Enabling Production-Level Computing on Linux and Windows Ethernet and Infiniband Clusters with MPI/Pro and ClusterController

Rossen Dimitrov, Srihari Angaluri, Anthony Skjellum
MPI Software Technology, Inc.,
101 S. Lafayette Str., Suite 33, Starkville, MS 39759
{rossen,srihari,tony}@mpi-softtech.com

ABSTRACT

This paper reviews key requirements for production-level cluster computing and how these requirements impact the architecture of message-passing middleware and resource managers used on clusters. In addition to providing high performance and scalability of target applications, system requirements include correctness, strong compliance with standards (both community and industry), robustness (stable operation under heavy and long-term loads), predictability, sufficient security, and efficient use of system resources. This paper distinguishes between research projects in the area of cluster computing, which typically address clusters from a scientific and research point of view, and commercial-grade solutions whose design and architecture are specifically aimed at meeting the requirements of production computing. MPI/Pro and ClusterController are presented here as examples of interoperable commercial-grade solutions that jointly address these issues. Perspectives on the current and future uses of clusters for parallel processing are given, and comparison of user experiences with commercial and freeware are mentioned.

INTRODUCTION

In the past several years, cluster computing has evolved from niche research efforts and demonstrations into a widely used solution for high-performance computing, meeting the needs of production-level environments. Today, numerous business and government organizations rely on clusters for performing their mission-critical parallel computations, whereas dedicated parallel computers were previously the norm. This transition has imposed new requirements for controlling the cost of ownership and predicting the quality and timeliness of the results generated by the cluster. Quality and maturity associated with previous-generation parallel and vector system environments need to recur in cluster systems for them to be widely adopted.

MPI [5] is the most widely used middleware API for message passing on clusters. Vendors of dedicated parallel multicomputers and hybrids/multiprocessors typically offer vendor-specific MPI libraries that are specifically tuned for the target platforms. MPICH [4], the Argonne/Mississippi State model implementation of MPI, and LAM [10] are two of the most popular freely available implementations of MPI used for clusters. These two implementations have been initiated as research efforts and are now maintained by government and/or academic organizations. By way of comparison, MPI/Pro is a commercial grade, vendor-independent MPI product offered by MPI Software Technology, Inc. MPI/Pro is designed to address the needs of clusters used for production purposes.

Resource manager and job scheduler systems are commonly used on production clusters. These systems allow multiple users to gain access to the shared cluster resources according to organization-specific policy. ClusterController is a scheduler for Windows and Linux. Many well-known freeware and a few commercial cluster schedulers exist besides ClusterController, such as PBS [11] in the freeware space, with PBS/Pro [12], and LSF [13] in the commercial space.

The remainder of this paper is organized as follows: Production computing requirements are reviewed, then a discussion of MPI/Pro is offered, explicating how such production requirements are realized; futures are also discussed. ClusterController is then overviewed, with discussion keyed again to production computing requirements and futures. The last three sections are a brief discussion of these products as used in combination, and software quality issues, followed by overall conclusions.

PRODUCTION COMPUTING REQUIREMENTS

Performance and scalability are the most important software requirements of a high-performance cluster computing system. In addition, a number of other requirements need to be met in a production cluster:

- Robustness and correctness,
- Compliance with standards,
- Efficient use of system resources,
- Compatibility with system and application-level software,
- Security of user authentication and impersonation,
- Reliability,
- Flexibility,
- Software evolution and upgrade path, and
- Commercial-grade support.

In contrast, software created in research projects rarely meets the requirements of production environments. The main goal of research projects must clearly be to demonstrate new capabilities and concepts, prove research hypotheses, and/or carry out a comparative analyses. The objective of such efforts is commonly scholarly publications, and they rarely have the objective to provide a complete software implementation ready for production use. The staff of research groups changes often, which results in limited continuity in the software design and architecture. In addition, research groups typically respond to requests from users on a best-effort basis, which cannot guarantee timely resolution of problems.

