ALCC Final Report

PROJECT: HPC Colony II
LEAD PI: Terry Jones <trj@ornl.gov>

Project Abstract

HPC Colony II has been a 36-month project focused on providing ***portable performance*** for leadership class machines—a task made difficult by the emerging variety of more complex computer architectures. The project investigated moving the burden of portable performance to ***adaptive system software***, thereby allowing domain scientists to concentrate on their field rather than the fine details of a new leadership class machine.

To accomplish our goals, we focused on adding intelligence into the system software stack. Our revised components include: new techniques to address OS jitter; new techniques to dynamically address load imbalances; new techniques to map resources according to architectural subtleties and application dynamic behavior; new techniques to dramatically improve the performance of checkpoint-restart; and new techniques to address membership service issues at scale.

In keeping with these goals, our ALCC objectives focused on demonstrating new levels of portability and performance/scalability on experiments consisting of extreme core counts. All Colony2 methodologies are designed to scale up to at least one million cores, and the unique resources made available through the ALCC program provided a matchless tool to test our designs.

Our primary experiment platform was Oak Ridge's Cray XK6/XK7 machine. The Oak Ridge machine offers very large core counts which is a critical component to our testing. Our secondary resource was Argonne's 160K core BlueGene/P machine; the remaining portion of our allocation was consumed on it. The BGP features multiple communication topologies (which is an important component to our load-balancing testing) and was used for both scalability/production runs, and for preparation/development runs.

Award Details

Duration:..... 1 year
OLCF processor hours: ..... 3 million
OLCF Amount remaining: 0
ALCF processor hours: ..... 3 million
ALCF Amount remaining: 0

Research Results

Performance of Optimized Message-Logging Protocol
Area: Fault Tolerance

The goal of this topic area is to measure the performance improvement of an optimized message-logging protocol. A major source of performance penalization of most message-logging protocols is the use of determinants. These bits of information are necessary to provide a correct recovery from a failure. During normal execution, the message-logging protocol creates, stores and sends determinants. The combine cost of all those operations varies from application to application, but it may be as high as 20% in some situations. Therefore, a strategy that avoids determinants is desirable to keep message-logging as an alternative to provide fault tolerance in the future. A new strategy that avoids the use of determinants uses high-level information from the programming language. In some cases, it is possible to avoid the creation of determinants altogether, removing a high percentage of the performance penalization of message-logging.

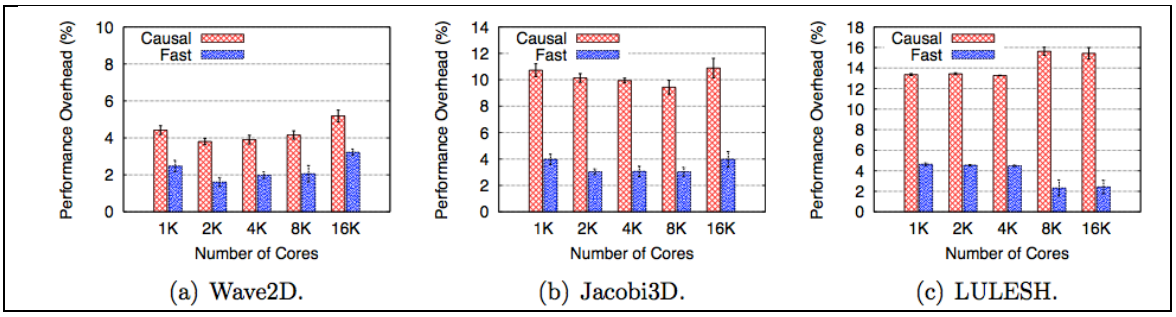

Figure 2: Causal versus Fast message-logging strategies for three applications.

