7X Performance Results – Final Report: ASCI Red vs. Red Storm

Joel 0. Stevenson, Robert A. Ballance, Karen Haskell, and John P. Noe, *Sandia National Laboratories* **and Dennis C. Dinge, Thomas A. Gardiner, and Michael E. Davis**, *Cray Inc.*

ABSTRACT: *The goal of the 7X performance testing was to assure Sandia National Laboratories, Cray Inc., and the Department of Energy that Red Storm would achieve its performance requirements which were defined as a comparison between ASCI Red and Red Storm. Our approach was to identify one or more problems for each application in the 7X suite, run those problems at two or three processor sizes in the capability computing range, and compare the results between ASCI Red and Red Storm. The first part of this paper describes the two computer systems, the 10 applications in the 7X suite, the 25 test problems, and the results of the performance tests on ASCI Red and Red Storm. During the course of the testing on Red Storm, we had the opportunity to run the test problems in both single-core mode and dual-core mode and the second part of this paper describes those results. Finally, we reflect on lessons learned in undertaking a major head-to-head benchmark comparison.*

KEYWORDS: 7X, ASCI Red, Red Storm, capability computing, benchmark

This work was supported in part by the U.S. Department of Energy. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States National Nuclear Security Administration and the Department of Energy under contract DE-AC04-94AL85000

1. Introduction

Background

The goal of the 7X performance testing is to assure Sandia, Cray, and DOE that Red Storm will achieve its performance requirements and to assess whether major applications will realize at least a seven-fold performance increase on the new Red Storm system relative to its predecessor ASCI Red. The focus is on problem sizes and processor counts representative of capability computing; i.e. single application runs that use 20% to 100% of the processors on ASCI Red.

The performance tests are defined as a comparison between ASCI Red and Red Storm. In general, the Red Storm contract calls for a series of speedup comparisons using selected applications at various problem sizes [1]. Our approach is to identify one or more problems for each application, run those problems at two or three processor sizes, and compare the results between ASCI Red and Red Storm.

The Red Storm supercomputer, originally built in 2005 with 2.0 Ghz single-core Opteron processors, received several upgrades to processors, memory, and system software in 2006 and 2007. Red Storm compute nodes are now 2.4 Ghz dual-core Opteron processors. Users have the option of running applications on both cores on each node of the system, or on only one core per node. The latter option is useful for those applications which are memory intensive and cannot abide having a node's memory partitioned between the two cores. During the course of executing 7X applications on Red Storm, results were collected in both modes.

Since the processor speeds were upgraded by 20% (from 2.0 Ghz to 2.4 Ghz) on Red Storm before results were gathered, the real target should now be 8.4X, not 7X. We will, however, continue to use the 7X descriptor

throughout this report when referring to performance testing.

The report is organized in the following manner: Section 2 and Section 3 discuss the guidelines for application/problem selection and application/problem runs, respectively. Section 4 contains a detailed listing of the applications under test with problem size information. Section 5 contains an overview of activities and roles and Section 6 discusses project and data management. 7X performance testing results are presented in Section 7 and Section 8 discusses single-core vs. dual-core results. Section 9 contains a summary and lessons learned in undertaking a major head-to-head benchmark comparison.

It will be useful to discuss the basic history, layout, and operation of ASCI Red and Red Storm in order to fully understand and contrast their performance.

ASCI Red

ASCI Red, the first computer in the Advanced Strategic Computing Initiative (ASCI) program, was built by Intel and installed at Sandia in late 1996. The design was based on the Intel Paragon computer. The goal was to deliver a true teraflop machine by the end of 1996 that would be capable of running an ASCI application using all memory and nodes by September of 1997. In December, 1996, three quarters of ASCI Red was measured at a world record 1.06 TFLOPS on MP LINPACK and held the record for fastest supercomputer in the world for several consecutive years, maxing out at 2.38 TFLOPS after a processor and memory upgrade in 1999 [2, 3, 4]. ASCI Red was decommissioned in 2006, shortly after completing the required 7X runs.

The ASCI Red supercomputer was a distributed memory MIMD (Multiple Instruction, Multiple Data) message-passing computer. The design provided high degrees of scalability for I/O, memory, compute nodes, storage capacity, and communications; standard parallel interfaces also made it possible to port parallel applications to the machine. The machine was structured into four partitions: Compute, Service, I/O, and System. Parallel applications executed in the Compute Partition which contained nodes optimized for floating point performance. The compute nodes had only the features required for efficient computation – they were not purposed for general interactive services. The Service Partition provided an integrated, scalable host that supported interactive users (log-in sessions), application development, and system administration. The I/O Partition supported disk I/O, a scalable parallel file system and network services. The System Partition supported initial booting and system Reliability, Availability, and Serviceability (RAS) capabilities. A block diagram of ASCI Red illustrating the Compute, Service, I/O, and System partitions is reproduced in Figure 1.

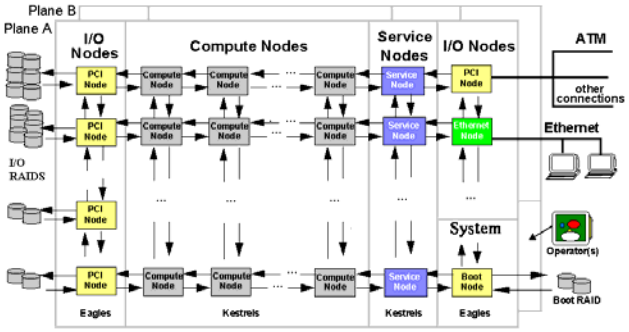

Figure 1. ASCI Red Block Diagram – Compute, Service, I/O, and System Partitions.

In normal operation, disconnect cabinets divided ASCI Red into two sides; unclassified and classified. In this situation, each side appeared as a separate plane in the mesh topology. In its full configuration, ASCI Red consisted of four rows, each with an unclassified end and a classified end. The unclassified ends of the rows appeared to the users as a single machine named Janus, and the classified ends of the rows appeared as a single machine named Janus-s. Each end was a significant parallel computer in its own right, with a peak computational rate of approximately 780 Gflop. Since each end was always connected to a LAN (barring catastrophic failure or other rare circumstance), files on both systems were always available. Each end had its own set of disks for file storage, service nodes to handle user logins, I/O nodes to handle I/O requests, and system nodes for system monitoring and control, in addition to the computational nodes, on which parallel applications ran.

The precise configuration was 1168 compute nodes on the unclassified end and 1166 compute nodes on the classified end. The middle section contained 2176 compute nodes and could be switched between the unclassified end and the classified end. Thus the total number of compute nodes on ASCI Red was 4510.

ASCI Red used two operating systems, the Teraflops Operating System on the Service, I/O, and System Partitions, and a Sandia-developed lightweight kernel (Cougar) on the Compute nodes. The Teraflops Operating System was Intel's distributed version of UNIX (POSIX 1003.1 and XPG3, AT&T System V.3 and 4.3 BSD Reno VFS) developed for the Paragon XP/S Supercomputer. It was a full-featured version of UNIX, used for boot and configuration support, system administration, user logins, user commands/services, and development tools. The operating system in the Compute Partition was Cougar, which was Intel's port of Puma, a light-weight operating system for the TOPS, based on the very successful SUNMOS system for the Paragon. SUNMOS, and subsequently Puma, were developed by Sandia National Laboratories (SNL) and the University of

New Mexico. Cougar was a very efficient and high-performance operating system providing program loading, memory management, message-passing support, some signal handling and exit handling, and run-time support for the supported languages. Cougar was very small, occupying less that 300 KBytes of RAM.