MPI/PRO

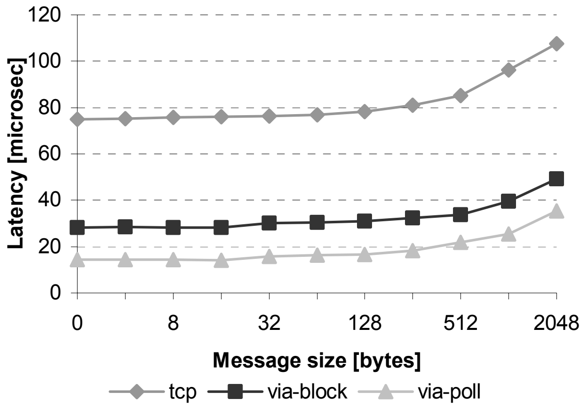
MPI/Pro is a high-performance, scalable implementation of the MPI 1.2 standard for clusters with Linux and Windows operating systems. MPI/Pro fulfills the need for a scalable programming environment based on a key community standard (MPI [4,5]). MPI/Pro supports communication over a variety of high-speed networks, such as Myrinet, VI Architecture, and Infiniband, as well as over traditional TCP/IP transport on Fast and Gigabit Ethernet. Efficient intra-box (SMP) communication is also supported. Following multiyear research in the message passing area and experimentation with freeware MPI libraries, MPI/Pro was created from first principles and has no legacy limitations associated with early implementations. This enabled an efficient and scalable design aimed at achieving specific architectural goals, involving processor overhead and overall better application performance. These capabilities have been shown to be unavailable in the architectures of model implementations of MPI [6,18].

Architecture

MPI/Pro has a number of architectural features that facilitate high-performance and scalability while imposing controlled processor overhead. Among these features are asynchronous completion notification and independent message progress. These features provide for an efficient overlapping of communication and computation, which is an important source for improving overall application performance. In addition, asynchronous notification and independent message progress enable MPI/Pro to achieve low latency and high-sustained bandwidth for long messages at a low CPU consumption, which frees a large portion of the CPU resources for useful computation on user applications [6,18].

MPI/Pro has an efficient multi-device architecture, developed based on many years of experience with the features and limitations of earlier freeware MPI implementations. Multiple devices enable the MPI library to simultaneously utilize different communication media offered by hierarchical systems. Such hierarchies are built from networked multiprocessor nodes or systems with multiple networks. Modern operating systems provide efficient inter-process communication (IPC) mechanisms between processes on one node. MPI/Pro utilizes these mechanisms through its SMP devices for Windows and Linux. The multi-device architecture of MPI/Pro allows all active devices to operate in an independent manner, which removes the performance dependency of faster devices on slower devices. MPI libraries with polling progress do not provide such isolation of their devices, which leads to a negative impact on performance and scalability [6,18].

MPI/Pro supports communication devices with both synchronous (polling) and asynchronous (blocking) modes of message completion notification. Polling notification achieves lower latency for short messages at the expense of increased CPU overhead. Blocking notification allows for a more efficient use of CPU resources (compared to polling) but results in higher message latency because of the processing involved in interrupt servicing. The difference of the two modes in bandwidth performance of medium- and long-size messages is negligible. A large number of parallel algorithm exchange messages with such sizes so these algorithms can clearly benefit from using the blocking synchronization mode by utilizing more CPU resources for useful computation [6]. MPI/Pro offers both modes of completion notification and allows users to choose the mode that fits best the specific needs of their siuations. MPI/Pro has a run-time mechanism for completion mode selection. Experimental results demonstrating this capability of MPI/Pro are presented in figures 1 and 2. These results are obtained with MPI/Pro's VI Architecture (VIA) device in polling and blocking mode using the Emulex cLAN network [17]. Figure 1 shows message latency in microseconds based on half of the round-trip time measured by a ping-pong test. Figure 2 demonstrates the bandwidth results in megabytes per second based on the same ping-pong test. For comparison purposes, the two charts also show the latency and bandwidth numbers obtained with MPI/Pro's TCP device using the TCP bindings of the cLAN driver. The experiments were carried out on two single-processor Dell Precision 530 workstations with Pentium 4 Xeon processors at 1.5 GHz and with 512 MB RDRAM running Windows 2000 Server. Emulex cLAN 4.2 drivers for Windows 2000 were used.



