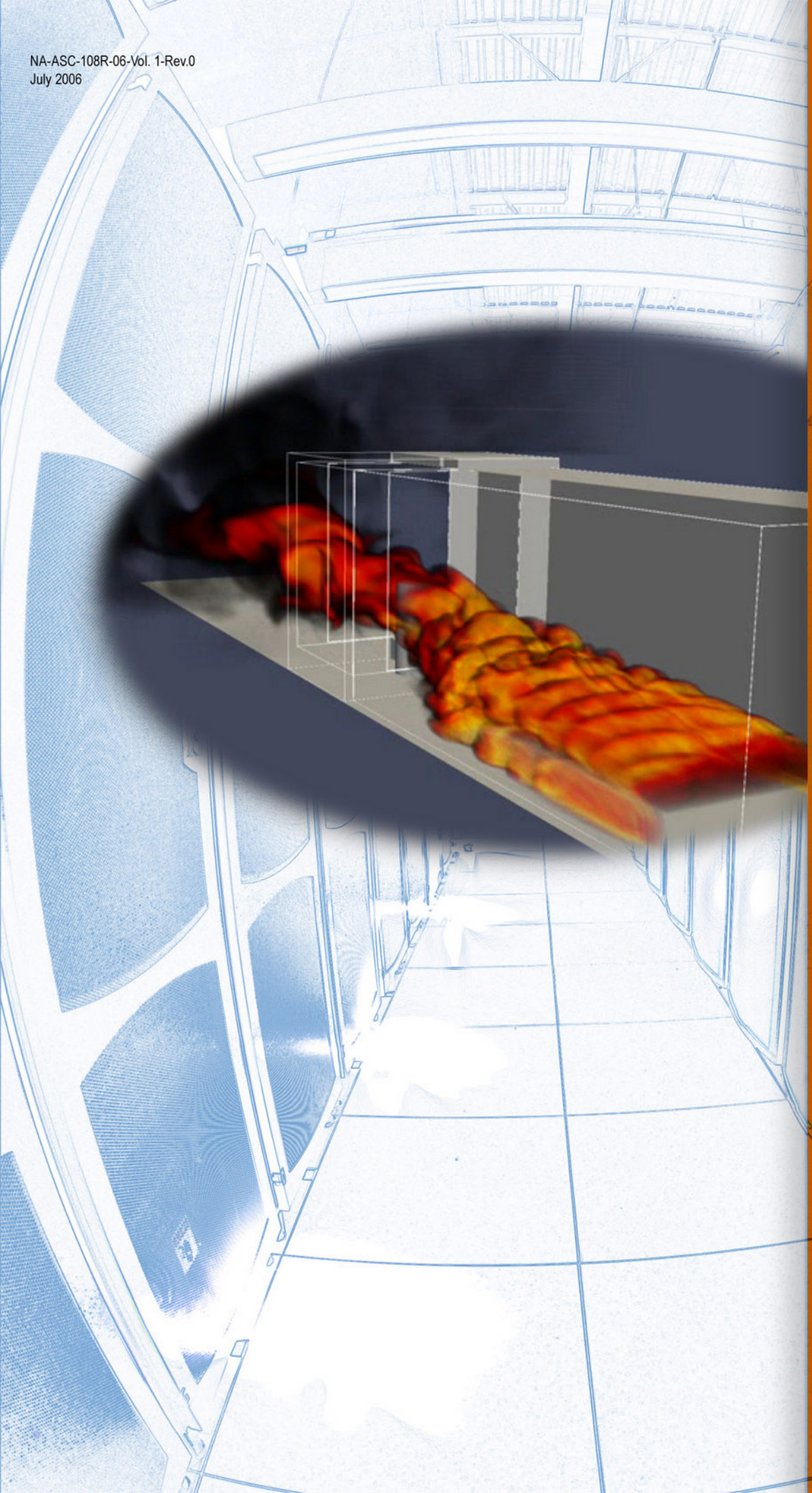
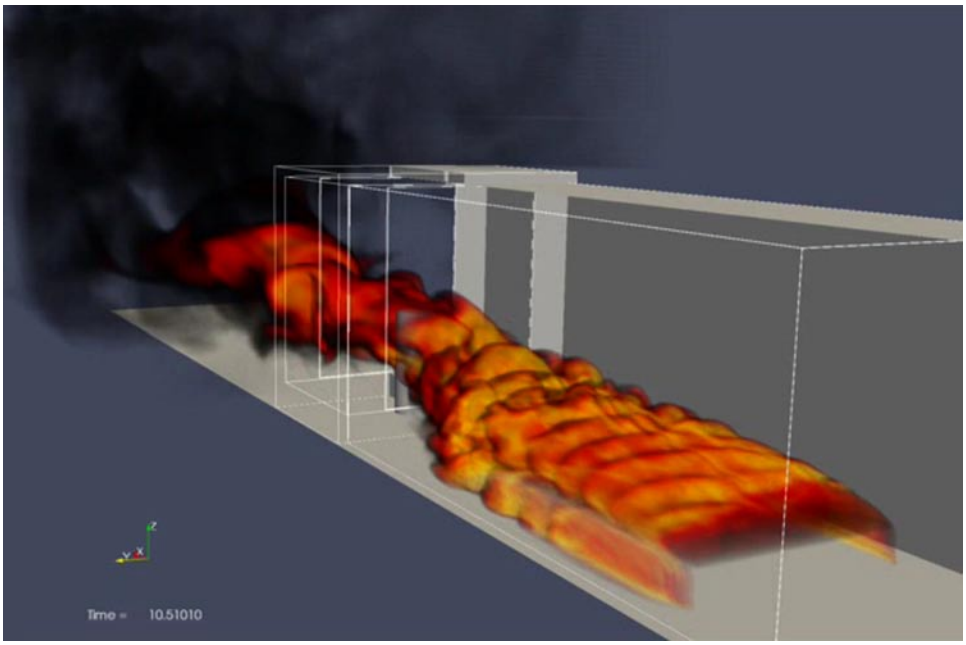
NA-ASC-108R-06-Vol. 1-Rev.0
July 2006