The experiments run on Intrepid revealed that the new protocol reduces the performance overhead by a big margin, compared to a traditional message-logging protocol. A collection of three iterative stencil programs were used in the experiments (Jacobi3D, Wave2D and LULESH). These programs differ in the amount of computation per iteration and the communication pattern. The optimized protocol reduced the performance overhead in these applications more than 50%, 66%, 75%, respectively, compared to a traditional message-logging protocol. After that reduction, the performance overhead of the protocol is lower than 4% for all the applications examined.

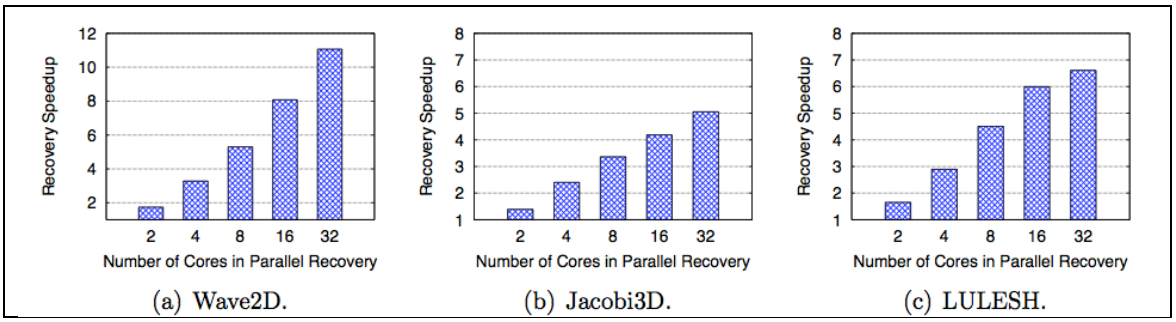

Figure 3: Recovery speedups made possible by parallel recovery.

Additionally, this new protocol was extended with the parallel recovery strategy of Charm++ to accelerate recovery by a factor of 10, 5, and 6 in Jacobi3D, Wave2D, and LULESH, respectively. Overall, the new strategy has a low overhead and provides a competitive strategy to provide resilience at exascale.

Publication Plan: these results were incorporated into the PhD thesis of Esteban Meneses and are publicly available through the library system of the University of Illinois.

Improved Clock Synchronization Protocol
Area: Coordinated Kernel Scheduling

A second topic area of our work was to investigate coordinated scheduling for machines without hardware support for global synchronized clock (e.g. Cray XT5, XK6, and XK7 architectures). We developed an improved software-based clock synchronization scheme that provides high precision time agreement among distributed memory nodes. The technique is designed to minimize variance from a reference chimer during runtime and with minimal time-request latency. Our scheme permits initial unbounded variations in time and corrects both slow and fast chimers (clock skew). An implementation developed within the context of the MPI message passing interface was designed, and time coordination measurements are presented below. To investigate our design, we began by measuring how nine nodes vary from a reference source when only NTP is employed. Figure 4 presents data for uncorrected variance and variance corrected with a linear fit.

