (including computer science and IT personnel) said they needed help from the Magellan staff. Most Magellan users were able to overcome the challenges of using these environments and would consider using these environments again.

11.4.5 Workflow Management

Another challenge in using cloud systems is developing a mechanism to distribute work. This is complicated by the fact that cloud systems like Amazon Web Services are inherently ephemeral and subject to failure.

Applications must be designed to dynamically adjust to compute nodes entering and leaving the resource pool. They must be capable of dealing with failures and rescheduling work. In traditional clusters, batch systems are routinely used to manage workflows. Batch systems such as Torque, Sun GridEngine, and Condor can and have been deployed in virtualized cloud environments [71, 49]. However, these deployments typically require system administration expertise with batch systems and an understanding of how to best configure them for the cloud environment. Grid tools can also play a role, but they require the user to understand and manage certificates and deploy tools like Globus. To lower the entry barrier for scientific users, Magellan personnel have developed and deployed Torque and Globus-based system images.

Each of our applications have used different mechanisms to distribute work. MG-RAST and the IMG task farmer are examples of locally built tools that handle this problem. Hadoop might be run on top of virtual machines for this purpose; however, it will suffer from lack of knowledge of data locality. STAR jobs are embarrassingly parallel applications (i.e., non-MPI codes), where each job fits in one core and uses custom scripts to handle workflow and data management.

A majority of our Magellan domain science users, despite previous experience with clouds and virtualization, ranked their experience with workflow management in virtualized cloud environments as *medium* to *difficult.*

11.4.6 Data Management

The last significant challenge is managing the data for the workload. This includes both input and output data. For the bioinformatic workloads, the input data includes both the new sequence data as well as the reference data. Some consideration has to be given to where this data will be stored and read from, how it will be transported, and how this will scale with many worker nodes. In a traditional batch cluster, the users would typically have access to a cluster-wide file system. However, with EC2-style cloud systems, there is a different set of building blocks: volatile local storage, persistent block storage associated with a single instance (EBS), and a scalable put/get storage system (S3). These are different from traditional cluster and HPC offerings that users are accustomed to, and hence our users made limited use of these storage systems. The image also can be used to store static data. Each of these options has differing performance characteristics that are dependent on the application. Thus, the choice of storage components depends on the volume of data and the access patterns (and cost, in the case of EC2). Similar to workflow management, a majority of our users noted that data management in these environments took some effort.

11.4.7 Performance, Reliability and Portability

Scientific applications need to consider performance and reliability of these environments. Our benchmarks (Chapter 9) show that the performance of an application is largely dependent on the type of application.

Cloud resources have also gained popularity since resources are immediately available to handle spikes in load etc. However, starting a set of virtual machines can take a few minutes and can be highly variable when trying to get a large number of VMs [66]. In case of synchronous applications, it will be necessary to wait until all the virtual machines are up; and the user gets charged in public clouds for the time that the machines are up, even if they are not being used.

High availability is often mentioned as one of the advantages of cloud resources. It is true that a number of cloud computing vendors such as Amazon provide various fault tolerance mechanisms, such as availability zones and regions, to enable highly fault-tolerant applications. However, the burden is largely on the end users to design their applications to be fault-tolerant, usually at a higher cost. Failures can occur at the hardware or software level, and good design practices to survive failures must be used when building software and services. The Amazon outage in April 2011 was triggered by an incorrect network configuration change made during maintenance that affected multiple services. Applications that used multiple regions were less impacted by the event. However, the common EBS plane impacted EBS activities. Thus cloud computing has many of the same availability challenges that impact HPC centers.

One of the advantages that virtual environments provide is the ability to customize the environment and port it to different sites (e.g., CERN VM [10]). However, our experience was that machine images are not fully portable and are still highly dependent on site configurations and underlying infrastructure.

11.4.8 Federation

One key benefit of cloud computing is its ability to easily port the software environment as virtual machine images. This enables users to utilize resources from different computing sites which allows scientists to expand their computing resources in a steady-state fashion, to handle burst workloads, or to address fault tolerance.

Both MG-RAST and the IMG pipeline were run successfully across both sites. However, application scripts were needed to handle registering the images at both sites, managing discrete image IDs across the sites, and handling distribution of work and associated policies across the sites.

A number of challenges related to federation of cloud sites are similar to the challenges in grid environments related to portability, security, etc.

11.4.9 MapReduce/Hadoop

Cloud computing has an impact on the programming model and other programming aspects of scientific applications. Scientific codes running at supercomputing centers are predominantly based on the MPI programming model. Ad hoc scripts and workflow tools are also commonly used to compose and manage such computations. Tools like Hadoop provide a way to compose task farming or parametric studies. Legacy applications are limited to using the streaming model that may not harness the full benefits of the MapReduce framework. The MapReduce programming model implementation in Hadoop is closely tied to HDFS, and its non-POSIX compliant interface is a major barrier to adoption of these technologies. In addition, frameworks such as Hadoop focus on each map task operating on a single independent data piece, and thus only data locality of the single file is considered.

Early Magellan Hadoop users were largely from bioinformatics and biomedical disciplines where the problems lend themselves to the MapReduce programming model. Magellan Hadoop users largely used streaming to port their applications into Hadoop, and there were a limited set of users who used Java and higher-level tools such as Pig. Hadoop users encountered some problems, and Magellan staff helped them to get started. HDFS was relatively easy to use, and applications gained performance benefits from being able to scale up their problems.

Overall, MapReduce/Hadoop was considered useful for a class of loosely coupled applications. However, the open-source Hadoop implementation is largely designed for Internet applications. MapReduce implementations that take into account HPC center facilities such as high-performance parallel file systems and interconnects as well as scientific data formats and analysis would significantly help scientific users.

Chapter 12

Cost Analysis

One of the most common arguments made for the adoption of cloud computing is the potential cost saving compared to deploying and operating in-house infrastructure. There are several reasons in support of this argument. Commercial clouds consolidate demand across a large customer base, resulting economies of scales that small departmental clusters cannot achieve. This includes lower number of FTEs per core, stronger purchasing power when negotiating with the vendor, and better power efficiency since large systems can justify investing more in the design of the cooling infrastructure. It is worth noting that large DOE HPC Centers also consolidate demand and achieve many of the same benefits. Low upfront costs and pay-as-you-go models are also considered advantages of clouds. Clouds allow users to avoid both time-investments and costs associated with building out facilities, procuring hardware and services, and deploying systems. Users are able to get access to on-demand resources and only pay for the services they use. Ultimately, whether a cloud offering is less costly than owning and operating in-house resources is very dependent on the details of the workload, including characteristics such as scaling requirements, overall utilization, and time criticality of the workload. In this chapter, we will review some of the cost model approaches, consider the costs for typical DOE centers, and discuss other aspects that can impact the cost analysis.

12.1 Related Studies

Walker [81] proposed a modeling tool that allows organizations to compare the cost of leasing CPU time from a cloud provider with a server purchase. The paper provides an analysis comparing three options for NSF's Ranger supercomputing resources, i.e., purchase, lease, or purchase-upgrade, and shows that a three-year purchase investment is the best strategy for this system. The analysis also considers a one-rack server and shows that leasing is the better option for the system. Carlyle et. al. [9] provide a case study of costs incurred by end-users of Purdue's HPC community cluster program and conclude that users of the cluster program would incur higher costs if they were purchasing commercial cloud computing HPC offerings such as the Amazon Cluster compute program. Hamilton published an analysis of data center costs in 2008 [38] and an updated analysis in 2010 [39]. His primary conclusion was that, despite wide-spread belief that power costs dominated data center costs, server costs remained the primary cost factor.

12.2 Cost Analysis Models

There are various approaches to conducting cost comparison as discussed above. We present three approaches, computing the hourly cost of an HPC system, the cost of a DOE center in a commercial cloud, and a cost analysis using the HPC Linpack benchmark as a stand-in for DOE applications. These three models are useful since they tackle the question from various dimensions—system, center, and user perspectives. This first approach translates the operational cost an HPC system into the typical pricing unit used in commercial clouds. The second essentially compares the costs of an entire center. The final approach takes an application centric approach.

12.2.1 Assumptions and Inputs

In all of the cost analyses, we have attempted to use the most cost effective option available. For example, based on our benchmarking analysis, the Cluster Compute offering is the most cost effective option from Amazon for tightly coupled MPI applications and even most CPU intensive applications, since the instances are dedicated resulting in less interference from other applications. Furthermore, most of the instance pricing works out to a roughly constant cost per core hour. For example, a Cluster Compute Instance is approximately 16x more capable than a regular small instance and the cost is approximately 16x more. So using smaller, less expensive instances isn't more cost effective if the application can effectively utilize all of the cores, which is true of most CPU intensive scientific applications. In contrast, many web applications under utilize the CPU, making small instances more cost effective for those use cases. For compute instances, we use a one year reserved instance and assume the nodes are fully utilized over the entire year to compute an effective core hour cost. With reserved instances, you pay a fixed up-front cost in order to pay a lower per hour cost. If the instance is used for a majority of the reserved period (one year in our analysis), this results in a lower effective rate. For example, an on-demand Cluster Compute instance costs $1.60 per hour, but a reserved instance that is used during the entire one year period results in an effective rate of $1.05 (a 30% reduction). We further divide this by the number of cores in the instance to arrive at an effective core-hour cost, which simplifies comparisons with other systems. Table 12.1 summarizes this calculation. It is worth noting that the lowest spot instance pricing is approximately 50% of this effective core hour cost. We do not use this offering as a basis for the cost analysis, since the runtime for a spot instance is unpredictable and application programmers need to design their applications to handle pre-emption, which would not match the requirements for our applications. However, spot pricing does provide an estimate of the absolute lowest bound for pricing, since it essentially reflects the price threshold at which Amazon is unwilling to offer a service.

For file system costs, we use elastic block storage to compute the storage costs for file systems. This most likely underestimates the costs since it omits the costs for I/O requests and the costs for instances that would be required to act as file system servers. S3 is used to compute the costs for archival storage. S3 uses a tiered cost system where the incremental storage costs decline as more data is stored in the system. For example, the monthly cost to store the first terabyte of data using reduced redundancy is $0.093 per gigabyte, while the monthly cost to store data between 1 TB and 49 TB is $0.083 per GB. For simplicity, we compute all S3 costs at the lowest rate. For example, since the NERSC archival system has 19 PB of data stored, we use Amazon's rate of $0.037 per GB (for a month) for data stored above 5 PB with reduced redundancy. The cost for transactions is also omitted for simplicity, but would further increase the cost of using the commercial offering. The pricing data was collected from the Amazon website on September 30, 2011.

12.2.2 Computed Hourly Cost of an HPC System

One of the more direct methods to compare the cost of DOE HPC System with cloud offerings is to compute the effective hourly cost per core hour. This makes it relatively straight forward to compare it with similar commercial cloud systems. However, determining this cost is problematic, since many of the costs used for the calculation are indirect or business sensitive. However, for the sake of comparison we will use Hopper, a Cray XE-6 system recently deployed at NERSC. This system was selected for comparison because it is a relatively recent deployment, is large enough to capture economy of scale, and is tuned for scientific applications relevant to the DOE-SC community. In lieu of providing detailed costs that may be business sensitive, we will use conservative values for the cost which are higher than actual costs. The Hopper contract has been valued at approximately $52M. We use the peak power of 3 MW of power (it typically uses around 2 MW) which translates into an power cost of $2.6M per year assuming a cost of $0.10 per KWHour. In general, $0.10

Table 12.1: Computing the effective core hour cost of a 1-year reserved Amazon cluster compute instance and a summary of other parameters used in the analysis. The pricing data was current as of September 30, 2011.