This combination of operating systems made it possible to specialize for specific tasks and standard programming tools to make the supercomputer both familiar to the user and non-intrusive for the scalable application. The machine provided a single system image to the user. The users perceived the system as a single UNIX machine even though the operating system was actually running on a distributed collection of nodes. To the user, the system had the look and feel of a UNIX-based supercomputer. All the standard facilities associated with a UNIX workstation were available to the user, yet the compute partition running the Cougar operating system had only those features required for computation. The Cougar operating system could therefore be small in size and very fast. The combination of these two operating systems was a powerful approach.

Since Cougar was a minimal operating system, system services and support for the interactive user were provided by the host operating system (in this case, the Paragon-derived UNIX OS running in the Service Partition). All access to hardware resources came from the Q-Kernel, the lowest-level component of Cougar. Above the Q-Kernel sat the process control thread (PCT), which ran in user space and managed processes. User applications sat at the highest level. As with most MPP systems, the basic programming model in Cougar was based on message passing. FORTRAN77, FORTRAN90, C and C++ were supported. The interactive debugger and performance analysis tools understood these languages and mapped onto original source code.

One interesting feature of ASCI Red concerned processor "mode". While message passing was used between nodes, shared memory mechanisms were used to exploit parallelism on a node. Each compute node had two processors. The second processor could be used in one of four modes:

- Proc 0 option – ignored the second processor – default mode – entire system RAM on the node was available to the application (256 MB)
- Proc 1 option – used the second processor as a communication co-processor.
- Proc 2 option – used the second processor to run an additional application thread.
- Proc 3 option – this mode treated each processor as a separate compute node – virtual mode – the processors shared memory so only half the system RAM was available to the application (128 MB).

The 7X testing was performed on ASCI Red in Proc 0 and Proc 3 modes only.

Red Storm

Red Storm, the follow-on computer to ASCI Red, was built by Cray and installed at Sandia in early 2005. Red Storm is a distributed memory, massively parallel supercomputer modeled on ASCI Red. Red Storm itself is a dual-headed machine, being split between classified (Red) and unclassified (Black) use. Each end is anchored in a specific network. The classified portion of Red Storm (redstorm-s.sandia.gov) is anchored in Sandia's Classified Network, the SCN. The unclassified portion of Red Storm (redstorm.sandia.gov) is anchored in Sandia's Restricted Network, the SRN.

The Red Storm architecture facilitates simultaneous usage on the unclassified and classified sides of the machine. In normal operation, disconnect cabinets divide Red Storm into two sides; unclassified and classified. The initial configuration was 2688 compute nodes on the unclassified end and 2688 compute nodes on the classified end. The middle section contained 4992 compute nodes and could be switched between the unclassified end and the classified end. Thus the total number of compute nodes was 10368.

A fifth row of cabinets was added in an August-October 2006 upgrade, bringing node counts to 3360 on the unclassified side and 3360 on the classified side. The middle section contains 6240 nodes. The total number of compute nodes is now 12960. Each compute node was upgraded to dual-core topology, bringing total processor count to 25920. Processor speed was upgraded from 2.0 Ghz to 2.4 Ghz.

In 2005, Red Storm was measured at 36 TF on MP LINPACK. Following the fifth row addition and upgrade from 2.0 Ghz single-core Opteron processors to 2.4 Ghz dual-core Opterons in 2006, Red Storm was measured at 101.4 TF on MP LINPACK. Red Storm was measured at 102.2 TF on MP LINPACK in 2007 [5].

Red Storm combines commodity and open source components with custom-designed components to create a system that can operate efficiently at immense scale. The basic scalable component is the node. There are two types of nodes; compute nodes run user applications and service nodes provide support functions, such as managing the user's environment, handling I/O, and booting the system. Basic internal services such as networking and file system access run on the specially designated and configured service nodes. Each compute node or service node is a logical grouping of a processor, memory, and a data routing resource.

Cray XT3 systems use a simple memory model: for applications distributed across numerous nodes, each instance of the application has its own processor and local memory. Remote memory is the memory on the nodes running the associated application instance – there is no shared memory.

The system interconnection network is the data-routing resource that Cray XT3 systems use to maintain

high communication rates as the number of nodes increases. The system interconnection network enables the system to achieve an appropriate balance between processor speed and interconnection bandwidth.

To a user, Red Storm appears as a collection of Linux-based login nodes that have access to the Red Storm file systems as well as the compute nodes. Activities such as compilation, job submission, and job monitoring are performed on login nodes. The collection of login nodes appears to the user as a single system.

At the core of Red Storm is the compute partition, where parallel jobs execute. Since the compute partition is switchable between the classified and unclassified ends of the machine, the actual size of the compute partition on either end will vary over time.

Each Red Storm compute node has dual-core topology. The 7X testing was performed on Red Storm in both SN and VN modes.

- SN option – ignore the second processor – default mode – entire system RAM on the node is available to the application.
- VN option – each processor is a separate compute node – only half the system RAM on the node is available to each processor.

The software environment is summarized as follows: Operating systems include Linux on service and I/O nodes (SuSE Enterprise Server), Catamount VN lightweight kernel on compute nodes, and Linux on RAS monitors. The run-time system includes a logarithmic job launch utility (yod), the node allocator (CPA), and the batch system workload manager (MOAB). The high performance file system is Lustre. The user environment includes PGI compilers (Fortran, C, C++), various libraries (MPI, I/O, Math, MPI-2), the showmesh utility for displaying node states and job layouts on the mesh, the Totalview debugger, and a performance monitor.

The lightweight compute node OS is fundamental to the Sandia architecture. It is essential for: (1) maximizing CPU resources, by reducing OS and runtime system overhead; (2) maximizing memory resources, with a small memory footprint and large page support; (3) maximizing network resources, with no virtual memory and physically contiguous address mapping; (4) increasing reliability, with a small code base and reduced complexity; (5) deterministic performance, with a high degree of repeatability; (6) scalability, for which OS resources must be independent of job size.

Other computing systems in the Red Storm Environment are used for job preparation (such as meshing) and visualization. Visualization can be performed on Red RoSE and Black RoSE, companion clusters to Red Storm that support classified and unclassified visualization and data services, respectively. High-speed data links ensure fast data migration between Red Storm and other systems inside Sandia's computing environment. Special scripts ensure that the data movement is both simple and robust.

While Red Storm has many characteristics in common with ASCI Red, it also differs in many ways. The system parameters for ASCI Red and Red Storm are summarized in Table 1 [3, 5]. Red refers to the Classified (SCN) side and Black refers to unclassified (SRN) side.

