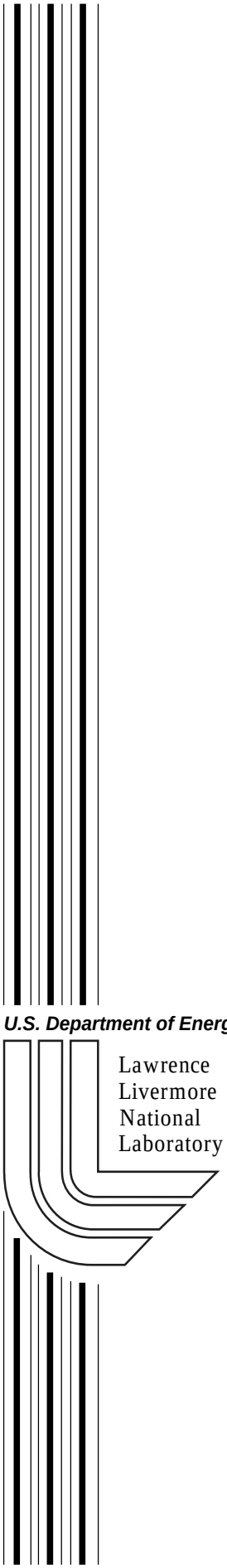
UCRL-ID-150021

Linux Project Report

Jim E. Garlick
Chris M. Dunlap



August 18, 2002

DISCLAIMER

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

1 Project Overview

The goal of the Linux Project at LLNL is to field production Linux clusters providing Livermore Computing (LC) users with the Livermore Model programming environment, described in Sec. 1.1. In the past, this was accomplished using variants of the UNIX operating system on proprietary hardware. By using Linux on near-commodity hardware, we believed that we could procure more cost-effective clusters, offer better support to our users, and create a platform from which open source system software could be developed jointly with strategic partners and leveraged by the high-performance computing (HPC) community at large.

The focus of the Linux Project is twofold. First, we address the technical gaps between Linux software and the Livermore Model through a combination of local software development efforts and collaborations. Second, we participate in the design, acquisition, and support of these clusters.

1.1 The Livermore Model

The LC scalable system strategy, known as the Livermore Model (Seager, 2002) and depicted in Fig. 1, is to provide a common application environment on all LC production clusters. This allows highly complex scientific simulation applications to be portable across generations of hardware architectures and between currently available operating systems. The strategy has provided a stable applications development environment since about 1992, when the Meiko CS-2 MPP was introduced at LLNL.

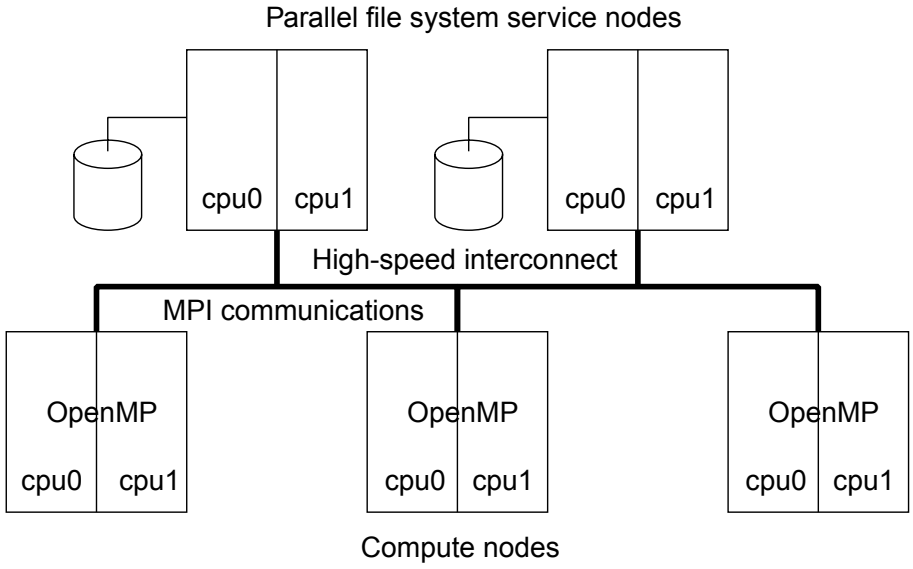


Figure 1. LC's scalable system strategy, the Livermore Model.

The idea of the Livermore Model is to abstract the parallelism and I/O services of the computing environment to a high level that evolves slowly. The abstraction is based on SMP compute nodes attached to a high-speed, low-latency message passing interconnect. Applications utilize compiler-generated OpenMP threads to exploit SMP parallelism and MPI message passing to exploit parallelism between nodes. Data sets can be stored on a common (POSIX interface) parallel file system, which may utilize striping across multiple file service nodes on the high-speed interconnect in order to achieve scalable performance characteristics.