ASC™

Total Cost of Ownership of Linux/ Open Source Software

NNSA
National Nuclear Security Administration



ON THE COVER:

This Parallel Volume Rendering of a cross-wind fire simulation shows the temperature of gases. This 150 million degree-of-freedom simulation uses loosely coupled SIERRA framework's codes: Fuego/ Syrinx/Calore. It models objects in cross-wind fire (one fluid and participating media radiation region; two conducting regions). The simulation was run on 2,048 Red Storm processors. An ASC supercomputer, Red Storm is located at Sandia National Laboratories in Albuquerque, New Mexico. The simulation is a part of a qualification test plan for system testing to be conducted at the new Thermal Test Complex Cross Wind Facility at Sandia.

The ParaView scalable visualization tool was used to generate the rendering, using 64 nodes of a Linux-based visualization cluster integrated with the Red Storm environment.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

National Nuclear Security Administration

SAND 2006-3866P
Issued by Sandia National Laboratories for NNSA's Office of Advanced Simulation & Computing, NA-114.

For more information, contact Robert Meisner at bob.meisner@nnsa.doe.gov

Advanced
Simulation &
Computing

Total Cost of Ownership of Linux/Open Source Software in the Advanced Simulation and Computing Program:

A Quantitative and Qualitative Analysis of the Strategy

Authors:

Robert Meisner, NA-114,
Office of Advanced Simulation and Computing
NNSA Defense Programs

Charles Slocomb
NNSA/SAIC

Doug East
Lawrence Livermore National Laboratory

Jim Ang
Sandia National Laboratories

M'hamed A. Jebbanema
Los Alamos National Laboratory

A Publication of the Office of Advanced Simulation & Computing,
NNSA Defense Programs


LINUX
POWERED

Total Cost of Ownership[1] of Linux/Open Source Software in the Advanced Simulation and Computing Program:

A Quantitative and Qualitative Analysis of the Strategy

INTRODUCTION

The Advanced Simulation and Computing Program (ASC) is the cornerstone of the Stockpile Stewardship Program (SSP)—one of the most highly integrated technical programs designed to maintain the safety and reliability of the U.S. nuclear stockpile. ASC provides simulation capabilities (in the absence of underground testing) and computational resources to (a) support the annual stockpile assessment and certification, (b) study advanced nuclear-weapons design and manufacturing processes, (c) analyze accident scenarios and weapons aging, and (d) provide the tools to enable Stockpile Life Extension Program (SLEPs) and the resolution of Significant Finding Investigations (SFIs). These responsibilities require a portfolio of balanced resources including hardware, simulation software, and computer science solutions.

The Advanced Simulation and Computing Program (ASC) is the cornerstone of the Stockpile Stewardship Program (SSP)—one of the most highly integrated technical programs designed to maintain the safety and reliability of the U.S. nuclear stockpile.

This report focuses on the latter and addresses the importance of adopting a common operating system across the three National Nuclear Security Administration (NNSA) defense laboratories that contribute to the ASC Program.

Value of a Common Operating System

The computational environment at the NNSA national defense laboratories that meets mission needs[2] must satisfy three primary requirements: performance, functionality, and usability and security. The performance of the overall computational system (e.g., compute, move data, store, and visualize) must be sufficient to meet the demands of weapons simulations. The functionality and usability of the systems (e.g., operating system, system software such as schedulers, and file systems and software that allows the users to make efficient use of the system such as compilers, debuggers, libraries, and performance analysis tools) must be adequate to enable weapons physicists and engineers to develop the detailed simulation models necessary to address weapons problems. The system security must be robust to ensure that classified work is protected. This must be done at a cost that aligns computing with the other priorities of the program to meet mission needs.