	ASCI Red	Red Storm
Compute Nodes (Red/Center/Black)	4510 (1166/2176/1168)	12960 (3360/6240/3360)
Compute Processors (Red/Center/Black)	9020 (2332/4352/2336) PII Xeon 333Mhz	25920 (6720/12480/6720) Opteron Dual-core 2.4Ghz
Service Nodes (Red/Black) Disk I/O Nodes (Red/Black)	52 (26/26) 73 (37/36)	640 (320/320) Service and I/O partition (login, service, I/O, administrative nodes)
System Nodes (Red / Black)	2 (1/1)	RAS and System Management Partition
Network Nodes (Red/Black)	12 (6/6) Ethernet ATM	100 (50/50) 10GigE to RoSE 20 (10/10) 1GigE to login nodes
Number of Cabinets	96 (76 compute/20 disk)	155 (135 compute/20 service and I/O)
Interconnect Topology	3-D Mesh (x,y,z) (38x32x2)	3-D Mesh (x,y,z) (27x20x24)
Architecture	Dist. Memory MIMD	Dist. Memory MIMD
Theoretical Peak Performance	3.15 TF	124.42 TF
MP-Linpack Performance	2.38 TF	101.4 TF (2006) 102.2 TF (2007)
Total Memory	1.21 TB	39.19 TB
System Memory B/W	2.5 TB/s	78.12 TB/s
Disk Storage (Total/per Color)	12.5 TB/6.25 TB	340 TB/170 TB
Parallel File System B/W (Total/per Color)	2.0 GB/s / 1.0 GB/s	100 GB/s / 50 GB/s sustained disk transfer rate
External Network (Total/per Color)	0.4 GB/s / 0.2 GB/s	50 GB/s / 25 GB/s sustained network transfer rate to RoSE
Interconnect B/W		
MPI Latency	15 *u*s 1 hop, 20 *u*s max	~4.78 *u*s 1 hop, ~7.78 *u*s max
Bi-Directional Link B/W	800 MB/s	9.6 GB/s
Minimum Bi-Section B/W	51.2 GB/s	4.61 TB/s
Full System RAS		
RAS Network	10 Mb Ethernet	100 Mb and 1 Gb Ethernet
RAS Processors	1 for each 32 CPUs	1 for each 4 CPUs
Operating System		
Compute Nodes	Cougar	Catamount VN
Service and I/O Nodes	TOS (OSFI)	Linux
RAS Nodes	VX-Works	Linux
Red/Black Switch		
Switches	2/row	4/row

Table 1. System Parameters – ASCI Red vs. Red Storm.

A block diagram of Red Storm illustrating the three functional hardware partitions 1) Compute, 2) Service and I/O, and 3) RAS/System Management is reproduced in Figure 2.

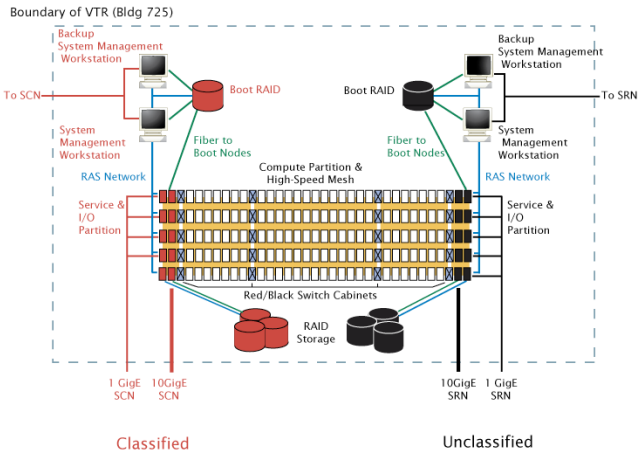

Figure 2. Red Storm Block Diagram – Three Functional Hardware Partitions 1) Compute, 2) Service and I/O, and 3) RAS/System Management.

2. Guidelines for Application and Problem Selection

The key criteria for application and problem selection are discussed at length in *The 7X Cookbook* [6]. One of the mandates of 7X testing is that applications and problem sets shall be "real". The 7X testing effort is attempting to characterize production job behavior by exercising applications-of-interest with production input files and algorithms. Each of the chosen applications is either a significant DOE production application or an idealized benchmark application that is based upon and closely resembles the behavior of a major DOE production application.

The same calculation will be run on ASCI Red and Red Storm. The primary metric is wall-clock time as measured by the elapsed time to execute the entire job script, including any pre and post processing [7]. The two calculations should give equivalent answers. The answers might not be numerically identical due to different sequences of operations, math libraries or numerical round-off, but the analysts should be comfortable that they are giving the "same" answer. For example, the answers may agree to only 6 significant digits.

Problems should be chosen to use as many ASCI Red resources (processor, memory) as possible in order to place reasonable stress on Red Storm. Problem sizes are deliberately chosen so that jobs run on ASCI Red should range from ~4-8 hours of wall-clock time [7]. Simplified geometries are preferred in order to simplify input file creation and to avoid meshing problems during benchmarking. All applications should use standard production capabilities including I/O, checkpoint/restart, and visualization files. When an application can be run using alternative algorithms, such as ALEGRA with and without contact, said application may have more than one benchmark problem in the suite.

We will test each application in "standard" mode (with the exception of PARTISN and SPPM where we will skip standard mode and proceed directly to stretch mode). We will also test each application in "stretch" and/or "maximum" mode. Figure 3 shows an overlay of node counts on ASCI Red vs. Red Storm to highlight the regimes under test.

1. Standard - the standard size should be easily run and accurately measured on both platforms. This standard will be used to calibrate the testing and to check for shifts in performance due to changes in the underlying system software. Standard refers to "Large – proc 0" on ASCI Red and "Small" on Red Storm.
2. Stretch - the stretch size will fully occupy the large configuration of ASCI Red. Stretch refers to "Large – proc 3" on ASCI Red and "Large (SN)/Small (VN)" on Red Storm. Problem sets will need to accommodate the reduced memory available in ASCI Red stretch mode.
3. Maximum - selected applications (CTH, ITS, PARTISN, SAGE, SALINAS, SPPM, UMT2K) may also be run in maximum size that requires an operational configuration of ASCI Red's entire compute node partition. Maximum refers to "Jumbo – proc 0 or Jumbo – proc 3" on ASCI Red and "Large (SN)/Small (VN) or Large" on Red Storm. Where maximum runs are not feasible on ASCI Red, the corresponding stretch sizes shall be considered sufficient.

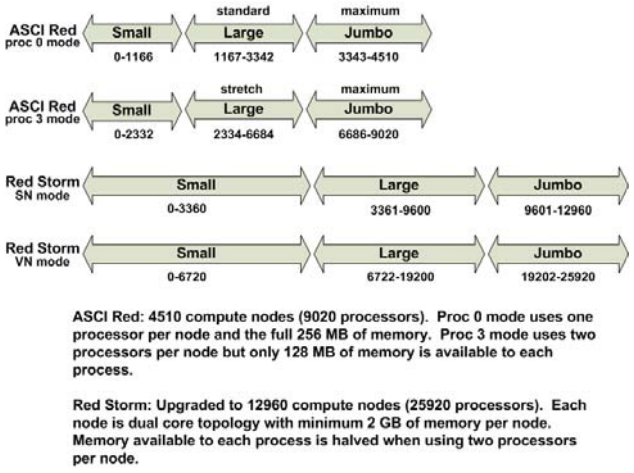

Figure 3. Standard, Stretch, and Maximum Modes on ASCI Red and Red Storm.

