Diagnostic Capabilities of the Red Storm Compliance Test Suite

Mike Davis, *Cray Inc.*

ABSTRACT: *The Red Storm Compliance Test Suite, originally developed to demonstrate the system's compliance with the requirements of the Red Storm Statement of Work (SOW), has also proven to be useful for diagnosing system hardware and software problems and for exposing regressions in performance following system hardware and software upgrades. This paper will describe the tests that comprise the suite and their specific diagnostic capabilities.*

KEYWORDS: XT3, Sandia, Red Storm, compliance, diagnostics

1. Overview

In mid-2002, Cray Inc. and Sandia National Laboratories entered into an agreement to develop a massively parallel computer system. The development program and the developed system were both named *Red Storm*. In late 2004, Cray introduced the Cray XT3™ computer system, a productized version of the Red Storm design. A current description of the Red Storm system configuration is described in the proceedings of a recent workshop held at Los Alamos National Laboratory.

A central guide of the Red Storm development program is the Red Storm Statement of Work (SOW). The SOW describes the capabilities of the system and, in many respects, how it must demonstrate them. It is comprised of 96 requirements in seven major categories. For over 20 of these requirements, compliance must be demonstrated by the execution of a software test. These software tests comprise the Red Storm Compliance Test Suite.

2. The Compliance Test Suite (CTS)

Many of the tests in the test suite are adapted from commonly-used tests available in the public domain and are cited below. Other tests were written specifically to address the requirements of the SOW. In the course of developing, validating, and demonstrating these tests, their value as diagnostics became apparent.

Every test reports summary performance metrics at the end of a run. One of these metrics is identified as the *key metric*. In addition, many tests compute an array of *component-level metrics* that apply on a per-node (or other component) basis.

Every test uses a performance target. In most cases, the target is accepted at runtime as a user parameter; otherwise, the target is predefined within the test. The units of the target vary for each test. Whenever prefixes are used to indicate a multiple or fraction of a unit (for example, *mega-*, *giga-*, *micro-*, and so on), a base-ten multiple or fraction is assumed.

The target values specified for diagnostic purposes are occasionally different from those used to demonstrate compliance with SOW requirements. The values discussed in this paper are those typically used in diagnostic situations.

Tests that derive component-level metrics also accept a deviation tolerance as a user parameter. This parameter is dimensionless, and usually represents a decimal fraction of a *nominal reference value*. The nominal reference value is the "better" of the performance target or the key metric, scaled down to the level of the component.

The deviation tolerances specified for diagnostic purposes are often different from those used to demonstrate compliance with SOW requirements. The values discussed in this paper are those typically used in diagnostic situations.

Every test performs a *key assessment*, or a comparison of the key metric and the performance target. In addition, for those tests that compute component-level metrics, a *deviation assessment* compares deviations from the nominal reference value against the deviation tolerance. A run is deemed noncompliant if either of these comparisons is unfavorable.

If the key assessment fails, the test reports the failure and, if component-level metrics exist, a component report is generated for each of the top ten most noncompliant

components. If the deviation assessment fails, a component report is generated for each noncompliant component. A component report includes the location of the component in the system, the appropriate metric value, and the nominal reference value.

The tests can be grouped into three types, based on scale, as follows:

1. The *scaled single-component test with summary metrics (SC)* can be run on a single component and produce meaningful result metrics. Indeed, some tests within this class were adapted from code that was originally written for a non-parallel system. When run on multiple components in parallel, the amount of work done by each component is fixed and equal, and the key metric is usually the performance rating of the least-performing component. The test also computes component-level metrics for deviation assessment. In this type of test there is no communication between components. Communication takes place only during summary metric computation.