1.2 Choosing Linux

In the past, the Livermore Model was implemented on proprietary UNIX platforms including Solaris, AIX, and Tru64. While this approach was successful, it had a few disadvantages. First, highly specialized proprietary hardware is expensive to purchase and maintain. Second, the HPC market has dwindled in comparison to the commodity market in recent years, making it increasingly difficult to convince proprietary vendors to implement HPC-specific features. This problem is exacerbated because this effort must be repeated for each vendor involved with a production cluster. Finally, because proprietary vendors also support a large commodity base, their help-desk support is optimized for commodity users. This makes it hard to get help, and there is often a long delay between problem identification and resolution.

Implementing the Livermore Model using Linux on near-commodity hardware has several advantages. First, commodity hardware has a substantial price/performance advantage over proprietary hardware (Sterling, 1997). Second, the open source nature of Linux lends itself well to HPC customization, and the results can be made portable across hardware platforms and can be freely shared. Finally, customization efforts build in-house expertise that can serve as the basis for an excellent support infrastructure.

2 Technical Challenges

Initial project investigations (McVoy, 1998) identified gaps between current open source software and the software required to implement the Livermore Model. Missing were a low-latency interconnect for message passing, a scalable parallel file system, a scalable resource manager for parallel programs, and cluster administration tools for providing a "single-system image." Four software projects and strategic partnerships were established to address these deficiencies.

2.1 Low-Latency Interconnect for Message Passing

The Linux Project was initiated with a partnership with Compaq and Quadrics to port the system software stack from the Compaq Sierra product line to identical hardware running Linux. The first technical challenge faced by the Linux Project was to port the Quadrics QsNet device drivers and associated software from Compaq Tru64 to Alpha/Linux.

This effort took place during 1999 and achieved QsNet performance and stability on Alpha/Linux that was comparable to that on Tru64. Figure 2 shows QsNet Elan3 MPI bandwidth for different message sizes on two different clusters. It shows that on identical hardware (i.e., the Tsunami chipset on the UP2000), QsNet under Linux slightly outperformed that under Tru64 and achieved a maximum bandwidth of 210 MB/s. On the Linux Pengra cluster (with the e7500 chipset), however, we achieved a maximum bandwidth of 320 MB/s.

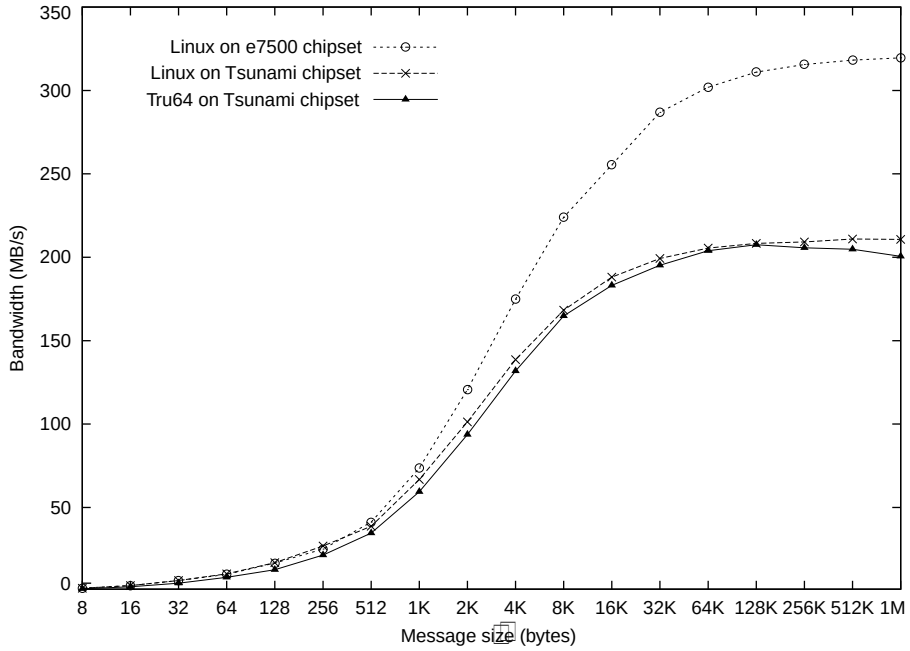


Figure 2. Quadrics QsNET Elan3 MPI ping-pong bandwidth.

Since this early effort, Compaq dropped out of the partnership, but the Laboratory's relationship with Quadrics has remained strong. Quadrics has recently released all of their software (except RMS) under an open source license, and they have realigned their business model toward Linux as their primary platform.

