

International Journal of High Performance Computing Applications

<http://hpc.sagepub.com>

Systems Administration

Anthony Skjellum, Rossen Dimitrov, Srihari Venkata Angaluri, David Lifka, George Coulouris, Putchong Uthayopas, Stephen L. Scott and Rasit Eskicioglu

International Journal of High Performance Computing Applications 2001; 15; 143

DOI: 10.1177/109434200101500207

The online version of this article can be found at:

<http://hpc.sagepub.com>

Published by:

 SAGE Publications

<http://www.sagepublications.com>

Additional services and information for *International Journal of High Performance Computing Applications* can be found at:

Email Alerts: <http://hpc.sagepub.com/cgi/alerts>

Subscriptions: <http://hpc.sagepub.com/subscriptions>

Reprints: <http://www.sagepub.com/journalsReprints.nav>

Permissions: <http://www.sagepub.com/journalsPermissions.nav>

SYSTEMS ADMINISTRATION

Anthony Skjellum¹
Rossen Dimitrov²
Srihari Venkata Angaluri²
David Lifka³
George Coulouris³
Putchong Uthayopas⁴
Stephen L. Scott⁵
Rasit Eskicioglu⁶

¹ MPI SOFTWARE TECHNOLOGY, INC. AND MISSISSIPPI STATE UNIVERSITY

² MPI SOFTWARE TECHNOLOGY, INC.

³ CORNELL THEORY CENTER

⁴ KASETSART UNIVERSITY, BANGKOK, THAILAND

⁵ OAK RIDGE NATIONAL LABORATORY

⁶ UNIVERSITY OF MANITOBA, CANADA

1 Introduction

As industry-standard computing platforms such as workstations and small-scale symmetric multiprocessor servers continue to show better and better price performance, more and more significant clustering projects are developing internationally. What separates these systems is how usable they are in terms of actually producing computations of value to their users and at what “comfort level” for their users. Manageability of a system can be the single most important factor in its practical usability. This is complicated by the current fact that many clusters are in all or in part themselves experimental computing systems for computer science studies of architecture and/or software while attempting to offer research and/or production platforms for their user base. Even for a single system, the emphasis can vary over time to satisfy sponsor requirements and evolving research ideas, whereas a production-oriented user base typically is most interested in application results, system availability, and stability.

The goals of various clustering projects are quite diverse, with extremes evident. First, there is the dedicated cluster for computer science research. The goal of this type of system is typically to undertake performance testing, benchmarking, and software tuning. This work is often done at the networking, application, and operating systems levels. These clusters are not typically used for computational science and quite often are in a state of flux. The opposite extreme is the production-computing environment. The tacit goal of these systems is to provide reliable computing cycles with dependable networking, application software, and operating systems. As noted above, systems may attempt both extremes, either by spatial division of the resources among emphases or by reallocation policies over time. Realistically, the entire range of work being done on clusters and the quality of the results generated on them are driven strongly by the tools available to the systems administrators. In this article, we describe some of the critical issues that arise when planning a cluster-computing installation and its ongoing support and maintenance.

No matter what the goals of a particular cluster-computing project may be, good systems manageability will directly equate to better results. Systems such as the IBM SP have been successful in providing these types of utilities in the past. However, similar and in many cases superior tools are being developed for clusters out of necessity. Even if the users are just experimenting with a small test cluster, the ability to verify and ensure consistency in hardware, application software, and system settings across the cluster’s resources from a single console will

“No matter what the goals of a particular cluster-computing project may be, good systems manageability will directly equate to better results.”

“With the ever-increasing performance of the Intel architecture and its broad market base, the cost/performance ratio of clusters of workstations continues to improve almost daily.”

improve the efficiency and reproducibility of results. It is equally important for a systems administrator to be able to install system images and layered software in a parallel fashion. All of these issues reduce the cost of ownership, which is an important aspect of the overall value of cluster computing to users, not just their initially low-cost nature.

2 System Planning

Several considerations must be made when planning a cluster installation. Generally, they are the types of computers to purchase, the type of networking interconnect, and the system software to use. All of these must be taken into consideration, and typically the driving factor for any particular configuration is the requirements of the applications that will be run on it. In this section, we describe some of the more popular options and try to focus on the application requirements that warrant their use.

2.1 HARDWARE CONSIDERATIONS

The idea of clustering has been around for well over 10 years. It originated as a way to leverage the investment of workstations in an organization. Because workstation technology was and continues to progress rapidly, it was found that linking workstations together with Ethernet networks was a good way to get near to traditional supercomputer performance. The performance could be delivered at a low entry price and furthermore expanded interest in applications that could run in parallel without the huge network performance of traditional parallel supercomputers. Also, it meant the machines, which were commonly left idle, such as desktops at night and on weekends, could putatively begin to be used in a productive manner during those periods.

With the ever-increasing performance of the Intel architecture and its broad market base, the cost/performance ratio of clusters of workstations continues to improve almost daily. Because of its origins, many people prefer to purchase “beige boxes” or inexpensive generic Intel-based workstations. This provides a low-cost/compute cycle ratio. What has changed is how people are using clusters. Today, organizations are starting to use clusters as their primary source of computational platform. In doing so, the demands for reliability, manageability, and supportability have dramatically increased. Once again, applications dictate what works and what does not. If the purpose of a cluster is hard-core computer science research in which drivers and software are being continually evolved, workstations may work just fine. As clusters are placed in production computing environments, where the research that is being

undertaken is chemistry, physics, finance, computational fluid dynamics, or other quantitative application areas and not necessarily only computer science, the issue of uptime becomes critical. To meet these needs, computing centers are looking toward high-end industry standard-based servers. The cost-benefit remains, but the additional up-front cost typically affords excellent hardware redundancy and maintenance programs that used to only be associated with traditional supercomputers and mainframes. Furthermore, as the size of the cluster begins to increase, many server vendors are providing space-efficient, rack-mountable solutions. These provide not only a smaller footprint but also organized cable, power, and cooling management. Figure 1 shows a common cluster cable plant. Figure 2 illustrates what is possible with a rack-mounted solution.

2.2 PERFORMANCE SPECIFICATIONS

Information about the type of workload intended to be incorporated in a cluster will help tailor the cluster's hardware to these needs. When building a cluster, it is important to choose hardware that is well suited to the prospective workload. Low-level details such as memory bus speed and PCI bus interfaces are just as important as processor speed or cache sizes. We outline several of these issues in the following subsections.

2.3 MEMORY SPEED AND INTERLEAVE

While faster memory is in general "always better," higher memory density is not always desirable. For example, consider a cluster that requires 256 MB per node, and each node has four available slots. While getting two 128-MB modules allows for later upgrades, using four 64-MB modules may provide increased memory bandwidth if the motherboard chipset supports memory interleaving. Upgradability is a competitive concern, if one preplans to do a midlife memory upgrade to a system, based on funding or expanding user requirements for in-core problem size support.

2.4 PROCESSOR CORE SPEED VERSUS BUS SPEED

Modern processor cores typically run at a multiple of the speed at which they interface with the rest of the system. For example, while a processor might run at 500 MHz, its bus interface might only be 100 MHz. On a code, which is memory intensive, for example, the cluster designer may wish to opt for a system with a lower core frequency but higher bus speed.