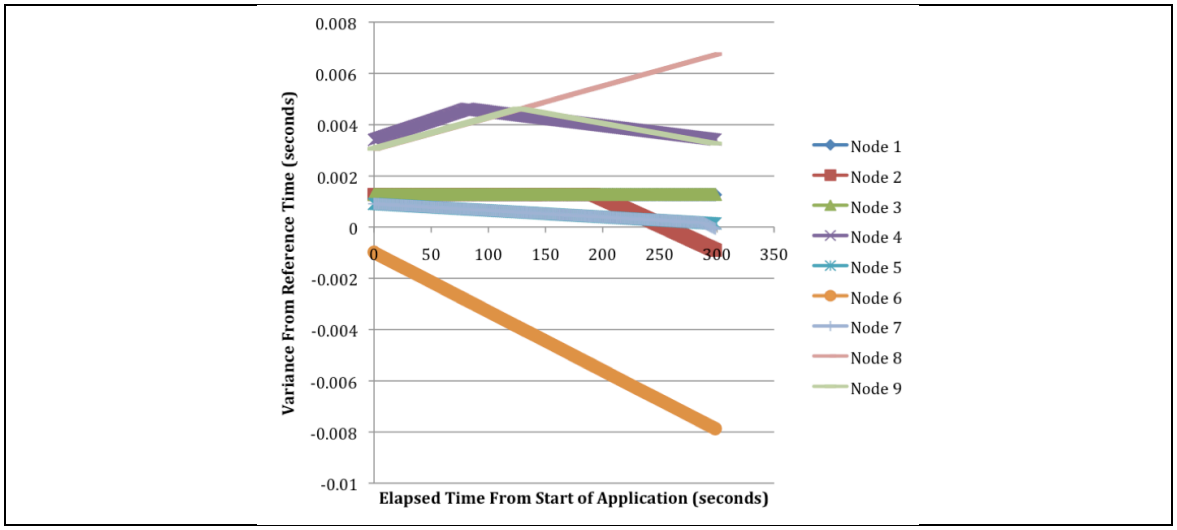

Figure 4: Time coordination without Synchronization Improvements

Given that the mean time variance for even a small set of nodes can reach 20.0 milliseconds under standard Network Time Protocol (NTP), the Colony project designed a new software-based synchronization protocol suitable for high performance computing environments. Several studies have sought to quantify the magnitude of scalability issues associated with operating system noise[1] [2]. For example, the Tau team at the University of Oregon has reported 23% to 32% increase in runtime for parallel applications running at 1024 nodes and 1.6% operating system noise. More recently, Ferreira et al. confirmed that a 1000 Hz 25μs noise interference (an amount measured on a large-scale commodity Linux cluster) can cause a 30% slowdown in application performance on ten thousand nodes. By tightly synchronizing the clocks on the compute nodes, it is possible to extend the system software to support co-scheduling (an effective technique to reduce the effects of noise on a parallel computation).

---

[1] T. Hoefler, T. Schneider, and A. Lumsdaine. Characterizing the Influence of System Noise on Large-Scale Applications by Simulation. *In International Conference for High Performance Computing, Networking, Storage and Analysis (SC'10)*, Nov. 2010.

[2] Terry Jones, Shawn Dawson, Rob Neely, William Tuel, Larry Brenner, Jeff Fier, Robert Blackmore, Pat Caffrey, Brian Maskell, Paul Tomlinson, and Mark Roberts, *Improving the Scalability of Parallel Jobs by adding Parallel Awareness to the Operating System*. Proceedings of Supercomputing 2003, Phoenix, AZ, November 2003.

With our ALCC allocation, the Colony team investigated a new point-to-point synchronization protocol (our previous methods required collective operations). Figure 5 shows a histogram of results for the new synchronization protocol on 20,480 Titan processors.