2.2 Scalable Parallel File System

As part of the Compaq partnership just described, Livermore and Compaq ported the Petal/Frangipani (Lee, 1996 and Thekkath, 1997) research file system to Linux and improved performance and stability, utilizing QsNet for transporting file system information at high bandwidth. As performance and stability drew near targets, this project was aborted because of business changes within Compaq and an end to the Livermore partnership. While Petal/Frangipani did not materialize as a viable file system for Linux, the effort resulted in improvements to the QsNet device drivers, an understanding of Linux high-performance disk I/O issues (Garlick, 2001), and a knowledge base that was used in subsequent file system efforts.

Our next file system effort emerged from an April 2001 ASCI PathForward request for proposal (RFP) for an SGS (scalable, global, secure) file system. That RFP resulted in the development of PathForward Lustre. Lustre (Braam, 2002) is an open source, distributed, object-based file system with a design informed by lessons learned from the Intermezzo, Coda, and GPFS file systems. Livermore established a contract with Peter Braam of Cluster File Systems, Inc. to collaborate on producing Lustre Lite, a scaled-back version of PathForward Lustre that implements a parallel file system for a single cluster.

Lustre uses Object Storage Targets (OSTs) to implement its object storage protocol. Livermore has partnered with BlueArc Corporation to build a storage appliance that implements the OST protocol. Other OST partnerships are being solicited from hardware vendors. A goal is to have Lustre Lite operating with BlueArc OSTs on the 1152-node Multiprogrammatic and Institutional

Computing Capability Resource (MCR) cluster scheduled for deployment at Livermore before the end of CY2002 (see Sec. 4.1).

Livermore's contribution to Lustre Lite includes supporting and advising Cluster File Systems on the basis of our experience with large systems. Livermore also contributes significantly to Portals NAL (Network Abstraction Layer) development, configuration and management infrastructure, SNMP monitoring of the file system, and other aspects of the project.

Livermore has also taken a key role in testing the file system during development. A rigorous testing framework is being developed to ensure that Lustre conforms to POSIX standards. This testing framework is also used to measure file system performance and to stress the code to expose bugs as early as possible. These efforts help to ensure that Lustre becomes a stable, well-designed file system for general use on Linux clusters.

2.3 Scalable Resource Manager for Parallel Programs

Cluster resource managers allocate resources to parallel programs and shepherd those programs through execution. At Livermore, a resource manager must interface with the Distributed Production Control System (DPCS), a center-wide metabatch and accounting system with connections to all LC resources, and must also manage interconnect resources. Because the Quadrics Resource Management System (RMS) was used with DPCS on the Meiko CS/2 and Compaq Sierra clusters and was the only resource manager capable of managing QsNet interconnect resources, it was the initial choice for Linux systems at Livermore.

Although RMS has improved enough over the course of the Livermore/Quadrics partnership that it is now highly scalable and reliable, it still has the major drawbacks that it is closed source and it is not portable to other interconnects we may deploy in the future. No resource managers available for Linux were highly scalable and portable to different cluster architectures, so the Simple Linux Utility for Resource Management (SLURM) Project (Jette, 2002) was initiated in late 2001. The primary effort is at Livermore, with participation from collaborators at Linux NetworX and Brigham Young University.

SLURM is designed to scale to thousands of nodes. It will be able to recover from a variety of failure modes without terminating workloads. Although it will initially support only IP-based communications and the QsNet Elan3 interconnect, it should be straightforward to add support for other interconnects. SLURM is being distributed under the GNU General Public License (GPL) (Free Software Foundation, 1991). It is anticipated that it will run on the 1152-node MCR cluster by the end of CY2002.

2.4 Cluster Administration Tools

Specialized tools are required to effectively administer large clusters of nodes. As the node count rises, it becomes increasingly advantageous to treat a set of nodes as a logical unit and to act on those nodes in parallel. Although open source solutions exist for some problems, they generally do not scale to the size of the clusters we deploy. We have therefore developed several tools to augment those already available in the open source community.

2.4.1 Pdsh

Pdsh is a multithreaded remote shell client capable of executing commands on remote hosts in parallel. Pdsh can use several remote shell services, including RSH, SSH, and Kerberos IV. It also handles common failure modes gracefully, such as the case of a target node being down or

slow to respond. Using Pdsh, a system administrator can execute commands across all nodes of a cluster as though it were a single machine.

Pdsh has been extended to run MPI jobs on the QsNet interconnect. This is useful for running parallel test jobs or interconnect diagnostics on a new system that does not yet have a resource manager installed.

2.4.2 Genders

Genders is a simple flat file database that represents the layout of a cluster and is used for cluster configuration management. A Genders file, containing a list of node names and their attributes, is replicated on each node in the cluster. Hard-coded node lists in scripts are replaced with Genders file lookups, thus allowing scripts to be generalized for use without modification in different cluster configurations.

A standard practice in LC system administration is to codify all changes made to a cluster in such a way that they can be quickly reapplied after a fresh installation. Genders facilitates this by providing an rdist distfile preprocessor that expands node lists using Genders queries. System files residing in a central repository under RCS control can be propagated to multiple clusters using this mechanism. With Genders, the distfiles need not change when cluster configurations change.