Fig. 1 A do-it-yourself cluster look and feel



Fig. 2 A professionally integrated cluster

2.5 PCI BUS SPEED AND WIDTH

PCI interfaces currently come in two widths (32 or 64 bits) and two speeds (33 or 66 MHz), with expansions to be developed. While most low-end cards will have a 32-bit/33-MHz interface, cards such as high performance network adapters will have wider or faster interfaces. If the motherboard does not support the wider or faster bus, maximum performance will not be obtained. The quality of PCI bus implementations is a hot topic among people considering network card performance because actual performance varies considerably from chipset to chipset. Investing time to be sure that the chipset works well with the cluster's planned interconnect is essential, as are any issues of multiple buses and the planned utilization of local I/O as well as networking. Newer systems do not immediately equate to better PCI performance, though this is the general trend.

2.6 MULTIPROCESSOR ISSUES

Depending on the cluster's workload, one may wish to choose single- or multiprocessor building blocks. For applications whose working set is smaller than the processor cache size, using multiprocessor machines can increase a cluster's aggregate performance significantly. For memory-intensive applications, however, there are trade-offs. Simple bus-based multiprocessor machines work well in two-processor configurations but do not scale well for higher numbers of processors. Crossbar-based or hybrid bus/crossbar systems deliver higher memory bandwidth and better scalability but are usually priced higher, and memory consistency management heightens the context switch times of such systems. Context switch time affects performance of thread-oriented applications and may somewhat affect applications that do a lot of I/O.

2.7 CLUSTER INTERCONNECT CONSIDERATIONS

Network interconnects are essential components of high performance computational clusters. Four major factors determine the importance of cluster interconnects:

1. Communication-intensive parallel applications require efficient data transfers even at low processor scales. Applications that spend a large portion of their execution time in communication benefit most from improvements in network speeds. Fine-grain algorithms that are sensitive to short-message latency and coarse-grain algorithms that are sensitive to peak bandwidth of bulk transfers re-

quire balanced and efficient communication systems.

2. Efficiency of the network system includes the effective drain on processor cycles associated with transfers resulting both from hardware and associated software stack. Furthermore, as technical applications often possess the ability to overlap communication and computation, network architectures with software stacks that support user-to-user overlap of communication and computation can reduce application runtimes significantly, even for large-scale systems (since problems are often scaled as cluster processor counts as scaled).
3. Large-scale clusters with number of processors exceeding 1000 are currently being deployed. Maintaining sufficient parallel efficiency at these scales requires highly optimized network interconnects.
4. The rapid growth of microprocessor speed and memory throughput allows for a cost-efficient increase of parallel performance. However, the effective benefit of this increase may be significantly diminished if the cluster is interconnected with an inadequate network that may become a performance bottleneck in latency, bandwidth, and/or consumption of CPU cycles to deliver needed performance.

3 Software Considerations

Hardware is not the only set of concerns that play a major role in the decision process when building a commodity cluster. The choice of the operating system and the other necessary software also affect the usability and manageability of the cluster. In most cases, it is the proposed use of the cluster that influences the decision for the software that goes into the system, including the operating system. Clusters present a logical extension and/or an equal environment to traditional multiprocessor and multicomputer (massively parallel) systems. Typically, clusters are chosen with price performance in mind, and the problems that are solved on clusters are those that are ported from multicomputer environments, though applications targeted especially for clustered architectures are slowly taking off. The issues surrounding the choice, features, and functionality of cluster operating systems are discussed in this section.

3.1 REMOTE ACCESS

How users and administrators access cluster resources remotely is an important consideration. Both UNIX and

Windows offer a variety of options. Windows 2000 has made great strides to provide the same types of tools that UNIX provides.

Windows Capabilities. Windows 2000 has a built-in telnet server. Under server and advanced server, it limits the number of telnet connections to two. Under the services for the UNIX toolkit from Microsoft, there is an alternative telnet service that does not limit the number of connections. One important factor that is currently lacking in the Windows telnet environment is a secure shell capability. However, one can use SSH to encrypt terminal server sessions. Terminal server from Microsoft only allows Windows-based clients to connect, while Citrix (<http://www.citrix.com/products/metaframe/>) has an add-on product, which provides terminal server clients for many UNIX flavors, as well as Macintosh, and a web client that permits connections from a standard web browser.

Microsoft Internet Information Services (IIS) provides FTP, web, news, and e-mail services for Windows 2000. IIS can be configured with various levels of security for each of the services it provides. One feature of IIS is its integration with the Windows security model. This makes the development of secure web sites easier.

Finally, Microsoft allows users to run new and existing X Windows applications. Products such as Hummingbird Exceed (<http://www.citrix.com/support/solution/SOL00128.htm>) and Cygnus (http://www.redhat.com/support/wpapers/cygnus_whitepapers.html) are designed for Windows platforms.

UNIX Capabilities. Because UNIX has been used in cluster environments longer than Windows, most of the access tools have been available for a while. Most UNIX versions provide SSH, Telnet, X Windows, and FTP. Integrating secure UNIX web sites can be accomplished using the Apache web server (<http://www.apache.org>), for example.

3.2 SYSTEM INSTALLATION

When managing a cluster of any size, it is extremely important to be able to ensure a consistent system image across all cluster nodes. This helps eliminate variables when trying to debug system configuration problems and user program errors. System tools such as DHCP go a long way toward making it possible to generate a system image and propagate it across all cluster nodes by minimizing the number of node-specific customizations, such as Internet Protocol (IP) addresses.

Windows Capabilities. Windows 2000 includes the capability to remotely install all cluster nodes and to distribute layered software. Remote Installation Service (<http://www.microsoft.com/technet/win2000/win2ksrv/>

“Typically, clusters are chosen with price performance in mind, and the problems that are solved on clusters are those that are ported from multicomputer environments, though applications targeted especially for clustered architectures are slowly taking off.”

remosad.asp) allows an administrator to generate a system installation image and then push it to new cluster nodes. There are third-party tools also, such as Norton's Ghost (http://www.symantec.com/techsupp/ghost/files_ghost) and Imagecast (<http://www.storagesoft.com/support/documentation.asp>), that perform the same function.

Windows 2000 also has a built-in capability to "package" a software installation and then distribute it to all cluster nodes. The way it is implemented is particularly convenient. Windows keeps track of where in the installation a particular package is on a node. If the network fails or the node crashes, Windows will resume the installation where it left off.

A real strength of the Windows system and software distribution is the way it has been integrated into the Windows 2000 operating system, making it easy for anyone to take advantage of.

UNIX Capabilities. The Linux installation basically consists of the following: describe the hardware, configure the software, and load, load, load. It can be tedious for one machine yet completely unbearable and unnecessary for the multiple, mostly identical, cluster nodes. There are presently three approaches to solving this problem:

1. The machine vendor-supplied disk duplication
2. The use of a network disk installation or image replication package
3. The use of a self-install prepackaged Beowulf2 cluster-in-a-box solution

While having a vendor duplicate a cluster node image sounds rather simple, there are still some problems to avoid. The first hurdle to clear is to obtain a machine identical to those to be used in the cluster. The best way to do this is to convince a vendor that an initial loan machine is part of the purchase prequalification process. When the machine arrives, install the operating system and the entire necessary programming environment that will reside on a cluster node. Then, back up that disk and either give the disk or the entire system to the vendor so that they may use their disk duplication system to copy the disk contents to all purchased cluster nodes. Another advantage of doing this, in addition to the apparent time saved, is that a vendor may keep the base disk image and provide all future machines of the same hardware configuration with the same software installation. There will be nothing further to do when the machines arrive if the cluster is set up to do dynamic IP via a DHCP server. However, if using statically allocated IPs, each machine will require its net-

work configuration to be reset after initial boot (and then be rebooted before use).

Another mechanism is to require the vendor to write the MAC address on the outside of each system box. This makes setting up static IP with a specific node numbering scheme much simpler. The alternative is to find it written on the NIC or to boot each machine one at a time and capture the MAC address as the machine starts.