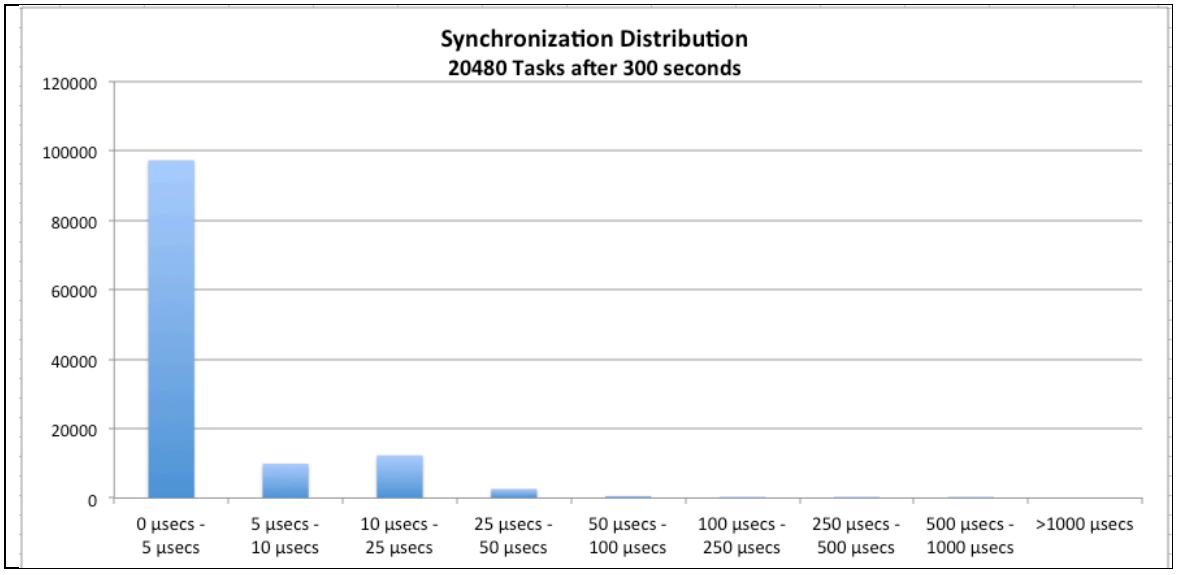

Figure 5: Time synchronization with Coordination improvements after 5 minutes. These figures represent the variance for synchronizing up to 20,480 nodes.

Publication Plan: these results are currently being written for publication, the venue is tbd.

Performance Evaluation of Meta Balancer
Area: Load Balance

Performance of an application is affected by load imbalance. Therefore load balancing is required to scale an application but performing load balancing incurs a cost. If the cost of load balancing is more than the benefit obtained from load balancing, it degrades the performance further. Meta-Balancer framework, implemented in Charm++, automatically identifies a load balancing period based on the application characteristics and cost of load balancing. Meta-Balancer has been previously shown to perform well up to 4096 cores on Jaguar. In this project we show the benefits of Meta-Balancer on up to 131,072 cores of Intrepid, BlueGene/P.