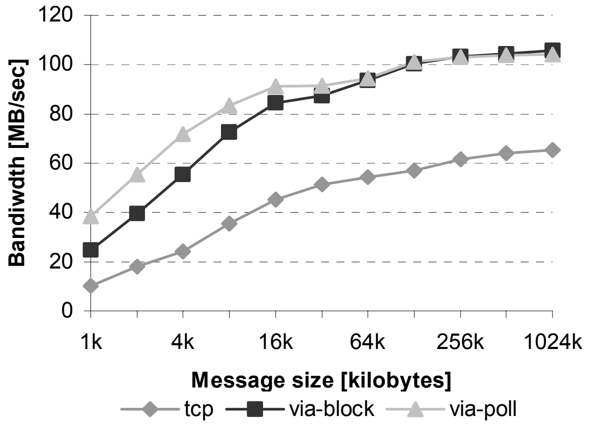


Figure 2. MPI/Pro bandwidth using different modes

MPI/Pro also has specific optimizations for collective communication algorithms, user derived data types, and persistent mode of communication. MPI/Pro is fully compliant with the strict interpretation of the MPI Progress Rule [4], which ensures timely progress of long messages when user applications perform computations without calling the MPI library for long periods of time.

MPI/Pro is also one of only a few MPI implementations that support multithreaded MPI applications; notably, this is a feature of dedicated parallel computer environments' MPI's. Thread-safety allows for a programming model with multi-level concurrency. Parallel applications can be designed so that they perform multi-threaded processing within a cluster node in order to exploit local parallelism while message-passing level parallelism is achieved through the MPI library using network communication between nodes. Thread-safety also enables MPI/Pro to operate efficiently in a hybrid-programming environment using a combination of MPI and OpenMP. OpenMP provides intra-node compiler-level parallelism. Multithreaded programming has widened its popularity as computer vendors offer low-cost machines with moderate processor count and common operating systems with highly optimized thread packages such as Windows and recently Linux. MPI/Pro allows all user threads to execute MPI calls concurrently. with internal concurrency in the MPI library itself.

Process startup

Under Windows, the remote process startup utilities of MPI/Pro are implemented over the Microsoft .NET Framework [9]. The framework allows for greater scalability and higher level of security than previous architectures. Other MPI implementations in the public domain do not evidently attach sufficient emphasis on security. The work here emphasizes authentication and authorization, rather than privacy, which remains for future work on all MPI systems.

The startup services utilize the advanced features of the .NET Framework and the C# (C Sharp) language [14] for security, process startup, management, and I/O redirection. Specifically, the MPI/Pro startup services are developed with the *remoting* infrastructure of .NET. Using remoting, any number of startup service instances can be launched on a single host. This allows for multiple instances of MPI/Pro jobs running on a single host in isolated sessions of their own, without interfering with the other MPI/Pro processes. The startup services also utilize the JobObject concept in Windows 2000 and XP operating systems to manage and clean up processes. The JobObject enables faster, more robust clean up procedure than any other technique of which we are aware.

