;login:
THE MAGAZINE OF USENIX & SAGE
August 2001 • Volume 26 • Number 5

Special Focus Issue:Clustering

Guest Editor: Joseph L. Kaiser

inside:

USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

Linux clustering for high-performance computing

by Ian Lumb

Ian Lumb is an integration architect at Platform Computing Inc. His interests include computing architectures, parallel computation and high availability.

ilumb@platform.com

Introduction

Traditionally, high-performance computing (HPC) has involved the use of monolithic mainframe, supercomputer, or symmetric multiprocessor (SMP) systems to solve the Grand Challenge problems of the physical sciences and mathematics. Classic HPC, however, has experienced a significant evolution on two fronts. First, it has been infused with interest from extra-classic disciplines ranging from high-tech and industrial engineering, to digital content creation and the life sciences. Second, the availability of commodity processors, in combination with low-latency, high-bandwidth interconnect technologies, has catalyzed the appearance of computing architectures based on clustering paradigms. Linux clustering through distributed operating systems, middleware, and hybrids thereof, represents the current focus.

Distributed Operating Systems

Clustering through distributed operating systems (D-OSes) is not "new." However, it has been Linux-based clustering solutions that have facilitated the commoditization of HPC in nontraditional disciplines. Although MOSIX (*http://www.mosix.org/* and *http://www.mosix.com/*) is notable in its own right as a D-OS, the current illustration of clustering via a D-OS in the Linux context is provided here via Beowulf clustering (*http://www.beowulf.org/*).

In 1994, Thomas Sterling and Donald Becker of CESDIS[1] created the first Beowulf cluster out of 16 Intel 486-generation systems interconnected via channel-bonded Ethernet. An instant success at the time, Beowulf clustering has reached beyond academic circles in its ability to generate attention and has become an acknowledged genre in HPC.

Beowulf clusters consist of:

- Commodity off-the-shelf (COTS) hardware
- LAN interconnect technology
- GNU/Linux operating system
- System software
- Programming environments

Arguably, the enabling aspect of Beowulf clustering derives from the system software that allows interconnected COTS-class hardware, each running its own instance of GNU/Linux, to function as a parallel compute engine.

Although individual GNU/Linux kernel instances abstract process manipulation through the notion of a process identifier (PID), no such abstraction exists for the case in which a parent process has forked child processes that execute on physically distinct systems, each running their own instance of the GNU/Linux kernel. Generally speaking, there is no concept of a "global PID" for clustered Linux systems. To address this shortcoming, the Beowulf system software incorporates a kernel modification to provide a distributed process space (BProc) which allows:

- PIDs to span multiple physical systems – each running its own instance of GNU/Linux

ACKNOWLEDGEMENTS

Through numerous discussions, the author's colleagues Bill McMillan and Chris Smith have indirectly contributed to this article.

Beowulf clustering placed distributed-memory parallel computation in the public domain by making it simultaneously accessible and realizable

- Processes to be launched on multiple physical systems – each running its own instance of GNU/Linux

The development of this distributed process space marks a significant infrastructure advancement in the provisioning of a clustered Linux environment for parallel computation.

Ultimately, the hardware, interconnect technology, operating system, and system software collectively provide an infrastructure for scientists and engineers to develop and execute applications which exploit the distributed-memory parallel computation paradigm. Both the parallel virtual machine (PVM, *http://www.epm.ornl.gov/pvm/*) and the message-passing interface (MPI, *http://www.mpi-forum.org/*) approaches to parallel computation are supported within the Beowulf framework. In addition, various GNU compilers and tools (e.g., editors, debuggers, profilers, etc.) provide a fairly complete development environment.

First-generation Beowulf clusters generated interest for a variety of reasons:

- They demonstrated that a substantial parallel computation infrastructure could be built from readily available, and inexpensive, hardware and software components.
- They leveraged key, existing open source software in the GNU/Linux operating system, plus the programming environments offered by PVM and MPI in tandem with the GNU compilers and tools.
- They demonstrated that a distributed-memory parallel computation approach could rival, and in some cases surpass, the performance characteristics of "more traditional" serial or shared-memory programming paradigms.