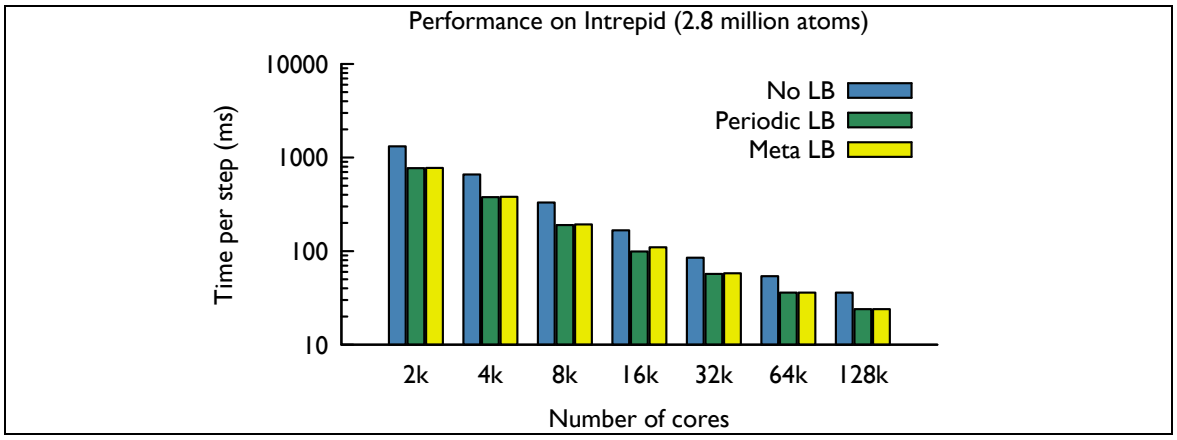

Figure 6: New Smart Runtime System Load Balancer

The performance of Meta-Balancer was evaluated on LeanMD benchmark and the results are plotted in Figure 1 above. LeanMD is a molecular dynamics simulation program written in Charm++. It simulates the behavior of atoms based on the Lennard-Jones potential. Load imbalance in LeanMD is due to the variation in number of atoms per cell as well as the slow migration of atoms. LeanMD was run for a system of 2.8 million atoms for 2000 iterations from 2048 to 131072 cores. The load balancing strategies used were GreedyLB for runs till 8k cores and HybridLB for larger runs. The HybridLB is a hierarchical strategy which uses GreedyLB and RefineLB strategies. We experimented with a range of hand tuned load balancing period. We find that if load balancing is done frequently, then the overhead of load balancing is more than the benefit but if load balancing is performed infrequently, then the performance is affected by load imbalance. Meta-Balancer is able to automatically identify optimal load balancing period without any input from the user. It is also able to scale to 131072 cores without any performance bottleneck.

Publication Plan: We are planning to submit this work to either JPDC or ACM TOPC journal.

Other Colony2 ALCC Highlights:

a. HPC Colony members are undertaking a post-project activity to pursue getting coordinated-scheduling advances adopted by HPC vendors. In addition, IBM is evaluating SpiderCast advances for possible HPC products.

b. Recent developments in Charm++ fault tolerance infrastructure permits to run Charm++ programs on top of an MPI library and simulate rank failures. This mechanism exports a function to the user to kill a rank. Using this technique, new fault tolerance methods and algorithms can be developed on top of Charm++. The approach scales up to 32K cores on BG/P and provides an almost-negligible restart time.

c. An April full-machine Jaguar test of a new Charm++ implementation provided impressive performance gains over an earlier version. The new version features a new network layer implementation designed for Cray's Gemini interconnect. Performance for a 100M atom NAMD run (PME every 4 steps) improved from a 26 milliseconds per step runtime over last year's MPI over SeaStar+ numbers, to a 13 milliseconds per step runtime with the new software and hardware.

d. We conducted research on scalable membership, attribute, monitoring and communication services that will enable sophisticated applications and general purpose cluster computing on high-performance computing systems with a very large numbers of processors. The SpiderCast system, that provides these services, is based on overlay and peer-to-peer technologies. SpiderCast will, on the one hand, utilize the unique architecture and networking features of Blue Gene/P [BGP08] to achieve top performance, and on the other hand, will develop broad scalable technologies for systems with hundreds of thousands of processors, which can be deployed on general cluster systems. Large scale experiments were conducted on the Blue Gene/P platform in the IBM Watson Research Center.