Reserved Instance Cost (1-year)	$4290 per year
Amortized cost per hour over full year (8760 hours)	$0.49 per hour
Hourly usage cost	$0.56 per hour
Total Effective Instance-Hour Cost	$1.05 per hour
Cores per instance	8 cores
Effective Core-Hour cost	$0.13 per hour
Elastic Block Storage Cost	$0.10 per GB-month
EBS Cost (TB-year)	$1229 per TB-year

KWHour is a high estimate, since most of the DOE laboratories have long-term power contracts for industry rates. Some commercial providers obtain even lower rates by locating data centers in areas with access to abundant power such as the Pacific Northwest. We will assume an annual cost for support and maintenance based on 10% of the contract value, or $5.2M per year. The contract includes hardware and software support including on-site hardware support staff. Therefore, many of the tasks that would require additional FTEs are captured in the support cost. We have included one FTE which is dedicated to supporting the system. It is worth noting that most of the activities performed by the administrator would still be required in a commercial cloud, since the work revolves around configuring and maintaining the environment for the users. We use typical DOE staff rates with overheads which translates into $300k per year. Therefore, the total annual operation cost is $8.1M per year. Finally, we look at the total capability delivered in a year. Hopper has 153,408 cores and runs at approximately 85% total utilization (which includes both job utilization and availability). This translates into approximately 1.1 billion utilized core hours over the course of a year. Here we assume the system is operated over a four year period and amortize the acquisition costs evenly over this period. Using the values above, Hopper effectively costs NERSC approximately $0.018 per core hour. This price includes the IO subsystem (i.e. the parallel scratch file system often included with the system) which is typically around 10% of the total system costs. If we do not consider the cost of the IO subsystem, it further reduces the price to approximately $0.015 per core hour. In comparison, a commercial IaaS offering that focuses on the same application space is between $0.10-$0.20 per core hour. In other words, the cloud cost is 7x-13x more expensive than existing current HPC systems cost DOE Centers to operate. These factors are relatively conservative since they do not include the potential impact on performance described in Chapter 9. Finally, while we selected Hopper for this analysis as its scale is similar to cloud systems, a similar analysis for other recent NERSC systems such as Magellan and Carver shows similar cost advantages, although not quite as high due to their smaller scale. It should be noted that the 2.93 GHz processors in the Amazon Cluster Compute Instance are compared with Hopper's 2.1 GHz AMD Magny-Cours processors. The processors use the same instruction set. A similar analysis for Intrepid, an IBM BlueGene/P at ALCF, was performed and yielded even lower costs; however, Intrepid uses an 850 MHz PowerPC processor. This is completely different architecture and it is difficult to make a direct comparison. Therefore the analysis for Intrepid is not included here. The analysis in Section 12.2.4 may be a more valid comparison for this system.

The system used in the analysis above is optimized for HPC applications with a high-performance interconnect, optimized HPC software stack, and a high-performance file system. Furthermore, these systems typically include contractual commitments by the vendor for delivered application performance. As seen in Chapter 9, a high-performance interconnect is critical to achieving good performance on even modest scale applications which are tightly-coupled (i.e. most MPI applications). Many of the commercial offerings, especially the least expensive, do not provide this capability. Furthermore, the parallel file system on Hopper is nearly 2 PB in size, capable of delivering over 70 GB/s of bandwidth, and uses a file system designed for

Table 12.2: Hourly Cost of a DOE HPC system

	Hopper
Total Acquisition Cost	$52,000,000
Annual Acquisition Cost (divided over 4 years)	$13,000,000
Maintenance & Support	$5,200,000
Power & Cooling	$2,600,000
FTEs	$300,000
Total Annual Costs	$21,100,000
Cores	153,408
Utilization	85%
Annual Utilized Core Hours	1,142,275,968
Cost per Utilized Core Hour (includes storage subsystem)	$0.018

HPC systems. Currently, there is no streamlined method to provide this type of capability in commercial IaaS cloud systems like Amazon. However, even using the best methods available in the existing IaaS system, creating a file system of this scale would add additional costs for storage and for additional instances to act as servers for the file system.

12.2.3 Cost of a DOE Center in the Cloud

Another cost analysis approach is to consider the total annual budget of a center and the resources it provides to the users and translate those into commercial cloud offerings. In Table 12.3, we show a summary of the approximate cost to operate NERSC and ALCF in the Amazon cloud. The computational systems are calculated using Amazon's Cluster Compute offering which provides the closest performance to traditional HPC systems. Based on the performance studies described in Chapter 9, using less expensive, less capable offerings would result in significantly lower performance, which would actually increase costs. In addition, the estimates use a 1-year reserved instance with the generous assumption that the cores are used for an entire year (see Section 12.2.1 for a more detailed description of the assumptions used in the pricing calculations). This results in effective core hour cost of $0.13 per core hour. The total core hours delivered by each system in a year is multiplied by this rate assuming 85% overall utilization (NERSC typically exceeds this value). The storage costs are calculated using $0.037 per GB Month and $0.05 per GB for transfers out of S3 (transfers into S3 are currently free). This is based on the marginal cost of single copy storage in Amazon S3. The file system costs are based on Amazon's Elastic Block Storage cost of $0.10 per GB Month. The data amounts are based on the current volume of data stored on disk and tape, not the available capacity. They do not account for the observed growth in data storage. For example, the annual growth in HPSS (tape) is around 70%. The calculated total annual cost to operate NERSC in the cloud is just over $200M, and to operate ALCF in the cloud is just under $180M. NERSC has a current annual budget of between $50M-$55M. ALCF is a relatively new facility and its budget has grown since it started. The budget shown in the table is an average over a 4 year period, using ALCF's actual budgets for 2008 through 2011. These budgets include all system acquisition costs, support and maintenance, power and cooling cost, system administration and management, and software licensing costs, as well as data analytics and visualization resources that are provided to the facility users.

While facilities costs are not directly included in NERSC's budget, and some are not directly included in ALCF's budget (data center electricity is included), they are incorporated in overhead charges and burdens and therefore indirectly included in the budget. However, the facilities costs are negligible. For example, if we use an industry average cost of $1,200 per square foot for data center space for a system the size of Hopper, we arrive at $3.6M. Assuming a 10 year life span for the data center, which is low, we arrive at $360k per year. This translates into approximately 3/100th of a cent per core hour or around a 2% increase in the

Table 12.3: Annual Cost of the DOE NERSC HPC Center in the cloud

NERSC			ALCF	
Cost in Cloud per Year			Cost in Cloud per Year	
· Hopper (152,496 cores)	$148,993,548		· Intrepid (163,840 cores)	$160,077,004
· Franklin (38,320 cores)	$37,439,885		· Surveyor (4096 cores)	$4,001,925
· Carver (3,200 cores)	$3,126,504		· Challenger (4096 cores)	$4,001,925
· HPSS (17 PB, 9 PB transferred)	$8,189,952		· HPSS (16 PB, 9PB transferred)	$1,062,297
· File Systems (2 PB Total)	$2,642,412		· File Systems (8 PB Total)	$10,066,329
Total Cost in Cloud	$200,392,301		Total Cost in Cloud	$179,209,482
NERSC Total Annual Budget (including support and other services)	$55,000,000		ALCF Total Annual Budget (including support and other services)	$41,000,000
NERSC Annual Budget for systems and related costs only	$33,000,000		ALCF Annual Budget for systems and related costs only	$30,750,000

estimate. In addition, the budgets for both centers include additional services and support which would not be captured in the cloud cost. Our users have indicated the need for these services for cloud solutions as well (Chapter 11). These additional services account for approximately 40% of NERSC's annual budget and 25% of ALCFs budget (calculated based on the staff at the centers who provide these additional services and support). Adjusting for these costs results in a comparison of approximately $200M for the commercial cloud versus $33M for the actual costs of NERSC to DOE, and a comparison of $180M for the commercial cloud versus $31M for the actual costs of ALCF to DOE.

12.2.4 Application Workload Cost and HPL Analysis

Another approach is to consider the cost of running a given workload in a commercial cloud and multiplying that out for the anticipated amount that the workload would be performed over a period of time. Chapter 9 showed the costs for bioinformatics workloads and STAR workloads. This approach works best when the workload can be well encapsulated and doesn't have extraneous dependencies on large data sets. For example, performing functional analysis on genomic data sets works well since the input data can be easily quantified and the run time for a given dataset can be easily measured.

Here we use the Linpack benchmark as a stand-in for the various applications that run at NERSC and ALCF. One reason for choosing this benchmark is it is a recognized benchmark that is published for many systems and removes many of the complexities of making a comparison across different system architectures. Also, it was chosen because it gives some favoritism to the Cloud systems since Linpack has high computational intensity and makes modest use of the interconnect compared with many real-world scientific applications using MPI (Section 9.1.5). For the comparison, we calculate the cost of a Teraflop Year. This represents the cost to deliver a full teraflop for an entire year. Table 12.4 shows a comparison of the cost of a Teraflop Year (based on Linpack) from Amazon, NERSC, and ALCF. All of the HPL results come from the published values listed on the TOP500 [80] website from the June 2011 list. The Amazon costs are based on 1-year reserved Cluster Compute Instances that are assumed to be fully utilized during the year (as described earlier). The NERSC and ALCF costs are based on each center's total annual budget which includes significantly more than just computation as noted in the section above. The comparison illustrates that for tightly coupled applications (even ones with high computational intensity like Linpack), the cost of a commercial cloud is historically much higher (over a factor of 4). This factor is lower compared with some of the other approaches, but, again these calculations use the entire center budgets which include

Table 12.4: Comparison of cost of a Teraflop Year between DOE Center and Amazon

	Amazon (1-year reserved instances)	NERSC	ALCF
HPL Peak	42TF	1361 TF (Hopper 1054 TF, Franklin 266 TF, Carver 41TF)	481 TF (Intrepid 459TF, Surveyor 11TF, Challenger 11TF)
Cost	\$7.5M/year for 42TF	\$55M/year (entire annual center budget)	\$41M/year (average annual center budget)
Cost per TF to operate for 1 year	\$179K/year per TF	\$40K/year per TF	\$85K/year per TF

services, storage, network, and other benefits. For example, if we isolate just the Hopper system (1054 TF) and use the annual costs from Section 12.2.2 (\$21.1M per year), we arrive at \$20k per TF-Year. The ALCF numbers demonstrate the impact of older equipment on cost analysis, as the ALCF systems are coming up on their fifth year of operations. These systems will soon be replaced with new, next generation IBM Blue Gene systems. A discussion of this impact is provided in the section on historical trends in pricing below.

There are scientific applications where the performance penalty of running in a virtualized environment including Cloud systems is less significant. For example, many high-throughput applications such as those used in bioinformatics run relatively efficiently in cloud systems since they are computationally intensive and can run independent of other tasks. One way of understanding the potential improvements for applications that run more efficiently in Clouds is to recalculate the Linpack result assuming the same efficiency across systems. Amazon's Linpack result achieved 50% of peak compared with 82% of peak for the NERSC and ALCF systems. If Amazon were able to achieve the same efficiency as the DOE HPC systems, then their cost would drop to \$109K/year per TF which is still higher than the DOE systems.

12.3 Other Cost Factors

In addition to the analysis above, we briefly discuss other factors that can impact costs and should be considered when performing a cost/benefit analysis for moving to a cloud model.

Utilization. Consolidation has been cited as one of the primary motivators for using cloud resources. Utilization of IT hardware assets has been reported at between 5% and 20% [5]. This is not a typical utilization rate at HPC Centers, as they already consolidate demand across a large user base. For example, DOE HPC Centers provide resources to hundreds of projects and thousands of users. Consequently, many DOE HPC Centers have very high utilization rates. In the above analysis, we have assumed a utilization of 85% which is relatively conservative.