In short, Beowulf clustering placed distributed-memory parallel computation in the public domain by making it simultaneously accessible and realizable.

Touted as the next-generation solution, the Scyld Beowulf clustering distribution (*http://www.scyld.com*) offers the following enhancements over its progenitor:

- Installation and administration improvements
- Efficient, single-point distributed process management
- Various 64-bit capabilities
- BProc-aware MPICH
- MPI-enabled linear algebra libraries and Beowulf application examples.

The second-generation Beowulf clustering solution provides a number of significant enhancements beyond what was available in the first-generation Beowulf clustering solution. Save for the BProc system software, a comprehensive framework to effectively manage resources that are distributed over a network remained absent. This significant shortfall is identified by D. F. Savarese and T. Sterling[2] as *The Software Barrier*:

> The earliest Beowulf-class systems were employed as single-user systems dedicated to one application at a time, usually in a scientific/engineering computing environment. But the future of Beowulf will be severely limited if it is constrained to this tiny niche.
>
> The need to enhance Beowulf systems usability while incorporating more complicated node structures will call for a new generation of software technology to manage Beowulf resources and facilitate systems programming.

They articulate the software barrier even more precisely in distributed resource management terms in the following:

Adequate system management may depend on the virtualization of all its resources. This will separate the user application processes from the physical nodes upon which the tasks are executed. The result is a system that dynamically adapts to workload demand, and applications that can perform on a wide range of system configurations trading time for space. Therefore, a new class of workload scheduler will be required, developed, and incorporated in most Beowulf systems. It will support multiple jobs simultaneously, allocating resources on a to-be-defined priority basis. It will also distribute the parallel tasks of a given job across the allocated resources for performance through parallel execution. Such schedulers are not widely available on Beowulfs now and will be essential in the future. They will incorporate advanced checkpoint and restarting capabilities for greater reliability and job swapping in the presence of higher priority workloads. Compilers to use the more complicated structures of the SMP nodes will be required as well to exploit thread level parallelism across the local shared memory processors. The software used on these systems will have to be generally available and achieve the status of de facto standard for portability of codes among Beowulf-class systems.

In addition to resource-management opportunities, there exists a tight dependency between BProc and the Linux kernel; even in the case of routine upgrades and/or patch application, this dependency needs to be considered, and implies system downtime. The architecture of BProc itself has caused scalability concerns[3] and introduced challenges in porting parallel debugging tools due to global-local PID mapping issues. As a Linux kernel modification, BProc is necessarily covered by the GPL. While this provides all the benefits that the open source approach has to offer, it makes it challenging for commercial Independent Software Vendors (ISVs, e.g., Scyld), Linux distribution providers and integrators, and traditional-UNIX system vendors to simultaneously add value and retain a differentiating edge.

Middleware

Middleware can also be used to deliver clustering solutions. In striking contrast to D-OSes, clustering via middleware does not require modifications to the Linux kernel. Instead, such middleware fundamentally provides some form of distributed process abstraction, including primitives for process creation and process control. Broadening the discussion beyond this point, and providing a clearer distinction from distributed-memory programming paradigms (e.g., MPI, PVM), is best presented through the framework of distributed resource management (DRM).

DRM is a class of middleware that facilitates the management of heterogeneous compute resources distributed over a network. DRM directly addresses the tension between supply and demand by matching an application's resource requirements with the compute resources capable of filling the need. By effectively arbitrating the supply-demand budget over an enterprise-scale IT infrastructure, subject to policy-driven objectives, DRM solutions allow organizations to derive maximal utilization from all available compute resources.

In general, DRM solutions make use of dynamic-load-state data to assist in making effective, policy-based scheduling decisions, and in applying utilization rules to hosts, users, jobs, queues, etc., all in real time. This dynamic-load-state capability has significant implications in distributed-memory parallel computing, since task-placement advice (i.e., which hosts are best suited for computational use) can be provided directly to the MPI application (on launching).