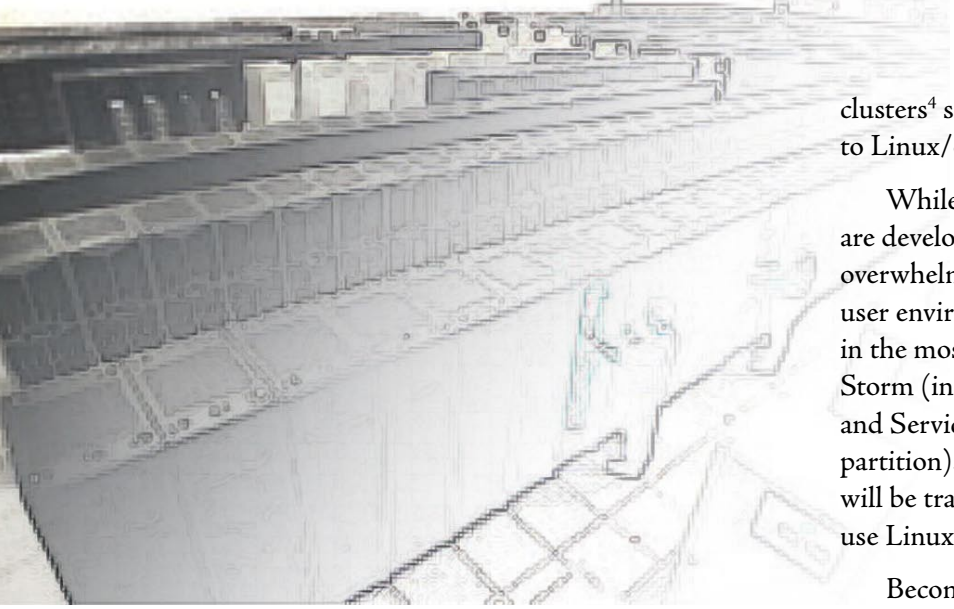
Central to meeting the above requirements is the acquisition of high-end computer systems that balance the total cost of ownership (including initial cost of the system, continuing software and hardware maintenance, and integration costs) with expectations of performance and usability. In section 2, we present a table that compares the costs of the commodity hardware / open source software strategy with the vendor systems strategy for all three NNSA weapons laboratories and demonstrates that the overall costs for Linux systems are substantially less than those for vendor systems.

The major elements of computer software that must be available are the operating system, system software that makes the system efficient in running user software, and software tools that allow the user to make efficient use of the system. Until recently, the only viable choice has been to acquire vendor hardware coupled with a vendor proprietary operating system. The remaining system software and software tools have been supplied by the vendor, by a third-party company or by one of the laboratories before doing their own development effort.

After acquiring the computer system, it must be integrated into the overall computing environment so that the system can be accessed by the user community

[1] See glossary for definitions. In our analysis, we assumed that the overhead costs (facility, consulting) are not significantly different for high-end Linux systems and high-end vendor systems, so those costs were not included.

[2] See *The ASC Strategy: the Next Ten Years*; NA-ASC-100R-Vol. #1 Rev. 0, August 2004



of the laboratory and so that the computing system has access to the laboratory's network storage system, visualization systems, and other services of the local computing environment. For the past decade, the vendor operating systems have been based on Unix, which has provided a significant benefit for integration and user interface. Even so, with each new procurement of a computer system, it has been necessary to learn the intricacies of the new system software and hardware, to integrate the system into the overall computing, network, visualization and storage environments, to develop the custom software that reflects the local needs, and to make the inevitable fixes to the proprietary software that come with using the system in a high-end scientific environment that is not duplicated anywhere else in the world. This repeated integration is costly and inefficient and puts a large burden on the end user, whose primary objective is to focus on the simulations and not to learn and debug a new computing system.

The laboratories are moving toward the use of open source software[3] to address the proliferation of divergent system software implementations and produce a more stable user interface that is consistent over multiple systems. With the Linux/open source environment, the envisioned ASC sites' strategy is to do the difficult integration work once, incorporate it into the open source software, and repeatedly use it when new high-end systems are acquired. The major assumption that drives the move to open source systems is that a stable environment controlled by the laboratories will increase user, system programmer, and system administrator productivity. The ASC capacity clusters[4] show the greatest potential for benefit from a move to Linux/open source systems.

While the capability platform[5] procurements are developed over many years and priority has been overwhelmingly given to performance over continuity of the user environment, efforts have been made to leverage Linux in the most recently acquired capability platforms, e.g., Red Storm (in its service partition and Reliability, Availability, and Serviceability subsystem), and BG/L (in its service partition). Much of the work done for capacity platforms will be transferable directly to new capability platforms that use Linux (as some will).