Security. Security is another area where outsourcing to the cloud is often expected to reap cost savings, since the responsibility for security can potentially be shifted to the commercial cloud provider. This may be more accurate for particular cloud models than others. For example, when outsourcing services like e-mail, the service provider can centrally manage applying updates, protecting against common spam and phishing attacks, etc. This can likely be done more efficiently and effectively than typical in-house operations can achieve. However, for infrastructure services the answer is more complicated. In an IaaS model, the commercial cloud provider is responsible for certain aspects of security such as physical security, network security, and maintaining security domains. However, the end user must still insure that the operations of the virtual resources under their control are compliant with relevant security requirements. Many of the security mechanisms are implemented and enforced at the host level, such as maintaining the security of the operating system, configuration management, and securing services. As illustrated in the case studies (Chapter 11), most scientific users are not experts in these areas and thus will require user support services that will require additional personnel and possibly training. In the end, overall security costs in clouds are unlikely to go down but will more likely just shift from one area to another.

Power Efficiency. Power efficiency is often noted as a reason commercial cloud providers are cost effective. Most of the power efficiency comes from consolidation which was discussed elsewhere. This consolidation results in higher utilization and less wasted power. In addition, warehouse scale data centers can easily justify the additional design and operations efforts to optimize the facilities and hardware to achieve improved power efficiency. Consequently, best-in-class commercial cloud providers boast of power usage effectiveness (PUE) of below 1.2 [35]. The PUE is the ratio of the total power used to operate a system to the amount of power used directly by the IT equipment. Higher PUEs are typically due to inefficiencies in cooling, power supplies, air handlers, etc. So a PUE below 1.2 is an example of a very efficient design. In comparison, a 2007 report cited an estimated average PUE of 2.0. More recent reports estimate a range between 1.83 and 1.92, but these surveys likely include results from commercial cloud data centers [54]. DOE HPC Centers have raised concerns about power requirements for large systems for nearly a decade. This was triggered by an observation of the growing costs of electricity to operate large HPC systems. Consequently, many of the large DOE HPC data centers are working towards improving their PUE and the overall efficiency of the systems. While DOE facilities may not quite match the best commercial warehouse scale data centers, recent deployments are approaching PUE values between 1.2-1.3. These efficiencies are achieved through novel cooling design, architectural choices, and power management. For example, the Magellan system at NERSC utilized rear door heat exchangers which used return water used to cool other systems. Another example is the cooling for the ALCF data centers which has been designed to maximize free cooling capabilities when weather conditions are favorable. With high efficiency chillers installed to allow for partial free cooling and centrifugal chiller cooling operations simultaneously, up to 17,820 KW-hr can be saved per day. Furthermore, new planned facilities, such as the LBNL Computational Research and Theory Building, will incorporate free-air cooling to further improve the energy efficiency. Looking towards the future, a major thrust of the DOE Exascale vision is to research new ways to deliver more computing capability per watt for DOE applications. This will likely be achieved through a combination of energy efficient system designs and novel packaging and cooling techniques.

Personnel. A large fraction of the staff effort at DOE HPC Centers does not go directly towards performing hardware support and basic system support but is, instead, focused on developing and maintaining the environment, applications, and tools to support the scientists. Consequently, many of these functions would still need to be carried out in a cloud model as well. Moving towards a cloud model where individual research teams managed their environment and cloud instances could actually increase the effort since many of those functions would go from being centralized at DOE centers or institutional clusters to being decentralized and spread across individual research groups. This could actually reverse the trend in DOE labs towards consolidating support staff for IT systems.

Storage. While most of the cost analysis has focused on the cost of computation in the Cloud, storage cost is also an important consideration. Calculating the TCO for a recent disk storage procurement at NERSC yields a cost of approximately $85 per TB-Year or $0.007 per GB-Month. This is over 14x less expensive than the current storage cost for Amazon's EBS of $0.10 per GB-Month. It is even 5x less expensive than Amazon's least expensive incremental rate for S3 with reduced redundancy ($0.037 per GB-Month). In addition, Amazon imposes additional charges for IO transactions and out-bound data transfers. For example, at Amazon's lowest published rates, transferring 10 TB of data out of Amazon would cost around $500. For some data intensive science applications, this could add considerable costs. Other cloud providers have similar costs. For example, Google's Cloud Storage offerings are basically the same ($0.105 per GB-Month for its lowest published rate). These higher costs do not factor in many of the performance differences discussed in Section 9.3 nor does the need for high-performance parallel file systems by many HPC applications. The cost, performance, and access models for storage are a significant barrier to adopting Cloud based solutions for scientific applications.

Productivity. One advantage of cloud computing can be increased productivity. For users who need the added flexibility offered by the cloud computing model, additional costs may be more than offset by the increased flexibility. Furthermore, in some cases the potential for more immediate access to compute resources could directly translate into cost savings. Section 12.5 discusses examples where this could occur. Potential increases in productivity should be weighed against any up-front investments required to re-engineer software or re-train staff.

Acquisition Costs. It is generally assumed that cloud providers pay significantly lower costs for hardware than typical IT customers. This assumption is reasonable given the purchasing power a cloud provider can exercise. Since cloud providers do not publish their acquisition costs, it is difficult to make direct comparisons. However, DOE centers use competitive procurements or close partnerships to procure their systems. These procurements are typically very large and command substantial discounts from both the integrator and component providers. These discounts may not be quite as high as for warehouse scale datacenters, but likely approach them. Furthermore, the purchasing contracts typically include acceptance criteria tied to application performance which significantly reduces risks.

12.4 Historical Trends in Pricing

One argument often made in favor of commercial clouds is that it enables customers to automatically reap the cost benefits of technology improvements. HPC customers have come to count on the "Moore's Law" improvements in computing capability which typically result in a doubling in capability per dollar approximately every two years. As a result, DOE centers have historically delivered average improvements in computing capability of 40%-80% per year with relatively flat budgets. However, this level of decreasing costs for compute performance is not currently observed in the commercial cloud space. For example, the cost of a typical compute offering in Amazon (m1.small) has fallen only 18% in the last five years with little or no change in capability. This translates into a compound annual improvement of roughly 4%. Meanwhile, the number of cores available in a typical server has increased by a factor of 6x to 12x over the same period with only modest increases in costs. While there are examples of cost reductions in commercial cloud services, such as free in-bound data transfers or lower storage cost for higher tiers, the overall pricing trends, especially for computation, still show only modest decreases. The new systems coming online at the ALCF provide an example of how DOE centers delivere increased computing capability with relatively flat budgets. The three new IBM Blue Gene/Q systems at the ALCF will provide an estimated total HPL Peak of 8.4PF (based on prototype Blue Gene/Q systems currently listed on the November 2011 Top500 list) at an anticipated annual budget of $62M/year. This results in a Cost per TF to operate for 1 year of $7.4K, which is 13-14x better than Amazon's calculated cost (Section 12.2.4), and is an 11x improvement over the current ALCF resource, Intrepid, which is around four years old.

12.5 Cases where Private and Commercial Clouds may be Cost Effective

There are some cases where moving a workload to a private or public cloud offering can be cost effective. We will briefly describe a few of these cases.

Unknown Demand. In the case of a new project or a new application area where the potential demand is still poorly understood, it can be cost effective to use a commercial cloud while the demand is quantified and understood. Once the level has been established, in-house resources can deployed with greater confidence. Similarly, if a new project's prospects are highly uncertain, clouds can be a cost effective option while the long-term fate is determined. In both cases, savings are achieved by avoiding investments in unneeded capacity.

Sporadic Demand. One of the more common cases for using commercial cloud offerings is when the demand is highly variable, especially if there are also time sensitive requirements for the service. In the analysis of DOE centers above, the demand is still highly variable. However, scientists can typically tolerate reasonable delays in the start of an application, especially if this results in access to more cycles. For cases where the demand must be quickly met, the ability to quickly add additional resources can mean the difference in being able to complete the project or not. One example is a DOE lab science project that is conceived suddenly that requires more resources than can be obtained quickly from a DOE HPC Center. DOE HPC Centers typically do not have a viable way to add the necessary resources in a short timescale, and the existing resources are heavily utilized, so the only way to make room for the new project would be to push out existing projects. Other examples are ones with real time-critical requirements, e.g., modeling an oil spill to direct efforts to contain the spill, hurricane tracking, or simulating a critical piece of equipment when it fails and causes an expensive resource to go down. Additionally, some experiments such as those at light sources and accelerators require or benefit from analysis resources in real-time. If those experiments only run a fraction of the time, they may benefit from on-demand cloud-type models. These kinds of opportunity costs can be difficult to quantify but should be considered when considering whether to move to a cloud model.

Facility Constrained. Some sites are severely infrastructure limited. This could be due to building restrictions, insufficient power at the site, or other limitations. In these cases, commercial offerings may be the only reasonable option available. However, if expansion is an option, the long-term cost should be considered. While infrastructure expansion can be costly, those costs can be amortized over a long period, typically 15 years.

The potential cost savings to the customer for these cases come from a few common sources. One is the potential ability to avoid purchasing and deploying computing resources when the demand is unclear. The other is operating resources that are only needed for infrequent periods of time which results in very low utilization. Once a project can maintain a reasonably high utilization of a resource, the cost savings typically vanish.

12.6 Late Update

As this report was being finalized, Amazon announced several important updates. Due to the timing of these announcements, we have left most of the analysis unchanged, but we considered it was important to discuss the impact of these changes on the analysis. There were three significant developments: an updated Top500 entry from Amazon, the release of a new instance type which was used for the Top500 entry, and new pricing. We will discuss each of these and their impact on the analysis.

Amazon's Top500 entry for November 2011 achieved 240 TF and number 42 on the list. More interesting than the absolute numbers or position is the improvement in efficiency to 68% of peak. On previous Top500 entries, Amazon had achieved approximately 50% of peak. This is likely due to better tuning of the Linpack execution. Traditional HPC systems typically achieve between 80%-90% of peak. Eventually virtualized systems may approach these efficiencies through improved integration with the interconnect and continued improvements in the virtualization stacks.

In parallel with the new Top500 result, Amazon announced a new instance type, *cc2.8xlarge.* This instance type is notable for several reasons. It is the first significant deployment of Intel Sandy Bridge EP. In addition to increasing the number of cores per socket to eight cores from four cores compared with the Nehalem processor used in the previous cluster compute instance type, the Sandy Bridge processor also effectively doubled the number of floating point operations per cycle. However, the processors used in the new instance type run at slightly lower clock rate (2.6 GHz versus 2.95 GHz). As a result of these differences, the new instance has a theoretical peak FLOP rate that is approximately 3.5x larger than the previous cluster compute instance type. Linpack is able to take full advantage of these increases. However, Linpack is not representative of all scientific applications. Thus, as shown in our benchmarking (Chapter 9), scientific applications do not typically realize the same benefits. The increase in FLOPs per cycle often occurs every other generation of processor from Intel, since it is typically a byproduct of a change in core architecture. This is the “Tock” of the “Tick-Tock” development cycle in Intel parlance. AMD typically follows a similar roadmap for their processor architecture. Consequently, these types of improvements typically occur every 3-4 years and will be seen in DOE centers in the next few years.

Amazon announced new pricing in November 2011. This lowered the price for some instance types and introduced new tiers of reserved instances. Some instances did not change in cost, e.g., the m1.small remains at $0.085 per hour. However, some of the higher-end instances did drop. For example, the on-demand price of a cc1.4xlarge instance (the type used in much of the analysis above) dropped by 19%. Amazon also expanded the reserved instances to three different tiers. The new tiers allow customers to match the reserved instance to the level of utilization. These different tiers essentially trade higher upfront costs for lower per-usage costs. The tier with the highest upfront costs, “Heavy Utilization Reserved Instances”, provides the lowest effective rate if the instance is fully utilized over the reservation period.

Applying the earlier analysis to the new instance type yields some interesting results. The per-core hour cost of a 1-year reserved instance of the new type results in $0.058 cents per core hour versus $0.13 per core hour before. This is partly due to a heavily discounted reserved instance cost for this instance type. For example, fully utilizing a 1-year reserved instance for most types provides a discount around 40% over the on-demand option. However, for the new instance type this discount is 63%. More impressive, the cost of a Teraflop Year (Section 12.2.4) drops to approximately $36k per TF-Year. This is a 5x improvement over the previous calculation. Roughly half of this improvement comes from the switch to Intel Sandy Bridge and the resulting doubling in FLOPs per cycle. Since Amazon essentially scales its pricing on the number of available cores in the instance, not floating point performance, there is no premium charge for the doubling of the FLOPs per cycle for the new instance type. Another fraction of the increase comes from the improvement in efficiency of the Linpack execution (68% versus 50%). The remainder comes from the drop in pricing and especially the heavily discounted reserved instance pricing.