Strong cryptographic techniques are employed for user credentials storage, retrieval, and transmission to the remote startup services. Asymmetric encryption techniques and dynamic public key pair generation are used for these purposes. The remote startup services and the clients are signed with public key pairs, so that only the authentic MPI/Pro executables will be able to successfully create instances of the startup services. As part of the feedback received from industry, users have emphasized security for commercial deployment of MPI

Redirection of the standard output and error streams from MPI/Pro processes is handled through the delegates and events mechanism of the C# language. Using this mechanism, the I/O is transmitted to the MPIRun client from the remote startup service in the form of asynchronous events. This prevents polling and buffering issues on part of the MPIRun client for I/O from the remote processes. This issue is also important in a production environment where overhead detracts from batch or gang scheduling, and even single-job scheduling efficiency.

Deployments

MPI/Pro is in use at hundreds of organizations worldwide. Two of the organizations that use currently MPI/Pro in production environments for offering shared high-performance cluster computing resources to a large number of users are the Cornell Theory Center (CTC, www.tc.cornell.edu) and the NSF Research and Engineering Center at Mississippi State University (ERC, www.erc.msstate.edu). A growing number of industrial sites also use MPI/Pro in production.

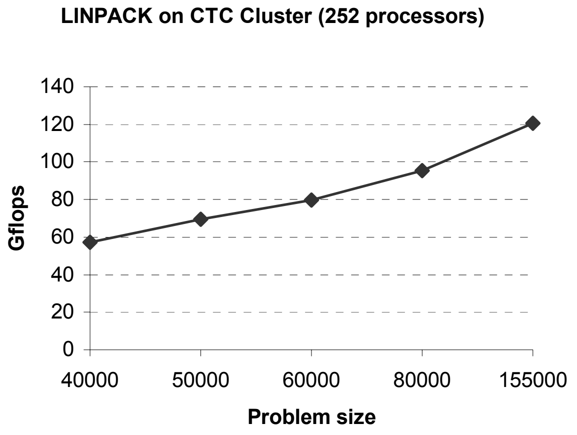


Figure 3. LINPACK performance on CTC cluster

CTC operates two (Velocity and Velocity+) 256-processor clusters running Windows 2000 and interconnected with Emulex cLAN. MPI/Pro for Windows+cLAN is installed on both clusters. ClusterController for Windows is used as resource manager and batch job scheduler on both clusters. The two Velocity clusters and a pool of serves for sequential jobs are used around the clock on a 7x24 basis. Hundreds of users run a large variety of parallel codes every day. The two Velocity clusters are part of CTC's strategy to replace traditional expensive multi-computer platforms with architectures based on COTS technologies [1]. Figure 3 shows the results of MP-LINPACK on Velocity+, which were used for ranking CTC in the November 2001 issue of the Top500 supercomputers list (www.top500.org) with a score of 120.70 Gflops obtained on 252 processors. Velocity+ consists of 128 dual-processor Dell PowerEdge 2450 servers with 733 MHz Pentium III processors and 2 GB RAM.

ERC operates two Linux clusters. The first cluster consists of a 165 dual-processor SGI 1100 servers with Pentium III processors at 1 GHz and 1.25 GB RAM. The cluster nodes are interconnected with a 100 Mbps FastEtehrnet network. Using this cluster (330 processors) and MPI/Pro 1.6.3 for Linux, ERC measured 140.50 Gflops on LINPACK and also entered the June 2001 Top500 list raked 158th. The second cluster consists of 128 IBM x330 dual-processors servers with the same processor and memory specifications as the SGI cluster. The operating system is Linux RedHat 7.2 with a custom built 2.4.9-13smp kernel. ERC used a subset (32 nodes) of the IBM cluster to perform performance comparison between MPI/Pro and MPICH (version 1.2.1) on the LINPACK benchmark. The results of this comparison are presented in Figure 4. BLAS with ATLAS 3.2.1 and gcc 2.95.3 compiler were used for the tests.