Becoming a part of the open source environment has other major advantages. The most obvious is that there is a large cadre of programmers, both inside the ASC complex and in the computing world at large, continually adding new features and fixing bugs for Linux and other open source software. This work for open source software makes the software more capable and is paid for by the institutions for whom the programmers work, in contrast to vendor software development, which is charged back to the end customer. Of course, the large and growing body of programmers who know the Linux/open source software can be hired by the national laboratories to work on their computers without the extensive training that may be required for vendor-proprietary software.

Continuity of systems is also cited as a critical issue by the weapons code developers and code physicists and, although there are no data to support it in this study, Linux/open source potentially will provide an enormous increase in user productivity over the long term.

System Integration

The major cost, besides the initial acquisition cost, of deploying a high-performance computing (HPC) system is not acquisition of the operating system or support software, but the integration required to make the target system work correctly and most efficiently in the environment of the laboratories. The integration work required to implement a Linux system is similar to that required for proprietary systems, but in the Linux case, the laboratories are the primary integrator and tester and do not depend on a vendor for essential integration work. Included in the deployment are the costs of integrating all the systems (storage,

[3]See Glossary for definition.

[4]See Glossary for definition.

[5]See Glossary for definition.

visualization) and software (compilers, debuggers, libraries, performance tools) to ensure that they work together and perform at a reasonable level.

For example, the integration of the Lightning system at Los Alamos required that several ingredients of the system merge for the first time—a new file system, the message passing software, and the system scheduling software. This integration requires both Linux system administrators and operating system developers to cooperate to uncover problems as they arise and to identify and fix functionality and performance issues. The same has been true for systems at Lawrence Livermore and Sandia.

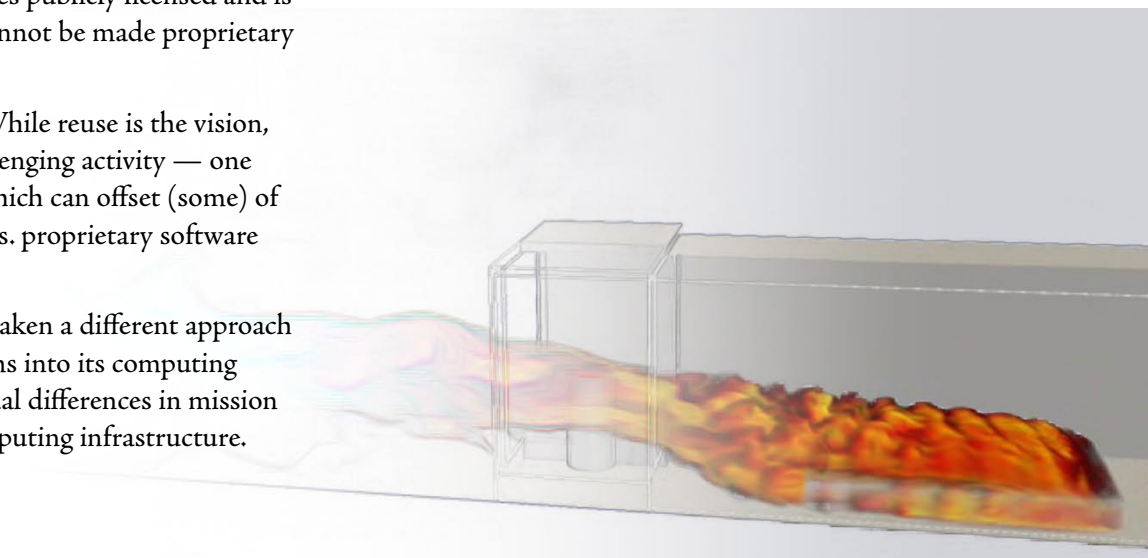
The defense laboratories do a significant amount of development work on Linux and other open source software to meet their own needs (as described above) and donate that software to the open source community. If the developed software is accepted into the open source Linux distribution, the maintenance issues for this software are reduced because then it does not need to be reintegrated with each new release. In the several years we have been using laboratory-produced software on our Linux system, a need to get industry's buy-in to new software developed has been recognized and is the mode of operation used for most new code development projects. This helps reduce the long-term support burden. Improvements to the software written by the laboratories' programming staff may be made by the whole community, increasing the leverage of the local system programmers. Of course if the laboratory-developed enhancements to the core Linux software are not accepted into the Linux distribution, it becomes the responsibility of the host organization to incorporate their changes into each new Linux operating system distribution with the accompanying configuration management challenges and incremental cost. Any software written for the Linux kernel distribution, whether accepted into the kernel distribution or not, becomes publicly licensed and is a part of the public domain. It cannot be made proprietary and cannot be restricted in use.

Reuse is the key to savings. While reuse is the vision, in actual practice, it can be a challenging activity — one that requires ongoing effort — which can offset (some) of the cost-benefits of open source vs. proprietary software approaches.

Each defense laboratory has taken a different approach to integrating Linux-based systems into its computing environments because of individual differences in mission requirements and in existing computing infrastructure.