Project Productivity

Primary

Publications –

- **Jones**, Terry. ***Linux Kernel Co-Scheduling and Bulk Synchronous Parallelism***. International Journal of High Performance Computing Applications (IJHPCA). Vol. 26, Issue 2, May 2012.
- **Jones**, Terry, & **Koenig**, Gregory. ***Clock Synchronization in High-end Computing Environments: A Strategy for Minimizing Clock Variance at Runtime***. Concurrency and Computation: Practice and Experience. Journal of Concurrency and Computation: Practice and Experience (J CCPE), Vol. 25, Issue 6, DOI: 10.1002/cpe.2868, pages 881-897, April 25, 2013.
- **Langer,** Akhil**, Venkataraman,** Ramprasad**,** & **Kále**, Laxmikant. ***Scalable Algorithms for Constructing Balanced Spanning Trees on System-ranked Process Groups***. In 19th European MPI Users' Group Meeting (EuroMPI 2012), Vienna, Austria, September 23-26, 2012.
- **Lifflander**, Jonathan, **Miller,** Phil**, Ramprasad** Venkataraman**, Anshu** Arya**, Terry** Jones & **Kále**, Laxmikant**.** ***Mapping Dense LU Factorization on Multicore Supercomputer Nodes***. In Proceedings of 26th IEEE International Parallel and Distributed Processing Symposium (IPDPS). May-2012. Shanghai, China.
- **Mendes**, Celso L., **Kále**, Laxmikant, **Rodrigues**, Eduardo R., & **Panetta**, Jairo. ***Adaptive and Dynamic Load Balancing for Weather Forecasting Models***. In Annual meeting of the Cray Users Group (CUG) 2012. May 2012. Stuttgart, Germany.
- **Meneses**, Esteban. ***Scalable Message-Logging Techniques for Effective Fault Tolerance in HPC Applications.*** PhD diss., University of Illinois at Urbana-Champaign, 2013.
- **Meneses**, Esteban, **Ni**, Xiang, & **Kále**, Laxmikant. ***A Message-Logging Protocol for Multicore Systems*.** In Proceedings of the 2nd Workshop on Fault-Tolerance for HPC at Extreme Scale (FTXS 2012). Jun-2012. Boston, MA.
- **Meneses**, Esteban, **Sarood**, Osman, & **Kále**, Laxmikant. ***Accessing Energy Efficiency of Fault Tolerance Protocols for HPC Systems.*** In Proceedings of 24nd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD). Oct. 2012. New York City, NY.
- **Menon**, Harshitha, **Acun**, Bilge, **Garcia**, Simon, **Sarood**, Osman, & **Kále**, Laxmikant. ***Thermal Aware Automated Load Balancing for HPC Applications.*** In Cluster Computing (CLUSTER), 2013 IEEE International Conference on. IEEE, 2013.
- **Menon**, Harshitha, **Jan**, Nikhil, **Zheng**, Gengbin, & **Kále**, Laxmikant. ***Automated Load Balancing Invocation based on Application Characteristics.*** In Cluster Computing (CLUSTER), 2012 IEEE International Conference on. IEEE, 2012.
- **Menon**, Harshitha, & **Kále**, Laxmikant. ***A Distributed and Dynamic Load Balancer for Iterative Applications***. In Proceedings of 2013 International Conference for High

Performance Computing, Networking, Storage and Analysis (SC'13). Nov. 2013. Denver, CO.

- **Ni**, Xiang, **Meneses**, Esteban, & **Kále**, Laxmikant. ***ACR: Automatic Checkpoint/Restart for Soft and Hard Error Protection***. In Proceedings of 2013 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'13). Nov. 2013. Denver, CO.
- **Ni**, Xiang, **Meneses**, Esteban, & **Kále**, Laxmikant. *Hiding Checkpoint Overhead in HPC Applications with a Semi-Blocking Algorithm*. In Cluster Computing (CLUSTER), 2012 IEEE International Conference on, pp. 364-372. IEEE, September, 2012. Beijing, China.
- **Sarood**, Osman, **Meneses**, Esteban, & **Kále**, Laxmikant. ***A 'Cool' Way of Improving the Reliability of HPC Machines.*** In Proceedings of 2013 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'13). Nov. 2013. Denver, CO.
- **Sun**, Yanhua, **Zheng**, Gengbin, **Mei**, Chao, **Bohm**, Eric J., **Kále**, Laxmikant, **Philips**, James C., & **Jones**, Terry. ***Optimizing Fine-grained Communication in a Biomolecular Simulation Application On Hybrid Cray XK6 System***. In Proceedings of 2012 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'12). Nov. 2012. Salt Lake City, UT.
- **Sun**, Yanhua, **Zheng**, Gengbin, **Kále**, **Laxmikant**, Jones, **Terry** R., & **Olson**, Ryan. *A uGNI-based Asynchronous Message-driven Runtime System for Cray Supercomputers with Gemini Interconnect*. In Proceedings of 26th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2012). May-2012. Shanghai, China.
- **Tock**, Yoav, **Mandler**, Benjamin, **Moreira**, José, and **Jones**, Terry. ***Design and Implementation of a Scalable Membership Service for Supercomputer Resiliency-Aware Applications***. Lecture Notes in Computer Science Volume 8097, 2013, pp. 354-366. DOI http://10.1007/978-3-642-40047-6 37. Print ISBN 978-3-642-40046-9.
- **Tock**, Yoav, **Mandler**, Benjamin, **Moreira**, José, and **Jones**, Terry. ***Design and Implementation of a Scalable Membership Service for Supercomputer Resiliency-Aware Applications***. Euro-Par 2013 Parallel Processing: 19th International Euro-Par Conference, Aachen, Germany, August 26-30, 2013: Proceedings. Springer, 2013.
- **Yu**, Li, **Zheng**, Ziming, **Lan**, Zhiling, **Jones**, Terry, **Brandt**, Jim, & **Gentile**, Ann. ***Filtering log data: finding the needles in the haystack.*** In 42nd IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012). Jun-2012. Boston, MA.
- **Zheng**, Gengbin, **Ni**, Xiang, & **Kále**, Laxmikant. ***A Scalable Double In-memory Checkpoint and Restart Scheme towards Exascale.*** In Proceedings of the 2nd Workshop on Fault-Tolerance for HPC at Extreme Scale (FTXS 2012). June-2012. Boston, MA.
- **Zheng**, Ziming, **Lan**, Zhiling, **Yu**, Li, & **Jones**, Terry. ***3-Dimensional Root Cause Diagnosis via Co-analysis.*** In Proceedings of 9th International Conference on Autonomic Computing (ICAC). Sep-2012. San Jose, CA.

