Conf-950985--3

UCRL-JC-120489
PREPRINT

Management Issues for High Performance Storage Systems

S. Louis
R. Burris

This paper was prepared for submittal to the
14th IEEE Symposium on Mass Storage Systems
Monterey, CA
September 11-14, 1995

March 1995

This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

RECEIVED

JUL 18 1995

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Management Issues for High Performance Storage Systems

Steve Louis

Lawrence Livermore National Laboratory
Livermore, California

Randy Burris

Oak Ridge National Laboratory
Oak Ridge, Tennessee

Abstract

Managing distributed high-performance storage systems is complex and, although sharing common ground with traditional network and systems management, presents unique storage-related issues. Integration technologies and frameworks exist to help manage distributed network and system environments. Industry-driven consortia provide open forums where vendors and users cooperate to leverage solutions. But these new approaches to open management fall short addressing the needs of scalable, distributed storage. We discuss the motivation and requirements for storage system management (SSM) capabilities and describe how SSM manages distributed servers and storage resource objects in the High Performance Storage System (HPSS), a new storage facility for data-intensive applications and large-scale computing. Modern storage systems, such as HPSS, require many SSM capabilities, including server and resource configuration control, performance monitoring, quality of service, flexible policies, file migration, file repacking, accounting, and quotas. We present results of initial HPSS SSM development, including design decisions and implementation trade-offs. We conclude with plans for follow-on work and provide storage-related recommendations for vendors and standards groups seeking enterprise-wide management solutions.

The management problem

Effective management of new information technologies, whether for high-performance computers, for the network, for the desktop, or for storage, cuts to the heart of today's corporate concerns. Rapid technology change, decreased budgets, and increased demand for reliability present new management challenges in all fields of computing. Hardware and software technology vendors must now provide efficient management solutions alongside their products to help meet these challenges.

This push to provide better management has generated an evolutionary advancement from purely reactive approaches to more proactive ones, sometimes utilizing knowledge-based expert systems. Today's organizations are also in the midst of a radical change in their computing and information infrastructures, from older, mainframe-based, highly centralized resources to new network-based, open distributed systems with many management perspectives. These trends lead to a new *enterprise-wide* view of management.

An enterprise-wide view

Initially, management applications and tools focused on networks and components on networks, such as hubs, bridges and routers. Next came management of host systems, predominantly mainframes. Management of applications running on mainframe systems followed. The subsequent transition to more open heterogeneous environments led to distributed applications consisting of multiple cooperating processes running on several nodes. The implementation of such distributed technologies is difficult, and managing them is daunting.

Formerly commonplace management activities such as event handling, centralized logging, performance monitoring, security, accounting, configuration control, and workload scheduling must now be handled across an entire enterprise using a highly distributed infrastructure. Increased user autonomy in these environments translates to loss of control over security, integrity, and backup, and to continuous interruption in the management environment. New open system choices introduced a combinatorial increase in management complexity and load-balancing problems. Consistent management tools began to be a necessity. Newer management solutions tend to be non-proprietary, use open protocols, and provide application programming interfaces (APIs). Proprietary management solutions that cannot be easily integrated into new open systems will be inadequate for rapidly changing environments.

MASTER
DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

According to many vendors, *integration* across the enterprise is the new Holy Grail of management and the solution to growing complexity. Integration across multiple protocols, multiple operating systems, multiple networks and multiple applications is the overarching goal. New technologies are required. Older, proprietary management platforms are slowly being replaced by newer open system technology directions, but open management platforms do not solve all integration problems. Diverse management applications must still be integrated with these open platforms and there are many options for integrating them (e.g., process launching, alarm trapping, command line interfaces, and APIs).

Complexity concerns

Integration may be the goal, but a looming problem (and hidden Achilles Heel of open distributed computing) is system scalability and corresponding management complexity. Networks have grown from small 10 megabit Ethernet networks to very large data pipes and switches. Unintelligent network nodes have become smart. Smart network nodes have expanded to themselves become entire networks. Management of networks (and some systems) used to be straightforward using the Internet-developed Simple Network Management Protocol (SNMPv1) [1]. Because of scalability concerns, there are now several follow-on activities suggesting how to expand SNMPv1's ability to handle thousands of management agents, sub-agents, and Management Information Bases (MIBs), and to monitor and control millions of managed objects. Unfortunately, many of these newer activities lead to vendor-proprietary schemes and protocols.

Other approaches to controlling complexity stress more automation and less manual intervention. Many vendors suggest adopting a *management-by-exception* philosophy instead of more common polling procedures. Event- and alarm-oriented management employs automated setting of alarm thresholds, alarm filtering criteria, and alarm correlation to help reduce complexity. Correlation means that multiple alarms are examined and analyzed by a (possibly automated) process for clues about what has happened and what appropriate responses might be. Another common technique is *drill-down*. This is a way to systematically examine more detailed layers of management information, thus preventing drowning in a flood of superfluous or misleading data. Most modern management tools provide drill-down capabilities, but not necessarily in consistent ways.