In recent years, Sandia's mission needs were focused on production capacity systems that have a balance of cost, stability, and performance for the size (n =1 to 128 nodes with dual processor nodes) of problems they address. This usage model aligns well with a large critical mass of industry and academic scientific computing users; that is, most jobs are between 4 and 64 nodes and can use standard-release Linux software. Since it was not necessary to add laboratory-developed capabilities for scalability performance to a vanilla software stack, the porting costs for most Sandia MPI applications was minimal. The recent addition of Thunderbird to Sandia's user environment requires that the baseline commodity Linux software stack be modified to improve the scalability of these systems to address problems beyond 1,000 processors.

Both Los Alamos and Lawrence Livermore require capacity computing systems with a very large number of processors compared to most other scientific organizations (their capacity systems currently have more than 1000 processors). Modifications to the released version of the Linux operating system may be necessary or desirable to ensure that the capacity cluster can be made useful in the high-end scientific computing environment of the defense laboratories. Supporting the system management, user job scheduling, and applications programming needs for such large systems have required modifications to the kernel that are not included in the standard release. This requires significant technical expertise. Lawrence Livermore attempts to minimize and isolate kernel changes so that the software provided by third-party vendors is unaffected. Los Alamos attempts to provide a more manageable computing environment at the expense of making kernel modifications, which require third-party vendors to test and then demonstrate that their software works with the modified operating system.



ANALYSIS

Each defense laboratory has experience with systems running Linux and has seen a qualitative and quantitative advantage to that approach. Many, if not most, of the vendors providing systems that the laboratories can use to address their problems use the Linux operating system, and the work that the laboratories have done can be incorporated into their environments.

Our total cost of ownership analysis for these systems is detailed in the Appendix, "Supporting Data." We have focused on the cost per teraFLOP for vendor systems and Linux systems. The total cost of support and integration for each type of system and total cost including hardware acquisition was provided by the three defense laboratories. The dollar values given in the table for the hardware and support costs are based on the actual acquisition costs as reflected in contracts with vendors. The other costs are estimated costs, based on the expert judgment of the laboratory representatives in this study.

The Appendix shows that the cost per teraFLOP of vendor systems is about twice the cost for Linux systems for the systems examined ($2,754k/teraFLOP for vendor systems versus $1,413k/teraFLOP for Linux systems). The Q system[6] is now 3.5 years old. If we value Q according to today's costs, using Moore's law,[7] today's cost would be about one-fourth of its original cost. That would make the vendor systems total cost of ownership (TCO) $2,182k/teraFLOP versus the Linux TCO of $1,413k/teraFLOP—still a significant difference.

Another comparison of interest comes from the columns labeled "Vendor System ongoing cost per year" and "Linux ongoing cost per year." These show that the ongoing cost per year (summed across the three laboratories) to operate the vendor systems is $8,867k per year and $5,402k/yr to operate the Linux systems. This is also a significant cost savings to operate the Linux systems for a year.

The following table summarizes the data gathered from the laboratories:

	Total cost of ownership	Out-year annual costs
Vendor	$2182k/teraFLOP	$8867k/yr
Linux/open source	$1413k/teraFLOP	$5402k/yr

Not all teraFLOPs provide the same simulation power, but we believe that this comparison, which uses information from systems that we own and with which we have experience, provides insights that allow us to make informed future decisions. Q, Purple and Red Storm were acquired as capability systems. The ASC Program, because of the nature of its mission requirements, needs systems that are the most capable at the time of acquisition. Since the vendors who provided them were willing to build a system that may have been one of a kind, total costs were higher than for those computer systems sold in large quantities to universities and industries.

The Linux systems acquired by the laboratories are also high-end. They typically use commodity parts to construct the system (including central processing units (CPUs), networks, and memories) and the laboratories assume more of the integration costs. These make the user environment complete and functional by doing essentially all of the integration into the target computing environment. The laboratory role also includes taking the Linux system that is not fully developed for high-end systems and ensuring that it is maintainable, that jobs can be efficiently scheduled across the large number of processors, that users are able to debug their software to run on a large number of processors. The laboratories also handle all the other tasks that are affected by the use of thousands of processors instead of tens of processors. Acquiring a system that is mostly comprised of commodity hardware and open source software makes the initial cost of the system lower but increases the burden on the laboratories to make the system work. Additional costs include testbeds that are acquired to give the system development staff a place to do development without unduly disrupting the work environment for software developers.

[6]See Glossary for definition.