3. Guidelines for Application and Problem Runs

The key criteria for application and problem runs are discussed at length in *The 7X Cookbook* [6].

For the purposes of the 7X testing, the versions of the applications run on both platforms should be identical, subject only to changes required to make the applications run on both systems (ASCI Red and Red Storm). This requirement implies that the 7X benchmarking team needs to carefully manage the source code for each application. The SNL and Cray teams will both need access to the source code and input decks to build binaries and execute the runs. System and support libraries (e.g. glibc) required for the application to link and run may differ between the systems. Numeric and message passing libraries may differ, but the differences should not include substantive algorithm changes unless such changes were induced by the Red Storm architecture.

SNL will port each application to Red Storm and provide Cray with access to the source code, on-site at Sandia, for the purpose of 7X compilation. All access to source code is subject to export control restrictions. Cray must obtain licenses for any other use of the 7X application source code.

SNL personnel will run the tests on ASCI Red, with Cray personnel witnessing the process and validating the results. Cray personnel will run the tests on Red Storm, with SNL personnel witnessing the process and validating the results.

The Red Storm hardware configuration and software stack was designed to minimize any need for application source code modifications when porting any application from ASCI Red to Red Storm. Consequently, any modifications required in the source code itself are of interest to both the Sandia Red Storm design teams and to Cray. Any changes introduced into the 7X application code base that were required so that the application will correctly compile and run on Red Storm need to be shared among the Red Storm design teams, including Cray.

The Makefiles and scripts used to build the applications may require changes during the port to Red Storm. For example, the compiler options and switches used may vary. Similarly, job launch options may differ. Such changes are acceptable and need only be identified and tracked as support to the end-users of Red Storm. However, the final configuration switches used for the 7X benchmark runs must be properly documented and logged into the status database.

Each application on Red Storm should be validated against known "gold standard" results provided by SNL, LANL, or LLNL as appropriate. Discrepancies in the output must be validated with designated Labs points of contact. Similar validation runs should be carried out on ASCI Red for any applications that are not currently in production on ASCI Red such as PARTISN, SAGE, SPPM, and UMT2K. Validation should include checkpoint/restart and visualization outputs.

The output of the benchmarking runs should be checked for proper completion, proper creation of output files, and approximate size of files. Where possible, the same validation procedures as used by the functional testing team will be used to validate the 7X runs. Each test case (a specific benchmark and size) will be run 2-3 times on ASCI Red and Red Storm.

Speedup for each benchmark size will be calculated by dividing the average of the runs on ASCI Red by the average of the runs on Red Storm. Speedup for each application will be calculated by dividing the arithmetic average of the benchmark speedups on ASCI Red by the arithmetic average of the benchmark speedups on Red Storm. Overall speedup will be measured as specified in the contract [1]: "The speedup of each of the applications above will be measured as a number, nominally around 7, and these numbers will be linearly averaged with equal weights."

For a given benchmark and size (e.g. SPPM on 4500 processors), all runs must be made with the same compiled binary. Final 7X testing runs shall be allocated exclusive use of the platform in order to eliminate any contention for machine resources (i.e. only one 7X application running on the mesh at a time and no other users on the platform during 7X testing).

4. 7X Application Suite

Ten applications comprise the 7X test suite: ALEGRA Contact, ALEGRA NoContact, CTH, ITS, PARTISN, PRESTO, SAGE, SALINAS, SPPM, and UMT2K. The ten applications and the twenty-five test problems in the 7X suite are shown graphically in Figure 4. Table 2 summarizes the test problem sizes for each application. Following Table 2, we describe each of these applications and test problems.

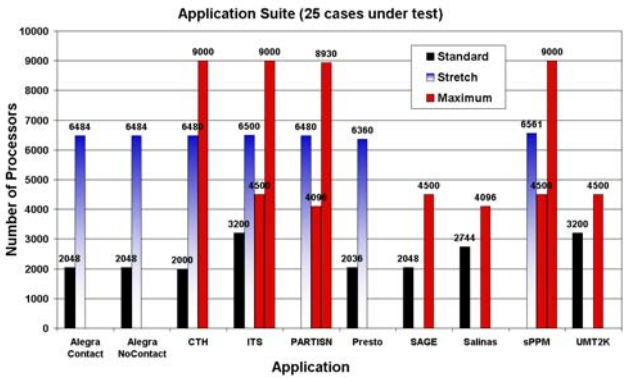

Figure 4. 7X Application Suite

App	Run	Size	ASCI Red	Red Storm
ALEGRA Contact	Standard	2048	Large – proc 0	Small
	Stretch	6484	Large – proc 3	Large (SN) Small (VN)
ALEGRA NoContact	Standard	2048	Large – proc 0	Small
	Stretch	6484	Large – proc 3	Large (SN) Small (VN)
CTH	Standard	2000	Large – proc 0	Small
	Stretch	6480	Large – proc 3	Large (SN) Small (VN)
	Maximum	9000	Jumbo – proc 3	Large
ITS	Standard	3200	Large – proc 0	Small
	Maximum	4500	Jumbo – proc 0	Large (SN) Small (VN)
	Stretch	6500	Large – proc 3	Large (SN) Small (VN)
	Maximum	9000	Jumbo – proc 3	Large
PARTISN	Maximum	4096	Jumbo – proc 0	Large (SN) Small (VN)
	Stretch	6480	Large – proc 3	Large (SN) Small (VN)
	Maximum	8930	Jumbo – proc 3	Large
PRESTO	Standard	2036	Large – proc 0	Small
	Stretch	6360	Large – proc 3	Large (SN) Small (VN)
SAGE	Standard	2048	Large – proc 0	Small
	Maximum	4500	Jumbo – proc 0	Large (SN) Small (VN)
SALINAS	Standard	2744	Large – proc 0	Small
	Maximum	4096	Jumbo – proc 0	Large (SN) Small (VN)
SPPM	Maximum	4500	Jumbo – proc 0	Large (SN) Small (VN)
	Stretch	6561	Large – proc 3	Large (SN) Small (VN)
	Maximum	9000	Jumbo – proc 3	Large
UMT2K	Standard	3200	Large – proc 0	Small
	Maximum	4500	Jumbo – proc 0	Large (SN) Small (VN)

Table 2. Test Problem Sizes.

ALEGRA (QSEM with Contact)

ALEGRA is used to simulate the dynamic material response of complex configurations [8]. It solves coupled physics problems in 2D or 3D using Lagrangian, Eulerian, and/or ALE coordinates. The code runs efficiently on massively parallel computers and contains a large variety of physics options including hydrodynamics, magnetohydrodynamics with external circuit coupling, radiation transport, thermal conduction, and dual ion and electron temperatures. The ALEGRA Contact problem is a Quasistatic electromechanics (QSEM) problem in which a curved impactor depoles a potted active ceramic element.

ALEGRA (QSEM without Contact)

The ALEGRA No Contact problem is a QSEM problem identical to the contact problem except the boundary condition is a prescribed displacement rather than an impactor. This eliminates the need for contact.