A final note here that cannot be overemphasized—find a knowledgeable Linux vendor. Although it may save money going to the lowest bidder, it may not be such a good deal if the vendor simply hits the power switch on a Linux box after the burn-in period without going through the proper shutdown procedure. You may find out whether they did this during the initial cluster power-on as a number of nodes will fail to boot without direct user interaction.

The second method (and one of the options if the cluster is not a new purchase) is to use one of the tools to do either a replicated network installation or network disk image duplication. Linux Utility for cluster Install (LUI) (<http://oss.software.ibm.com/developerworks/opensource/linux/projects/lui/>) is an open-source tool developed by IBM to perform a replicated network install across cluster nodes. VA SystemImager (<http://systemimager.sourceforge.net/>) is a network disk image duplication tool available as open source from VA Linux.

LUI installation consists of unpacking a tar file, soon to be available via RPM, onto the machine that is to become the LUI server. The LUI server will be the machine from which cluster builds will be orchestrated and will maintain a database of cluster configurations and resources. To use LUI, the server, the cluster (client) nodes, and the resources to load onto the client cluster nodes must be configured. LUI provides a Graphical User Interface (GUI) for this purpose. The server information required is the following:

- Machine name
- IP address
- Net mask

The cluster node information required is as follows:

- Machine name
- IP address
- Net mask
- MAC address
- Fully qualified hostname

The optional node information includes the following:

- The default route
- Installation gateway
- Number of processors
- PBS string

The last two data fields are included as a result of collaboration with the Open Source Cluster Application Resources (OSCAR) (<http://www.epm.ornl.gov/oscaror> <http://www.OpenClusterGroup.org>) project and are used for the automated OSCAR install of OpenPBS (<http://www.openpbs.org>). Various resources may be specified, including the following:

- Disk partition table
- User-specified exit code
- File system
- Kernel and kernel map
- Ramdisk
- RPM
- Source (copied as a file)

The final GUI setup step requires that resources be associated with each cluster node. A future release of LUI will include a grouping feature so that nodes may be grouped together and then the group associated with a set of resources. To effect node installation, one need to only network boot the cluster client nodes; LUI automatically installs the specified resources. This of course requires a PXE-enabled network card (<ftp://download.intel.com/ial/wfm/pxespec.pdf>) and BIOS capable of its use. LUI has a number of significant advantages over disk image solutions—one is that it stores the configuration, very small in comparison to a disk image, and rebuilds each time for installation. Because the LUI server need only be online during the installation process, a computer outside of the normal cluster resources may be used to “service” the installation of the actual cluster nodes. This machine could be the cluster administrator’s desktop or notebook machine. One negative aspect of LUI is that although it has the ability to move any software package across the cluster during installation, it does not have the ability to automatically install that package on each cluster node. This is a major drawback with respect to passing an entire computing environment across the cluster.

VA Systemimager is a network disk image duplication tool developed by VA Linux. Systemimager must be in-

stalled on the image server, and the Systemimager client must be installed on the master client. The image server will be the machine to initiate the image duplication process as well as holding the storage of all disk images. The master client is the cluster node that is configured and maintained as the example node to be distributed to all other cluster nodes. In addition to the installation of the appropriate Systemimager packages on the image server and master client, all the final software needed on the cluster must be installed on the master node before it is loaded across the cluster. This process is similar to that used with the vendor disk duplication discussed earlier. Once the master client is fully configured, its image may be pulled from the master client to the image server. All cluster nodes may now be booted via a diskette, CD, or the network; each will pull the specified image from the image server. As with LUI, a PXE-enabled network card and BIOS are required for the network boot option.

While neither LUI nor Systemimager provides all the desired capabilities, they complement one another in numerous ways and, when combined with other tools, provide the ability to extract the best of each. This is the approach taken with the OSCAR project.

The third method of building a complete cluster-computing environment is that of using one of the self-install prepackaged Beowulf cluster-in-a-box solutions. A number of companies, including Scyld (<http://www.scyld.com>) and Turbolinux (<http://www.turbolinux.com>), have started to market commercial cluster solutions that include some type of auto-install feature. However, these options tend to contain proprietary solutions for building a Beowulf cluster and are rather expensive (as compared with a no-cost solution). OSCAR is an open-source consortium solution that provides an auto-install cluster-on-a-CD that contains the current “best of practice” for clusters consisting of less than 64 nodes. OSCAR consists of an LUI-initiated install and subsequent configuration of cluster nodes with the following cluster-computing environment: M3C (<http://www.epm.ornl.gov/torc/M3C/M3Chomepage>), C³ (<http://www.epm.ornl.gov/torc/C3>), OpenPBS, SSH/SSL, DHCP, NFS, TFTP, RSYNC (<http://rsync.samba.org>), Systemimager, Parallel Virtual Machine (PVM), and MPICH. The configuration consists of providing LUIs required and optional information via the LUI GUI. Installation and configuration then take place in a similar fashion to a standard LUI installation. The difference from the LUI-only installation is that when finished, the entire cluster-computing environment is configured.

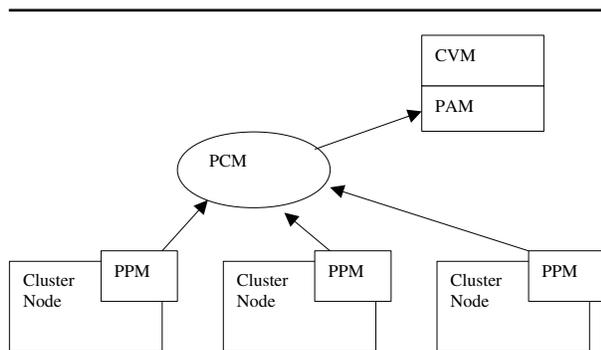


Fig. 3 Organization of typical system performance monitoring tools (PPM = Performance Probing Module, PCM = Performance Collection Module, PAM = Performance API Library Module, CVM = Control and Visualization Module)

3.3 SYSTEM MONITORING AND REMOTE CONTROL OF NODES

System performance monitoring is the act of collecting the cluster's system performance parameters, such as the node's CPU utilization, memory usage, I/O, and interrupts rate, and presenting them in a form that can be easily retrieved or displayed by the system administrator. This feature is important for the stable operation of large clusters since it allows the system administrator to spot potential problems earlier. Also, other parts of the system's software, such as task scheduling, can benefit from the information provided (e.g., using it to perform better load balancing).

Structure of Monitoring Systems. The basic system-monitoring tool consists of the components shown in Figure 3.

- **Performance Probing Module (PPM):** This is part of the monitoring system that interfaces with the local operating system to obtain the performance information. The information obtained is later sent or collected by other monitoring subsystems. In many implementations, PPM also performs certain checking-on-node conditions and generates event notification to indicate the occurrence of certain conditions.
- **Performance Collection Module (PCM):** This part of the monitoring system acts as an information collector that collects information from every node and stores it for later retrieval. One of its main functions is to cache performance information that reduces the overall impact of monitoring on the system being monitored.
- **Performance API (Application Programming Interface) Library Module (PAM):** This is a library that allows a programmer to write applications to access the performance information.
- **Control and Visualization Module (CVM):** This is an application that presents the performance information to the user in a meaningful graphical or text format. Another function that the CVM performs allows a user to control the set of nodes and information that the user wants to see. One example of this module from software, called SCMS (Uthayopas, Maneesilp, and Ingongnam, 2000), is illustrated in Figure 4.

Obtaining the Performance Information. There are many techniques being used by the developer to obtain performance information from the systems. These techniques are summarized in Figure 5.