The comparison between MPI/Pro and MPICH shows that MPI/Pro consistently delivers higher application performance than MPICH; the increase is in the range of 3-6%. This demonstrates that the quality and the architecture of the MPI middleware can have a significant effect on the performance of the entire parallel system. This impact has been observed to be stronger on clusters with higher processor counts and when a high-speed cluster interconnect is used, and on more realistic applications than the LINPACK benchmark. Applications that are designed to take advantage of the architectural features of MPI/Pro will be able to extract yet more performance. The 100 Mbit/s network is the system with the lowest opportunity for a commercial-grade MPI to overwhelm a freeware system in a pure micro-benchmark capacity, so this represents a lower bound on the value of using commercial-grade middleware.

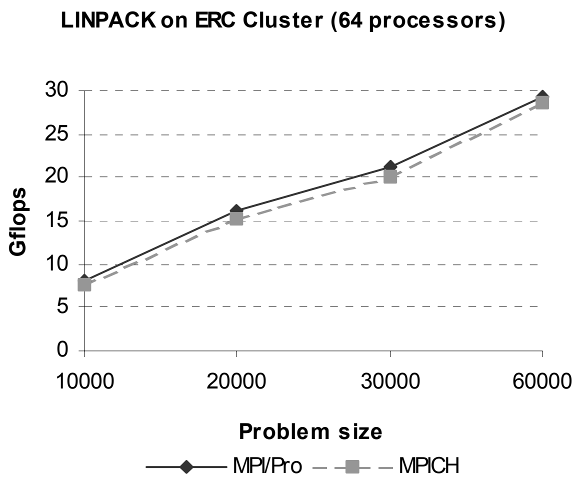


Figure 4. Comparison between MPI/Pro and MPICH

Future plans
MPI Software Technology is currently developing a next generation MPI implementation under the pre-product-release name ChaMPIon/Pro. ChaMPIon/Pro builds on the successes of MPI/Pro and specifically emphasizes highly ultra-scale parallel systems. Currently, ChaMPIon/Pro is being developed for the highly scalable DOE ASCI platforms. MPI-2 functionality is another major emphasis of ChaMPIon/Pro. At present, fully compliant support for parallel file I/O (MPI I/O) is available for NFS and PVFS. In development are one-sided communication and dynamic process management. ChaMPIon/Pro emphasizes ultrascale, production computing, and hence is a driver for wider adoption of massive parallelism in production computing.

As far as a disruptive network technology that MPI can exploit, InfiniBand [7] is viewed as a key-next generation high-speed, scalable interconnect that would enable multi-thousand way clusters. MPI/Pro and its follow-on ChaMPIon/Pro are designed to take full advantage of InfiniBand features. MPI/Pro with InfiniBand support is in development currently, tracking existing hardware and emerging scalable solutions in this important networking space.

InfiniBand is a new industry specification intended to deliver high bandwidth and scalability of message and data I/O through a point-to-point, channel-based, switched-fabric technology [8,15]. InfiniBand eliminates certain performance bottlenecks of shared-bus-based technologies by utilizing switched-fabric architecture. The data link rate of the full duplex communication channels is specified at 2.5 Gbps. Channels with one, four, and twelve links are to be available. The building components of the InfiniBand fabric are Host Channel Adapters (HCA), Target Channel Adapters (TCA), and fabric switches. The HCAs are equipped with Direct Memory Access (DMA) engines that perform fast memory read and write operations as well as protection and address translation. InfiniBand offloads communication-related processing from the CPU, thus enabling communication protocols with low CPU overhead, unlike traditional networking technologies using TCP/IP based protocols.

In complement to new networking technology support, future versions of MPI/Pro on the .NET Platform will be integrated with various other Microsoft applications such as Excel. At the same time, the MPI/Pro Web services architecture will enable interaction with the startup services over the Web.

CLUSTERCONTROLLER

ClusterController (CCS) is a resource management and job scheduling system designed primarily for production quality computing on clusters of commercial off-the-shelf (COTS) computers. ClusterController was originally developed for Microsoft Windows-based clusters, and a version targeted for Linux clusters nearing release. ClusterController enables batch mode job processing by employing scheduling algorithms, which improve the resource utilization and job turn-around times for users. One of the important attributes of ClusterController is the minimal CPU overhead of its services, which makes it suitable for production-grade clusters. ClusterController for Windows uses state-of-the-art Windows-specific technologies, such as the Component Object Model (COM+) and the Microsoft .NET Framework. ClusterController for Linux, on the other hand, uses technologies based on CORBA, Java, and Java Server Pages (JSP).