The bottom line

Inattention to management detail results in ineffective and inefficient system design. Lack of appropriate management function places a huge burden on users and support personnel to cope with new problems. Almost everything in a system can be logged or transformed into blinking icons, but what does a manager really want to see and do? There are those who believe we

should make everything look like a network, and manage them with existing network management tools. However, for real success, there should be a shift from managing networks and systems to management of the applications that run on those networks and systems. In the next section, we describe some current and emerging management technologies and standards that may help realize this shift.

Competing technologies

Understanding current open management technologies takes effort. There are management platforms, management tools, management toolkits, management models, and management frameworks. Some technologies are viewed as *off-the-shelf* commodity items. Others are not yet well formed. Still others are in open conflict.

Management platforms

Since network management is viewed as the most evolved technology and closest to a commodity, we start there. Several network management platforms exist. Most are based on UNIX with Hewlett-Packard OpenView, IBM NetView for AIX, and SunSoft Encompass as notable market leaders. Other vendors, such as Cabletron, NetLabs, OpenVision, and Peer Networks provide platform-related supporting technologies. Platforms provide basic management services, such as easy-to-use graphical user interfaces (GUIs), resource discovery and mapping capabilities, data browsers and editors, alarm capability, and protocol support. They differ in their service implementation methods and advanced services offered. Advanced services include alarm correlation, distributed management capabilities, resource modeling features, and support for non-Internet Protocol (non-IP) communications.

A limitation of management platforms is that they are not much more than enhanced IP network monitors. Platforms primarily act as data collectors without capabilities to do much with those data. They provide an infrastructure for other management tools, but not necessarily a comprehensive solution. Applications are still needed to meet specific needs. Most platforms also lean heavily toward networking issues at the expense of distributed systems, applications, and integrated enterprise management.

Management models

Models for management, especially in networks, can be *intrusive* or *non-intrusive*. In an intrusive model, management processes simply go out and *kick* objects to see if they are still *alive*. This works well for non-intelligent devices, but can impact performance of communication networks if polling activities are excessive. In a non-intrusive model, objects have abilities to tell a management process when something of interest

is happening. This kind of model works well for computers and intelligent equipment. Non-intrusive models usually have less impact on network performance.

Recently, systems management extensions have been developed for network management models and network management protocols. Typical system and application functions are automated scheduling, backup and restore, tape management, security management, inventory management, print management, performance management, software distribution, and user/group management. However, introducing system and application extensions to standard networking models raises new issues.

The original SNMP protocol supports simple GET and SET attribute operations and has a trap facility for alarm handling. It relies on a single SNMP port for each system. SNMP agents listen on that single UDP port for management requests. There are severe limitations using a monolithic agent for all possible combinations of system and application components in a complex environment, hence the recent activity in new sub-agent protocols. Limited security in SNMPv1 is also considered a major problem for managing systems and applications.

Potential improvements to SNMP have been explored to address limitations in its original design and inherent network outlook. Agent multiplexing and

enhanced interface schemes for SNMP are described in various Internet RFCs [2,3]. SNMPv2 [4] addresses several of the shortcomings in SNMPv1, including security [5].

Another network management model that has been extended to include systems and applications concepts is the ISO OSI management model [6,7]. The OSI model, while richer and more complex than SNMP, has not found acceptance in the form of widespread use in the United States.

One area that remains consistent through most management models, whether for networks, systems, or applications, is the use of objects to encapsulate the behavior of manageable entities within a system. An object can be viewed as a set of state values and operations that together represent object behavior in response to requests. The state values of an object are sometimes called the object's attributes. The object operations are normally accessible through well-defined interfaces. A managed object is an object to which a management policy applies and whose behavior can be monitored and/or changed by a manager (see Figure 1). Managed objects can represent hardware, software, data structures, or non-computational entities such as people.

Figure 1. Generalized object model for management applications.

Standards and consortia

Management integration approaches exist in the form of several standards, frameworks, and industry consortia. Some of these efforts have found wide acceptance, others hold promise for yet-to-be-realized capabilities, and a few have stalled or have been disbanded. Many times there is little coordination between these approaches, forcing vendors to track multiple efforts and confusing users about what standards are important.

The IETF (Internet Engineering Task Force)

community that developed SNMP believes it is more helpful to consider standards based upon merit, not pedigree [8], and that there are important distinctions between *de jure* and *de facto* standards. The former are legitimate by force of law, but the latter become legitimate by popular acceptance. *De jure* standards can, and do, fail to gain popular use. Usable standards, whether *de jure* or *de facto*, require multiple vendor involvement and convergence. Motivation to support a standard may be driven by several competing factors, including finding a profitable extension to traditional markets, selling more of an existing product, or winning a related war with

competitors.

"When is a standard really a standard?" The following anecdote, taken from an article on the development of a new SNMP MIB [9], illustrates the difficulty of vendor convergence in the standards process:

"Anyone who has been in the computing industry awhile is all too familiar with the way that standards sometimes go when they go bad (we'll use the term *Widget* as the hypothetical technology being standardized):

"Vendors can each develop their own Widgets, and then try to make them de facto standards. This may result in one winner, but usually leaves a group of die-hards who cannot embrace it, for one reason or another, and who continue to cling to their own Widgets.