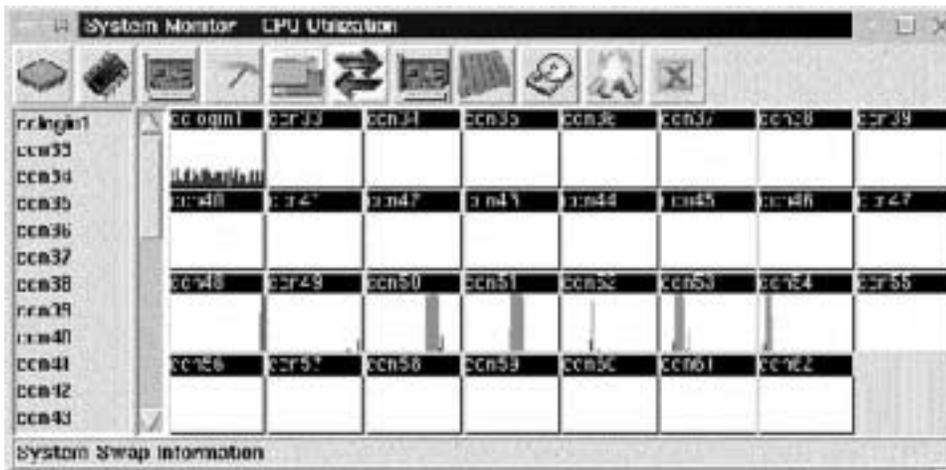


Fig. 4 Control and Visualization Module portion of SCMS (showing CPU load on 31 of 256 nodes on Chiba City Linux cluster, Argonne National Laboratory)

Probing by Direct Access to Kernel Memory. In this approach, the data are obtained by reading them directly from a known address (using an interface such as `/dev/kmem`). This approach makes probing depend on the kernel version. Even a slight change in the kernel can cause a failure of probing the system. To avoid this problem, most modern kernels usually provide a system performance interface API. The advantage of this approach is the speed of information collection.

Some recent tools (Astalos and Hluchy, 2000) try to make fast access to kernel information by using a custom device driver that directly accesses kernel information but keeps the application interface API fixed. The benefit of this is fast access and kernel independence. The device driver, however, must be kept updated with kernel changes. This is rather difficult considering the fast pace of kernel development in the Linux environment.

Probing by File System Interface. Most UNIX systems, including Linux, currently have an interface through a virtual file system named `/proc`. Information from inside the kernel such as CPU, process, file, and networking statistics will appear in `/proc` for users to access. The advantages of this approach are better portability and kernel independence. The location of information found in `/proc` tends to be the same for all kernel versions. Moreover, the speed of access is comparable to direct kernel access, since `/proc` is a virtual interface through the Virtual File System that goes directly into kernel memory. Some

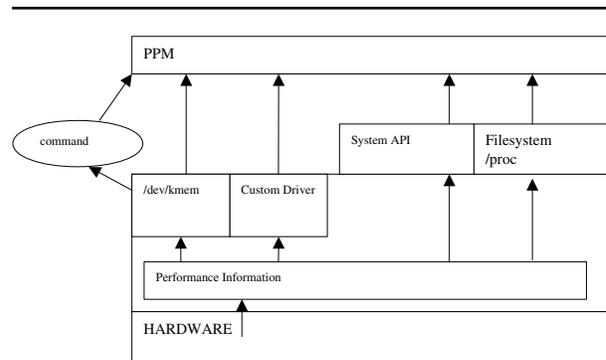


Fig. 5 An example of how to read system information (PPM = Performance Probing Module, API = Application Programming Interface)

projects such as SMILE (<http://smile.cpe.ku.ac.th/research/scms1.2.2/>) also developed a library layer (Hardware Abstraction Layer—HAL) on top of `/proc` that allows better portability and ease of programming.

Probing by System APIs. Certain operating systems, such as Microsoft Windows, provide APIs such as Windows Management and Instrumentation (WMI) (<http://www.microsoft.com/hwdev/WMI/>) to gather system information. The advantages of using such an API are ease of programming and kernel independence. WMI is implemented as a distributed COM object that can easily be called from almost any programming and scripting language under Windows. There are similar APIs available in the UNIX environment but none quite as comprehensive or consistently implemented as WMI.

Probing by System Command. Some UNIX-based tools use the result generated from the execution of standard UNIX command such as `uptime` or `ps` to probe system performance. This can provide portability between families of UNIX systems. However, this is an inefficient approach, since it places a heavy load on the system every time the commands are executed. This approach is by and large used in primitive tools or used in early development prototypes before more efficient methods are implemented.

Collecting the Performance Information. Once the system information has been read on each node, it must be delivered to the application that requests it.

A choice to have the application perform the collection of information itself or have some subsystem collect and maintain the information for the application needs to be made. Most monitoring systems opt for the latter case, since there are many benefits to be gained, such as reducing the network traffic and ease of management for multiple sessions. But its choice may limit scalability, as will be discussed later.

The act of collecting the information itself is a collective operation (e.g., `gather/scatter`). So far, most monitoring systems rely on point-to-point communication from nodes to the information collector. High-level communication systems, such as Message-Passing Interface (MPI), have well-defined and efficient collective operations. The use of MPI might not be possible yet, since MPI is not dynamic enough to cope with some situations. For instance, if the monitoring system used MPI, when one of the nodes dies, every MPI process must be terminated and the monitoring system must be restarted. Currently, MPI has no clear interface for the interaction with processes outside the MPI scope. So, it is not possible at this time to write a hybrid and flexible MPI-based system management tool.

Scalability. Many works on monitoring systems (Uthayopas, Maneesilp, and Ingongnam, 2000; Astalos and Hluchy, 2000; <http://vacm.sourceforge.net>) are based on a centralized performance collection paradigm. This simple implementation works efficiently for a small- to medium-sized cluster (16-64 nodes). However, the trend is to build ever-larger clusters. Currently, 64- to 256-node systems are not uncommon. Some clusters have more than 1000 nodes (Reisen et al., 1999; Bennett et al., 1999). With these large clusters, the centralized approach will be inadequate. One technique suggested by Liang, Sun, and Wang (1999) is to use the cascading hierarchy of monitoring domains. In this approach, the monitoring is divided into domains. Each domain consists of a set of nodes and a *monitor proxy* that receives monitoring requests from higher domains, merges and sends information upward, and handles fault events and forwards them to the upper level proxy.

Optimizing the Network Traffic. There are many techniques being used by various performance monitoring systems to optimize network traffic. Examples of these techniques are the following:

- *Adjusting the monitoring frequency:* The function of the monitoring system is to present the current system state of the cluster. This tends to be a nontrivial task, since the state of the cluster consists of hundreds of continuously changing variables. Hence, the precision of monitoring depends on the frequency with which the monitoring system collects the information. However, more up to date information usually means a higher frequency of information collection. This will generate a high traffic load for the system. Some systems use a PCM to separate the frequency of requesting the information by the application program from the real collection frequency of information cluster nodes. This approach allows the tuning of the level of accuracy required against the network traffic.
- *Data set selection:* In a practical system, a system administrator usually wants to observe only a certain number of performance parameters. Therefore, allowing the users to select only information they are interested in can reduce the amount of data movement. This support is typically built into the API and user interface to permit users to mask/filter the information they need. This mask or filter must be sent to each node to notify PBM so it sends back only information needed. Certain Boolean operations such as AND or OR might be needed to combine the masks from multiple applications running concurrently and requesting the infor-

mation from the monitoring system at the same time. This feature appears in Co-Pilot Overview (<http://www.sgi.com/software/co-pilot/>).