- Presentations –
  - Celso L. Mendes, Laxmikant V. Kale, Eduardo R.Rodrigues, Jairo Panetta. *Adaptive and Dynamic Load Balancing for Weather Forecasting Models*. In Annual meeting of the Cray Users Group (CUG) 2012. May 2012. Stuttgart, Germany.
  - Esteban Meneses, Xiang Ni and L. V. Kale. *A Message-Logging Protocol for Multicore Systems*. In Proceedings of the 2nd Workshop on Fault-Tolerance for HPC at Extreme Scale (FTXS 2012). Jun-2012. Boston, MA.
  - Gengbin Zheng, Xiang Ni and Laxmikant V. Kale. *A Scalable Double In-memory Checkpoint and Restart Scheme towards Exascale.* In Proceedings of the 2nd Workshop on Fault-Tolerance for HPC at Extreme Scale (FTXS 2012). June-2012. Boston, MA.
  - Jonathan Lifflander, Phil Miller, Ramprasad Venkataraman, Anshu Arya, Terry Jones and Laxmikant Kale. *Mapping Dense LU Factorization on Multicore Supercomputer Nodes.* In Proceedings of 26th IEEE International Parallel and Distributed Processing Symposium (IPDPS). May-2012. Shanghai, China.
  - Li Yu, Ziming Zheng, Zhiling Lan, Terry Jones, Jim Brandt, and Ann Gentile. *Filtering log data: finding the needles in the haystack.* In 42nd IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012). Jun-2012. Boston, MA.
  - Yanhua Sun, Gengbin Zheng, L. V. Kale, Terry R. Jones and Ryan Olson. *A uGNI-based Asynchronous Message-driven Runtime System for Cray Supercomputers with Gemini Interconnect*. In Proceedings of 26th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2012). May-2012. Shanghai, China.
  - Ziming Zheng, Zhiling Lan, Li Yu and Terry Jones. *3-Dimensional Root Cause Diagnosis via Co-analysis*. In Proceedings of 9th International Conference on Autonomic Computing (ICAC). Sep-2012. San Jose, CA.
  - Terry Jones. HPC Colony – Unconstrained Performance at Exascale. Exascale Research Conference. Oct-2012. Arlington, VA.