"A group of vendors can form the Open Widget Consortium, defining a standard that everyone is then supposed to adopt. Usually, though, the Consortium excludes one or more key vendors, who then form the rival Integrated Distributed Widget Group. Dueling press releases follow.

"A formal standards body takes up the process of Widget standardization. Depending on how the standards body functions, this may take months to complete, or years, or it may never complete at all, eventually being eclipsed by newer Widget technology. Indeed, sometimes this gridlock may serve the interests of one or more of the parties, who don't really want a standard to be adopted at all.

Examples of *less-than-successful* open distributed management efforts abound (e.g., UI Atlas, OSF DME, COSE systems management). Nevertheless, the standards process continues. Recent activities (standards and otherwise) that may have an impact on future storage systems and management architectures are described below.

Recent developments

The Management Integration Consortium (MIC) is a group recently formed by vendors to define a common management data model and management data access mechanism [10]. This effort is more a working forum for vendors rather than an official standards body. Stated goals of MIC are to develop a standardized database schema and APIs that facilitate data exchange and sharing between management applications. One motivation of MIC is to have third-party vendors drive the schema and API definition instead of major platform vendors. MIC believes that official standards bodies are not well-equipped to deliver simplicity, and that platform vendors may have more incentive to deliver differentiation than integration.

While not targeted toward management, the Object Management Group's Common Object Request Broker Architecture (CORBA) and the OSF's Distributed Computing Environment (DCE) are examples of existing technologies that address heterogeneity and interworking in large, distributed computing infrastructures.

Open Distributing Processing (ODP) is also a new effort attempting to solve problems of software interaction by proposing a common framework for distributed systems. This framework, called the Reference Model for Open Distributed Processing (RM-ODP), is now an international standard [11] built upon several years of distributed computing research and experience with commercial technologies like CORBA and DCE. It describes an architecture within which support of distribution, interworking, interoperability and portability can be integrated. CORBA, DCE, and RM-ODP only peripherally address management issues, but are important because they shape the distributed environments that enterprise management solutions must support.

SNMP was recently used to standardize management for relational database management systems. Leading RDBMS vendors, together with users, consultants, and IETF members produced a standard SNMP MIB for managing an RDBMS [12]. This activity is an example of a new activity that fulfilled its charter in less than six months, providing a standard before any proprietary solution became entrenched. The standard addresses basic functionality and was developed to discover information about all brands of DBMSs, such as vendor identity, name, and administrator, and to characterize (by size) the number users and activity levels. Private vendor-developed MIBs are allowed for application-specific drill-down inquiries.

Newer management models that integrate existing SNMP standards, databases, file systems, storage facilities, and user application management have begun to emerge. Some models suggest using RPC technologies rather than SNMP (see Figure 2). RPCs are used to communicate between enterprise components and a systems management platform running enterprise-wide management applications. However, RPC-based models can also have scalability problems, such as resource discovery issues when tens of thousands of agents, MIBs, and objects are present in distributed environments.

Storage system problems

Traditional goals for network and systems management are focused toward minimizing service downtime and maximizing service availability, reliability, and security. Recently, other goals have appeared, including lower administrative overhead and support of overall business objectives. All these goals also apply to storage services. While several technologies exist to help networks and systems attain their management goals, well-focused storage system management (SSM) applications and tools are hard to find.

Due to the lack of adequate SSM tools, many storage systems lack a coherent approach to management, sometimes providing an eclectic mix of management function based on local site requirements and peculiarities of system implementation. In the high-performance storage community, storage systems were historically developed in-house and not purchased from commercial

suppliers. The high-performance storage system ran as an independent entity. Thus, the need for compatibility or integration with other management applications in the

enterprise was not well-recognized.

Figure 2. SNMP and alternative RPC-based management models.

Reference models and frameworks

One effort that examined SSM requirements was the IEEE's Storage System Standards Working Group (SSSWG). Using the site management ideas of the original Mass Storage System Reference Model Version 4 (MSSRMv4) [13], storage system vendors, researchers, developers, and users discussed what might constitute a standardized framework for SSM, specialized functions of SSM, and how SSM relates to other management technologies [14]. These views incorporate both managed objects and management by policy (see Figure 3).

Progress toward SSM standardization is slow. A difficulty faced by IEEE SSSWG was that many participants were not vendors. Instead, they were motivated system developers and managers who faced the reality of SSM as a continuing concern at their own sites. Another difficulty is the existence of other management standardization activities that, while not specifically addressing storage, are attempting to provide standards for the management infrastructure needs of general distributed systems, thus complicating a scope definition for SSM standards.

A standardized framework for distributed storage system implementations and distributed storage system management is needed. The IEEE MSSRM is a step in this direction. An SSM framework does not mandate

specific management roles and rules, but rather provides an extensible structure in which to implement and enforce site-specific storage system policies and procedures. Frameworks can provide flexibility, scalability and a foundation for rapidly responding to change. However, frameworks will not solve specific problems. Instead, they enable and facilitate good solutions. Most existing frameworks use objects, as they are scalable, secure, reliable and support reuse. Unfortunately, there are not yet any widely accepted standards or frameworks for distributed applications management.