In striking contrast to D-OSes, clustering via middleware does not require modifications to the Linux kernel

A remote-execution service is required to allow computational tasks to be communicated over a network

As indicated previously, a remote-execution service is required to allow computational tasks to be communicated over a network. Although the standard remote-shell infrastructure (i.e., rsh-rshd-rexec) offers some possibilities, a more comprehensive service is required to enable:

- Authenticated communications over a network
- A high degree of transparency in maintaining the user's execution environment
- Task control with respect to limits, signal passing, etc.
- An environment for the task to execute in on the remote server

Although DRM solutions do not need to provide a distributed-process space, task-tracking mechanisms are required. Thus application-task identifiers act as a handle to the individual (parent and child) processes that collectively constitute a distributed application. In addition to providing a unique identifier for application control, such cluster-wide identifiers can be used in monitoring, manipulating, reporting, and accounting contexts.

Through the introduction of elements, the task identifier can be further generalized to the level of a one-dimensional array. This abstraction allows the same executable to be run with different inputs, while being referencable as a unit. In some circles, this approach is referred to as parametric processing.

DRM solutions typically employ a policy center to manage all resources, e.g., jobs, hosts, users, queues, external events, etc. Through the use of a scheduler, and subject to predefined policies, demands for resources are mapped against the supply for the same in order to facilitate specific activities.

The extension of DRM solutions to support the programming, testing, and execution of parallel applications in production environments requires:

- Complete control of the distributed processes making up a job in order to ensure that no processes will become un-managed. This effectively reduces the possibility of one parallel job causing severe disruption to an organization's entire compute infrastructure.
- Vendor-neutral and vendor-specific MPI interfaces
- The ability to leverage a policy-driven DRM infrastructure that is cognizant of dynamic load state

Challenges specific to the management of MPI parallel applications include the need to:

- Maintain the communication connection map
- Monitor and forward control signals
- Receive requests to add, delete, start, and connect tasks
- Monitor resource usage while the user application is running
- Enforce task-level resource limits
- Collect resource usage information and exit status upon termination
- Handle standard I/O

To illustrate the value of parallel-application management for developers of MPI applications, consider the following example of fault tolerance. It is beyond the present scope of the MPI, and indeed PVM, to take into account transient situations (e.g., a host that exhibits a kernel panic and crashes) that inevitably occur while an application is in the execution phase.[4] Such situations will affect some of the processes involved in the execution of the MPI-based application. If the application does not include some mechanism to address such situations, it is possible for the remainder of the application to run

to completion and deliver incomplete and (potentially) meaningless results; the effect of this situation is compounded when attempts are made to interpret the results. Although DRM solutions cannot enable fault tolerance in MPI-based applications to the degree that resynchronizations and reconnections are made possible, such middleware can trap and propagate signals, thus affording a significantly improved degree of management during execution – all without the need for additional coding (beyond exception handlers) by the MPI application developer.

As described in this section on clustering via middleware, Platform Computing's Load Sharing Facility (LSF, *http://www.platform.com*) is the only provider of a comprehensive DRM solution. Components of the DRM solution are provided by Sun's GridEngine (*http://www.sun.com/software/gridware/*) and Veridian's Portable Batch System (PBSPro, *http://www.pbspro.com/*); an open source version of the Portable Batch System (PBS) also exists (*http://www.openpbs.org*). TurboLinux's EnFuzion (*http://www.turbolinux.com/products/enf/*) offers a parametric processing solution.

Hybrid Solutions

Clustering via distributed operating systems or middleware shouldn't be taken to imply strict exclusivity. In fact, examples exist in which hybrid approaches have been employed. In one such case, Platform's LSF leverages SGI Array Services (*http://www.sgi.com/software/array/*) and SGI's implementation of the MPI, respectively, for distributed-process-space management and vendor-MPI leverage in the case of parallel computing. With the tremendous interest in Linux clustering solutions from both the open source and commercial ISV, integrator, and system-vendor communities, it is expected that such hybrids will continue to appear.

Summary

In addition to their price-performance appeal, clustering solutions can rival the throughput capacity and capabilities of legacy mainframes, supercomputers, and high-processor-count SMPs. The recent creation of a Top 500 list dedicated to cluster computing (*http://clusters.top500.org/*) can be regarded as one indication of the viability of this approach to high-performance computing. Both distributed operating systems and middleware have proven capable of delivering Linux clustering solutions; hybrid solutions have also been identified as likely to be of increasing value and prevalence in the not-too-distant future. The choice of approach will ultimately need to take into consideration a multiplicity of factors – e.g., total cost of ownership, importance of the service (which distinguishes needs from the interests of the hobbyist to the data center), desired characteristics of the service, etc.