Architecture
The first, "beta" version of ClusterController was developed in early 1999. The original architecture was based on research conducted as part of a migration strategy from IBM SP environments to Microsoft Windows clusters at CTC. The major goal was to make the migration process as painless as possible to the users of the IBM EASY-LL batch-processing environment at CTC. Consequently, ClusterController evolved primarily as a batch-processing environment for serial and parallel jobs on the new Windows research clusters at the CTC. Work on the CTC version is described in Chapter 17 of [16].

The architecture and implementation is based on Windows principals, APIs, and services, as opposed to being a software port from Unix to Windows, since Windows offers many useful distributed services This an important strategy in guaranteeing a production quality, low overhead, and secure computing environment for commercial grade applications. The key architectural concepts of CCS are Job management, Resource management, Scheduling, Security, and Impersonation.

Job management
CCS incorporates several client and server applications, which enable cluster users and administrators to seamlessly interact and manage the jobs and resources on the cluster. The JobManager service controls the CCS job queue by accepting new jobs from the users, updating the job status, and removing jobs from the queue, either upon their successful completion or

because of premature termination initiated by the users or administrators. The Jobmanager service also interacts with the Scheduler service by way of asynchronous event notifications upon changes to the job and resource status queues. Such asynchronous events significantly reduce the CPU overhead of the CCS services by preventing continuous polling for status changes in the queues. This enables the services to handle thousands of jobs in a scalable fashion without causing any drastic performance side effects.

Resource management
One of key aspects of cluster management is predictable, controlled, audited, and efficient use of the cluster resources. CCS ensures that cluster resources are subject to these constraints by incorporating deterministic resource scheduling and monitoring. The resources are primarily owned by the CCS services and users are allowed access to the cluster only though the submission of proper job requests. Interactive access to the cluster nodes is dynamically granted and revoked by the Scheduler service, and access is restricted to the duration of an active job. The scheduler service also launches batch-mode jobs directly on the cluster nodes without any user intervention. The cluster administrators can interact with the services in order to make dynamic modifications to the resource status and other attributes.

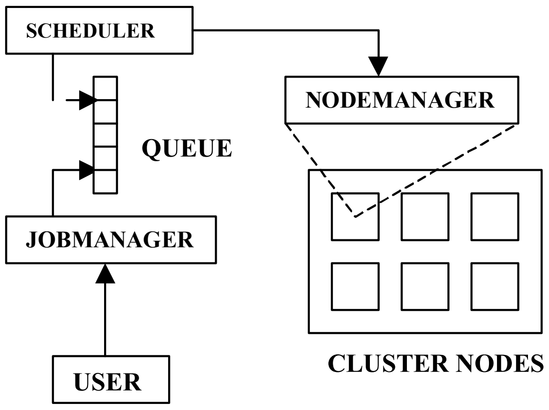


Figure 5. Conceptual view of CCS services

Scheduling
The CCS Scheduler service uses the Backfill scheduling algorithm in order to deterministically schedule user jobs from the queue. The Backfill algorithm is a modified form of the first come first served (FCFS) strategy, which achieves better resource utilization and job turn-around times over the FCFS algorithm. Backfill works by scheduling such jobs in the queue whose requirements can be immediately met, and it is not required to preempt any executing jobs or preventing any jobs waiting ahead in the queue from executing. Consequently, CCS Scheduler service is able to precisely specify the latest time before which a particular job in the queue will start execution. The Scheduler initiates job startup and cleanup after the jobs finish execution. Other scheduling algorithms are also supported in and form of "pluggable" code written in Perl. Future revisions of CCS will support custom schedulers written using any scripting language, such as VBScript or Jscript.