High-performance distributed storage systems share many of the management concerns of other distributed applications, but also have important management issues not necessarily found elsewhere. These storage issues must be addressed today, and solutions must be integrated smoothly into future open distributed management standards and technologies.

Storage issues

Management services are sometimes grouped into functional categories such as fault, configuration, security, accounting, and performance management. The OSI management model uses this categorization. The IEEE MSSRM adds transaction management. RM-ODP uses a slightly different grouping deemed appropriate for ODP systems - namely configuration, quality of service,

accounting, performance, and policy definition. Security functions are not included in RM-ODP management, but management interactions are subject to policy control. Although functional groupings are similar for most management services, the specific requirements of storage systems are quite different from management of communications equipment or small systems.

Unlike failing networks or systems that can be rerouted, reloaded, rebooted, or replaced, storage system

components that fail represent the potential loss of irreplaceable, permanent data. Hence, fault management in storage includes comprehensive procedures for backup and recovery of critical data, both user data and system metadata. The integrity of the system metadata may also need to be continuously maintained through integrated transaction management facilities.

Figure 3. Conceptual view of IEEE SSSWG storage system management

The most important issues for storage systems involve hierarchical storage management (HSM) operations. Data caching and migration (movement of data between different storage levels), and media repacking (consolidation of data to reclaim unusable space) are common operations. Management of these operations constitute much of the day-to-day administrative and operational procedures for high-performance storage, yet are not well addressed by current network and system management tools. SSM requires an ability to define policies that ensure HSM operations meet specific site requirements.

Another issue for storage systems, especially those that are highly distributed, is flexible server and resource configuration control. This includes not only basic functions to start, stop, and display server components, but also to find out what components exist, how to remove them, and how to create them. Automatic discovery capabilities are commonly used in network management applications to display iconic views of active network components. Similar capabilities are needed for

storage systems, but must be capable of tracking up to millions or billions of objects of vastly different type (e.g., bitfiles, virtual volumes, physical volumes, and cartridges).

Once resources are defined (or discovered) and configured to suit one's needs, there still needs to be security or policy control over the use of configured resources and management operations. Security for some sites is paramount because stored data might be classified or sensitive. Security violations need to be detected and reported. Management might maintain logs of violations and storage system requests of security interest.

Because storage systems contain fixed resource capacities (i.e., finite allocable space on usable devices), a method of user control or coercion is necessary to prevent abuse of limited resources. Some systems employ an accounting process that charges for the resources consumed. An alternative to charging is to use quotas that automatically prevent access to resources when a user (or group) limit is reached.

Quality of service (QoS) is a set of client-understood

attributes, expressed in a client-understood language, describing expected characteristics of a particular service. QoS is used to help clients observe or influence the collective behavior of distributed system resources. QoS parameters for storage might include minimum and maximum file sizes, allowable I/O operations, remote vaulting capability, data transfer rate, perceived delay, availability, reliability, and security. QoS in storage systems can be implemented as a fixed or dynamic hierarchy of devices, each level in the hierarchy having a different combination of QoS characteristics.

Cost-effective utilization of storage resources and construction of proper policy needs adequate performance monitoring capabilities. This will require intelligent definition and use of measurement data (metrics). Storage system designs should clearly define and correctly balance the level of instrumentation detail within the system's components. Tools should be provided for throttling, filtering, collecting, verifying, and correlating data. There are no standards for storage-system-specific metrics, although X/Open's Performance Management Working Group (PMWG) has defined several metrics for UNIX I/O operations, and disk and tape devices [15].

Management of large storage systems has historically been performed by a dedicated operations staff or by knowledgeable systems programmers well-grounded in storage configuration, maintenance, and intervention procedures. As distributed processing and distributed management place more demand on existing staff, operation and management of storage systems will likely fall to the those that manage networks and systems. This presents problems if these new administrators are not familiar with storage system management issues. The need for storage systems to provide proactive management capability and smoother integration with other management systems will be required.

In summary, a well-constructed storage system should meet the following SSM concerns:

- Hierarchical storage management
- Fault detection and data recovery
- Resource configuration control
- Secure management operations
- Accounting and usage limits
- Quality of service flexibility
- Balanced internal instrumentation
- Proactive/self-managed components

We describe in the next section how these requirements were addressed in a modern, high-performance, distributed storage system, in the presence or absence of suitable management technologies for storage.

HPSS case study

HPSS, a development effort of the National Storage Laboratory (NSL), is a new high-performance storage system for data-intensive applications and large-scale computers. Key elements of HPSS design were

requirements for modularity of software components and an integrated SSM capability. Each software component is responsible for a well-defined set of storage objects, and acts as a service provider for those objects. Two specialized management components, the *System Manager* and the *Data Server*, together comprise HPSS SSM, which monitors and controls the resources of the system.