2.4.3 ConMan

Because of cost, space, and cooling constraints, large clusters of PCs do not have monitors and keyboards attached to each node. Remote administration of the system becomes an issue when the cluster is physically located in a remote machine room. Because the system console may occasionally be the only means of configuring the BIOS, receiving Power-On Self-Test (POST) error messages, or performing system administration tasks (e.g., when a node experiences networking problems or fails to enter multi-user mode), we deploy clusters in which all nodes are accessible via a serial console connection.

ConMan is a console management program designed to support a large number of console devices and simultaneous users. ConMan currently supports local serial devices and remote terminal servers (via the Telnet protocol). It provides for a mapping of symbolic names onto physical console devices and allows all output from a particular console device to be logged to a file. Clients can connect to these virtual consoles in a read-only (monitor), read-write (interactive), or write-only (broadcast) mode. If a console is already in use, the client can either join with the existing clients or steal console “write” privileges. An Expect library allows scripts to be quickly developed and executed across multiple consoles in parallel.

2.4.4 PowerMan

When a cluster is booted, power to individual nodes may need to be staged to prevent circuit breakers from tripping. The system administrator may want to boot the entire cluster, an individual rack, or an individual node. And as with console management, remote power control becomes an issue when the cluster is remotely located.

PowerMan is a power management program designed to support a wide variety of power control and monitoring devices accessible via a TCP network connection. Because there is no standard protocol for interfacing with a power control device, PowerMan provides a flexible configuration that can be adapted to almost any hardware. It is capable of querying both plug and power supply

output status when available, and it can power-on, power-off, power-cycle, and hard-reset individual nodes or node ranges.

2.4.5 YACI

Livermore Linux clusters currently run with a full copy of the Red Hat operating system on each node. This was favored over a shared root file system approach because of Livermore system administration practices, concerns about performance and reliability of network file servers, and concerns about maintaining the integrity of Red Hat packaging. However, initial cluster deployment or operating system upgrades required a scalable method of imaging the operating system onto a large number of nodes. Because no available open source solutions (including VA System Imager and LUI) met our needs, we initiated an effort to build what we call Yet Another Cluster Installer (YACI) (Minich, in press).

A YACI cluster installation begins with the creation of disk partitions. RPMs are installed in a staging partition within a chrooted environment and are converted to compressed tar images ("tarballs"). Cluster installation then consists of installing the management node(s) from a YACI CD-ROM, gathering MAC addresses to construct a DHCP configuration, and network booting a stand-alone image onto the remaining nodes. This stand-alone image is used to partition the local disk and copy over tarballs of the disk partitions.

This approach turns out to be quite scalable. YACI was used to install the 128-node Adelie cluster, from scratch, in about an hour. This included bootstrapping the management node and performing post-installation configuration.

3 Strategic Challenges

In addition to the technical challenges listed above, there were several strategic challenges associated with migrating to Linux and open source. First, our desire for releasing locally developed software under an open source license was at odds with Laboratory practices concerning intellectual property. Second, we needed a mechanism for pushing local changes made to open source packages back into the mainstream release so as to minimize patch management overhead. Finally, we needed an overall strategy for transforming a collection of software packages into a cohesive production environment for LC.

3.1 Embracing Open Source

A core strategy of our project has been for the use of open source software wherever possible. This provides a hedge against "change in support" status from a particular vendor. Access to the source code gives us the ability to fix bugs quickly. It also allows us to incorporate HPC features that vendors are unwilling to support due to the unprofitable nature of a small niche market with demanding requirements.

The Director of the DOE Office of Simulation and Computer Science directed us to release our code to the open source community whenever possible. One of the reasons the Advanced Simulation and Computing (ASCI) Program has a strategic interest in the development of open source software is because the "open source distributed development model yields important contributions to the state of the art with significant leverage of the Government investment" (Reed, n.d.).

We release our software under terms of the GNU GPL. This allows others to leverage (and ideally contribute to) our work while ensuring that the source code remains open. By advocating

the GPL, we encourage our collaborators to do the same. Ultimately, this creates more open source software from which LLNL and other HPC sites can benefit.

Open-sourcing our software has provided other benefits. It simplifies the distribution of our software, which we can make available via our external Web site (http://www.llnl.gov/linux/). It also allows us to work more closely with our collaborators. For example, when we recently reported a hardware bug to one of our vendors, they were able to reproduce the problem quickly by downloading our software (which had uncovered the bug) from the web; by the next day, they had provided us with a firmware update correcting the problem.