The DOE systems used in the previous analysis were deployed over a year ago. ALCF’s Intrepid was deployed almost 4 years ago. So, much of the improvement in the Amazon’s values come from the deployment of a very new architecture. Ultimately, the DOE Labs and the commercial cloud vendors are relying on the same technology trends to deliver improved performance over time. As DOE Centers deploy new technologies like Intel Sandy Bridge and AMD Interlagos, their costs will drop in a similar manner. DOE deployments are typically timed about three years apart at each center, with these deployments staggered across centers. In addition, centers will often perform mid-life upgrades to systems to remain closer to the technology edge. This combination of strategies enables DOE to deliver a portfolio of systems that closely tracks the technology improvements. For example, the Mira system is projected to achieve a cost less than $8k per TF-Year when it is deployed next year. This is approximately 4x better than Amazon’s improved result, and represents an improvement of around 10x over the previous ALCF system, Intrepid, which was deployed around 4 years ago. Eventually, DOE HPC centers and commercial cloud providers like Amazon are likely to track each other, aiming for cost-effectiveness. However, the pricing changes in the commercial cloud will not address the various other challenges in moving scientific computing and HPC workloads to cloud offerings, such as workflow and data management challenges, high-performance parallel file systems, and access to legacy data sets (see Sections 6.4 and 11.4).

The recent announcement by Amazon highlights several lessons. One, it shows that Amazon is responding to feedback from the HPC community on improving their offerings in this space. This process started with the introduction of the Cluster Compute and GPU offerings over a year ago and continues with the recent announcement of the new instance type and associated pricing. Secondly, it demonstrates the importance of tracking the changes in the cloud space and routinely updating cost analysis to ensure that the appropriate choices are being made.

12.7 Summary

There are many documented and advertised examples of cost savings achieved by moving to clouds. However, most of these examples come from the IT space where system utilization is extremely low, demand is very dynamic, and response time is critical. DOE labs and DOE HPC Centers have spent decades optimizing systems to meet the needs of science with a watchful eye on cost effectiveness. Consequently, we see that DOE Centers are cost effective when compared to commercial clouds. This is a result of high utilization, operational efficiency, energy efficiency, and highly optimized systems. As labs and science projects weigh the option to move applications to the cloud, they should closely consider the potential up-front costs and operational costs of moving. DOE Labs should focus first on consolidating resources to increase economies of scale, lower operational costs, and improve utilization.

Chapter 13

Conclusions

Cloud computing has gained traction in the last few years in serving the needs of commercial applications, especially web applications. The goal of the Magellan project funded through the U.S. Department of Energy (DOE) Office of Advanced Scientific Computing Research (ASCR) was to investigate the role of cloud computing for scientific workloads. The Magellan project adopted an *application-driven* approach to evaluate what performance and reliability applications can expect in cloud environments, the stability of current private cloud software, which user support models are best suited to cloud solutions, any security implications in these environments, the use of the MapReduce programming model and usability of these environments, as well as cost efficiency of the cloud business model. In addition to answering computer science research questions, Magellan resources played a key role in producing important science results for a diverse set of projects including MG-RAST (a metagenomics analysis server), the Joint Genome Institute, the STAR experiment at the Relativistic Heavy Ion Collider, and the Laser Interferometer Gravitational Wave Observatory (LIGO).

Below we summarize the key results of Magellan:

- Open-source virtualized cloud software stacks have significantly improved over the course of the project, but they still have gaps that must be addressed before they are ready for production use. These cloud software stacks would benefit from additional development, hardening, testing, and customization to support DOE center policies and scientific workload needs.

- DOE centers typically manage the complex software stacks of scientific users and provide user support for code optimizations, etc., to leverage the hardware benefits. The cloud model provides users the flexibility of configuring their own software stacks but this is often complex and tedious and requires some level of system administration expertise. Thus, to efficiently use cloud software stacks, additional user training is required. The user support model will need to be revisited to address specific user support needs, including providing base images, guiding users through complex tasks, and maintenance of the images.

- Cloud environments present some unique security challenges. User-controlled images, dynamic networks, and the transient nature of virtual machines expose additional security risks compared to traditional DOE Center usage models. Constant monitoring and capturing critical system events and logs, coupled with running an intrusion detection system, can mitigate some of these risks. However, DOE Centers will need to take a closer look at current security practices and policies since the cloud model is substantially different from the current HPC model.

- Applications with minimal communication and I/O are able to achieve similar performance in cloud environments as HPC environments. The performance impact for HPC applications comes from the absence of high bandwidth, low latency interconnects in virtualized environments. Thus a majority of current DOE HPC applications even in midrange concurrencies is unlikely to run efficiently in today's cloud environments. Similarly, I/O intensive applications take a substantial hit when run inside virtual environments.

- Cloud programming models such as MapReduce and the resulting ecosystem show promise for addressing the needs of many data-intensive and high-throughput scientific applications. However, current tools have gaps for scientific applications. The MapReduce model emphasizes the data locality and fault tolerance that are important in large systems. Thus there is a need for tools that provide MapReduce implementations which are tuned for scientific applications.
- Current cloud tools do not provide an out-of-box solution to address application needs. There is significant design and programming required to manage the data and workflows in these environments. Virtual machine environments require users to configure and create their software images with all necessary packages. Scientific groups will also need to maintain these images with security patches and application updates. There exist a number of performance, reliability, and portability challenges with cloud images that users must consider carefully. There are limited user-side tools available today to manage cloud environments.
- One way clouds can achieve cost efficiency is though consolidation of resources and higher average utilization. DOE Centers already consolidate workloads from different scientific domains and have high average utilization, typically greater than 90%. Even with conservative cost analysis, we show how two DOE Centers are more cost-efficient than private clouds.

As noted in the final point, a key benefit of clouds is the consolidation of resources. This typically leads to higher utilization, improved operational efficiency, and lower acquisition cost through increased purchasing power. If one looks across the scientific computing landscape within DOE there are variety of models for how scientists access computing resources. These cover the full range of consolidation and utilization scales. At one end of the spectrum is the small group or departmental cluster. These systems are often under-utilized and represent the best opportunity to achieve better efficiency. Many of the DOE National Laboratories have already taken efforts to consolidate these resources into institutional clusters operating under a variety of business models (institutionally funded, buy-in/condo, etc.). In many ways, these systems act as private clouds tuned for scientific applications and effectively achieve many of the benefits of cloud computing. DOE HPC centers provide the next level consolidation, since these facilities serve users from many institutions and scientific domains. This level of consolidation is one reason why many of the DOE HPC centers operate at high levels of utilization.

Clouds have certain features that are attractive for scientific groups needing support for on-demand access to resources, sudden surges in resource needs, customized environments, periodic predictable resource needs (e.g., monthly processing of genome data, nightly processing of telescope data), or unpredictable events such as computing for disaster recovery. Cloud services essentially provide a differentiated service model that can cater to these diverse needs, allowing users to get a virtual private cluster with a certain guaranteed level of service. Clouds are also attractive to high-throughput and data-intensive workloads that do not fit in current-day scheduling and allocation policies at supercomputing centers. DOE labs and centers should consider adopting and integrating features of cloud computing into their operations in order to support more diverse workloads and further enable scientific discovery. This includes mechanisms to support more customized environments, but also methods of providing more on-demand access to cycles. This could be achieved by: a) maintaining idle hardware at additional costs to satisfy potential future requests, b) sharing cores/nodes typically at a performance cost to the user, and c) utilizing different scheduling policies such as preemption. Providing these capabilities would address many of the motivations that lead scientists to consider cloud computing while still preserving the benefits of typical HPC systems which are already optimized for scientific applications.

Cloud computing is essentially a business model that emphasizes on-demand access to resources and cost-savings through consolidation of resources. Overall, whether cloud computing is suitable for a certain application, science group, or community depends on a number of factors. This was noted in NIST's draft document on Cloud Computing, "Cloud Computing Synopsis and Recommendations", which stated,

> *Inherently, the move to cloud computing is a business decision in which the business case should consider the relevant factors some of which include readiness of existing applications for cloud deployment, transition costs and life-cycle costs, maturity of service orientation in existing infrastructure, and other factors including security and privacy requirements.*

Cost-saving benefits from cloud computing depends on a number of factors and will need to be analyzed carefully for each scenario. The NIST document echoes this thought, *"Whether or not cloud computing reduces overall costs for an organization depends on a careful analysis of all the costs of operation, compliance, and security, including costs to migrate to and, if necessary, migrate from a cloud."* Our detailed performance analysis, use case studies, and cost analysis shows that DOE HPC centers are significantly more cost-effective than public clouds for many scientific workloads, but this analysis should be reevaluated for other use cases and workloads.

Magellan Leads

Kathy Yelick, Susan Coghlan, Brent Draney, Richard Shane Canon

Magellan Staff

Lavanya Ramakrishnan
Iwona Sakrejda
Scott Campbell
Tina Declerck
Adam Scovel
Anping Liu
Piotr T. Zbiegiel
Paul Rich

Collaborators

Nicholas J. Wright, Richard Bradshaw, Shreyas Cholia, Linda Winkler, John Shalf, Jared Wilkening, Harvey Wasserman, Narayan Desai, Krishna Muriki, Victor Markowitz, Shucai Xiao, Keith Jackson, Nathan M. Mitchell, Jeff Broughton, Michael A. Guantonio, Zacharia Fadikra, Levi J. Lester, Devarshi Ghoshal, Ed Holohann, Elif Dede, Tisha Stacey, Madhusudhan Govindaraju, Gabriel A. West, Daniel Gunter, William E. Allcock, David Skinner, Rollin Thomas, Karan Bhatia, Henrik Nordberg, Wei Lu, Eric R. Pershey, Vitali Morozov, Dennis Gannon, CITRIS/University of California, Berkeley, Greg Bell, Nicholas Dale Trebon, K. John Wu, Brian Tierney, Brian Toonen, Alex Sim, Kalyan Kumaran, Ananth Kalyanraman, Michael Kocher, Doug Olson, Jan Balewski, STAR Collboration, Linda Vu, Yushu Yao, Margie Wesley, John Hules, Jon Bashor

Acknowledgements

This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357, funded through the American Recovery and Reinvestment Act of 2009. Resources and research at NERSC were funded by the Department of Energy from The American Recovery and Reinvestment Act of 2009 under contract number DE-AC02-05CH11231.

Bibliography

[1] G. Aldering, G. Adam, P. Antilogus, P. Astier, R. Bacon, S. Bongard, C. Bonnaud, Y. Copin, D. Hardin, F. Henault, D. A. Howell, J. Lemonnier, J. Levy, S. C. Loken, P. E. Nugent, R. Pain, A. Pecontal, E. Pecontal, S. Perlmutter, R. M. Quimby, K. Schahmaneche, G. Smadja, and W. M. Wood-Vasey. Overview of the Nearby Supernova Factory. In J. A. Tyson & S. Wolff, editor, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 4836 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 61–72, Dec. 2002.

[2] N. Ali, P. Carns, K. Iskra, D. Kimpe, S. Lang, R. Latham, R. Ross, L. Ward, and P. Sadayappan. Scalable I/O forwarding framework for high-performance computing systems. In *IEEE International Conference on Cluster Computing (Cluster 2009)*, 2009.

[3] K. Antypas, J. M. Shalf, and H. Wasserman. NERSC-6 workload analysis and benchmark selection process. Technical report, LBNL, 2008.