HPSS overview

The NSL and HPSS are industry and U.S. Department of Energy collaborations organized to develop and commercialize new technologies for high-performance distributed storage [16]. Principal developers for HPSS are the U.S. Department of Energy's Lawrence Livermore, Los Alamos, Oak Ridge and Sandia National Laboratories, and IBM Government Systems. Other developers include Cornell University, NASA Lewis, and NASA Langley Research Centers. A major motivation for HPSS was to develop a high-speed storage system providing scalability to meet demands of new high-performance computer applications where vast amounts of data are generated, and to meet the needs of a national information infrastructure [17].

HPSS has a networked-centered architecture [18] and its software components are based on IEEE MSSRMv5 (see Figure 4). The architecture includes a high-speed network for data transfer and a separate network for device control and management. The control and management network uses DCE RPC technology. Control and data transfer networks may be physically separate or shared in actual implementations [19]. Another feature of HPSS is support for both parallel and sequential input/output (I/O). The I/O architecture is designed to scale as technology improves by using data striping and multiple data movers as a parallel I/O mechanism.

HPSS was designed to support data transfers from hundreds of megabytes up to a gigabyte per second. File size scalability must meet the needs of billions of data sets, some potentially terabytes in size, for total storage capacities in petabytes. The system must also scale geographically to support distributed systems with hierarchies of distinct storage systems. Management components of HPSS were also designed to scale along with the rest of the system.

Management philosophy

Concepts from current management standards and technologies were incorporated into the HPSS SSM approach where feasible. Initial design began with abstract managed object definitions and was based on widely accepted management model constructs. These definitions were subsequently refined after meeting with server development teams. This helped in developing an overall approach to uniform software server development and a flexible instrumentation process. HPSS is written in C, not an object-oriented language, but objects are used to represent resources and components throughout the

system. This provided a good conceptual view of what needed to be managed. We focused on administrative and operational control of the system with the understanding that those who would be exercising control were not necessarily system software developers nor experts in HPSS storage internals.

The highest priorities for HPSS SSM were to develop necessary management functionality. As a result, HPSS SSM has a robust set of fault, configuration, security, accounting, QoS, HSM, and instrumentation capabilities. Since system and software vendors have not

yet agreed on comprehensive standards for management APIs or protocols, we avoided using a specific protocol such as SNMP for management communication between the servers and the System Manager. We, instead, relied on simple RPC-based management interactions, similar to other inter-server communication in HPSS. We believe that the SSM architecture currently in place is easily adaptable to what finally emerges as the protocol and integration strategy of choice for enterprise-wide management of distributed systems and applications.

Figure 4. HPSS software architecture.

A decision was also made to avoid a reactive management style, where problems are only identified after the system is in serious trouble. Instead, a proactive management philosophy was used. SSM can query almost any internal server information through the display of managed object attributes. An ability to provide management by policy was also built into the SSM design. Policies in HPSS are implemented as metadata that can be modified by administrators and operators to suit the specific needs of a site. For example, migration and repacking of files are controlled through identifiable policies associated with HPSS service and storage classes. Policies are also used for HPSS accounting and security management. Servers are written such that management decisions are not hard-coded into software but can, instead, be determined from policy files or from management requests.

Administrative management concerns were shielded from servers where possible. For example, to satisfy

requirements for a quality of service capability, SSM provides tools for administrative definition and management of HPSS Class of Service (COS), a data structure that helps identify performance, media, and usage attributes of resource objects managed by the servers. COS is used to ensure that HPSS meets varying client expectations for service quality [20]. These administratively managed COS definitions are internally mapped to lower-level *storage class* attributes more easily understood by the servers that perform actual resource management. This helps isolate server developers from external management processes unrelated to basic server functionality. Similarly, administrative filtering of log messages, events, and alarms sent by servers is performed external to servers (it is performed by a logging service infrastructure). Comparable approaches to shielding application developers from complex management concerns have been used by others [21].

Data structures

To properly manage distributed HPSS servers and resources, configuration and state data describing the servers must be consistent. To meet this goal, necessary managed objects and metadata were defined, together with interfaces for object manipulation and communication.

Entities in HPSS are of two types: (1) servers, and (2) resources managed by servers. Servers are described by two types of managed object: (1) common, and (2) specific. The common server managed object contains standardized server information such as state data. Its structure is generic and shared by all servers managed by SSM. Most servers are also described by an additional specific server managed object containing information relevant only to that server. In a more object-oriented system implementation, the specific server information could be used to define a new server subclass that inherits all generic server information from a common server superclass. APIs were implemented to allow SSM to get and set attributes for each type of server managed object.

There are also several managed objects containing information about non-server HPSS entities representing logical and physical resources, such as drives, cartridges, virtual volumes, physical volumes, storage segments, and bitfiles. Attributes of these managed objects include information useful in determining overall system status and performance. APIs to get and set attributes of these managed objects were also implemented.

To satisfy the requirement for flexible resource configuration, each server and some non-server entities (e.g., cartridges and drives) have configuration-specific information associated with them. This configuration information is stored as metadata under the control of an internal HPSS Metadata Manager. The Metadata Manager, based on Transarc's Encina product, also helps provide transaction management capabilities in HPSS [22]. APIs to create, read, write, update, and delete configuration metadata are available to SSM and to the other servers, thus ensuring the consistency of system metadata.