3.2 Interacting with the Open Source Community

The problem of convincing proprietary software vendors to implement HPC features has a parallel in the open source community. When an open source package such as the Linux kernel is aimed at serving the needs of a diverse community, it is often difficult to convince package maintainers to incorporate HPC-specific changes into their mainstream releases, even if those changes are submitted in a fully realized form. Package maintainers understandably do not want to overcommit themselves by allowing their software to become bloated and overly complex and thereby fail to effectively support their main user base.

With open source, end-users can maintain a set of local changes they add on to each new release of the software. Source code version control systems help automate the process of retargeting patches, but as the number of local changes grows, this activity and the associated testing become a significant burden.

One option for addressing this problem is for software developers to become more visible and trusted in the open source community so that their changes are more likely to be accepted by package maintainers. This approach has met with limited success, in part because it is time consuming and tends to be prioritized lower than more tangible project deliverables.

A successful approach for us has been to form partnerships with companies and individuals already enjoying this rapport. For example, Livermore has hired Red Hat to provide an on-site consultant, and this consultant has gotten several changes incorporated into Red Hat's Linux distribution. Careful choice of alliances for software projects can also help. Peter Braam of Cluster File Systems and several of his employees are well known in the Linux kernel community and have gotten Lustre file system–related changes incorporated into the Linux kernel. The Compaq alliance to port Petal/Frangipani (see Sec. 2.2) failed in similar efforts, however.

3.3 Distributing CHAOS

Early on, we identified software integration testing and release discipline as problem areas for deploying Linux in production. In response, Livermore initially joined the Open Source Cluster Application Resources (OSCAR) (Ferri, 2002) consortium, which makes a software distribution for clusters. After a few months, it became clear that Livermore was not achieving much leverage from the collaboration and that a new strategy should be sought. For example, a significant portion of the OSCAR effort was directed at providing a broad base of users and environments with turnkey administration software for small clusters. LC systems, on the other hand, tend to be large, the workload and environment fairly homogeneous, and the system administration staff capable of interacting with systems at a deep level. Furthermore, most of software distributed with OSCAR was not needed by Livermore, and some software, especially

quasi-open packages such as the Portable Batch System (PBS), created a legal and support burden for OSCAR collaborators that diverted scarce resources from Livermore big-system issues.

After finding similar problems with cluster offerings such as NPACI Rocks and Scyld Beowulf, we decided to draw on our OSCAR experience and build a cluster operating environment that would narrowly focus on the LC environment. The Clustered High Availability Operating System (CHAOS) (Garlick, in press) effort was the result.

The CHAOS environment, based on Red Hat Linux, includes all the software the Linux Project develops or has otherwise identified for use on production Linux clusters. Because CHAOS is released only within Livermore, it can contain proprietary software (e.g., the Quadrics RMS) and can be narrowly targeted at our system administration staff, scientific user base, data center environment, and hardware platforms. CHAOS can therefore be developed with much less effort than a general-purpose environment such as OSCAR.

CHAOS has a well-defined software life cycle, which so far has been used to deploy two releases in production. CHAOS releases are loosely synchronized to Red Hat Linux distribution releases (approximately every 6 months), but they are also strongly driven by the need to support new production hardware. Each release is tested, packaged, documented, and deployed first on development clusters and then on progressively larger production clusters. The CHAOS team uses a source code version control system (CVS) and bug tracking system (GNATS), and it is surveying regression testing frameworks for use on future releases.

Another part of the CHAOS model is a strong relationship between system administrators and software developers. CHAOS releases are planned, built, and released by the two teams working together, and both teams can be brought to bear on problems arising in production.

4 Results

As discussed in Sec. 1, the Linux Project is ultimately concerned with deploying production Linux clusters to provide LC users with the Livermore Model programming environment. The Linux Project can be considered successful to the extent that these clusters are being actively utilized.

Since fall 2001, we have fielded several clusters. They have delivered the anticipated price/performance benefits, and users have been able to use the software environment effectively.

4.1 Production Systems

All Linux Project efforts before about July 2001 were focused on Alpha hardware. The first production Linux cluster, Furnace, purchased in June 2001, consists of 64 Alpha Processors, Inc. CS20 nodes. Each CS20 has two 833-MHz EV68 Alpha CPUs. Production Alpha/Linux systems (e.g., the Compaq LX cluster, which had no high-speed interconnect or parallel file system), were already on the floor prior to Furnace, but Furnace was intended to be the first Linux cluster providing the full Livermore Model with QsNet.

Just as Furnace was delivered, the Intel Pentium 4 processor, with surprisingly good SPECfp performance, was released. This and other changes within Compaq led Livermore to abruptly alter its processor strategy from Alpha to Intel. It was deemed necessary to focus development efforts on only one platform. The Furnace cluster was therefore deployed without its QsNet

Elan3 interconnect and is now operated as a loosely coupled cluster with a stock Red Hat environment similar to that of the LX cluster. Linux Project efforts were then refocused on evaluating Intel-based platforms.