- *Data merging*: In Liang, Sun, and Wang (1999), a hierarchical structure of monitoring was used. A benefit of this, apart from higher scalability, is the ability to merge redundant information collected in each level of the hierarchy. By merging this information, the traffic is reduced.

Reducing the Intrusiveness. Performance monitoring must be minimally intrusive to the system it is monitoring. The sources of intrusiveness are the reading of system information and the execution of event notification. When reading the system, information using `/proc` usually causes a low impact on the system, since this requires only the memory access. However, in Astalos and Hluchy (2000), the level of intrusiveness was reduced further by using a custom driver that collects system information and sends all of it directly to the probing module. The trade-off between low intrusiveness and portability is an issue here. For the execution of event notification, the intrusiveness usually depends on the complexity of rule, number of events, frequency of checking, and efficiency of implementation. The only optimization possible is to select an appropriate number of events and the best interval of checking. In general, one possible technique is to distribute the execution of certain parts of the monitoring system to dedicated hosts (e.g., assigning one node to be a management node and then executing PCM, PAM, and CVM on that node).

Web Technology and Cluster Management/Monitoring Systems. Web-based technologies provide many benefits to users such as the following:

- Internet access to system management and monitoring capabilities
- Browser-based user interface
- Capability to interact with the monitoring system through low-speed communication channel

A typical web-based monitoring system is illustrated in Figure 6 (this design is used by KCAP [<http://smile.cpe.ku.ac.th/research/kcap/>]).

In this configuration, a dedicated node called management node runs the PCM and web server. The Java applet is the interface to connect to the real-time monitoring system. As the user accesses a certain web page, the Java applet is loaded. This applet makes the connection back to the PCM, gets the system information, and presents it to the user



“Performance monitoring must be minimally intrusive to the system it is monitoring. The sources of intrusiveness are the reading of system information and the execution of event notification.”

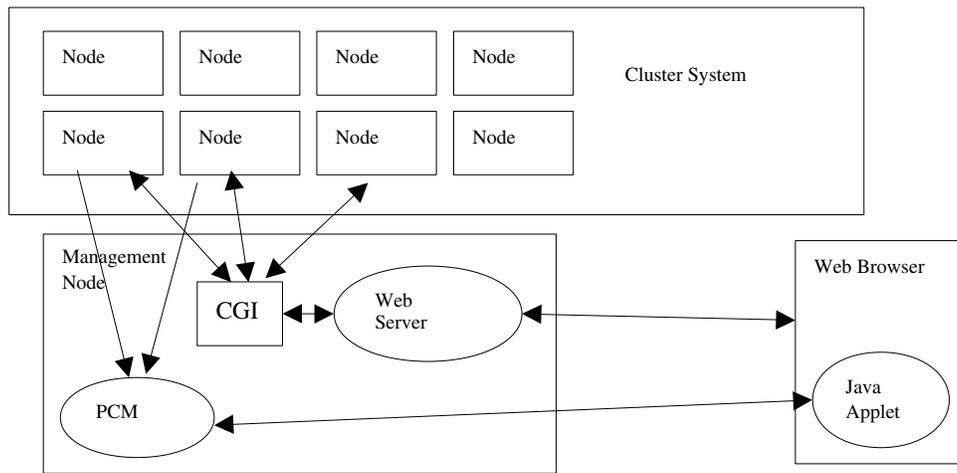


Fig. 6 Structure of a web-based management and monitoring system (PCM = Performance Collection Module)

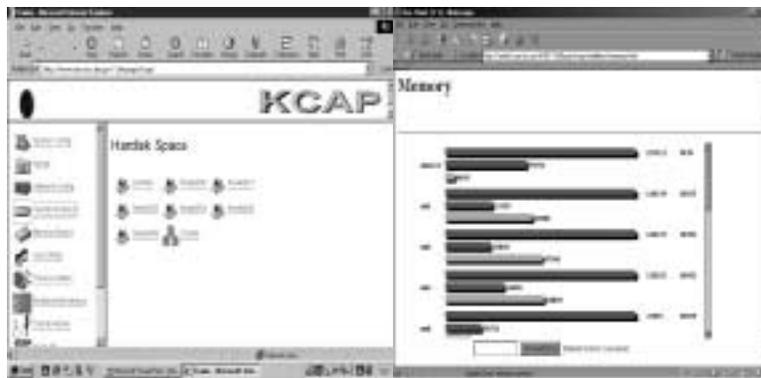


Fig. 7 Example of a web-based management and monitoring system

graphically. For some other services, such as checking the number of users and the number of running processes, CGI-bin can be used instead. Nevertheless, the system administrator must set up security to allow the remote execution on the cluster nodes. This may not be ideal when high security is required of the system.

Emerging web standards such as VRML97 (<http://www.web3d.org>) can be used to increase the information delivered to the user by allowing a user to navigate around the system to be monitored. By interfacing Java using the External Authoring Interface (<http://www.web3d.org/WorkingGroups/vrml-eai/>), dynamics behavior of the

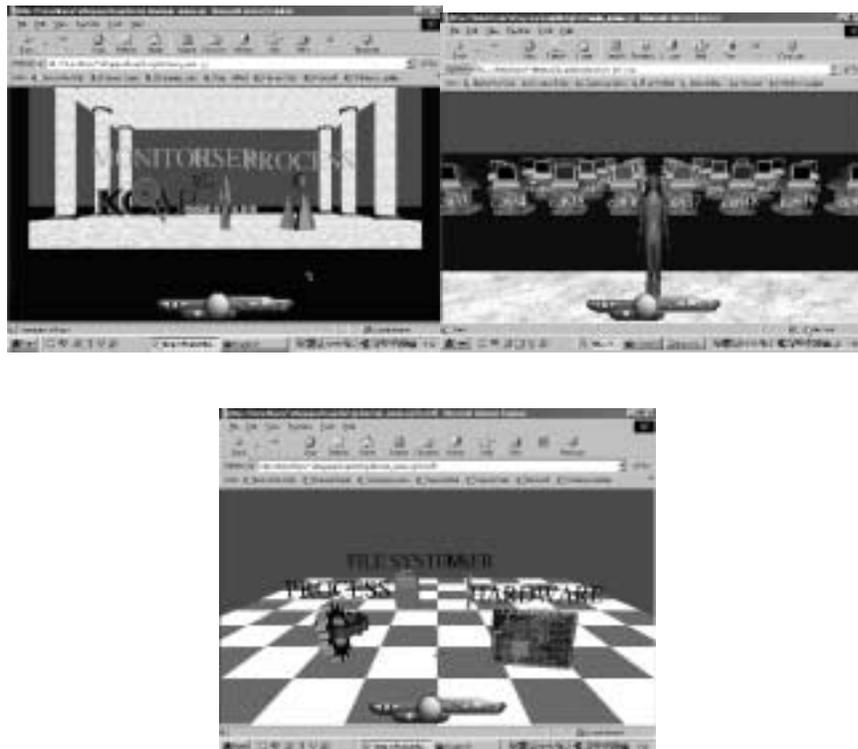


Fig. 8 Walk-in visualization of Chiba City generated by KCAP3D: (a) 3D icon showing global function, (b) a cluster farm, (c) inside a node

cluster can be displayed in real time. This method was used in the KCAP3D (<http://smile.cpe.ku.ac.th/research/kcap/>) package to visualize a cluster. In this system, the cluster configuration is converted into navigable VRML using a Perl-based parser. This approach allows users to select and view components that they want to interact with in 3D.