[4] Apache Hadoop `http://hadoop.apache.org/core`. `http://hadoop.apache.org/core`.

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[6] Community Atmopshere Model. `http://www.cgd.ucar.edu/csm/\\models.atm-cam`.

[7] CAM3.1. `http://www.ccsm.ucar.edu/models/atm-cam/`.

[8] S. Canon, S. Cholia, J. Shalf, K. Jackson, L. Ramakrishnan, and V. Markowitz. A performance comparison of massively parallel sequence matching computations on cloud computing platforms and hpc clusters using hadoop. In *Using Clouds for Parallel Computations in Systems Biology Workshop, Held at SC09*, 2009.

[9] A. Carlyle, S. Harrell, and P. Smith. Cost-effective hpc: The community or the cloud? In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 169 –176, 30 2010-dec. 3 2010.

[10] Cern Virtual Machines. `http://rbuilder.cern.ch/project/cernvm/releases`.

[11] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. BigTable: a distributed storage system for structured data. In *OSDI '06: Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*, pages 15–15, Berkeley, CA, USA, 2006. USENIX Association.

[12] P. G. Constantine and D. F. Gleich. Tall and skinny qr factorizations in mapreduce architectures. In *Proceedings of the second international workshop on MapReduce and its applications*, MapReduce '11, pages 43–50, New York, NY, USA, 2011. ACM.

[13] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI '04*, pages 137–150, 2004.

[14] E. Dede, M. Govindaraju, D. Gunter, and L. Ramakrishnan. Riding the elephant: Managing ensembles with hadoop. In *4th Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS)*, 2011.

[15] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good. The cost of doing science on the cloud: the montage example. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pages 1–12. IEEE Press, 2008.

[16] J. Demmel, L. Grigori, M. F. Hoemmen, and J. Langou. Communication-optimal parallel and sequential qr and lu factorizations. Technical Report UCB/EECS-2008-89, EECS Department, University of California, Berkeley, Aug 2008.

[17] N. Desai, R. Bradshaw, and J. Hagedorn. System management methodologies with bcfg2. *;login;*, 31(1):11–18, February 2006.

[18] N. Desai and C. Lueninghoener. *Configuration Management with Bcfg2*. Usenix Association, Berkeley, CA, 2008.

[19] Amazon Elastic Block Store. `http://aws.amazon.com/ebs/`.

[20] Amazon Elastic Compute Cloud. `http://aws.amazon.com/ec2/`.

[21] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S. Bae, J. Qiu, and G. Fox. Twister: A runtime for iterative mapreduce. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 810–818. ACM, 2010.

[22] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox. Twister: a runtime for iterative mapreduce. In *HPDC*, pages 810–818, 2010.

[23] C. Evangelinos and C. Hill. Cloud Computing for parallel Scientific HPC Applications: Feasibility of running Coupled Atmosphere-Ocean Climate Models on Amazons EC2. *ratio*, 2(2.40):2–34, 2008.

[24] Z. Fadika, E. Dede, M. Govindaraju, and L. Ramakrishnan. Benchmarking mapreduce implementations for application usage scenarios. *Grid 2011: 12th IEEE/ACM International Conference on Grid Computing*, 0:1–8, 2011.

[25] Z. Fadika, E. Dede, M. Govindaraju, and L. Ramakrishnan. Mariane: Mapreduce implementation adapted for hpc environments. *Grid 2011: 12th IEEE/ACM International Conference on Grid Computing*, 0:1–8, 2011.

[26] Z. Fadika and M. Govindaraju. Lemo-mr: Low overhead and elastic mapreduce implementation optimized for memory and cpu-intensive applications. *Cloud Computing Technology and Science, IEEE International Conference on*, 0:1–8, 2010.

[27] Federal Risk and Authorization Management Program (FedRAMP) Document `http://www.gsa.gov/portal/category/102371`.

[28] M. Fenn, S. Goasguen, and J. Lauret. Contextualization in practice: The clemson experience. In *ACAT Proceedings*, 2010.

[29] I. Foster, T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, and X. Zhang. Virtual clusters for grid communities. In *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, pages 513–520. Citeseer, 2006.

[30] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1 –10, nov. 2008.

[31] Fuse: File system in userspace. `http://fuse.sourceforge.net`.

[32] Futuregrid website. `https://portal.futuregrid.org/`.

[33] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 29–43, New York, NY, USA, 2003. ACM.

[34] D. Ghoshal, R. S. Canon, and L. Ramakrishnan. I/o performance of virtualized cloud environments. In *The Second International Workshop on Data Intensive Computing in the Clouds (DataCloud-SC11)*, 2011.

[35] Google Data center efficiency. `http://www.google.com/about/datacenters/inside/efficiency/power-usage.html`.

[36] Y. Gu and R. L. Grossman. Sector and sphere: the design and implementation of a high-performance data cloud. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 367(1897):2429–2445, June 2009.

[37] C. P. Guok, D. W. Robertson, E. Chaniotakis, M. R. Thompson, W. Johnston, and B. Tierney. A user driven dynamic circuit network implementation. *DANMS08*, 2009.

[38] J. Hamilton. Cost of Power in Large-Scale Data Centers `http://perspectives.mvdirona.com/2008/11/28/CostOfPowerInLargeScaleDataCenters.aspx`.

[39] J. Hamilton. Overall Data Center Costs `http://perspectives.mvdirona.com/2010/09/18/OverallDataCenterCosts.aspx`.

[40] S. Hazelhurst. Scientific computing using virtual high-performance computing: a case study using the Amazon elastic computing cloud. In *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*, pages 94–103. ACM, 2008.

[41] HPCC benchmark web page: `http://icl.cs.utk.edu/hpcc/`.

[42] HPSS Website. `http://www.hpss-collaboration.org/`.

[43] K. Ibrahim, S. Hofmeyr, and C. Iancu. Characterizing the performance of parallel applications on multi-socket virtual machines. In *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, pages 1 –12, may 2011.

[44] IOR Benchmark. `http://dl.acm.org/citation.cfm?id=1413413`.

[45] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. In *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, EuroSys '07, pages 59–72, New York, NY, USA, 2007. ACM.

[46] K. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. Wasserman, and N. Wright. Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud. In *2nd IEEE International Conference on Cloud Computing Technology and Science*, 2010.

[47] K. R. Jackson, K. Muriki, L. Ramakrishnan, K. J. Runge, and R. C. Thomas. Performance and cost analysis of the supernova factory on the amazon aws cloud. *Scientific Programming*, 19(2-3):107–119, 2011.

[48] J. Jenkins, P. Balaji, J. Dinan, N. F. Samatova, and R. Thakur. Enabling Fast, Non-contiguous GPU Data Movement in Hybrid MPI+GPU Environments. In *In submission*.

[49] G. Juve and E. Deelman. Scientific workflows and clouds. *Crossroads*, 16:14–18, March 2010.

[50] A. Kalyanaraman, W. R. Cannon, B. Latt, and D. J. Baxter. Mapreduce implementation of a hybrid spectral library-database search method for large-scale peptide identification. *Bioinformatics*, 2011.

[51] K. Keahey. Cloud Computing for Science. In *Proceedings of the 21st International Conference on Scientific and Statistical Database Management*, page 478. Springer-Verlag, 2009.

[52] K. Keahey, R. Figueiredo, J. Fortes, T. Freeman, and M. Tsugawa. Science clouds: Early experiences in cloud computing for scientific applications. *Cloud Computing and Applications*, 2008, 2008.

[53] K. Keahey, T. Freeman, J. Lauret, and D. Olson. Virtual workspaces for scientific applications. In *Journal of Physics: Conference Series*, volume 78, page 012038. Institute of Physics Publishing, 2007.

[54] J. G. Koomey. Growth in Data Center Electricity use 2005 to 2010. Technical report, Stanford University, 2011.

[55] W. Kramer, J. Shalf, and E. Strohmaier. The NERSC Sustained System Performance (SSP) Metric. 2005.

[56] J. Lauret, M. Walker, S. Goasguen, and L. Hajdu. From Grid to cloud, the STAR experience. *SciDAC 2010 Proceedings*, 2010.

[57] W. W. Lee. Gyrokinetic particle simulation model. *J. Comp. Phys.*, 72, 1987.

[58] J. Li, D. Agarwal, M. Humphrey, C. van Ingen, K. Jackson, and Y. Ryu. eScience in the Cloud: A MODIS Satellite Data Reprojection and Reduction Pipeline in the Windows Azure Platform. In *IPDPS*, 2010.

[59] H. Liu and D. Orban. Cloud mapreduce: A mapreduce implementation on top of a cloud operating system. In *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGRID '11, pages 464–474, Washington, DC, USA, 2011. IEEE Computer Society.

[60] Mellanox unstructured data accelarator (uda). `http://www.mellanox.com/content/pages.php?pg=hadoop`.

[61] Milc website. `http://physics.indiana.edu/~sg/milc.html`.

[62] J. Napper and P. Bientinesi. Can cloud computing reach the top500? In *Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*, pages 17–20. ACM, 2009.

[63] Nist cloud. `http://csrc.nist.gov/groups/SNS/cloud-computing/`.

[64] L. Oliker, A. Canning, J. Carter, J. Shalf, and S. Ethier. Scientific computations on modern parallel vector systems. In *Proc. SC04: International Conference for High Performance Computing, Networking, Storage and Analysis*, Pittsburgh, PA, Nov 6-12, 2004.

[65] F. T. C. on a Cloud Computing Cluster: Using Parallel Virtualization for Large-Scale Climate Simulation Analysis. Daren Hasenkamp and Alex Sim and Michael Wehner and Kesheng Wu. In *2nd IEEE International Conference on Cloud Computing Technology and Science*, 2010.

[66] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema. An early performance analysis of cloud computing services for scientific computing. *Delft University of Technology, Tech. Rep*, 2008.

[67] M. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel. Amazon S3 for science grids: a viable solution? In *Proceedings of the 2008 international workshop on Data-aware distributed computing*, pages 55–64. ACM, 2008.

[68] Phloem website. `https://asc.llnl.gov/sequoia/benchmarks/PhloemMPIBenchmarks\_summary\_v1.0.pdf`.

[69] X. Qiu, J. Ekanayake, S. Beason, T. Gunarathne, G. Fox, R. Barga, and D. Gannon. Cloud technologies for bioinformatics applications. In *Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers*, pages 1–10. ACM, 2009.

[70] L. Ramakrishnan, R. S. Canon, K. Muriki, I. Sakrejda, and N. J. Wright. Evaluating interconnect and virtualization performance for high performance computing. In *Proceedings of 2nd International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computing Systems (PMBS11)*, 2011.

[71] L. Ramakrishnan et al. VGrADS: Enabling e-Science Workflows on Grids and Clouds with Fault Tolerance. In *Proceedings of the ACM/IEEE SC2009 Conference on High Performance Computing, Networking, Storage and Analysis, Portland, Oregon*, Portland, Oregon, November 2009.

[72] L. Ramakrishnan, P. T. Zbiegel, S. Campbell, R. Bradshaw, R. S. Canon, S. Coghlan, I. Sakrejda, N. Desai, T. Declerck, and A. Liu. Magellan: experiences from a science cloud. In *Proceedings of the 2nd international workshop on Scientific cloud computing*, ScienceCloud '11, pages 49–58, New York, NY, USA, 2011. ACM.

[73] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis. Evaluating mapreduce for multi-core and multiprocessor systems. In *Proceedings of the 2007 IEEE 13th International Symposium on High Performance Computer Architecture*, pages 13–24, Washington, DC, USA, 2007. IEEE Computer Society.

[74] J. Rehr, F. Vila, J. Gardner, L. Svec, and M. Prange. Scientific computing in the cloud. *Computing in Science and Engineering*, 99(PrePrints), 2010.

[75] M. C. Schatz. CloudBurst: highly sensitive read mapping with MapReduce. *Bioinformatics*, pages 1363–1369, June 2009.

[76] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1 –10, May 2010.