In August 2001, the Parallel Capacity Resource (PCR) clusters were purchased. PCR was produced by an alliance between Silicon Graphics, Inc. and Linux NetworX, a small, Utah-based company that has produced commercial Linux clusters since 1997. PCR consists of three clusters: a 128-node production cluster (Adelie), an 88-node production cluster (Emperor), and a 26-node development cluster (Dev). Each PCR compute node has two 1.7-GHz Intel Pentium 4 CPUs and a QsNet Elan3 interconnect. Both production clusters are now in General Availability and have implemented all of the Livermore Model except the parallel file system. Instead, four dedicated, high-performance BlueArc NFS servers are used for storing data sets.

In July 2002, the MCR clusters were purchased. MCR is being produced by Linux Networx, this time without Silicon Graphics. It consists of an 1152-node production cluster and a 26-node development cluster. Each compute node has two 2.4-GHz Intel Pentium 4 CPUs. The production cluster uses a QsNet Elan3 federated switch consisting of 12 node switches and 4 trunk switches—the first of this configuration to be produced. Our intent is for MCR to utilize the Lustre Lite file system. Livermore has contracted with BlueArc to build a battery of Lustre OST appliances for MCR—another first.

Around this same time, PCR users requested a small cluster for the unclassified network that would allow them to develop their code in a less restrictive environment before running on the classified PCR systems. In response, the Pengra cluster was also procured in July 2002. It has 16 nodes, each with two 2.2-GHz Intel Pentium 4 CPUs and a QsNet Elan3 interconnect. Of particular interest is Pengra's cost-effectiveness; it was inexpensive to field because it was purchased without integration support from an outside vendor.

4.1.1 Interconnect Cost Effectiveness

The Linux clusters we have deployed all use the Quadrics QsNet Elan3 interconnect, but we have used three switch configurations depending on the size of the cluster. Pengra uses a single 16-port switch, whereas each PCR cluster uses a single 128-port switch. Because of its size, MCR uses a two-tiered federated switch.

In a federated switch, individual nodes are connected to node switches, which are in turn connected to higher-level trunk switches. Full bisection bandwidth can be preserved using 64U64D switches, where the 128 ports on a switch are divided into 64 up links and 64 down links. This topology scales up to 1024 ports using 16 node switches and 8 trunk switches. By sacrificing some of the bisection bandwidth, 32U96D switches can be used to scale up to 1536 ports using 16 node switches and 4 trunk switches. MCR achieves its 1152-node interconnect configuration using 12 node switches and 4 trunk switches.

Figure 3 shows the price per switch port for these configurations. The single 128-port switch is the most cost-effective; the full bisection bandwidth 1024-port federated switch is the most expensive. MCR (1152 ports) achieves a good compromise, sacrificing some bisection bandwidth in order to scale to a larger size using fewer trunk switches, thereby reducing the cost per port.

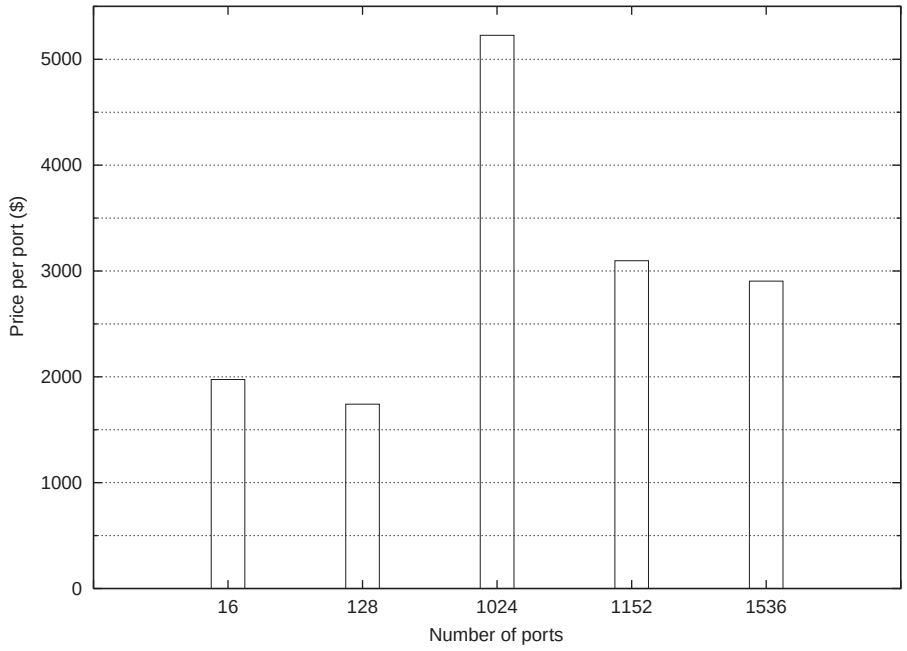


Figure 3. Cost effectiveness of QsNet Elan3 interconnect configurations.