4 Remote Management: Tools and Technology

On a UNIX-based system, any software that resided on each node can be easily accessed using X Windows. However, the situation is somewhat different for UNIX commands. In the past, remote execution has generally been done using traditional commands such as `rsh` and `rlogin`. The system administrator, however, needs to relax security to allow cluster-wide execution of these commands. This is fine for clusters that are behind firewalls, but it might present a security risk for open clusters. It should be noted that

the speed of `rsh` execution depends on how users are validated. In the systems that use a centralized server, such as NIS, executing hundreds of `rsh` sessions on remote nodes will slow due to the interaction with NIS. So, many systems solve this problem by replicating control files, such as `/etc/passwd`, `/etc/group`, and `/etc/shows` instead. The advantage of this is twofold. First, because authorization is done locally, remote access for short-lived tasks is speeded up substantially. Second, by removing the centralized NIS bottleneck, a user can login to any node to interact with the central NIS service.

Traditional UNIX commands do not support the model of parallel computing. There is an effort to extend and standardize UNIX command over the parallel computing platform's Parallel Tools Consortium project called Scalable UNIX Tool (Gropp and Lusk, 1994). The current implementation of this tool is based mostly on shell scripts, which are slow and do not scale well. There are two major requirements for fast implementation of scalable UNIX tools:

- Fast process startup
- Efficient collective communication

For the fast process startup, this seems to be the duty of cluster middleware and operating systems. Collective operations seem to be at the heart of cluster tools. There is a need for tools to start tasks on multiple nodes and collect the results quickly to a central point. Therefore, fast collective communications are extremely important. MPI (<http://www-unix.mcs.anl.gov/mpi/mpich/>; <http://www.mpi.nd.edu/lam/>) and PVM implementations seem to be a good choice for tools to implement fast and scalable UNIX commands.

Naming is also another important issue. To operate efficiently in cluster environment, the tools must have the following:

- The ability to name one object (e.g., node, file)
- The ability to name set of related object (e.g., `ps` to a set of nodes)

Traditional UNIX commands rely on regular expression to address single and multiple objects. The Scalable UNIX Tools project also proposes the use of enhanced regular expression to handle the selection of nodes for command execution. However, this is not enough for large clusters, which might not use uniform naming. In this case, packages such as SCMS handle this by using GUI instead. The advantage of this is that users can freely se-

lect unrelated nodes for command execution. This tool indicates growing importance of the GUI component to the system administration tool.

4.1 CLUSTER COMMAND AND CONTROL (C³)

The Cluster Command & Control (C³) tool suite from Oak Ridge National Laboratory (ORNL) provides capabilities that may be used to perform various system administration and user tasks across single and multiple clusters for tasks such as the following:

- Remote monitoring and control of nodes
- Copy/move/remove files
- Remote shutdown/reboot of nodes
- Security maintenance
- Parallel execution and invocation of any cluster-wide task

The true value of the C³ tools was shown recently when applying a security patch across the 64-node ORNL cluster HighTorc (<http://www.epm.ornl.gov/torc>). The patch was applied in parallel via the `cl_exec` command in less than 1 minute.

Eight command-line general user tools have been developed toward this effort with both a serial and parallel version of each tool. The serial version aids in debugging user tasks as it executes in a deterministic fashion, while the parallel version provides better performance. Briefly, these tools are as follows:

- `cl_pushimage` is the single machine *push* in contrast to the Systemimager *pull* image solution.
- `cl_shutdown` will shut down or reboot nodes specified in command arguments.
- The `cl_pushimage` and `cl_shutdown` are both root user system administrator tools.

The other six tools—`cl_push`, `cl_rm`, `cl_get`, `cl_ps`, `cl_kill`, and `cl_exec`—may be employed by any cluster user at both the system and application levels.

- `cl_push` enables a user to push individual files or directories across the cluster.
- `cl_rm` will permit the deletion of files or directories on the cluster.
- `cl_get` copies cluster-based files to a user-specified location.

- `cl_ps` returns the aggregate result of the `ps` command run on each cluster node.
- `cl_kill` is used to terminate a given task across the cluster.
- `cl_exec` is the C³ general utility in that it enables the execution of any command across the cluster.

The parallel version of each tool is invoked by appending a “t” to the end of the command. For example, the parallel version of the `cl_push` command is `cl_push.t`. From the command line, each C³ tool may specify cluster nodes to operate on as individual cluster nodes or a file node list or use the C³ default file `/etc/c3.conf` node list.

`cl_pushimage` enables a system administrator logged in as root to *push* a cluster node image across a specified set of cluster nodes and optionally reboot those systems. This tool is built on and leverages the capabilities of Systemimager. While Systemimager provides much of the functionality in this area, it fell short in that it did not enable a single point *push* for image transfer. `cl_pushimage` essentially *pushes* a request to each participating cluster node to *pull* an image from the image server. Each node then invokes the *pull* of the image from the outside cluster image server.

`cl_shutdown` avoids the problem of manually talking to each of the cluster nodes during a shutdown process. As an added benefit, many motherboards now support an automatic powerdown after a halt—resulting in an “issue one command and walk away” administration for cluster shutdown. `cl_shutdown` may also be used to reboot the cluster after pushing a new operating system across with `cl_pushimage`.

`cl_push` allows the user to *push* files and entire directories across cluster nodes. `cl_push` uses `rsync` to push files from server to cluster node.

The converse of `cl_push` is the `cl_get` command. This command will retrieve the given files from each node and deposit them in a specified directory location. Because all files will originally have the same name, only from different nodes, each file name has an underscore and IP or domain name appended to its tail. IP or domain name depends on which is specified in the cluster specification file. Note that `cl_get` operates only on files and ignores subdirectories and links.

`cl_rm` is the cluster version of the `rm` delete file/directory command. This command will go out across the cluster and attempt to delete the file(s) or directory target in a given location across all specified cluster nodes. By default, no error is returned in the case of not finding the target. The interactive mode of `rm` is not supplied in

`cl_rm` because of the potential problems associated with numerous nodes asking for delete confirmation.

The `cl_ps` utility executes the `ps` command on each node of the cluster with the options specified by the user. For each node, the output is stored in `/$HOME/ps_output`. A `cl_get` is then issued for the `ps_output` file, returning each of these to the caller with the node ID appended per the `cl_get` command. The `cl_rm` is then issued to purge the `ps_output` files from each of the cluster nodes.

The `cl_kill` utility runs the `kill` command on each of the cluster nodes for a specified process name. Unlike the `kill` command, `cl_kill` must use the process name as the process ID will most likely be different on the various cluster nodes. Root user has the ability to further indicate a specific user in addition to process name. This enables the root to kill a user’s process by name and not affect other processes with the same name but run by other users. The root user may also use signals to effectively do a broad-based kill command.

`cl_exec` is the general utility tool of the C³ suite in that it enables the execution of any command on each cluster node. As such, `cl_exec` may be considered the cluster version of `rsh`. A string passed to `cl_exec` is executed “as is” on each node. This provides flexibility in both the format of command output and the arguments passed into each instruction. The `cl_exec.t` provides a parallel execution of the command provided.

5 Scheduling Systems

It is not an unusual practice that users manually reserve the nodes they need for running their application in the absence of any specialized tools by announcing their intent to use these nodes through e-mail or a message board to users prior to starting their job. Such practice is acceptable as long as everyone abides by the protocol and resolves any usage conflicts in a friendly manner. However, as organizations grow larger and the complexities in hardware, software, and user base increase, these kinds of arrangements may not be too efficient or effective at all. What is needed is an automated tool that will accept user requests to reserve cluster resources and execute their jobs automatically, without much human intervention. This collection of tools is part of any well-designed scheduling system.