- Awards
  a. winner of HPC Challenge Award at SC' 2011
  b. IEEE Computer Society Sidney Fernbach Award: Laxmikant Kale

- Software Products

Software title	Current Version	Brief Description	Date of Last Release
Hierarchical load balancer module	Charm++ 6.5.0	plug-in	3/13/13
In-memory checkpoint auto-restart module	Charm++ 6.5.0	enhanced features	3/13/13
Team based load balancer	Charm++ 6.5.0		3/13/13
Causal message-logging module	Charm++ 6.5.0	newly created	3/13/13
Hi-Precision Synch-ronized Global Clocks	OpenMPI 1.4.4	high precision global synchronized clocks	5/20/12
Parallel Coordinated Scheduling	Linux 2.6.32.59	gives Linux kernel parallel awareness for coordinated scheduling	5/1/11
Spider Cast	SpiderCastCPP 1.0	A C++ scalable implementation that provides a membership service and group communication services for HPC environments.	5/1/12

Notes:

- In addition to our on-team involvement with IBM, we are working with Cray to ensure our work on ORNL's Titan machine results in a commercially available technology. This work is being funded by ORNL and includes close involvement with both HPC vendors. A pathforward plan has been developed to release coordinated scheduling technology for future machines.
- Some parts of this research have been incorporated to the public distribution of the Charm++ software infrastructure, which is available in both source and binary formats. In particular, a new release of Charm++ (v.6.5.0) was made available recently, through the Charm++ download website: http://charm.cs.uiuc.edu/software/
- SpiderCast is currently identified by IBM as an internal asset. As such, it is a candidate for inclusion in some IBM products and/or continued development of advanced features.

Center Feedback

- Please answer as applicable: Has the support received from the following been beneficial to your project team? Cite examples if possible
  - User Assistance Center
    - Utilized for new login ids and early access questions
    - All interactions were seen as very favorable from our viewpoint
  - Scientific Computing Group
    - Due to the nature of our research, we did not interact with the Scientific Computing Group
  - Visualization and Analysis Team
    - Due to the nature of our research, we did not interact with the Visualization and Analysis Team
- Any additional feedback from your project team for the centers?

Code Description and Characterization

Approach

Much of the computational infrastructure that we develop is written in C++, and is part of the Charm++ software distribution, available from http://charm.cs.uiuc.edu. Charm++ source is freely available under a simple free non-commercial-use license. For the applications that we have utilized (mainly NAMD), the dominant language is also C++, although there are pieces written in C due to legacy reasons. Some of these applications may benefit from specialized libraries (e.g. FFT routines), but this is not essential for the tests to be conducted.

Pattern of usage

Our typical pattern of usage on Leadership Computing Facilities is to conduct short runs on a large number of cores. These runs are repeated extensively, to test different combinations of settings and parameters, but each individual run typically consists of just a few iterations or timesteps of the full-scale application. This mode is generally sufficient to assess performance of critical parts in the application, and to test possible ways to improve performance. However, since our focus is scalable performance, via automated intelligent runtime, we need to run the tests and strategies on as large a fraction of the system as is feasible. For these experiments, we used OLCF's reservation system to request whole machine access. Weekly runs allow us time to analyze data, make improvements, and from time to time, design new strategies, before running experiments again. We believe that such allocations to "sharpen the

saw" by improving scaling across a wide range of future applications will optimize the usage of LCF resources in the future. In effect, our allocation will pay for itself through improvements in efficiency of production runs.

Job Characterization

Our efforts have focused on extending its scalability from 32K to 250K+ cores. Besides the scaling tests, we will also utilized a portion of our allocation (approximately 25% of the resource for scaling tests) to prepare and debug our software for the scaling runs. These preparation/development runs did not need dedicated access to the machine. Scaling tests were performed on Charm++ and the Colony Linux kernel.

- If possible and useful, please indicate which of the following algorithmic motifs appear in each of your major production codes.

Code Name	**Dense Linear Algebra**	**Sparse Linear Algebra**	**Monte Carlo**	**FFTs**	**Particles**	**Structured Grids**	**Unstructured Grids**	**AMR**
	⃠	⃠	⃠	⃠	⃠	⃠	⃠	⃠