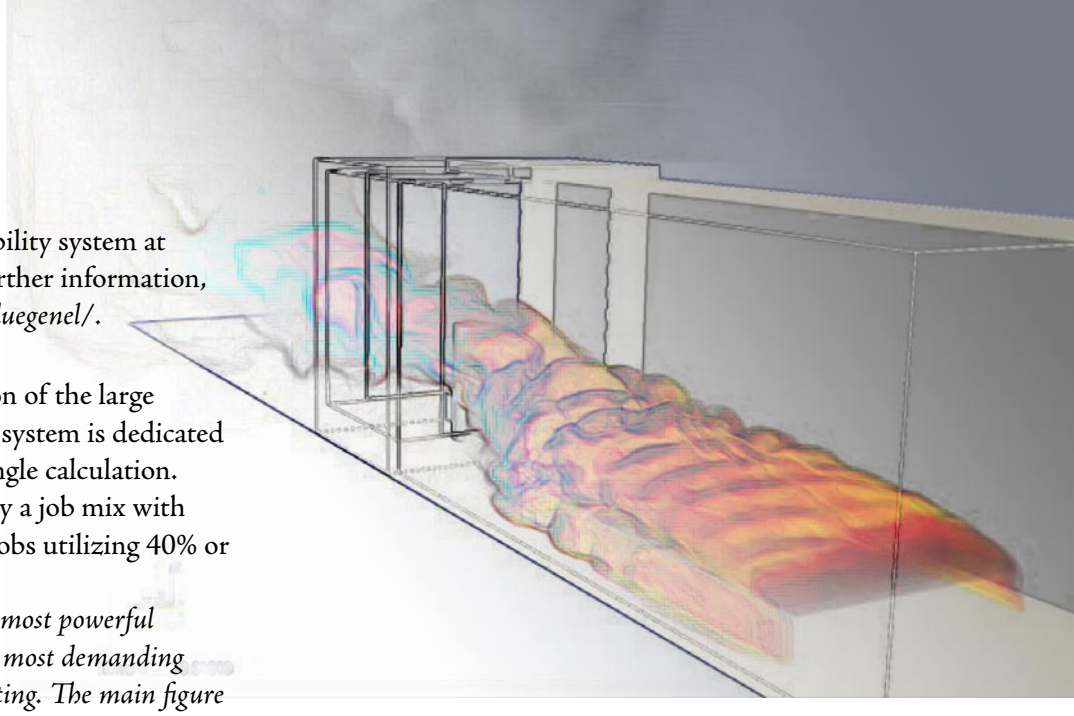
[7]See Glossary for definition.

SUMMARY

This study shows that the total cost of ownership for Linux systems is considerably less than that for vendor systems. Comparing the vendor proprietary systems with the Linux systems implemented at the laboratories, even when we bring the value of Q to what it would cost today, shows that the cost to implement and operate Linux systems is about 35% less than implementation of a vendor system and the cost to operate the Linux system is about 39% less.

With the new Tri-Laboratory Capacity Computing (TLCC), in which the integration costs will be done once and the successive systems will be able to fully utilize the software, the Linux capacity systems should be even less expensive. There is a major advantage to amortizing the integration costs over multiple sets of hardware, and reuse of the software will considerably reduce the cost of ownership. If the laboratories were also able to better share the software that they have developed for Linux, the total cost to the NNSA would be even further reduced.

Glossary



BG/L—Blue Gene Light, an ASC capability system at Lawrence Livermore Laboratory. For further information, see *http://www.llnl.gov/asc/platforms/bluegenel/*.

Capability platforms—a classification of the large parallel computing systems wherein the system is dedicated to, or capable of being dedicated to, a single calculation. Capability machines are characterized by a job mix with few simultaneous jobs, with individual jobs utilizing 40% or more of the system's compute nodes.

Capability computing *- The use of the most powerful supercomputers to solve the largest and most demanding problem, in contrast to capacity computing. The main figure of merit, metric, in capability computing is time to solution.*

Capacity clusters—a classification of parallel computing systems that are not used as capability machines. A job mix of many simultaneous jobs characterizes capacity machines. Historically, today's capability platforms become tomorrow's capacity machines as technology progresses.

Capacity computing *- The use of smaller and less expensive high-performance systems to run parallel problems with more modest computational requirements, in contrast to capability computing. The main figure of merit, metric, in capacity computing is the cost/performance ratio.*

Lightning system—Lightning, a Linux cluster system at Los Alamos National Laboratory, has 1,408 dual-processor AMD Opteron nodes with a Myrinet interconnect and a peak speed of 13.3 teraFLOPs. Lightning has 77 terabytes of temporary high-speed parallel storage and enjoys an award-winning architecture developed at Los Alamos (Science Appliance) and a software suite (Clustermatic) that can completely manage a cluster.

Linux—An operating system developed by Linus Torvalds, a student at the University of Helsinki in Finland. Linus' interest in Minix, a small UNIX system led to the development of the Linux Kernel which is at the heart of all Linux systems.

For more information, see *http://www.linux.org/info/index.html*

Moore's Law—The observation made in 1965 by Gordon Moore, co-founder of Intel, that the number of transistors per square inch on integrated circuits had doubled every year since the integrated circuit was invented. Moore predicted that this trend would continue for the foreseeable future.