Similar to the two managed objects representing servers, there are two categories of configuration metadata: (1) common, and (2) specific. The variables contained in common metadata include items like server name, a universal unique identifier, and node and security information. Specific metadata are unique to each server. Some HPSS servers require notification whenever their configuration metadata is changed. Other servers prevent SSM from changing configuration data while the server is running.

Management system design

There are three primary elements to the HPSS SSM design: the operations GUI (graphical displays), the Data Server, and the System Manager (see Figure 5). The System Manager communicates with all HPSS servers to acquire state information and to relay operator commands.

The Data Server receives information from the System Manager and formats it for display to the GUI through graphical display runtime APIs. The Data Server also receives operator requests and converts them to a form the Manager can use. Any Data Server accesses to Encina are made on the Data Server's behalf by the System Manager.

Operations provided by the System Manager to the Data Server include starting, stopping, and modifying states of servers, configuration of Encina files, importing and exporting physical media, creation of new virtual volumes, control of storage devices and jobs, viewing and updating managed objects, and delogging. In SNMP terms, the System Manager primarily acts as an intelligent master agent for the HPSS system. Servers may be viewed as sub-agents communicating management information through the System Manager.

The Data Server (or other client in a Data Server role) makes contact with the System Manager with a *CheckIn* function. The System Manager returns a unique identifier for the Data Server client and several lists that describe known servers, drives, and service classes. All alarms, events, and notifications issued to SSM by other servers (through the logging service) are received by the System Manager and forwarded to the Data Server as appropriate. Requirements for filtering of log data are met through administrative definition of logging policies.

One of SSM's major goals is ease of use for the administrator/operator. To achieve this, we implemented all management functions through a GUI with a standardized look-and-feel. Effective layout of a user interface can be tedious since each datum must be interpreted and placed on the screen, so a commercial layout tool, Sammi, from Kinesix, was chosen. Sammi was also used for storage system management in NSL-UniTree, a previous system developed at the NSL [23].

As a predecessor to HPSS, NSL-UniTree shares some HPSS high-level SSM design. Both systems use a general Health and Status display and both separate the Data Server and System Manager functions. However, the NSL-UniTree GUI is more limited than HPSS and does not use DCE for communication or Encina for metadata management. There are no actual managed objects in NSL-UniTree and the operator interface was not designed to be the sole mechanism for control of the system.

SSM operation and information flow

Many SSM operations, such as starting and stopping servers, are typical of other distributed application management tasks. These basic SSM operations are good candidates for an enterprise-management integration strategy, as they should be easily portable to existing management platform technologies. Below is a stop/start server example.

Assume the system is already running and a new operator has arrived. The operator must first log into the system utilizing a logon screen and security features that prevent unauthorized access to privileged management operations. All SSM operations have an internal security

level associated with them. This helps meet the requirement for administrative policy control over management operations. Having logged on, the operator is presented with a Health and Status Display presenting overall status information with statistics and an aggregated system state represented in both colors and words. In addition, this window has icons representing the aggregated operational state of HPSS devices, servers, and storage entities.

Let us now suppose that the server icon shows red; something is wrong. The operator can perform a drill-down operation by clicking on that icon to get a *sick list* containing a list of servers that have problems. When the operator clicks on one of the servers in the list (another drill-down operation), a new window will be presented containing the common and specific managed objects for that server, together containing all the relevant information maintained for that server.

The operator also has historical information available in the form of alarm and event messages. These are contained in separate windows accessible through a menu bar at the top of the Health and Status display. The

operator can open those windows at any time and scroll through them. If the desired information is unavailable, the operator can also perform a *delog* operation -- dump an active or archived log file containing information that may not have been routed to the SSM servers. Delogging operations can filter log messages on server type, time, alarm severity, and other criteria.

After studying all available information, the operator has determined that the sick server must be killed and restarted. The operator can either tell the server to stop through changing a field in the HPSS Server Managed Object, or can make a menu selection that will present a list of servers from which to select the one(s) to stop. After the server has been stopped, the operator can start it again through a menu selection process. When the server comes back up, the operator can make another menu selection to display the managed objects representing that server to ensure the server is functioning properly. The flow of information through HPSS when starting a server is briefly described below.

Figure 5. HPSS storage system management.

First, the command is fielded by the Sammi run-time system which presents it to the Data Server. The Data Server uses internal information it maintains to formulate a request to the System Manager. The System Manager instructs the Startup Daemon (a special process on each HPSS node) to start that server, and tells it where the common and specific configuration information can be found. Once the server starts, it reads its required configuration information from the Encina-maintained metadata.

The System Manager establishes communication with the newly started server and retrieves the managed objects, which it then passes to the Data Server. The Data Server uses status information contained in the managed objects to determine overall system status and to remove the now-healthy server from the sick list. When the operator asks to see the managed objects, the Data Server fields the request and instructs the window manager and Sammi to open and fill in the windows.