4.1.2 Cluster Cost Effectiveness

Figure 4 shows the cost effectiveness of our production clusters.

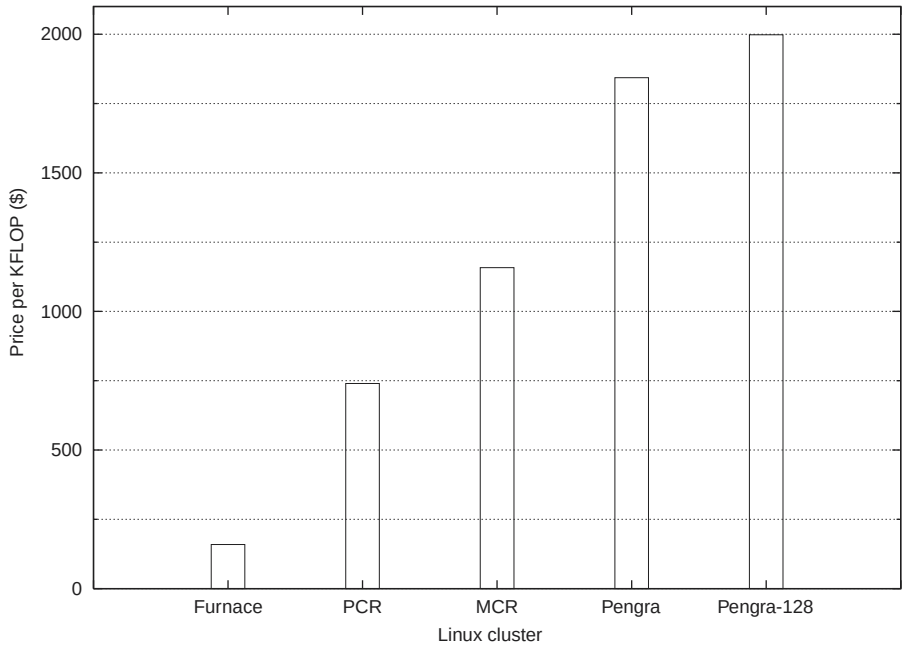


Figure 4. Cost effectiveness of the Linux production clusters.

The Furnace cluster was procured in June 2001. Around that time, we began investigating the IA32 line; the future of the Alpha line was in doubt, and the newly released 1.5-GHz Pentium 4 matched the SPECfp2000 benchmark of the 833-MHz Alpha for one-fifth the price (Seager,

2001. This led to the August 2001 procurement of the PCR IA32 clusters, which were almost five times more cost-effective than Furnace.

Now, almost a year later, we have begun procuring the MCR cluster. Although its federated switch cost per port is about 75% more than that of a single-switch configuration, this is more than offset by the savings attributable to Moore's Law and to greater economies of scale. At the same time, we have procured the small Pengra cluster. Although the cost per port of its interconnect is substantially less than that of MCR, most of the savings arise from the fact that we have performed the cluster integration ourselves. This is possible because Pengra is a relatively small cluster and because we have accumulated a vast amount of experience from the deployment of the Furnace and PCR clusters.

Pengra-128 (see Fig. 4) is a hypothetical cluster based on the same hardware as Pengra. Instead of using a 16-port switch, however, we take advantage of the cost-per-port savings of the 128-port switch. This hypothetical cluster is of interest as the most cost-effective configuration that could be fielded with the technology adopted so far by the Linux Project.

4.2 User Experience

User satisfaction with the production Linux clusters is high. Lin Yang, in his work with a Green's function simulation method (Yang, 2002), has obtained better performance on PCR than on ASCI White (IBM SP). Don Blackfield reports 18 times better performance using 120 nodes of PCR than using 64 nodes of TC2K (Compaq Sierra with QsNet) (Blackfield, 2002). Maria Caturla reports consistently better performance on her molecular dynamics code (MDCASK) on PCR than with the same number of nodes on ASCI White and ASCI Blue-Pacific, while scaling between 8 and 128 processors (Caturla, 2002). PCR is well utilized, and those scientists making effective use of the machine are now encouraging their colleagues to begin using this resource.

At the time this report was written, the only Generally Available tightly coupled Linux clusters resided on the classified network, so application performance information is somewhat limited. This fall, the MCR cluster will bring the Linux cluster environment to a whole new class of users and unclassified applications, so more feedback is expected. This feedback is vital to the success of the project and will be used to plan the next generation of Linux Project efforts.