CTH

CTH is a multimaterial, large-deformation, strong shock wave, solid mechanics code developed at Sandia National Laboratories [9]. CTH has models for multiphase, elastic viscoplastic, porous and explosive materials. Three-dimensional rectangular meshes; two-dimensional rectangular and cylindrical meshes; and one-dimensional rectilinear, cylindrical and spherical meshes are available. CTH uses second order accurate numerical methods to reduce dispersion and dissipation and to produce accurate, efficient results. CTH is used for studying armor/antiarmor interactions, warhead design, high explosive initiation physics, and weapons safety issues. The test problem is the shock physics in 3D of a large conical shaped charge.

ITS

The Integrated Tiger Series (ITS) code permits Monte Carlo solution of linear time-independent coupled electron/photon transport radiation transport problems, with or without the presence of macroscopic electric and magnetic fields of arbitrary spatial dependence [10,11]. Physical rigor is provided by employing accurate cross sections, sampling distributions, and physical models for describing the production and transport of the electron/photon cascade from 1.0 GeV down to 1.0 keV. Mulitgroup ITS Version 5.0 (April 1, 2002) contains (1) improvements to the ITS 3.0 continuous-energy codes, (2) multigroup codes with adjoint transport capabilities, (3) parallel implementations of all ITS codes, (4) a general purpose geometry engine for linking with CAD or other geometry formats, and (5) the Cholla facet geometry library. The 7X runs will perform the Starsat MITS test with CAD flow and geometry and ACIS simulation mode.

PARTISN

The Parallel Time-dependent S_N (PARTISN) code package is designed to solve the time-independent or dependent multigroup discrete ordinates form of the Boltzmann transport equation in several different geometries [12]. PARTISN provides neutron transport solutions on orthogonal meshes with adaptive mesh refinement in 1D, 2D or 3D. Much effort has been devoted to making PARTISN efficient on massively parallel computers. The package can be coupled to nonlinear multiphysics codes that run for weeks on thousands of processors to finish one simulation. The test problem is "Sntiming", in which flux and eigenvalue convergence are monitored by PARTISN.

PRESTO

PRESTO is a Lagrangian, three-dimensional explicit, transient dynamics code for the analysis of solids subjected to large, suddenly applied loads [13]. PRESTO is designed for problems with large deformations, nonlinear material behavior, and contact. There is a versatile element library incorporating both continuum and structural elements.

The contact algorithm is supplied by ACME. The contact algorithm detects contacts that occur between elements in the deforming mesh and prevents those elements from interpenetrating each other. This is done on a decomposition of just the surface elements of the mesh. The contact algorithm is communication intensive and can change as the problem progresses.

The brick walls problem consists of a number of rectangular bricks, each meshed using 3x3x6 elements. The bricks are stacked in an alternating fashion in a plane to produce a wall which is three elements thick. Four of these walls are lined up in the thin direction. The walls are then given a sudden pressure loading such that they compress against each other. Since all of the bricks are meshed independently, they interact with each other through contact on their outer surfaces. Each brick is located on one processor so the only communication for the finite element portion of the code is for the determination of the length of the next time step. As the problem grows with the number of processors, the contact problem also grows. Although there is no analytic solution for this problem, it provides a large amount of contact with respect to the number of elements. There are 1.67 times as many faces to be considered in contact as there are elements, so the cost of contact dominates the computation. This serves as an excellent test to exercise large-scale global contact and to demonstrate the parallel scaling of the algorithm.

SAGE

SAIC's Adaptive Grid Eulerian (SAGE) hydrocode is a multidimensional, multimaterial hydrodynamics code with adaptive mesh refinement that uses second-order accurate numerical methods [14]. SAGE represents a large class of production computing applications at Los Alamos National Laboratory (LANL). It is a large-scale parallel code written in Fortran 90 and uses MPI for interprocessor communications. It routinely runs on thousands of processors for months at a time on capability computing systems in the DOE complex. The test problem is an asteroids simulation of 45 degree, 3D, granite asteroid impact into a stratified medium of water, calcite, granite crust, and mantle.

SALINAS

SALINAS is a massively parallel implicit structural mechanics/dynamics code aimed at providing a scalable computational workhorse for extremely complex finite element (FE) stress, vibration, and transient dynamics models with tens or hundreds of millions of degrees of freedom (dofs) [15]. The SALINAS software predicts vibrational loads for components within larger systems, design optimization, frequency response information for guidance and space systems, and modal data necessary for active vibration control. SALINAS is used to predict mechanical response in normal and hostile STS1 environments for RB2 systems and missiles. The software is a tool for understanding and predicting structural response. It is used for both production type calculations and for research and development, especially with respect to development of joint and interface models.

The test problem is a transient dynamics problem based on one unit cube model. The cube will first be decomposed into subcubes using an nsub x nsub x nsub partition. Then each cube will be meshed using nelem x nelem x nelem hex8 elements. The x=0 face will be clamped, and x=1 face will have an x-directional load. The cube starts at the origin (0,0,0) and extends to (1,1,1). The faces are parallel to the three coordinate directions (x,y,z). We use "pmesh" to create the models on the fly.

SPPM

PPM (Piecewise Parabolic Method) is a 3-D hydrodynamics code used to model a wide range of shock physics problems [16]. It performs PPM hydrodynamics in Lagrangian style using a Riemann solver. A simple gamma-law equation of state is used, and an initially uniform grid with either periodic or continuation boundary conditions is assumed. The SPPM benchmark solves a 3D gas dynamics problem on a uniform Cartesian mesh, using a simplified version of PPM, hence the "s" for simplified [17, 18]. The code is written to simultaneously exploit explicit threads for multiprocessing shared memory parallelism and domain decomposition with message passing for distributed parallelism. It represents the current state of ongoing research which has demonstrated good processor performance, excellent multi-threaded efficiency, and excellent message passing parallel speedups all at the same time. The SPPM program was written in Fortran77 with all system dependent calls taking place through C. It uses a small number of MPI routines for communication between nodes.

The hydrodynamics algorithm involves a split scheme of X, Y, and Z Lagrangian and remap steps which are computed as three separate passes or sweeps through the mesh per timestep, each time sweeping in the appropriate direction with the appropriate operator. Each such sweep through the mesh requires approximately 680 FLOPs to update all of the state variables for each real mesh cell. Message passing is used to update ghost cells with data from neighboring domains three times per timestep and occurs just before each of the X, Y, and Z sweeps. Multiple threads are used to manipulate data and update pencils of cells in parallel.

UMT2K

The UMT benchmark is a 3D, deterministic, multigroup, photon transport code for unstructured meshes [19]. UMT 1.2, referred to as UMT2K for clarity, performs exactly the same physics as previous versions of UMT (i.e., UMT 1.1, referred to as UMT98) but now includes additional features that are commonly found in large Lawrence Livermore National Laboratory (LLNL) parallel applications. These features include mixed MPI and OMP support for large-scale parallelism, an OMP-based C computation kernel called from an MPI-based Fortran90 driver, a new mechanism for synthetically generating very large distributed meshes, a parallel checkpoint/restart mechanism, and graphics output files. The transport code solves the first-order form of the steady-state Boltzmann transport equation. The equation's energy dependence is modeled using multiple photon energy groups. The angular dependence is modeled using a collocation of discrete directions, or "ordinates." The spatial variable is modeled with an "upstream corner balance" finite volume differencing technique. The solution proceeds by tracking through the mesh in the direction of each ordinate. For each ordinate direction all energy groups are transported, accumulating the desired solution on each zone in the mesh. Hence, memory access patterns may vary substantially for each ordinate on a given mesh, and the entire mesh is "swept" multiple times. Note, however, that having the energy group loop on the inside significantly improves cache reuse, because all of the geometrical information related to sweeping an ordinate direction is the same for each energy group.