The SSM GUI and menu system provide additional

management capabilities such as system configuration, alarm and event filtering, cartridge import, and creation of physical and virtual volumes. Sometimes these storage system operations are complex. Defining new cartridges means defining the Storage Server resource management data structures for the cartridges, which include physical volumes, virtual volumes, and storage segment maps. When a chain of related operations like this must take place, an attempt is made to make the administrative interface as simple as possible. Unlike the starting and stopping of servers, a relatively common distributed application operation, management operations for the definition of new cartridges and related storage resources will not likely be standardized by current management standards activities.

Implementation experiences

The HPSS project is a collaborative effort and represented a new "distributed development" approach for the participants. The geographic dispersion of the developers, and the lack of clear standards to guide the SSM implementation process, created some unique experiences.

Not all development teams progressed through design, object definition, implementation, and test at the same rate. As a consequence, some teams produced early prototypes for SSM functionality, and some teams produced later ones. These SSM capabilities were not necessarily identical from server to server because of requirement and design changes as the overall implementation progressed. There were differences in routine name, status reporting, and alarm and event definition. These issues were successfully resolved during final implementation and system integration.

Several implementation issues concerning more complex administrative tasks in HPSS were not decided until late in the development process. For example, some objects in HPSS are shared between servers. This led to several issues. Which server really owns the object? How should the managed object(s) be structured? What does an operator have to know about the details of the sharing? Who should do the translation between the object an administrator can view and the possibly multiple objects comprising the resource in HPSS metadata?

SSM must communicate with all other servers, but it is not an infrastructure service (as are other HPSS routines or tasks that must communicate with all servers). It is instead more of a superstructure, and it is in SSM that all information from all servers comes together. It is SSM that needs to synthesize the information to produce new information (such as an overall system status). As a consequence, SSM is one of the places where inconsistencies in use of common data or functions become visible. This is comparable to the enterprise management problem where several distributed applications must all communicate management information in a consistent way to a central management platform. Standard management data structures such as

those under development at MIC would ease this process.

During final SSM design, coding, and integration test, it was necessary to identify and remedy all remaining inconsistencies, clarify SSM functionality, and revise code. Fortunately, the design of the system, particularly the set of APIs and managed objects, remained highly stable through the server integration process. This stability was, in part, the result of careful initial design work, and was to be of help developing the final SSM implementation.

In the design of HPSS Version 2, the developers used the lessons from Version 1 work. Because of more familiarity with SSM and geographically distributed collaborations, server writers were able to give earlier and continuing consideration to the way their server should be managed.

Conclusions

There are similarities and differences in management of networks, systems, applications, and high-performance storage. Storage systems present management requirements not necessarily addressed by existing management technologies. When similarities exist, we believe it is sensible to leverage previous work or existing technology. For unique characteristics of high-performance storage it may be necessary to develop new function.

In HPSS, we used existing approaches, including managed objects, GUIs, RPCs, and transaction management where practical, and developed our own mechanisms where necessary. HPSS provides policy management of HSM migration and repack operations, system metadata integrity through an integrated Metadata Manager, flexible resource configuration, event and alarm capabilities, and authorization control for administrative operations. HPSS also provides service quality management through its Class of Service data structures, and thorough server instrumentation for current and future measurement, monitoring, and proactive self-management.

Version 1 of HPSS underwent final integration test in early 1995 and was installed at several developer sites. A hybrid version (incorporating Version 1 and new disk support features from Version 2) was successfully demonstrated at Supercomputing '94. HPSS Version 2 is scheduled for release in late 1995. It will include several new SSM features lacking in Version 1, including new support for disk resources, migration, purge, repack, and accounting operations.

There are SSM areas that we would like to improve upon in future releases. These include a more easily managed class-of-service capability, better ways to utilize policy-driven management, easier control of devices that are shared between servers, more powerful accounting (including quotas), better performance monitoring, and resource discovery capabilities. Some areas targeted for future extension are being pursued in system and applications management areas, so it is likely that emerging technologies can be leveraged for use in HPSS. One step toward an enterprise integration strategy might

be to introduce SNMP agent or sub-agent code into the HPSS servers to allow simple management operations to be controlled by existing network management platforms using SNMP protocols. Several vendors offer such agent code for insertion into non-SNMP or other legacy applications.

Because high-performance distributed storage is a small market (compared to more widespread backup and restore applications), gathering a critical mass of vendors to produce workable standards in this area is difficult. This is evident for components at the client end of the IEEE Mass Storage System Reference Model, where dividing lines between storage, file systems, operating systems, and applications become fuzzy. An obstacle for specific storage system management standards is the large push for open management standards, integration solutions, open frameworks, and convergence in overlapping computing areas. Isolating storage-system-specific management topics from general management is hard, but nevertheless, should still be addressed.

A suggested approach for successors to the IEEE SSSWG (or any groups interested in SSM) is to follow the model used in creation of the RDBMS MIB. This was not a true de jure standards effort, but due to its well-defined scope, perceived need, and vendor acceptance was successful within reasonable time frames. Until SNMP agents and managers disappear, (unlikely in the U.S.), it may be reasonable to develop an overall storage system MIB. MIBs for specific storage devices have already been developed by individual vendors, but an overall storage system MIB has not been attempted.