In subsequent years, the pace slowed down a bit, but data density has doubled approximately every 18 months, and this is the current definition of Moore's Law, which Moore himself has blessed. Most experts, including Moore himself, expect Moore's Law to hold for at least another two decades. *(Source: Webopedia)*

National Defense Laboratories—Los Alamos National Laboratory (Los Alamos, New Mexico), Lawrence Livermore National Laboratory (Livermore, California), and Sandia National Laboratories (Albuquerque, New Mexico).

NNSA—National Nuclear Security Administration, a semi-autonomous agency within DOE.

Open source software—Computer programs or operating systems for which the source code is publicly available are referred to as open-source software. Inherent in the open source philosophy is the freedom of a distributed community of programmers to modify and improve the code. The most widely known example of open-source software is the Linux operating system. For more information, see *http://iet.ucdavis.edu/glossary.cfm*

Glossary (continued)

Purple—Purple is an ASC capability system located at Lawrence Livermore National Laboratory:100-teraFLOPs system. A huge machine, it is based on symmetric shared-memory multiprocessors (SMP) containing more than 12,000 next-generation IBM Power5 microprocessors.

Q system—A 30-teraFLOPs ASC supercomputer system located at Los Alamos National Laboratory was installed in 2000 as a 12.3-teraFLOPs system.

Red Storm—A 40-teraFLOPs ASC capability system, located at Sandia National Laboratories.

TeraFLOP—A teraFLOP is a measure of a computer's speed and can be expressed as:

- A trillion floating point operations per second
- 10 to the 12th power floating point operations per second (FLOPs)
- 2 to the 40th power flops.

Today's fastest parallel computing operations are capable of teraFLOP speeds. Scientists have begun to envision computers operating at petaflop speeds.

Thunderbird—A 4,096-node Dell high-performance capacity computer cluster located at Sandia National Laboratories will provide more than 8,000 processors of compute capacity. The aggregated capacity of the computer will have approximately 24 terabytes memory and a speed of 60 teraFLOPs.

Total Cost of Ownership (TCO)—Below are two definitions:

1. Model that helps IT professionals understand and manage the budgeted (direct) and unbudgeted (indirect) costs incurred for acquiring, maintaining and using an application or a computing system. TCO normally includes training, upgrades, and administration as well as the purchase price. Lowering TCO through single-point control is a key benefit of server-based computing. *www.e-formation.co.nz/glossary.asp*

2. Refers to the administrative costs associated with computer hardware and software purchases, deployment and configuration, hardware and software updates, training, maintenance, and technical support. *www.asu.edu/it/w2k/glossary.html*

APPENDIX— Supporting Data

In the following tables, we estimate the total cost of ownership for the last high-end system that was integrated into the computing environment for each of the laboratories and for the last high-end Linux system that was integrated. Each entry in the table reflects the sum of the costs at each individual laboratory. The cost per teraFLOP is the sum of the costs for an assumed seven-year lifetime divided by the total teraFLOPs for the systems analyzed.

The column headings describe the phases for the vendor systems and for the Linux systems.

The first columns, for "Vendor System Integration" and for "Linux Systems Integration" summarize the integration costs. Integration, for the purposes of this discussion, is the work that is necessary to be done on a system that makes it ready for limited availability to users.

The second column, "Vendor System Cost Through the End of Vendor Agreement") gives the costs after the initial system integration and through the acquisition agreement. For Red Storm, that is through the 4th year. For Q and Purple that is through the 3rd year. The third column ("Vendor System Ongoing Cost to End of 7-Year Period") gives the costs from the end of the period covered by the vendor acquisition agreement to the system's end of life (defined here as seven years). So the total lifetime cost for vendor systems is the sum of the first three columns including the acquisition cost.

The Linux systems do not have the same complexity in the acquisition agreement, so the ongoing costs start right after the integration period and the TCO is the sum of the integration costs for the first year, the next six years, and the initial acquisition cost. Again the system's lifetime is assumed for simplicity to be seven years (although it would be remarkable to cost-effectively run these systems for so long).

We have also estimated the expected integration cost of the TLCC procurement, the goal of which is to provide the same basic capacity computing platform to all three defense laboratories. The TLCC platforms may be a different architecture (different processor, different interconnect) than the Linux systems already integrated and so the integration costs will be higher. The plan is to acquire multiple systems with this architecture to minimize the initial integration costs of the subsequent systems.

The columns labeled "Vendor System Ongoing Cost Per Year" and "Linux Ongoing Cost Per Year" reflect the yearly cost to operate the systems after the initial costs of integration and other system services that have been included as part of the initial acquisition cost (such as maintenance and system analysts) have been completed. In this analysis, we are assuming that the ongoing cost to maintain future Linux systems will be the same as for past Linux systems.

The components of cost of ownership that we have identified and attempted to quantify in the table include the following:

System Programmers—Laboratory programmers who make changes or additions to the operating system or the user environment or do integration work that requires deep knowledge of the system involved. Usually the system programmer staff is responsible for all of the systems of a particular kind. For example, the Linux system programmers are responsible for all of the Linux systems at a site, and when changes are made, those changes are relayed to all the Linux systems. This is true for both vendor-supplied systems and for Linux systems. For purposes of this analysis, we have estimated the amount of effort that was required for the integration of the system (usually requiring more system programmer effort) and the amount of effort that is required to keep the system running.