Apart from scheduling user jobs, a scheduler is expected to arbitrate access to the cluster to optimize the resource usage, provide security, and perform accounting. Based on the particular job’s resource requirements, the

scheduler allocates an optimized set of resources dynamically from the available set of resources. This optimization is deemed to be important, especially when the cluster consists of a large set of heterogeneous resources, and administration wants to impose a certain group policy on the resource usage that the scheduler should take into consideration before allocating the resources to the job.

More often, the scheduling software is integrated into the cluster management (Baker, Fox, and Yau, 1996) middleware itself, so that the scheduler can make its resource allocation decisions based on feedback received from the management modules—mainly about resource availability and their load characteristics. The feedback may be used by the scheduler to statically load balance the cluster nodes, apart from deciding the feasibility of running the job on them. In this respect, the cluster scheduler undergoes the same decision process the typical operating system scheduler undergoes:

- Which job should it execute next from the ready queue(s)?
- Are there enough resources to meet the job requirements?
- How to allocate the most optimal resources to the job?
- Does the owner of the job have enough privileges to use the resources?
- Does the job meet its accounting restrictions?
- If preemption is permitted, which running job to preempt?

Note that some or all of these decisions are strongly influenced by one or more of the following factors:

- Underlying cluster architecture
- Operating system support
- Administrative policy on resource usage
- Support for multiple queues and/or multiple jobs per node
- Degree of heterogeneity in the cluster
- Support for priorities on jobs

A variety of scheduling technologies are available, ranging from basic batch job schedulers to dynamic load balancers. Most of the schedulers are designed to work in a clustered as well as in a multiprocessor environment:

- CODINE (<http://www.gridware.com/product/codine.htm>), for instance, can operate in a heterogeneous environment to schedule the nodes in a clustered

symmetric multiprocessor (SMP) as well as the processors on a Vector supercomputer. It also provides tools for managing the cluster policy and dynamically load balancing the jobs.

- CONDOR (Basney and Livny, 1999), on the other hand, works to improve the throughput of the resources by scheduling jobs on idle workstations. It can also checkpoint and dynamically migrate jobs across workstations.
- The Portable Batch System (PBS) (<http://pbs.mrj.com/overview.html>) has a flexible scheduler module that can be replaced by the site-specific scheduler to incorporate the site's own scheduling policy. The Basic Scheduling Language is a scripting language provided in PBS to be used to write the custom scheduler, apart from using other more familiar languages such as C and Tcl. PBS also supports the creation of multiple job queues, so that access control lists can be used on these queues to control which users have access to submit jobs to these queues. The scheduler can be redesigned to process the queues in a prioritized fashion.

6 Conclusions

Many interesting and complex issues are involved in the creation and utilization of commodity clusters, and system administration drives much of these issues. Clusters of small scale (e.g., less than 32 interconnected nodes) are widespread, and larger systems (e.g., up to 1000 or more interconnected nodes) are becoming prevalent today. All the major building blocks needed for successful clustering exist somewhere or another:

- Cost-effective workstations and small-scale SMP servers
- High-speed networks with low-latency, high-bandwidth, and acceptable CPU overhead
- Operating systems (with choice of Linux, other UNIX variants, or Windows)
- Middleware for message passing based on the MPI standard
- Cluster scheduler batch and interactive systems
- Software and system understanding tools
- System maintenance tools for clusters and appropriate device drivers

Successful projects are evidently those that take a balance of cost, component-oriented risk, capability per system, and scale to deliver to their users the best production environment possible for the investment made or that clearly

emphasize the “research” nature of clusters for computer science–centric work. System administration’s role in selecting, operating, and presenting the cluster resource to users cannot be underestimated in the success of a given system.

The state of the art today involves a lot of picking and choosing in each area of concern, whether one does a “do it yourself cluster” or works with an integrator/cluster OEM. The ability to break out the capital and maintenance cost of each item (some of which are free under the Linux strategy) allows users to control their investments according to their expected uses. However, the self-integration approach can also lead to problems and higher cost of ownership and perhaps more expenses in terms of human resources for administration. The quality of the underlying software and hardware components and the quality of their integrated behavior all drive the percentage of available cycles that will actually be available to be used by the audience of a specific cluster installation. Quick upgrade, recovery, and other factors enabled by diagnostic and support tools help this area. Well-understood procedures for debugging and repairing clusters, enabled by maintenance tools, are clearly essential to system administration.

Typical cluster sites dedicated as production platforms definitely receive benefit from packaged solutions and support, while many academic and research clusters thrive on the mix-and-match combination of freeware, low-cost networks, and inexpensive workstation components (“beige boxes”). Software tools, middleware, and maintenance tools, both freeware and commercial software, are emerging to address the Linux, other UNIX, and Windows markets. Significant expansion of options and of packages among software tools, systems, and networks are likely to emerge in the future, offering packaged cluster solutions that include system maintenance tools. It is possible to offer the familiar “UNIX command line” and tool chain both in Linux and under Windows; Windows offers other possibilities that may prove attractive as cluster users move into enterprises and need to integrate with desktop systems or use desktop systems as so-called supercomputers at night.

At present, the freeware world still relies heavily on software ported from earlier generations of multicomputers and multiprocessors; there are not enough new freeware items being created, nor do existing freeware products necessarily track hardware technology changes fast enough for cluster users. This causes a benefit gap that is evidently only going to be filled by software created based on demand and a market willing to pay for support. Training and general knowledge of system administrators are likely to increase over time, driven strongly by the

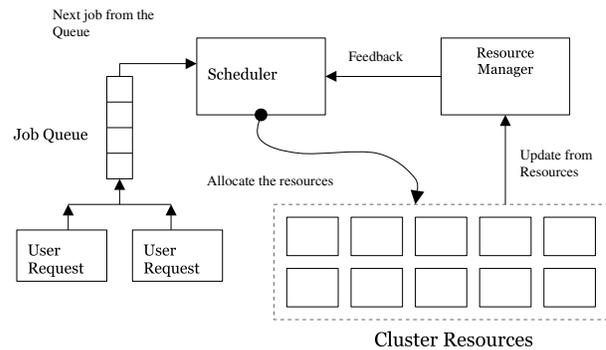


Fig. 9 Conceptual architecture of a cluster scheduler and its managed resources

“The state of the art today involves a lot of picking and choosing in each area of concern, whether one does a “do it yourself cluster” or works with an integrator/cluster OEM.”

growing user base and popularity of clusters. Interest among integrators and other OEMs to provide complete solutions are obviously growing. The willingness of users to run their applications on clusters will only widen if the system maintenance tools, middleware, and runtime environments grow in their user friendliness, allowing users with less and less cluster-specific knowledge to reap significant benefits while remaining relatively novice about the cluster's complexity. Clearly, many more killer applications and avid users will emerge as clusters become easier to use and well integrated into computational environments with common features, capabilities, and well-defined maintainability. The options, choices, and state of the art described in this article point system administration in directions that, it is hoped, will lead to such easier-to-use cluster environments in the not-too-distant future.

BIOGRAPHIES

Anthony Skjellum received a B.S. from the California Institute of Technology in 1984 in physics, an M.S. in chemical engineering in 1985, and a Ph.D. in chemical engineering with a computer science minor in 1990. He has worked in scientific and high performance computing for the past 17 years, the last 8 of which have been at Mississippi State University as a professor of computer science, and 3 earlier at the DOE Lawrence Livermore National Laboratory. He has made specific contributions in the area of portable, high performance message-passing specifications and systems (MPI Forums, MPI/RT Forum), including the widely used MPICH software jointly designed at Mississippi State University and Argonne National Laboratory. His research group at Mississippi State has created a number of freeware libraries for scientific computing, including MPICH as well as object-oriented middleware for sparse, parallel, and sequential linear algebra (PMLP). In 1996, he formed MPI Software Technology, Inc., which pursues commercial off-the-shelf software for message passing and mathematical libraries. He has received funding from NSF, NASA, DARPA, DOD, DOE, Intel, and others for research and advanced prototyping in scientific and high performance computing.