As numerous activities serve to enhance the capabilities of the Linux kernel, and powerful 64-bit processors like API NetWorks Alpha (*http://www.apinetworks.com*) and Intel's Itanium (*http://www.intel.com/itanium/*) are increasingly commodified, the possibilities for clustering solutions increase significantly.

Federating geographically distributed clusters to aggregate resources, or providing access to specialized resources remotely, has been brought into focus in recent times through the notion of the Grid.[5] Much like the ubiquitous, highly available electrical power grid, the global computing grid allows challenging problems in HPC to be addressed. Various academic research (e.g., the Globus Project, *http://www.globus.org*, and the Legion Project, *http://www.cs.virginia.edu/~legion/*) and commercial (e.g., Applied Meta, *http://www.appliedmeta.com/*, and Platform's LSF MultiCluster, *http://www.platform.com*) ventures are already realizing the Grid. Because meta-com-

In addition to their price-performance appeal, clustering solutions can rival the throughput capacity and capabilities of legacy mainframes, supercomputers, and high-processor-count SMPs

NOTES

1. The Center of Excellence in Space Data and Information Sciences (CESDIS) is a division of the University Space Research Association (USRA), located at the Goddard Space Flight Center in Greenbelt, Maryland. CESDIS is a NASA contractor, supported in part by the Earth and Space Sciences (ESS) project. The ESS project is a research project within the High Performance Computing and Communications (HPCC) program.

2. D.F. Savarese, T. Sterling, "Beowulf," in R. Buyya, ed., *High Performance Cluster Computing*, vol. 1, 1999, pp. 625–645.

3. D.H.M. Spector, *Building Linux Clusters* (O'Reilly & Associates, Sebastopol, CA, 2000).

4. K. Dowd, C.R. Severance, *High Performance Computing*, 2nd ed. (O'Reilly & Associates, Sebastopol, CA, 1998).

5. I. Foster, C. Kesselman, eds., *The Grid: Blueprint for a New Computing Infrastructure* (Morgan Kauffman Publishers, 1999).

puting necessitates increased collaboration between all stakeholders, standardization efforts such as the Global Grid Forum (*http://www.gridforum.org*) and the New Productivity Initiative (*http://www.newproductivity.org/*) are expected to be of increasing importance.

Note Added In Proof

Since this article was originally written, the author has been apprised of the following matters noteworthy of communication:

Although Scyld has recently released a significant update of its own Beowulf clustering solution (*http://www.scyld.com/page/products*), and in addition to MOSIX, newcomers SCore (*http://pdsw3.trc.rwcp.or.jp*) and CPLANT (*http://www.cs.sandia.gov/cplant*) also merit serious consideration for clustering via distributed operating systems.

The Two-Kernel Monte (*http://www.scyld.com/products/beowulf/software/monte.html*), or the use of diskless compute nodes, can serve to reduce the strong kernel interdependency noted in the case of clustering via distributed operating systems.

GridEngine is to join OpenPBS as an Open Source middleware contribution from Sun Microsystems, Inc.

The New Productivity Initiative has recently released for public comment a draft of its API for distributed resource management (*http://www.newproductivity.org./pdf/RefModel-V1.pdf*).

Under the auspices of the Department of Defense (DoD, United States), the High Performance Computing Modernization Project (HPCMP) has recently realized a significant production computing grid implementation (*http://www.platform.com/solutions/whitepapers*). Nine of the DoD HPCMP's twenty-one distributed compute facilities are involved in a Phase I implementation that is aimed broadly at improving the organization's overall use of compute capacity, and enhancing their ability for compute capability. This project utilizes a number of DRM technologies from Platform Computing Inc..

The joint agreement between Compaq Computer Corporation and Intel Corporation (*http://www.compaq.com/newsroom/pr/2001/pr2001062501.html*) promises to infuse the Itanium processor family with the proven HPC capabilities of the Alpha processor. This fusion of commodity and technology should eventually invigorate the possibilities for Linux clustering in HPC.