[77] D. Skinner. Integrated Performance Monitoring: A portable profiling infrastructure for parallel applications. In *Proc. ISC2005: International Supercomputing Conference*, Heidelberg, Germany, 2005.

[78] Spec website. `https://portal.futuregrid.org/`.

[79] J. Thatcher et al. NAND Flash Solid State Storage for the Enterprise - An In-depth Look at Reliability. Technical report, SNIA, 2009.

[80] Top500 Supercomputer Sites. `http://www.top500.org`.

[81] E. Walker. The Real Cost of a CPU Hour. *IEEE Computer*, 42(4), 2009.

[82] G. Wang and T. E. Ng. The impact of virtualization on network performance of amazon ec2 data center. In *Proceedings of IEEE INFOCOM*, 2010.

[83] Windows azure. `http://www.microsoft.com/windowsazure`.

[84] N. J. Wright, W. Pfeiffer, and A. Snavely. Characterizing parallel scaling of scientific applications using IPM. In *The 10th LCI International Conference on High-Performance Clustered Computing*, March 10-12, 2009.

[85] From Clusters To Clouds: xCAT 2 Is Out Of The Bag. `http://www.linux-mag.com/id/7230/`.

[86] S. Xiao, P. Balaji, Q. Zhu, R. Thakur, S. Coghlan, H. Lin, G. Wen, J. Hong, and W. chun Feng. Transparent Virtualization of Graphics Processing Units. In *In submission*.

[87] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association.

Appendix A

Publications

Selected Presentations, Tutorials and Posters

2011

1. Discussion of Infrastructure Clouds: A NERSC Magellan Perspective, Lavanya Ramakrishnan, MAGIC Meeting, September 2011.
2. The Real Future of HPC Cloud Computing, Jeff Broughton, HPC User Forum, September, 2011.
3. Clouds for HPC: A NERSC Magellan Perspective, Keith Jackson, ISC-Cloud, September, 2011.
4. Cloud Computing: A Scientific Perspective, Keith Jackson, NSA Headquarters, July, 2011.
5. Hadoop Tutorial, Lavanya Ramakrishnan and Shane Canon, SciDAC Tutorial, July 2011.
6. Eucalyptus Tutorial, Iwona Sakrejda, Discovery 2015: HPC and Cloud Computing Workshop, June 2011.
7. Eucalyptus Overview, Lavanya Ramakrishnan, Discovery 2015: HPC and Cloud Computing Workshop, June 2011.
8. NERSC/Magellan Cloud computing overview, Shane Canon, Discovery 2015: HPC and Cloud Computing Workshop, June 2011.
9. Evaluating Cloud Computing for Science, Lavanya Ramakrishnan in Discovery 2015: HPC and Cloud Computing Workshop, June 2011.
10. Magellan: Evaluating Cloud Computing for Science, Shane Canon, High End Computing Interagency Working Group, June 2011.
11. Magellan: Evaluating Cloud Computing for Science, Lavanya Ramakrishnan, Invited talk at Lawrence Livermore National Lab, June 2011.
12. Debunking Some Common Misconceptions of Science in the Cloud, Shane Canon, Invited talk at ScienceCloud, June 2011.
13. Magellan: Experiences from a Science Cloud, Lavanya Ramakrishnan, Paper presentation at ScienceCloud 2011, June 2011.
14. Science in the Cloud: Exploring Cloud Computing for Science, Shane Canon, Invited talk at MoabCon, May 2011.

15. Magellan, An Experiment in Cloud Computing for Science, Susan Coghlan and Shane Canon, presented at ASCAC March 2011.

2010

1. Science in the Clouds: A View from Berkeley, Kathy Yelick, Keynote, ISC Cloud, September 2010
2. Hadoop for Scientific Workloads, Lavanya Ramakrishnan, Invited Talk at IBM Research Almaden, July 2010.
3. Hadoop for Scientific Workloads, Lavanya Ramakrishnan, Hadoop Summit, July, 2010.
4. Magellan, Shane Canon, Discovery 2015: Cloud Computing Workshop, July 2010.
5. Cloud Computing Overview, Lavanya Ramakrishnan, Discovery 2015: Cloud Computing Workshop, July 2010.
6. Magellan, Shane Canon, National Human Genome Research Institute (NHGRI) Workshop on Cloud Computing, May 2010.
7. Cloud Computing for Science, Keith Jackson, Internet2 Joint Techs Meeting, Salt Lake City, Utah, February, 2010
8. Running BLAST on HPC Systems and in Hadoop, Shane Canon, JGI/Argonne HPC Workshop on January, 2010.

2009

1. A Performance Comparison of Massively Parallel Sequence Matching Computations on Cloud Computing Platforms and HPC Clusters, Lavanya, Using Hadoop, in Using clouds for parallel computations in systems biology, Held in conjunction ACM/IEEE SC2009 Conference on High Performance Computing, Networking, Storage and Analysis, Portland Oregon, November 2009.

Magellan Research Publications

1. L. Ramakrishnan, R. Canon, K. Muriki, I. Sakrejda, and N. Wright, Evaluating Interconnect and Virtualization Performance for High Performance Computing. SIGMETRICS Performance Evaluation Review 40, 2 (2012) [Also appeared as paper in 2nd International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS11) held as part of SC11, Seattle, WA, 2011].

2. D. Ghoshal, R. S. Canon and L. Ramakrishnan, "I/O Performance of Virtualized Cloud Environments", The Second International Workshop on Data Intensive Computing in the Clouds (DataCloud-SC11), Held in conjunction with SC—11, Seattle, WA, 2011 [**Best Paper Award**].

3. E. Dede, M. Govindaraju, D. Gunter and L. Ramakrishnan, "Riding the Elephant: Managing Ensembles with Hadoop", The 4th Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS) 2011, Held in conjunction with SC—11, Seattle, WA, 2011.

4. S. Xiao, P. Balaji, Q. Zhu, R. Thakur, S. Coghlan, H. Lin, G. Wen, J. Hong, and W. Feng, "Efficient Migration Capabilities for Virtual GPU Based Computing", Under preparation.

5. S. Xiao, P. Balaji, Q. Zhu, R. Thakur, S. Coghlan, H. Lin, G. Wen, J. Hong, and W. Feng, "Transparent Virtualization of Graphics Processing Units", Under review.

6. Z. Fadika, E. Dede, M. Govindaraj u, and L. Ramakrishnan, Mariane: Mapreduce implementation adapted for hpc environments, Grid 2011: 12th IEEE/ACM Interna- tional Conference on Grid Computing, vol. 0, pp. 18, 2011.

7. Z. Fadika, E. Dede, M. Govindaraju, and L. Ramakrishnan, Benchmarking mapre- duce implementations for application usage scenarios, Grid 2011: 12th IEEE/ACM International Conference on Grid Computing, vol. 0, pp. 18, 2011.

8. L. Ramakrishnan, P. T. Zbiegel, S. Campbell, R. Bradshaw, R. S. Canon, S. Coghlan, I. Sakrejda, N. Desai, T. Declerck, and A. Liu, Magellan: Experiences from a science cloud, in Proceedings of the 2nd international workshop on Scientific cloud computing, ScienceCloud 11, (New York, NY, USA), pp. 4958, ACM, 2011.

9. R. Bradshaw and P. T. Zbiegiel, "Experiences with eucalyptus: deploying an open source cloud", In Proceedings of the 24th international conference on Large installation system administration (LISA'10). USENIX Association, Berkeley, CA, USA, 1-16

10. K. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. Wasser- man, and N. Wright, Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud, in 2nd IEEE International Conference on Cloud Computing Technology and Science, 2010 [**Best Paper Award**].

11. Y. Simmhan, E. Soroush, C. van Ingen, D. Agarwal, and L. Ramakrishnan, Brew: Blackbox resource selection for e-science workflows, in Fifth Workshop on Workflows in Support of Large-Scale Science (WORKS10), 2010.

12. Y. Simmhan and L. Ramakrishnan, Comparison of resource platform selection ap- proaches for scientific workflows, in 1st Workshop on Scientific Cloud Computing, co- located with ACM HPDC 2010 (High Performance Distributed Computing), Chicago, Illinois, 2010.

13. K. Jackson, L. Ramakrishnan, R. Thomas, and K. J. Runge, Seeking supernovae in the clouds: A performance study, in 1st Workshop on Scientific Cloud Computing, co-located with ACM HPDC 2010 (High Performance Distributed Computing), Chicago, IL, June 2010 [**Best Paper Award**].

14. L. Ramakrishnan, K. Jackson, S. Canon, S. Cholia, and J. Shalf, Defining Future Platform Requirements for e-Science Cloud (Position paper), in ACM Symposium on Cloud Computing 2010 (ACM SOCC 2010), Indianapolis, Indiana, June 2010.

Selected Magellan User Presentations and Papers

Here are a list of presentations and papers that resulted from use of Magellan resources.

1. Finding Tropical Cyclones on a Cloud Computing Cluster: Using Parallel Virtualization for Large-Scale Climate Simulation Analysis, D. Hasenkamp, A. Sim, M. Wehner, K. Wu, *Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom2010)*, 2010
2. Finding Tropical Cyclones on Clouds, D. Hasenkamp under the supervision of A. Sim, M. Wehner, K. Wu. *Won Third place in ACM Student Research Poster Competition,Super Computing (New Orleans, LA)*, 2010. .
3. ESG and Cloud Computing with an Experience from Exploring Cloud for Parallelizing Tropical Storm Tracking, *Expedition Workshop, GSA Office of Citizen Services, Intergovernmental Solutions*, Oct. 2010.
4. Interactively exploring data over the Wide Area, J. Insley, M. Hereld, E. Olson, V. Vishwanath, M. Norman, and R. Wagner, *Proceedings of the IEEE Symposium on Large Data Analysis and Visualization (LDAV 2011)*, 2011.
5. Modeling Early Galaxies Using Radiation Hydrodynamics, R. Harkness, D. R. Reynolds, M. L. Norman, R. Wagner, M. Hereld, J. A. Insley, E. C. Olson, M. E. Papka and V. Vishwanath *Proceedings of the IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC 2011)*, 2011.
6. Opportunities and Challenges in Running Scientific Workflows on the Cloud, Yong Zhao, Ioan Raicu, Xubo Fei, and Shiyong Lu, *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC 2011)*, 2011. In press.
7. Resource Sharing Barriers: A Mechanism for Load Balancing Multilevel Parallel Computations, M. H. Arafat, J. Dinan, S. Krishnamoorthy, P. Sadayappan and P. Balaji, *In preparation.*
8. Efficient Inter-node Communication in High Performance GPU Systems, A. Aji, P. Balaji, D. Buntinas, J. Dinan, R. Thakur and W. Feng, *In preparation.*
9. Enabling Fast, Non-contiguous GPU Data Movement in Hybrid MPI+GPU Environments, J. Jenkins, P. Balaji, J. Dinan, N. Samatova, and R. Thakur, *In review.*
10. Efficient Collective Communication in Hybrid MPI+GPU Systems, L. Wesolowski, P. Balaji, J. Dinan, R. Thakur and L. Kale, *In preparation.*
11. Efficient Intra-node Communication in Hybrid MPI+GPU Systems, F. Ji, P. Balaji, J. Dinan, R. Thakur and X. Ma., *In preparation.*
12. Enabling Urgent Computing within the Existing Distributed Computing Infrastructure, Nicholas Dale Trebon, *Ph.D. Dissertation*, University of Chicago, August 2011.
13. Adaptive data placement for Hadoop in non-dedicated distributed envrionment, H. Jin, X. Yang, X. -H. Sun and I. Raicu, *In preparation.*
14. TAU Performance System, S. Shende, *Presentation at 12th DOE ACTS workshop, Berkeley, CA*, Aug 2012.
15. RHIC Real time data reconstruction using Magellan cloud computing, *OSG All Hands, Harvard Medical School*, Boston, March 2011.