Rossen Dimitrov is a Ph.D. candidate in computer science at Mississippi State University. He received an M.S. in computer science from Mississippi State University in 1997 and a B.S. in electrical engineering from HIMEE Varna, Bulgaria, in 1992. His interests are in advanced networking and message-passing research, parallel and distributed computing, network protocols, and security in distributed high performance systems. He is an expert in the area of gigabit-per-second networks and the Virtual Interface (VI) Architecture—an industry standard for high performance system-area networks. He has experience in developing Microsoft Windows NT device drivers; he created the first Windows NT driver for Myrinet PCI boards. He has also been in-

involved in studying communications efficiency of the interaction between network interface adapters and host platforms and operating systems, which included the development of Myrinet Control Programs. He has studied the challenges that new high performance message-passing systems impose on protection and security in distributed environments. He has also participated in the PacketWay research effort. Recently, he developed an MPI-1 implementation for VI architecture networks, studying the impact of different protocols and transmission modes on MPI performance. Also, he participated in the development of MPI SMP and TCP devices for Windows NT and Linux operating systems. Currently, he is involved in the design of IMPI and MPI-2 implementations. He has expertise in the areas of operating system concepts, low-level network and system interaction, network protocols, and multithreaded communications systems design.

Srihari Angaluri holds a bachelor's degree in computer science from Osmania University and is working toward a master's in computer science at Mississippi State University. He also works as a software engineer at MPI Software Technology, Inc. His research interests include cluster job scheduling, resource management, and object-oriented interoperable computing. During his work at the Cornell Theory Center (CTC), Ithaca, New York, he developed tools for job scheduling and resource management, which are currently being used to manage the AC3 velocity cluster at CTC, one of the largest Windows clusters in the world. He is currently involved in developing interoperable cluster management tools with COM+ and CORBA-based technologies.

David Lifka is an associate director at the Cornell Theory Center (CTC). He has played a major role in the formation of the Advanced Cluster Computing Consortium, AC3, formed in conjunction with Dell, Microsoft, and Intel. He manages the systems group at CTC, which runs the production Windows 2000 clusters there. With a Ph.D. in computer science from the Illinois Institute of Technology, Dr. Lifka has previously worked at Argonne, where he wrote the EASY scheduler used at IBM SP sites worldwide. Most recently, he developed Cluster Controller, a new resource management system for Windows 2000 clusters, now commercially available from MPI Software Technology, Inc. He is on the advisory boards of the IEEE Task Force on Cluster Computing and Giganet, Inc.

George Coulouris began working at the Cornell Theory Center (CTC) as an undergraduate in 1996. After working for several years supporting the Center's IBM SP, he became involved with the development of Cluster CoNtroller for Windows NT. After completing his undergraduate work in 1999, he accepted a full-time position at CTC as a systems programmer. In addition to systems and network administration, his responsibilities include performance evaluation of IA32 and IA64 systems and porting scientific software to CTC's cluster environment. He participates in user training workshops by giving talks on optimization, tuning, and numerical libraries. He also provides con-

sulting services to users wishing to parallelize and tune their software.

Putchong Uthayopas received bachelor's and master's degrees in electrical engineering from Chulalongkorn University and master's and Ph.D. degrees in computer engineering from the University of Louisiana at Lafayette. His major research interest is in parallel/distributed computing, cluster computing, and parallel software tools. He founded a research group called parallel research group in Thailand in 1996 to explore the cluster-computing technology and application. His research group built a popular cluster management tool called SCMS that was downloaded and used around the world. Recently, his team integrated a 72-node Beowulf cluster called PIRUN at Kasetsart University, which is the most powerful supercomputing system in Thailand. He is now actively involved in cluster-computing activities and serves as a regional executive member of the IEEE Task Force on Cluster Computing. During the summers of 1998, 1999, and 2000, he was a visiting scholar at Argonne National Laboratory. During that time, he worked with an MPICH team on state-of-the-art message-passing system technology. He is a member of the IEEE and Upsilon Pi Epsilon.

Stephen L. Scott is a research scientist in the Distributed Computing Research Group of the Computer Science and Mathematics Division of Oak Ridge National Laboratory. In 1996, he received a Ph.D. in computer science from Kent State University. At Oak Ridge, his responsibilities include directing research and development efforts in high performance scalable cluster computing. His primary research interest is in experimental systems with a focus on high performance, scalable, distributed, heterogeneous, and parallel computing. He is a founding member of the Open Cluster Group—dedicated to bringing current “best practices” in cluster computing to all users via a self-installing cluster-on-a-CD suite. He is also a member of the ACM, the IEEE Computer Society, and the IEEE Task Force on Cluster Computing. Prior to attending graduate school, he worked in various industry positions, including one as principal with a business-to-business chemical database startup company.

Rasit Eskicioglu received a B.Sc. in chemical engineering from Istanbul Technical University and an M.Sc. in computer engineering from Middle East Technical University (Turkey). He is currently writing his long overdue Ph.D. thesis. He has

been in academia, teaching and doing research and development mainly in systems area, for more than 20 years. His research interests include operating systems, parallel and distributed systems, computer architecture, high performance cluster computing, and high-speed networks. His recent work involves investigating ways to make software DSM systems more efficient and scalable using high-speed, programmable interconnects. Most recently, he is investigating storage-area network architectures for highly available file systems. He is a member of the ACM, the IEEE Computer Society, and USENIX.

REFERENCES

- Astalos, J., and Hluchy, L. 2000. CIS—A monitoring system for PC clusters. In *Proceedings of EuroPVM/MPI2000, LNCS 1908*, edited by J. Dongarra et al., 225-232. Heidelberg: Springer.
- Baker, M. A., Fox, G. C., and Yau, H. W. 1996. Review of cluster management software. *NHSE Review* 1 (1).
- Basney, J., and Livny, M. 1999. Deploying a high throughput computing cluster. In *High Performance Cluster Computing*, edited by R. Buyya. Englewood Cliffs, NJ: Prentice Hall.
- Bennett, F., Koza, J., Shipman, J., and Stiffelman, O. 1999. Building a parallel computer system for \$18,000 that performs a half peta-flop per day. Paper presented at the Genetics and Evolutionary Computation Conference, July, Orlando, FL.
- Gropp, W., and Lusk, E. 1994. Scalable UNIX tools on parallel processors. In *SHPCC94: Proceedings of the Scalable High Performance Computing Conference 94*. Computer Society Press.
- Liang, Z., Sun, Y., and Wang, C. 1999. ClusterProbe: An open, flexible and scalable cluster monitoring tool. In *IWCC99: Proceedings of the International Workshop on Cluster Computing 99*, Australia.
- Reisen, R., Fisk, L., Hudson, T., and Otto, J. 1999. Cplant*. Paper presented at the Extreme Linux Workshop, USENIX Annual Technical Conference, June, Monterey, CA.
- Uthayopas, P., Maneesilp, J., and Ingongnam, P. 2000. SCMS: An integrated cluster management tool for Beowulf cluster system. In *Proceedings of the International Conference on Parallel and Distributed Proceeding Techniques and Applications 2000 (PDPTA 2000)*, Las Vegas, NV.