The code works on unstructured meshes, which it generates at run-time using a two-dimensional unstructured mesh (read in) and extruding it in the third dimension a user-specified amount. This allows the generation of a wide variety of input problem sizes and facilitates "constant work" scaling studies. The MPI-based parallelism in the Fortran portion uses mesh decomposition to distribute the mesh across the specified MPI tasks. The OMP-based parallelism in the C kernel then divides the ordinates among the OMP threads. This C kernel's computation time typically completely dominates the execution time of the benchmark.

5. Activities and Roles

Completion of the 7X benchmarking task requires cooperation among SNL, LLNL, LANL, and Cray.

SNL Responsibilities: Code Teams

The code teams will designate code releases to be used and identify test problems for each application, along with problem sizes and ASCI Red running modes. Problem sizes should include those needed for testing and scalability studies on Red Storm. Input files must be developed for each problem and size and the code teams must also work with the 7X systems team to port applications to Red Storm. Code teams will provide the 7X team with instructions on how to compile each application and how to assemble the input files if the actual files are not provided directly. Code teams will also provide either full source code snapshots or access to a source code repository in which the code to be used is appropriately tagged for later retrieval.

SNL Responsibilities: 7X Team

The 7X team will organize and manage the performance testing effort. A data repository will be created for all testing information. The team will develop, test, and document the benchmarking procedures and work with the code teams to develop input files, problem sizes, and running modes. The 7X team will also provide assistance to SNL code teams in porting the 7X applications to Red Storm and work with LANL and LLNL points of contact to port non-SNL applications to Red Storm. The 7X team will execute all official baseline runs on ASCI Red, run and validate each problem set at its standard size on Red Storm, and partner with Cray engineers to complete all necessary official runs and validations on Red Storm.

LLNL and LANL Responsibilities

LLNL and LANL teams will designate the code releases to be used and assist SNL in identifying test problems, problem sizes, and ASCI Red running modes for their candidate applications. The LLNL and LANL teams will also assist SNL in developing input files for each test problem/size and provide the 7X team with instructions on how to port/compile each application on Red Storm.

Cray Responsibilities

Cray will work with SNL to ensure that all applications compile and run on Red Storm for all problems and sizes. Cray engineers will partner with the 7X team to compile binaries for Red Storm and execute the official runs on Red Storm. Official runs will be executed with designated SNL personnel as witnesses.

6. Project and Data Management

Many data items need to be tracked during the 7X benchmarking. Two data repositories will be used: 1) relational database for projects, status, and result management 2) file repository for document, build, input, and output file management.

Project Database

A relational database has been developed that will store all relevant status and result information. The information in this database will be used to track both status and to extract final results. For example, when a data run occurs, the database will be used both to set up the run and to log relevant information about the results. The database currently uses a PostgresSQL [20] server

located on a development workstation. Interfaces to the relational database were developed in Perl (for command-line use). Web-based status reporting can be added if required. Preliminary status reporting interfaces have been developed that allow a user to extract data from the project database and format it as graphs, charts, or tables using Unix-based tools.

Test Management

A simple XML-based scripting language has been developed that will allow the 7X testers to specify, run, and log the results of each benchmark. The implementation of this language is called *rst*. Using *rst*, one can

- Specify a test to be run.
- Compile application binaries and capture extended output for later inclusion into the database.
- Run a test in either batch or interactive mode and capture run-time information for later inclusion in the database.
- Write results data back into the database.

The tool has been designed to minimize impact on the HPC engines where it runs. In particular, it can run without requiring access to a database server when compiling applications or when running the actual tests.

File Management

A Sourceforge [21] project repository has been created to support all of the file management necessary for running and reviewing the benchmarks. Sourceforge is a collaborative software development tool that supports web-based interactions, collaborative communications and file sharing. Underlying the web interface is a source code management system based on CVS [22]. Access to files can be controlled and limited to certain users via role-based access controls. The input files, build scripts, and run scripts for each application and benchmark will reside on the Sourceforge site. When a benchmark needs to be executed, the files can be retrieved, the application built (if necessary) and the test run. The resulting test output files will be pushed back into the Sourceforge repository, while the test results will be logged into the relational database.

7. Results – How Much Faster is Red Storm on the 7X Applications?

An effort was made to set up the test problems so that each would require ~4-8 hours wall-clock execution time on ASCI Red and, therefore, about 1 hr. on Red Storm. This goal was largely met, as seen in Figure 5, although it was not possible to scale the PARTISN test problem to that level. The SALINAS test problems ran for slightly more than an hour on Red Storm, as the SALINAS speedups on Red Storm were only a factor of 6 to 8 over ASCI Red, lower than for all of the other applications.

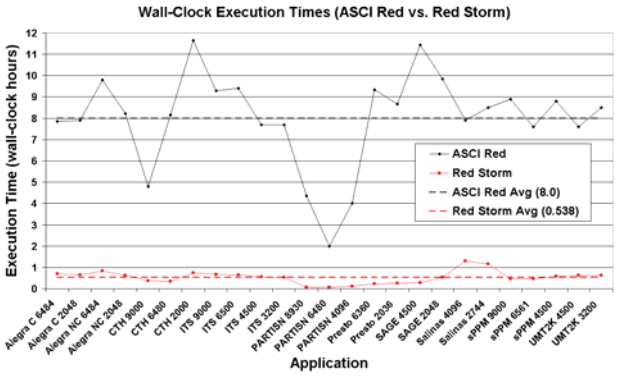

Figure 5. Application Execution Times on ASCI Red and Red Storm.

We have not thoroughly investigated the cause for the lower than expected speedup for SALINAS, however we observed that the total Finite Element Tearing and Interconnecting (FETI) solution time to wall-clock time was much closer to one in the ASCI Red runs than for the Red Storm runs. For example, a typical 2744 proc 0 mode run on ASCI Red, took 30419 wall-clock seconds with 29025 seconds spent in the FETI solve step which equates with 95% of the wall-clock time spent in the FETI solve. By comparison, a typical 2744 VN mode run on Red Storm took 4733 wall-clock seconds to complete with 3400 seconds spent in the FETI solve step. The FETI solve step occupied only 72% of the total wall-clock time. We will need to profile SALINAS on Red Storm to determine what is occurring during the "unproductive" 28% (1333 seconds).