Security
CCS security is tightly integrated with the Windows Domain security model. The users are authenticated to the Windows domain in which the cluster is a member, prior to accepting and launching their jobs on the cluster. The JobManager service accepts and stores the cluster user's credentials in a secure location in such as way that only authorized CCS services can locate and retrieve the credentials. Communication among the CCS services and clients takes place via secure communication channels, and using encrypted messages.

Impersonation
In batch mode of operation, the CCS Scheduler service is responsible for allocating the required resources as well as for automatically launching the job on behalf of the user. The Scheduler service contacts the Impersonator service on the master node, where the job will be initially launched. The Impersonator service is dynamically configured to execute as the launching user by the Scheduler service, so that the operations carried out as part of the batch job are actually performed in the security context of the user, as opposed to the Scheduler service's security context. The Scheduler service is capable of retrieving the stored credentials of the user, so that it can configure the Impersonator service to use the credentials. Impersonation ensures high security by way of preventing malicious users from unauthorized use of the cluster resources.

Batch jobs
CCS defines a convenient notation and syntax to specify users' batch job requirements. The resource specification language is based on the Windows batch command syntax. Using the batch files users can specify their heterogeneous resource, space, and time requirements, along with their Windows domain account information. CCS services will parse the batch files, extract the requirements, and compare them with the resource entries in the status queues. The entries in the status queues are also defined in such a way that the Scheduler's heterogeneous resource matching algorithm can extract the required information in a simple and natural way, thereby scaling the scheduling algorithm to thousands of jobs.

User interaction
As clusters have entered the mainstream computing arena, clusters of sizes ranging from 16 to 64 nodes (with 2 to 4 processors per node) or more are commonly found. Tools enhancing cluster usability, availability, and manageability are of high importance in order to achieve the highest job throughput. ClusterController recognizes this fact and includes a rich set of user interfaces, which enable cluster administrators and users to efficiently and seamlessly interact with the cluster. These tools are being designed using Windows Forms and Java in the respective Windows and Linux ClusterController versions. Using these tools the administrator can remotely view the status of any node in the cluster, as well as view the job(s) executing on a selected node.

Future versions on Windows
There is a growing need for cluster management utilities on Windows clusters. The current trend in the Windows cluster market is the provision of anytime, anywhere access to services and data to the users. CCS is moving in this direction as a step forward with its integration with the Microsoft .NET Platform. CCS services are being transformed into Web services, and a

rich set of user interface tools is under development, which will enable cluster monitoring and user interaction over the web. A resource specification meta-language is under development, which is based on XML. The language will enhance portability, as well as it will improve the efficiency of the CCS scheduling algorithms.

Future versions on Linux

With the initial success of ClusterController on Windows, and the growing popularity of Linux for clustering, a version of ClusterController is currently under development for Linux. CCS for Linux will include the entire current features available in the Windows version, as well as some additional capabilities. CCS for Linux has been completely redesigned and developed from first principles with currently available distributed and Internet technologies. The implementation is done over CORBA 3 with JacORB, a publicly available Java-based implementation of CORBA. CCS Linux also uses XML technology for resource requirement specification and resource descriptions. Advanced features such as Event Channels and Asynchronous messaging are also incorporated for efficient communication. Security is implemented over the Java Secure Socket Extension (JSSE).

MPI/PRO AND CLUSTERCONTROLLER

CCS and MPI/Pro on Windows interact closely with each other on various aspects. Users of MPI/Pro are not required to explicitly register their credentials with MPI/Pro, when submitting their batch jobs to CCS. MPI/Pro automatically fetches user credentials from the CCS JobManager service. In addition, users are not required to specify the list of nodes to be used for MPI/Pro jobs. MPI/Pro communicates with the CCS services to automatically determine the list of nodes allocated to a user's job. The standard output and error streams from the MPI/Pro jobs will also be redirected by the CCS services into files stored in the user's working directory. Hence, using MPI/Pro with CCS saves time and effort on the part of the users, as well as provides a secure and controlled environment for MPI/Pro jobs. Duplicate efforts and security holes will thereby be reduced in production settings.