16. Offloading peak processing to Virtual Farm by STAR experiment at RHIC, *ACAT 2011, Burnel Uni, London, UK*, September 2011.

17. STAR on the Cloud, *, STAR S&C, Brookhaven National Lab*, May 2011.

18. RHIC Real time data reconstruction using Magellan cloud computing, Jan Balewski, et al., *OSG All Hands Meeting*, March 2011, `https://indico.fnal.gov/contributionDisplay.py?sessionId=1&contribId=11&confId=3627`

19. Offloading peak processing to Virtual Farm by STAR experiment at RHIC, Jan Balewski, et al., *ACAT*, Sept. 2011, `http://indico.cern.ch/contributionDisplay.py?contribId=58&sessionId=6&confId=93877`

20. Wrangler: Virtual Cluster Provisioning for the Cloud, Gideon Juve and Ewa Deelman, Short paper, *Proceedings of the 20th International Symposium on High Performance Distributed Computing (HPDC 2011)*, 2011.

21. Experiences Using Cloud Computing for A Scientific Workflow Application, Jens-S. Vckler, Gideon Juve, Ewa Deelman, Mats Rynge, G. Bruce Berriman, *Proceedings of 2nd Workshop on Scientific Cloud Computing (ScienceCloud 2011)*, 2011.

22. MapReduce implementation of a hybrid spectral library-database search method for large-scale peptide identication. A. Kalyanaraman, W.R. Cannon, B. Latt, D.J. Baxter. *Bioinformatics*, 2011.

23. TSQR, AtA, Cholesky QR, and Iterative Refinement using Hadoop on Magellan, Austin R. Benson, *Class Project Report CS 267, UC Berkeley*, May 2011

Appendix B

Surveys

NERSC Magellan - Cloud Computing Interest Detailed Form

We are collecting additional data from users on their applications to understand the needs in cloud environments. Please fill out the form.

---

* Required

Name *

Email address *

NERSC username

NERSC repo

Title *

Which of the following cloud computing features do you find most attractive. Please check all that apply. *

- Access to additional resources
- Access to on-demand (commercial) paid resources closer to deadlines
- Ability to control software environments specific to my application
- Ability to share setup of software or experiments with collaborators
- Ability to control groups/users
- Exclusive access to the computing resources/ability to schedule independently of other groups/users
- Easier to acquire/operate than a local cluster
- Cost associativity? (i.e., I can get 10 cpus for 1 hr now or 2 cpus for 5 hrs at the same cost)
- MapReduce Programming Model/Hadoop
- Hadoop File System
- User interfaces/Science Gateways: Use of clouds to host science gateways and/or access to cloud resources through science gateways

Please add any other details as to why cloud computing might be attractive to your group.

Application Details

Identify DOE Office and Program of your application *

- Advanced Scienitific Computing Research - Advanced Mathematical Sciences
- Advanced Scientific Computing Research - Computer Sciences
- Biological and Environmental Research - Biology Systems Science
- Biological and Environmental Research - Biological Systems Science -SciDAC
- Biological and Environmental Research - Climate and Environmental Sciences
- Biological and Environmental Research - Climate/Environment - SciDAC
- Basic Energy Sciences - Chemical Sciences
- Basic Energy Sciences - Geosciences
- Basic Energy Sciences - Material Sciences
- Fusion Energy Sciences - Fusion Base Program
- Fusion Energy Sciences - Fusion SciDAC
- High Energy Physics - Accelerator Physics
- High Energy Physics - Astrophysics
- High Energy Physics - Lattice Gauge Theory
- High Energy Physics - High Energy Physics Theory

- Nuclear Physics - Accelarator Physics
- Nuclear Physics - Astrophysics
- Nuclear Physics - Lattice Gauge Theory
- Nuclear Physics - Nuclear Theory
- Advanced Networking Initiative (ANI) Project
- Other:

Identify the category that your application falls under. *
- Dense linear algebra
- Sparse linear algebra
- Spectral methods (FFT)s
- Particle Methods
- Structured Grids
- Unstructured or AMR Grids
- MapReduce
- Other:

How do you manage the data for your application? Please check all that apply *
- Use and expect a parallel file system
- Use and expect local disk
- Large amounts of static data is required at the site where application runs
- Input data arrives from offsite
- Output data needs to be pushed out to a different location

Provide rough orders of magnitude on running time of a single application run *

How many runs do you typically perform and how frequently? *

How many cores do you need for a single run? *

Provide rough orders of magnitude on your memory requirements *

Provide rough orders of magnitude of persistent disk requirements *

What is the typical input data size for a single run of your application? * Please enter a number and units of data size

What is the typical output data size for a single run of your application? * Please enter a number and units of data size

What are bandwidth requirements in or out of NERSC? *

Does your application require a specific operating system or version of the operating system? Please specify *

Does your application require specific kernel versions or modules? Please specify *

Does your application require specific libraries? Please specify. *

Would you like to specify any additional details about the application?

Use of Cloud Computing

Describe how many CPU hours you will need for validating cloud computing to be a valid platform choice (proof of concept runs)? *

Describe how many CPU hours you will need for using cloud computing on an ongoing basis? *

Virtual machines are a popular mode of operation in cloud computing that tends to affect some applications in terms of performance. Will this affect your application. Please specify *

- Yes, I anticipate it will run slower but the other features of cloud computing make it more attractive to me
- Yes, it can be a problem
- No
- Maybe
- Don't know

Cloud computing relies on pre-configured images for the OS and software. The nodes do not have any persistent state. Check all that apply that affects your application. *

- My code-base changes often and I need to recompile my code periodically.
- The data is updated often
- I need to save large amounts of data from each run

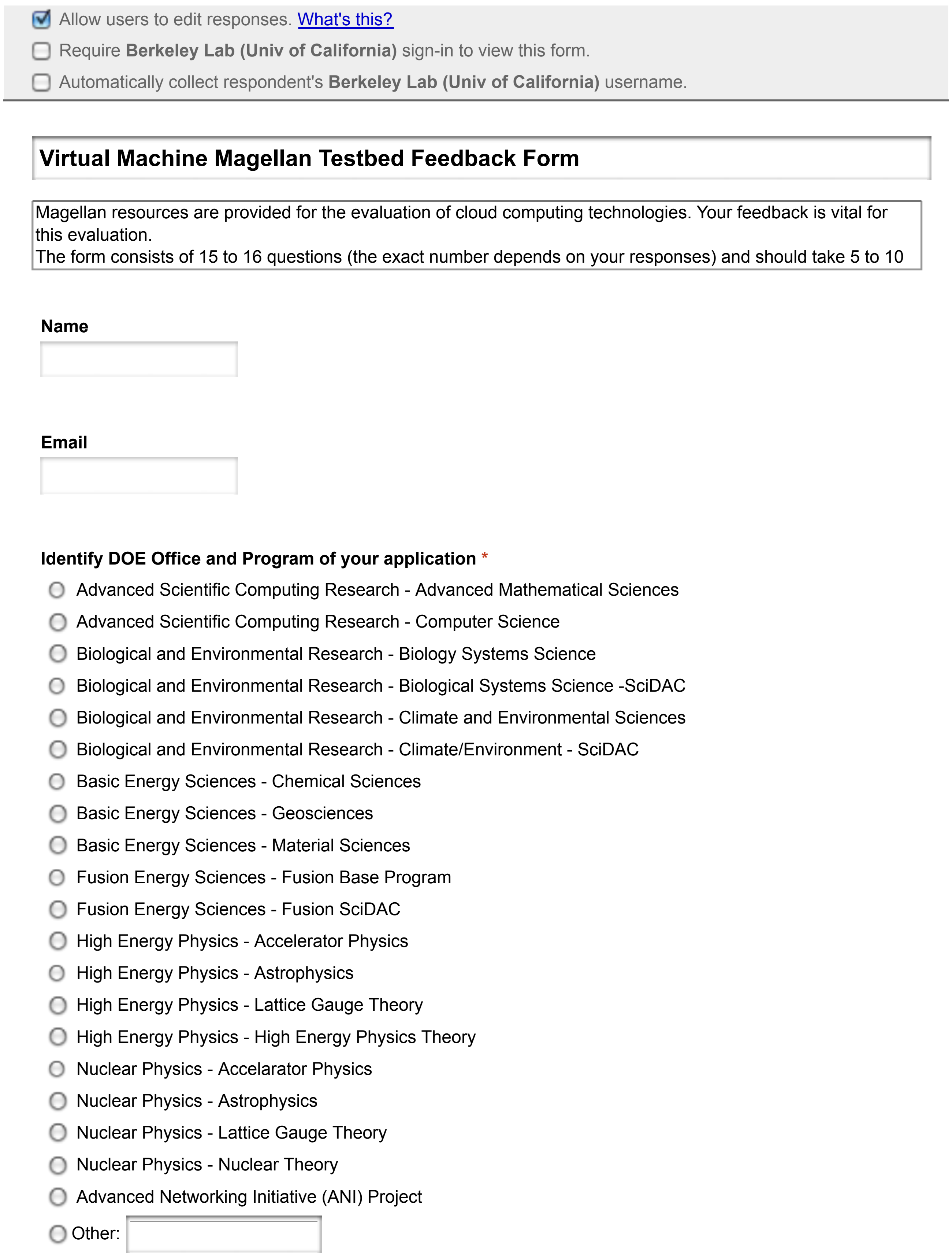
Will there be other users or collaborators who can use your cloud setup to run their experiments? *

- Yes
- No
- Maybe

Are there others in your scientific community interested in or looking at cloud computing? *

- Yes
- No

Additional comments.



[✓] Allow users to edit responses. What's this?

☐ Require **Berkeley Lab (Univ of California)** sign-in to view this form.

☐ Automatically collect respondent's **Berkeley Lab (Univ of California)** username.

Virtual Machine Magellan Testbed Feedback Form

Magellan resources are provided for the evaluation of cloud computing technologies. Your feedback is vital for this evaluation.
The form consists of 15 to 16 questions (the exact number depends on your responses) and should take 5 to 10

Name

Email

Identify DOE Office and Program of your application *

- ○ Advanced Scientific Computing Research - Advanced Mathematical Sciences
- ○ Advanced Scientific Computing Research - Computer Science
- ○ Biological and Environmental Research - Biology Systems Science
- ○ Biological and Environmental Research - Biological Systems Science -SciDAC
- ○ Biological and Environmental Research - Climate and Environmental Sciences
- ○ Biological and Environmental Research - Climate/Environment - SciDAC
- ○ Basic Energy Sciences - Chemical Sciences
- ○ Basic Energy Sciences - Geosciences
- ○ Basic Energy Sciences - Material Sciences
- ○ Fusion Energy Sciences - Fusion Base Program
- ○ Fusion Energy Sciences - Fusion SciDAC
- ○ High Energy Physics - Accelerator Physics
- ○ High Energy Physics - Astrophysics
- ○ High Energy Physics - Lattice Gauge Theory
- ○ High Energy Physics - High Energy Physics Theory
- ○ Nuclear Physics - Accelarator Physics
- ○ Nuclear Physics - Astrophysics
- ○ Nuclear Physics - Lattice Gauge Theory
- ○ Nuclear Physics - Nuclear Theory
- ○ Advanced Networking Initiative (ANI) Project
- ○ Other:

Which Magellan sites did you use? *

- ☐ NERSC Eucalyptus
- ☐ Argonne Eucalyptus
- ☐ Argonne OpenStack

What was your overall experience with the cloud solution? *

If you used more than one of the cloud solutions, answer for your most recent usage.