Figure 6 shows the speedups achieved on Red Storm relative to ASCI Red. An average speedup of 20X is observed across the test suite, far above the hoped-for seven-fold improvement. Three caveats are in order: (1) the average speedup is unduly influenced by the extremely large speedup (65X) measured for PARTISN (and, to a lesser extent, by SAGE) on Red Storm. We can speculate that ASCI Red may have been in a degraded state when the PARTISN runs were made, but this cannot be proven since the system is no longer available; (2) processor speeds were upgraded by 20% on Red Storm before these results were gathered, so the real target should now be 8.4X, not 7X; (3) Although our intent was to perform all testing on ASCI Red and Red Storm in "exclusive" mode (i.e. only one 7X application running on the mesh at a time and no other users on the platform during 7X testing), the Red Storm testing was almost never "exclusive". We often ran several 7X applications concurrently and other users were allowed to run jobs on Red Storm during the 7X testing due to program milestone needs. Our testing on ASCI Red was always "exclusive". This may indeed have disadvantaged Red

Storm performance results; however, we cannot determine the extent of the effect.

If we discard the maximum and minimum speedup values (65X for PARTISN 8930 processors and 6X for SALINAS 4096 processors), we obtain a 19X average speedup. If we discard the highest two values (65X for PARTISN 8930 processors and 42X for SAGE 4500 processors), we still obtain an 18X average speedup, well above the 7X target.

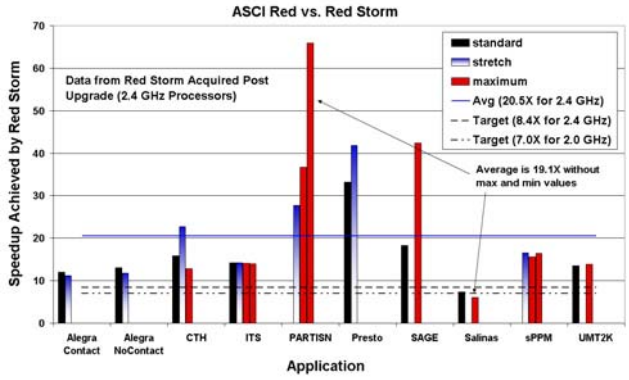

Figure 6. Application Speedup - ASCI Red vs. Red Storm

8. Results – Single-Core vs. Dual-Core Comparison on Red Storm

The recent upgrade of Red Storm to dual-core sockets has provided the option of specifying either one or two cores per socket when launching an application. As noted above, the 7X tests can be performed on Red Storm in either SN or VN mode: (1) the SN option, which is the default, ignores the second core and makes all user memory on the node available to the application; (2) the VN option, which treats each core as a separate compute node and makes only half the user memory on the node available to each core. If applications can run efficiently in VN mode on Red Storm, this frees up sockets for other applications.

In Figure 7, we compare the Red Storm results in terms of execution time for SN and VN runs of the test problems, as well as a few pre-upgrade runs (2.0 Ghz single-core Opteron processors). Run times in yellow were performed pre-upgrade (2.0 Ghz single-core). Run times in blue were obtained on 2.4 Ghz dual-core, but the second core of each node was left idle. Run times in red used both cores on the node and only required half as many compute nodes as the run times in blue, freeing nodes for other production work.

Most of the applications are demonstrating a small-to-modest performance hit (5-30%) for using the second core in VN mode. The average efficiency drop was 17% for VN mode vs. SN mode (post-upgrade). PARTISN is again an outlier with the largest dual-core performance penalty in the test suite. Interestingly, the 6484 processor ALEGRA No Contact test shows a very slight performance acceleration in VN mode relative to the same number of cores in SN mode.

Pre-upgrade runs were available for three applications in the 7X suite. These runs were performed using the 2.0 Ghz single-core Opteron processors that were in place prior to the Red Storm system upgrade. ITS shows a speedup commensurate with the 20% increase in processor speed due to the upgrade, but little benefit is seen for UMT2K and SAGE for upgrading to the 2.4 Ghz dual-core Opteron processors.

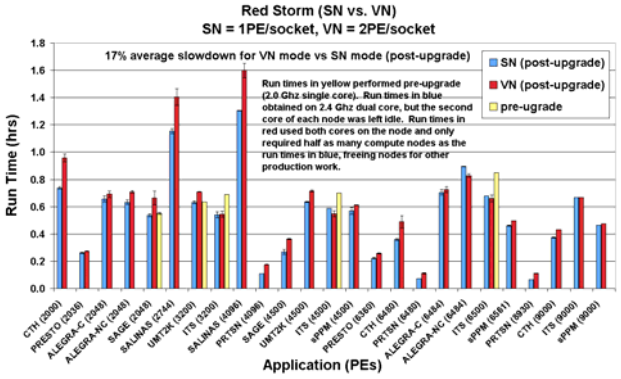

Figure 7. Comparison of SN and VN Results on Red Storm (includes pre-upgrade 2.0 Ghz single-core results for three applications).

9. Summary and Lessons Learned

In preparation for the testing and acceptance of the Red Storm system, a suite of ten applications/benchmarks were developed to assess whether major applications would realize at least a seven-fold performance increase on the new system relative to its predecessor. This methodology has subsequently proven quite valuable in addressing diverse performance issues: e.g. the benefits of processor and memory upgrades, particularly the benefits of dual-core processors. The impending 2008 upgrade of Red Storm to quad-core Opteron processors will provide another opportunity to demonstrate the usefulness of the 7X suite to track performance across single, dual, and quad core processors.

Red Storm has achieved its requirement of 7X performance over ASCI Red, posting an average speed-up of 20X. We find that although most of the individual applications show at least a 12-fold to 15-fold performance improvement over the ASCI Red system, there are interesting outliers: PARTISN shows run time speed-ups of up to 65X while SALINAS manages only a 6X-8X performance increase. The results validate Red Storm as a capability platform for major scientific and engineering codes on 2K-10K processors.

We also compared single-core (SN) and dual-core (VN) runs on Red Storm to investigate the efficiency that

users might experience when utilizing both cores on the node. Dual-core performed well on the 7X applications, often completing in nominally the same time as single-core runs. The average efficiency drop was 17% for VN mode vs. SN mode with most of the applications demonstrating a small-to-modest performance hit (5-30%) for using the second core in VN mode. The results validate the efficacy of the dual-core upgrade, as most of these applications make efficient use of the second core. Applications that can run efficiently in VN mode on Red Storm have the potential to free up sockets for other applications.

The availability and applicability of this test suite to answer design questions and evaluate upgrade options, such as the dual-core upgrade, further validates the need for evaluation of capability-class, massively parallel systems with real applications.

Many of the 7X applications are routinely used to benchmark and evaluate other new systems, e.g. highly parallel cluster systems that are acquired to serve as capacity computing systems. However, there are some serious limitations to this methodology. Several of the applications discussed here require major porting efforts whenever a new system is to be tested. This is particularly true of the Sierra framework-based applications, such as CALORE and PRESTO, as well as other large, modern, object-oriented applications such as ALEGRA. Some applications can require a week or two to be built for a new system, even if no portability issues are encountered.

When the comparison testing is spread out over a long period of time, it will undoubtedly be necessary to adjust to changes in the computing environment. Upgrades to the operating system, compilers, file systems, etc. can prove quite challenging. The application code may not compile the first time out of the chute with a new compiler. Application codes also "evolve", which is also quite challenging when striving for some level of test consistency over time.