SOFTWARE QUALITY

Both MPI/Pro and ClusterController are subjected to rigorous verification and validation procedures. This procedure emphasizes correctness, compliance with standards, and robustness. Users of MPI/Pro and ClusterController perform round-the-clock operations. Applications commonly run for days or weeks. Robustness and quality of the implementation are critical features of the cluster software for these applications. Therefore, a strict software engineering approach has been necessary in order to deliver cluster software that can meet the high requirement of production-grade computing. Important concerns include dimensions of testing that stress systems involving long-term runs, runs at scale, and runs with third-party, supposedly unrelated activities on-going, all of which apparently cause "gotchas" in production, if not tested. This process, while well developed remains an art in addition to its roots in well-grounded software engineering principles.

CONCLUSIONS

In this paper, we have delineated requirements for commercial grade cluster computing, emphasizing examples from our own experience – MPI/Pro and ClusterController. We have considered both Windows and Linux as target environments for such parallel processing. We have indicated that performance and scalability are primary but not the only issues that drive success of cluster software, and indicated how designs of the abovementioned commercial packages achieve many of these goals today. We indicated briefly the issues of software interactions, and software quality.

REFERENCES

[1] Cornell Theory Center, "Cornell Theory Center Migrates to Dell Clusters", Dell Power Sol., Vol. 4, 2001, pp: 34-37.

[2] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A High-performance, Portable Implementation of the Message Passing Interface Standard", Journal of Parallel Computing, No. 22, Vol. 6, 1996, pp: 789-828.

[3] R. Dimitrov and A. Skjellum, "Efficient MPI for Virtual Interface (VI) Architecture", Proceedings of the 1999 International PDPTA Conference, Las Vegas, Nevada, June 1999, Vol. 6, pp: 3094-3100.

[4] Message Passing Interface Forum. 1994. "MPI: A Message-Passing Interface Standard", International Journal of Supercomputer Applications, No. 8 (3/4), pp: 165-414.

[5] Message Passing Interface Forum, "MPI-2 Standard", 1998, http://www.mpi-forum.org/docs/mpi-20.ps.

[6] R. Dimitrov, "Overlapping of Communication and Computation and Early Binding: Fundamental Mechanisms for Improving Parallel Performance on Clusters of Workstations", Ph.D. Dissertation, Mississippi State University, 2001. http://library.msstate.edu/etd/show.asp?etd=etd-04092001-231941

[7] InfiniBand Trade Association, "The InfiniBand Architecture", 2001, http://www.infiniband.org/specs.

[8] Y. Peled and G. Risi, "InfiniBand – IBM Perspectives and Products", Micronews, Vol. 7, No. 1, 2001.

[9] Microsoft .NET Framework, http://www.microsoft.com/net.

[10] LAM Message Passing Library. http://www.lam-mpi.org.

[11] Portable Batch Scheduler (PBS). http://www.openpbs.org.

[12] Commercial-grade PBS: PBSPro. http://www.pbspro.com.

[13] Load Sharing Facility (LSF), Platform Computing, http://www.lsf.com,

[14] Introduction to C#, http://msdn.microsoft.com/vstudio/nextgen/technology/csharpintro.asp.

[15] Infiniband Standard/Industry Trade Association, http://www.infinibandta.org.

[16] Sterling, T, ed, Beowulf Clustering with Windows, MIT Press, 2001.

[17] Emulex cLAN, http://www.emulex.com/products/vi/index.html

[18] R. Dimitrov, A. Skjellum, "An Analytical Study of MPI Middleware Architecture and Its Impact on Performance," submitted to *Parallel Computing,* May 2002, in review.