4.3 Support Model

The support model for Linux within LC is similar to that for other platforms, except that local developers replace vendor hotlines at the back-end. A round-the-clock operations staff monitors the systems and pages the on-duty system administrator when something goes awry. System administrators attempt to resolve the problem, but they can page developers if needed. Users experiencing problems with the system or needing a feature contact the LC Hotline for front-end assistance; the Hotline forwards the information to system administrators and developers as needed. Numerous e-mail lists are used by users and support staff for communicating downtimes and software updates and for general system discussions.

5 Future Directions

Although successful, the Linux Project effort still has significant areas for improvement. The Lustre Lite parallel file system and the SLURM resource manager will both be deployed in fall 2002. These are substantial efforts that will require fine-tuning under production stresses. The CHAOS effort will continue to produce releases driven by design evolution, Red Hat updates,

and support for new production hardware. Formal integration testing will be added to the CHAOS software life cycle as job postings are filled. At 1152 nodes, MCR will test the scalability of Linux Project software by an order of magnitude, as will follow-on clusters expected in late 2002 and early 2003. Scalability will undoubtedly pose challenges that will require efforts whose scope and magnitude are difficult to predict.

In the longer term, 64-bit architectures such as IA64 and x86-64 will be evaluated for production readiness. Other interconnect architectures will be explored, including QsNet Elan4 and Infiniband. Lustre (the full PathForward implementation) will evolve toward production readiness. Additional partnerships with storage vendors to produce OST appliances will be solicited. Support within the Linux kernel for HPC features such as checkpoint restart and gang scheduling will be sought.

References

Blackfield, D.T. "Performance Study: Adelie versus Berg." Lawrence Livermore National Laboratory, CA, viewgraph presentation (June 4, 2002).

Braam, P.J., et al. *The Lustre Storage Architecture*. Cluster File Systems, Inc., Mountain View, CA (Sept. 17, 2002), from http://www.lustre.org/docs/lustre.pdf.

Caturla, M.J., A. Kubota, and J.S. Stölken. "PCR Linux Science Runs." Lawrence Livermore National Laboratory, CA, internal report (January 2002).

Ferri, R.C. "The OSCAR Revolution." *Linux Journal*, Issue 98 (June 1, 2002), from http://www.linux.com/article.php?sid=5559.

Free Software Foundation. *GNU General Public License*. Version 2, June 1991, from http://www.gnu.org/licenses/gpl.html.

Garlick, J.E. *Building a High Performance Raw Disk Subsystem for Alpha/Linux*. Lawrence Livermore National Laboratory, CA, UCRL-ID-144213 (June 30, 2001).

Garlick, J.E. and C.M. Dunlap. *Building CHAOS: an Operating Environment for Livermore Linux Clusters*. Lawrence Livermore National Laboratory, CA (in press).

Jette, M.A. and M.A. Grondona. *SLURM: Simple Linux Utility for Resource Management*. Lawrence Livermore National Laboratory, CA, UCRL-MA-147996-REV (August 28, 2002).

Lee, E.K. and C.A. Thekkath. "Petal: Distributed Virtual Disks." In *Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (ACM, Cambridge, MA, 1996) 84–92. Available from http://www.thekkath.org/papers/petal.pdf.

McVoy, L. *Scalability of the Linux Operating System to Large-Scale Parallel Computers*. Bitmover, Inc., San Francisco, CA (August 4, 1998).

Minich, R.M., T.E. D'Hooge, and M.J Miller Jr. *The YACI User's Guide*. Lawrence Livermore National Laboratory, CA (in press).

Reed, W.H. "Release of Open Source Software." U.S. Department of Energy, National Nuclear Security Administration, Office of Simulation and Computer Science, Defense Programs Memorandum (n.d.) [2001].

Ridge, D., D.J. Becker, P. Merkey, and T.L. Sterling. "Beowulf: Harnessing the Power of Parallelism in a Pile-of-PCs." In *Proceedings of the IEEE Aerospace Conference* (IEEE, Piscataway, NJ, 1997) vol. 2, 79–91.

Seager, M.K. "FY01 Production Clusters: Options for DNT Consideration." Lawrence Livermore National Laboratory, CA, viewgraph presentation (April 11, 2001).

Seager, M.K., et al. *Multiprogrammatic and Institutional Computing Capability Resource: Statement of Work*. Lawrence Livermore National Laboratory, CA, UCRL-CR-148022 (April 19, 2002).

Thekkath, C.A., T. Mann, and E.K. Lee. "Frangipani: A Scalable Distributed File System." In *Proceedings of the 19th ACM Symposium on Operating Systems* (ACM Press, New York, 1997) 224–237. Available from http://www.thekkath.org/papers/frangipani.pdf.

Yang, L. *Metals and Alloys Group, Green's Function Simulation Method*. Lawrence Livermore National Laboratory, CA, UCRL-WEB-148418, from http://www-phys.llnl.gov/Research/Metals_Alloys/Methods/GreenFunc/.

University of California
Lawrence Livermore National Laboratory
Technical Information Department
Livermore, CA 94551