When standing up any new parallel computing system, an argument could be made for using an appropriate subset of the large complex application codes in addition to simpler application/benchmark codes for quick portability. We see a need for compact applications based on "real" applications, and there are recent research and development efforts to create new compact applications, so that testing and evaluation of new systems and potential procurements can be done in a timely manner [23].

References

1. *Contract for Red Storm*, Internal Document 32124, Sandia National Laboratories, Albuquerque, NM, Sept. 23, 2002.
2. Paul Hommert, Dona Crawford, Rena A. Haynes, George W. Davidson, William J. Camp, Doug Brown, Arthur L. Hale, Juan C. Meza, *Accelerated Strategic Computing Initiative*, SAND96-2659C, Sandia National Laboratories, Albuquerque, NM, 1996.
3. James L. Tomkins, *The ASCI Red TFLOPS Supercomputer*, SAND96-2659C, pg 17-18, Sandia National Laboratories, Albuquerque, NM, 1996. An excerpt from this report can be found at http://www.sandia.gov/ASCI/Red/RedFacts.htm
4. Timothy G. Mattson and Greg Henry, *The ASCI Option Red Supercomputer*, Intel Corporation, 1997.
5. R. A. Ballance, Red Storm Web Site, Sandia National Laboratories, Albuquerque, NM, 2008. http://redstormweb.sandia.gov/RedStorm
6. R. A. Ballance, *The 7X Cookbook*, Version 1.5.1, Internal Document, Sandia National Laboratories, Albuquerque, NM, Jan. 10, 2005.
7. *Preliminary Cray Test Document for Red Storm*, Draft Cray Internal Document, Dec. 11, 2003.
8. T. A. Haill, et al., *Multi-dimensional z-pinch calculations with Alegra*, Pulsed Power Plasma Science, IEEE, Las Vegas, NV, June 2001.
9. E. S. Hertel, et al., *CTH: A Software Family for Multidimensional Shock Physics Analysis*, Proceedings 19th International Symposium on Shock Waves, **1**, 274ff, Universite de Provence, France, 1993.
10. Brian C. Franke, Ronald P. Kensek and Thomas W. Laud, *ITS Version 5.0: The Integrated TIGER Series of Coupled Electron/Photon Monte Carlo Transport Codes with CAD Geometry*, SAND2004-5172, Sandia National Laboratories, Albuquerque, NM, 2004.
11. M. Rajan, et al., *Performance Analysis, Modeling and Enhancement of Sandia's Integrated TIGER Series (ITS) Coupled Electron/Photon Monte Carlo Transport Code*, Proceedings of LACSI Symposium, Santa Fe, NM, Oct. 2005.
12. *Transport Methods CCS-4*, Los Alamos National Laboratory, Los Alamos, NM.
13. J. Richard Koteras and Arne S. Gullerud, *Presto User's Guide Version 1.05*, SAND2003-1089, Sandia National Laboratories, Albuquerque, NM, 2003.
14. D. J. Kerbyson, et al., *Predictive Performance and Scalability Modeling of a Large Scale Application*, Proceedings of the ACM-IEEE International Conference HPC and Networking (SC01), Nov. 2001.
15. Garth Reese, *Salinas – Strategic Vision*, Internal Document, Sandia National Laboratories, Albuquerque, NM, April 2, 2008.
16. P. Colella and P. R. Woodward, *The Piecewise Parabolic Method (PPM) for Gas-Dynamical Simulations*, J. Comput. Phys., 54, pg. 174-201, 1984.

17. J. Owens, *The ASCI sPPM Benchmark Code*, Lawrence Livermore National Laboratory, Livermore, CA, 1996.
18. T. Spelce, *Early Performance Results from the LLNL/NNSA Purple Computer*, UCRL-PRES-22309, SCICOMP 12, Boulder, CO, July 17-21, 2006.
19. B. Chan, *The UMT Benchmark Code*, Lawrence Livermore National Laboratory, Livermore, CA, 2002,
20. Bruce Momjian, *PostgresSQL: Introduction and Concepts*, Addison-Wesley, Reading, MA, USA, 2001.
21. VA Software, *Sourceforge 3.1 User Guide*, 2002.
22. Per Cederqvist, *Version Management with CVS*, Version 1.12.5.
23. D. A. Bader et al., *Designing Scalable Synthetic Compact Applications for Benchmarking High Productivity Computing Systems*, Cyberinfrastructure Technology Watch, **2** (4B), 1-10, Nov. 2006.

Acknowledgments

The authors thank Courtenay Vaughan, Bob Benner, John Van Dyke, Sue Goudy, Mahesh Rajan, and Hal Meyer for their assistance with compiling, configuring, and troubleshooting on ASCI Red and Red Storm. Many thanks also to the ASCI Red (Frank Jaramillo, Paul Sanchez, Mike Martinez, Sean Taylor) and Red Storm system administrators and support staff for their assistance. Thanks also to Mark Hamilton for assistance in setting up the Sourceforge repository.

The authors thank Cray engineers Paul Burkhardt, Doug Enright, and Ron Pfaff for their assistance with compiling and optimizing the codes for Red Storm runs.

Sue Goudy, Sue Kelly, Mike McGlaun, Jim Tomkins, and Courtenay Vaughan have all provided help, suggestions, and guidance as the predecessor [6] to this report was assembled. However, the authors are solely responsible for any errors or omissions.

We also thank the application code developers for their assistance: Brian Franke (ITS), Garth Reese (SALINAS), Riley Wilson (SALINAS), Galen Gizler (SAGE), John Daly (SAGE), Kevin Brown (PRESTO), Arne Gullerud (PRESTO), Allen Robinson (ALEGRA), Rich Drake (ALEGRA), and Josh Robbins (ALEGRA).

About the Authors

Robert A. Ballance is a Principal Member of Technical Staff at Sandia National Laboratories. Since joining Sandia in November, 2003, he has been deeply involved in the preparations for the delivery, acceptance testing, and production deployment of Red Storm, Sandia's newest capability computing platform. He is currently the System Manager for Red Storm. Prior to joining Sandia, he honed his operations skills while serving as the Manager of Systems and Systems Research at the Center for High-Performance Computing at the University of New Mexico. Dr. Ballance received his Ph.D. in Computer Science from the University of California, Berkeley (1989); a Masters in Computer Science from the University of Michigan, Ann Arbor (1978); and a Bachelors of Science in Mathematics from the University of North Carolina, Chapel Hill (1976). Bob can be reached at Sandia National Laboratories, Albuquerque, NM, USA, E-mail: raballa@sandia.gov.

Joel 0. Stevenson has been employed a total of 15 years at Sandia National Laboratories from 1986-1997 and 2005-present. Joel was Co-Founder and Director of Software Engineering for Peak Sensor Systems from 1997-2005. Joel and partners successfully developed, patented, and commercialized a portfolio of semiconductor process technologies (20 issued patents) and gained broad-based interest and acceptance in the technologies across a large number of semiconductor device manufacturers. Joel received his Masters in Computer Science from the University of New Mexico (1992) and a Bachelors of Science in Chemistry from Eastern New Mexico University (1984). Joel is committed to providing high quality, high value service to a broad range of customers and is a relative newcomer to High Performance Computing. Joel can be reached at Sandia National Laboratories, Albuquerque, NM, USA, E-Mail: josteve@sandia.gov.