The impetus behind enterprise-wide management and standardized approaches is gaining momentum. There are still storage-system-specific issues for large, distributed environments. We trust that the individuals, vendors, consortia, and formal standards bodies investigating management technologies and storage system applications will consider how to bring high-performance storage management issues, such as those addressed in HPSS, into the mainstream. The result will be useful management tools and applications for high-performance distributed storage and, consequently, much less need for high-end developers to continue down an independent path.

Acknowledgments

This work was, in part, performed by the Lawrence Livermore National Laboratory and the Oak Ridge National Laboratory under the auspices of U.S. Department of Energy Cooperative Research and Development Agreements, and by IBM Government Systems under Independent Research and Development and other internal funding. We wish to acknowledge the contributions of Vicky White and Dan Million of Oak Ridge for their work on the HPSS System Manager and Data Server. We also wish to acknowledge Tyce McLarty of Los Alamos, and Tracy Tran and John Nguyen of IBM Government Systems for early work on SSM design.

References

1. J. Case, M. Fedor, M. Schofstell and C. Davin, Simple Network Management Protocol (SNMP, RFC 1157, May 1990.
2. M. Rose, SNMP MUX Protocol and MIB, RFC 1227, May 1991.
3. G. Carpenter and B. Wijnen, SNMP-DPI Simple Network Management Protocol Distributed Program Interface, RFC 1228, May 1991.
4. J. Case, K. McCloghrie, M. Rose and S. Waldbusser, Introduction to version 2 of the Internet-standard Network Management Framework, RFC 1441, April 1993.
5. J. Galvin and K. McCloghrie, Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1446, April 1993.
6. ISO/IEC 10040, Information Technology—Open Systems Interconnection—Systems Management Overview, 1991.
7. ISO/IEC 10165-1, Information Technology—Open Systems Interconnection—Structure of Management Information, Part 1: Management Information Model, 1991.
8. D. Crocker, Standards the IETF Way, The Interoperability Report, Vol. 7, No. 9, September 1993.
9. R. Purvy, Using SNMP to Manage Relational Databases, Conference Proceedings of the Enterprise Management Summit, San Jose, CA, November 14-18, 1994.
10. E. Olinger, ed., Open Repository Specification - MIC0001: Meta-Schema Specification, Management Integration Consortium Draft, February 1995.
11. ISO/IEC JTC1/SC21/WG7, Reference Model for Open Distributed Processing (RM-ODP), ITU-TS Rec. X.901-904 | ISO/IEC 10765-1, -2, -3, -4, 1994.
12. D. Brower, R. Purvy, A. Daniel, M. Sinykin and J. Smith, Relational Database Management System (RDBMS) Management Information Base (MIB) using SMIV2, RFC 1697, August 1994.
13. S. Coleman and S. Miller, eds., Mass Storage System Reference Model, Version 4, IEEE Technical Committee on Mass Storage Systems and Technology, May 1990.

14. R. Garrison, et al. (eds.), Reference Model for Open Storage Systems Interconnection: Mass Storage Reference Model Version 5, IEEE Storage System Standards Working Group (P1244), September 1994.
15. L. Traister, ed., An Introduction to the UMA (Universal Measurement Architecture) Specifications, Version 1.0, Performance Management Work Group (PMWG), X/Open, April 1994.
16. R. Coyne, and R. Watson, The National Storage Laboratory: Overview and Status, Proc. Thirteenth IEEE Symposium on Mass Storage Systems, Annecy, France, June 13-16, 1994.
17. R. Coyne, H. Hulen, and R. Watson, The High Performance Storage System, Proc. Supercomputing '93, Portland, OR, November 15-19, 1993.
18. D. Teaff, R. Coyne, and R. Watson, The Architecture of the High Performance Storage System, Fourth NASA GSFC Conference on Mass Storage Systems and Technologies, College Park, MD, March 28-30, 1995.
19. R. Hyer, R. Ruef, and R. Watson, High Performance Direct Network Data Transfers at the National Storage Laboratory, Proc. Twelfth IEEE Symposium on Mass Storage Systems, Monterey, CA, April 26-29, 1993.
20. S. Louis, and D. Teaff, Class of Service in the High Performance Storage System, Third IFIP TC6 International Conference on Open Distributed Processing, Brisbane, Australia, February 21-24, 1995.
21. J. Weinstock, Management of Distributed Applications: A Developer's Perspective, Conference Proceedings of the Enterprise Management Summit, San Jose, CA, November 14-18, 1994.
22. D. Fisher and T. Tyler, Using Distributed OLTP Technology in the High Performance Storage System, Proc. of the 14th IEEE Computer Society Mass Storage Systems Symposium, Monterey, CA, September 11-14, 1995.
23. S. Louis and S. Hyer, Applying IEEE Storage System Management Standards at the National Storage Laboratory, Proc. Twelfth IEEE Symposium on Mass Storage Systems, Monterey, CA, April 26-29, 1993.