System Administrators (managers)—Laboratory computing experts who ensure that the system is configured to meet laboratory needs and who are responsible for keeping the system running efficiently and meeting user needs. System administrators usually are responsible for multiple systems. For this analysis, we have estimated the amount of effort that was required to initially make the system usable (usually requiring more system administrator effort) and the amount of effort that is required to keep the system running.

Vendor Analysts—Vendor representatives who work at the laboratories whose responsibilities include interfacing with the vendor, ensuring the correct operation of the system, and acting as an expert resource to laboratory people. There are vendor analysts for both vendor proprietary and for Linux systems.

System Software Purchase—The original cost of the operating system software.

Hardware/Software Maintenance Dollars—These are costs, either that the laboratories paid a vendor or that were incurred in house for software and hardware maintenance.

Third-Party Software Purchase— Cost of purchasing the third party software that runs on the system, such as debuggers, schedulers, compilers, and libraries.

Open Source Software— (Tools, file systems, I/O development and integration). Cost of people for integration and on-going support for open source software.

Interconnect Support—Includes network and interconnect software support, such as MPI

Hardware Acquisition Cost—is the cost of the hardware only part of the system. It is a number that we derived from the total vendor bid for the system by subtracting from the total cost those costs that are not directly hardware, such as hardware maintenance, software costs, software maintenance, and vendor support personnel.

The following table shows the total tri-lab costs comparing vendor proprietary system costs with initial and continuing Linux/open source costs.

Notes:

1. Costs are in $k.
2. Full-Time Equivalent (FTE) costs normalized to years @ $280 per FTE.
3. Vendor systems: Purple 100TF, Red Storm 41.5 TF, Q 20 TF. The sum is 161.5 TF.
4. Linux Systems: Thunder 19.94TF, NWCC 16.5 TF, Lightning 13.3TF. The sum is 49.74TF.

TCO Components	Phases							
	Vendor System Integration	Vendor System Cost Through End of Vendor Agreement	Vendor System Ongoing Cost to End of 7-Year Period	Vendor System Ongoing Cost Per Year	Linux System Integration	Tri-laboratory Capacity Computing Integration	Linux Ongoing for 6 years Beyond Year of Acquisition	Linux Ongoing Cost Per Year
System Programmers	1,050	3,906	2226	1022	946	2696	5460	910
System Administrators (managers)	2,975	6,300	7980	2940	1470	2030	13020	2170
Vendor Analysts	0	3,000	3000	1000	409	409		
System Software Purchase	0		0	0	0	25	300	50
Hardware/Software Maintenance $s	2360	13050	9132	3044	125	360	6594	1099
Third Party Software $s	590	553	413	161	750	750	2052	
Open Source Software (Tools, file systems, I/O development and integration)	560	1540	980	420	747	1587	1848	308
Interconnect Support	560	980	700	280	519	1079	2130	355
Hardware Acquisition Cost	382,983				32,912			
Total	391,078	29,329	24431	8867	37,878	8936	32412	5402
Lifetime Cost			444,838				70,290	
Total Cost of Ownership/TF			2754				1,413	

Note 1: The SNL Vendor System Integration is for Red Storm.
Note 2: The SNL Vendor System Ongoing is per Cray contract and support of SNL developed capabilities.
Note 3: The SNL Linux Integration is for NWCC (Nuclear Weapons Commodity Cluster), a commodity turn key cluster similar to a 1SU TLCC system.
Note 4: The SNL Linux Ongoing is for NWCC (Nuclear Weapons Commodity Cluster), a commodity turn key cluster similar to a 1SU TLCC system.
Note 5: The SNL Linux Ongoing maintenance fees are per HP contract.
Note 6: The SNL Linux + 1 generally assumes activities to extend scale commodity clusters to address mid-range capability problems.

Note 6: The LLNL Vendor System Integration is for Purple.
Note 7: The LLNL Vendor System Ongoing is per Purple contract.
Note 8: The LLNL Linux Integration is for Thunder.
Note 9: The LLNL Linux Ongoing costs (esp. staffing) attempt to isolate or amortize only those associated with Thunder.
Note 10: The LLNL Linux + 1 assumes a TLCC-like architecture which is new to LLNL and therefore has a similar level of integration complexity as Thunder.

Note 11: The LANL Vendor System Integration is for Q.
Note 12: The LANL Vendor System Ongoing is per Q contract.
Note 13: The LANL Linux Integration is for Lightning.
Note 14: The LANL Linux Ongoing is for Lightning.


LINUX
POWERED

Thunderbird

Sandia National Laboratories

Lightning

Los Alamos NATIONAL LABORATORY

Thunder

Lawerence Livermore National Laboratory