- ○ Excellent, worked seamlessly
- ○ Good, there were some problems but we could overcome them
- ○ Medium, there were some problems we were able to overcome and some that are still an issue
- ○ Poor, was able to use but there are some challenges that we were unable to overcome that is necessary for long term use
- ○ Unable to use, unable to overcome some fundamental challenges

Would you use a cloud system again? *

- ○ Yes
- ○ No
- ○ Maybe

How did the performance on the cloud testbed compare with other systems you run your application on? *

- ○ Much better performance
- ○ Little better performance but not significant
- ○ Similar
- ○ Slightly less but not significant
- ○ Significantly less performance

Did you have previous experience (before Magellan) with cloud infrastructure or software? *

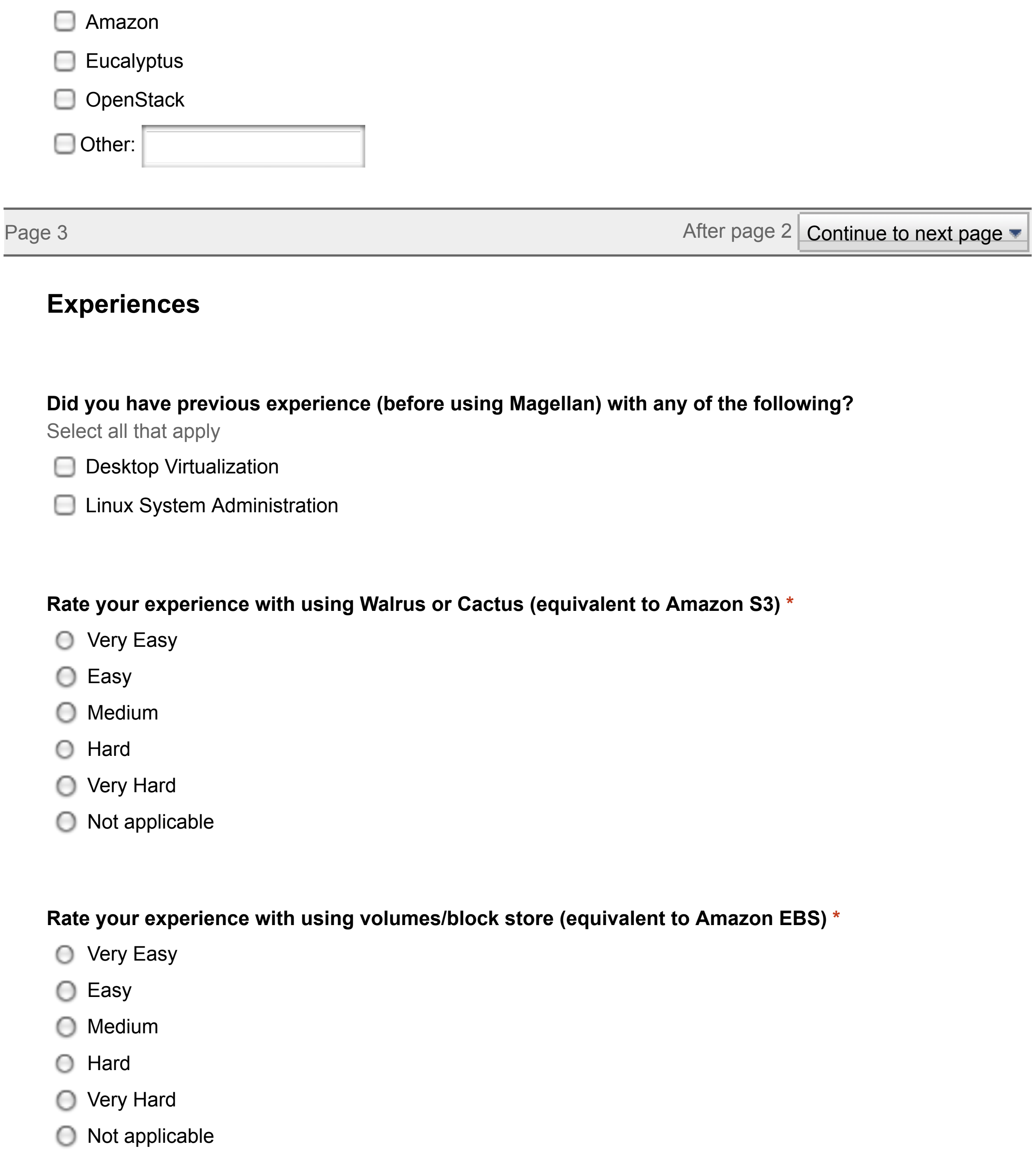
- ○ Yes
- ○ No

Page 2 After page 1 Continue to next page

Note: "Go to page" selections will override this navigation. Learn more.

Previous Cloud Experience

If you answered yes to the above question, which ones? *



☐ Amazon

☐ Eucalyptus

☐ OpenStack

☐ Other:

Page 3 After page 2 Continue to next page

Experiences

Did you have previous experience (before using Magellan) with any of the following?

Select all that apply

☐ Desktop Virtualization

☐ Linux System Administration

Rate your experience with using Walrus or Cactus (equivalent to Amazon S3) *

○ Very Easy

○ Easy

○ Medium

○ Hard

○ Very Hard

○ Not applicable

Rate your experience with using volumes/block store (equivalent to Amazon EBS) *

○ Very Easy

○ Easy

○ Medium

○ Hard

○ Very Hard

○ Not applicable

Rate your experience with creating new images *

- ○ Very Easy
- ○ Easy
- ○ Medium
- ○ Hard
- ○ Very Hard
- ○ Not applicable

Page 4 After page 3 Continue to next page

Application Design

Rate your experience with data management and movement on Magellan *

- ○ Very Easy
- ○ Easy
- ○ Medium
- ○ Hard
- ○ Very Hard

How difficult was it to manage your workflow?

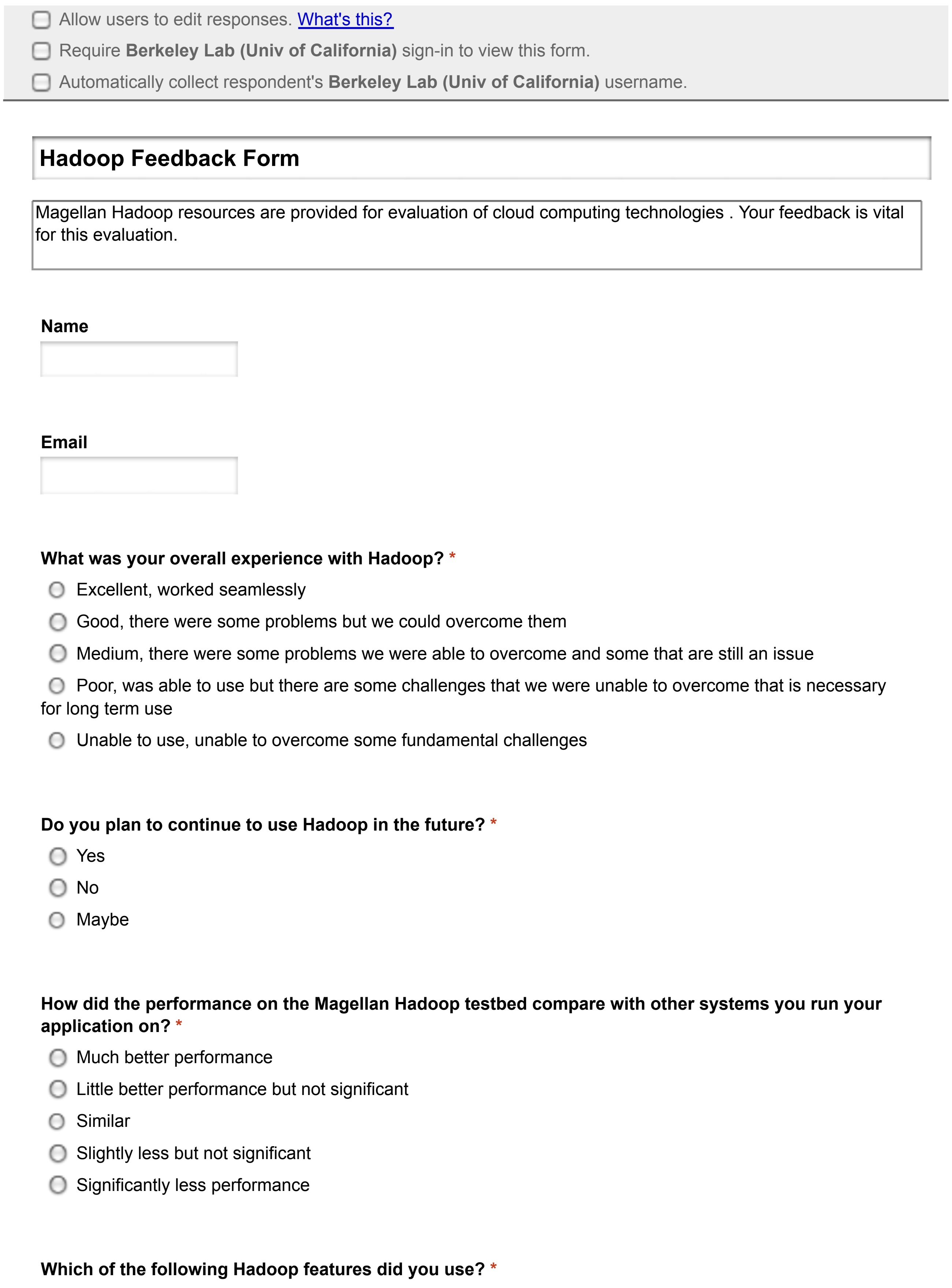
- ○ Very Easy
- ○ Easy
- ○ Medium
- ○ Hard
- ○ Very Hard

How important was user support/system adminstration to your effort? *

- ○ Not needed at all, I was able to do everything on my own
- ○ A little help was needed, but overall I was able to manage on my own
- ○ Moderate help was needed, it would have been possible but difficult to manage without help from the sites
- ○ Extensive help was needed, I could not have done it without help from user support/system adminstration

Please provide any other feedback on using Magellan virtual machines

Please list any publications/reports/presentations resulting from use of Magellan resources



☐ Allow users to edit responses. What's this?

☐ Require **Berkeley Lab (Univ of California)** sign-in to view this form.

☐ Automatically collect respondent's **Berkeley Lab (Univ of California)** username.

Hadoop Feedback Form

Magellan Hadoop resources are provided for evaluation of cloud computing technologies . Your feedback is vital for this evaluation.

Name

Email

What was your overall experience with Hadoop? *

○ Excellent, worked seamlessly

○ Good, there were some problems but we could overcome them

○ Medium, there were some problems we were able to overcome and some that are still an issue

○ Poor, was able to use but there are some challenges that we were unable to overcome that is necessary for long term use

○ Unable to use, unable to overcome some fundamental challenges

Do you plan to continue to use Hadoop in the future? *

○ Yes

○ No

○ Maybe

How did the performance on the Magellan Hadoop testbed compare with other systems you run your application on? *

○ Much better performance

○ Little better performance but not significant

○ Similar

○ Slightly less but not significant

○ Significantly less performance

Which of the following Hadoop features did you use? *

- ☐ Hadoop Streaming
- ☐ Hadoop Native Programming in Java
- ☐ Hadoop Pipes
- ☐ Pig
- ☐ HBase
- ☐ Hadoop with GPFS
- ☐ Other:

Do you have access to other Hadoop clusters? If yes, please specify details

Did you have previous experiencce with Hadoop before using Magellan Hadoop resources? *

- ○ Yes
- ○ No

What was your experience using HDFS? *

- ○ Very Easy
- ○ Easy
- ○ Medium
- ○ Hard
- ○ Very Hard

How difficult was it to manage your workflow in Hadoop? *

- ○ Very easy
- ○ Easy
- ○ Medium
- ○ Hard
- ○ Very hard

How important was user support/system administration to your effort?

- ○ Not needed at all, I was able to do everything on my own
- ○ A little help was needed but overall I was able to manage on my own
- ○ Moderate help was needed, it would have been possible but difficult to manage without help from the sites

○ Extensive help was needed, I could not have done it without help from user support/system adminstrator help

Please provide any other feedback on using Magellan Hadoop?

Please list any publications/reports/presentations resulting from use of Magellan Hadoop resources