2. The *scaled component-group with summary metrics (CG)* can be run on a small group of related components and will produce meaningful result metrics. In most cases, the relationship between the components in a component group is topological (for example, nodes that share a common network link, nodes in the same mesh plane, and so on). In some cases, the relationship is conformal (for example, service nodes that have Ethernet risers, service nodes that service a particular file system, and so on). Tests make use of controlling driver scripts that derive group memberships based on the contents of fairly static databases (for example, `/etc/hosts` files, hard-coded topology tables in header files, and so on). Utility scripts are maintained as part of the test framework to generate and customize these databases for a given system configuration.

   When a test of this type is run on multiple component groups in parallel, the amount of work done by each group is fixed and equal, and the key metric is, usually, the performance of the least-performing group. In some cases, an additional metric is reported – the performance of the groups working in concert. Such performance is based on an elapsed time measurement of last-out less first-in (LO –FI), and is compared against a global target. An unfavorable comparison is reported as a non-compliance. The scaling of such a test is constrained by these additional requirements, and is called *LOFI scaling*. In many cases, the test also computes group-level metrics for deviation assessment. Such metrics are usually collected and stored by a single component within each group, identified as the *group leader*. Communication in the test proper is usually within groups, and it is usually the performance of this communication that is measured and tested for compliance. In order to maximize the fairness and accuracy of these measurements, the test is configured to minimize the potential for "cross-talk" among groups. Ancillary communication, global or among group leaders, occurs during the setup and wrap-up phases of the test to establish groupings and to produce the summary metrics.

3. The *single metric (SM)* test runs on the entire complement of available components in parallel, and produces a single result metric. Component-level metrics are either not derivable from user level or not of particular interest.

These tests can also be grouped according to the part(s) of the system that they cover. This coverage-style grouping is often used to set the order in which to run the tests during a test session, and is used here to describe the individual tests.

2.1 Compute Node Hardware Tests

These tests are scaled single-component (that is, compute node) tests with summary metrics.

Opteron ™ Processor (Core) Inventory, Signature, Speed (104)

There are two key metrics in this test. The first is the processor signature (model, family, and stepping) as obtained by the CPUID instruction: the target is the signature value defined in a hard-coded data structure within the program, and the deviation tolerance is zero. The second is the processor speed (gigahertz): the target is 2.4, and the deviation tolerance, in clocks per million, is 100.

Memory Capacity and Accessibility (105)

The key metric in this test is the amount of user-accessible memory (gigabytes) on the node: the target is 1.9, 3.0, or 4.0, and the deviation tolerance is 0.005. (Currently the test must be run on separate sections of the machine in series. This is because of memory configuration differences among the compute nodes.)

Memory Latency (305)

The key metric in this test is the latency (nanoseconds) to access a single word in memory, using an access pattern that bypasses both L1 and L2 cache: the target is 80, and the deviation tolerance is 0.005. The test is based on work done by Iyer et al.

Memory Bandwidth (307)

The key metric in this test is the bandwidth (gigabytes/second) between processor and memory on the STREAM triad kernel: the target is 4.0 or 4.2, and the deviation tolerance is 0.005. (Currently the test must be run on separate sections of the machine in series. This is because of memory configuration differences among the compute nodes.)

2.2 Compute Partition Interconnect Network Hardware Tests

These are scaled component-group tests with summary metrics. The component group in question is a collection of compute nodes.

MPI Latency (204)

For this test, a component group (see Figure 2.2.1) consists of a pair of compute nodes (in blue) separated by a single network link. The key metric is one-half of the maximum time (in microseconds) to perform a ping-pong communication of a zero-byte message across the link between the nodes using MPI. The number of samples from which the maximum ping-pong is taken must equal or exceed 10,000. The target time is 11.5, and the deviation tolerance is 0.01.

The scaling of this test is constrained to comply with detailed requirements and to ensure that execution in parallel does not unfairly perturb the timings. To comply with requirements, ancillary ping-pong traffic (also called *other work*) is executed concurrent with, but in a direction orthogonal to, the traffic being measured (also called *primary work*); and every node performing primary work (a *primary node*, in blue, in Figure 2.2.1) also handles the routing of other work (dashed lines).

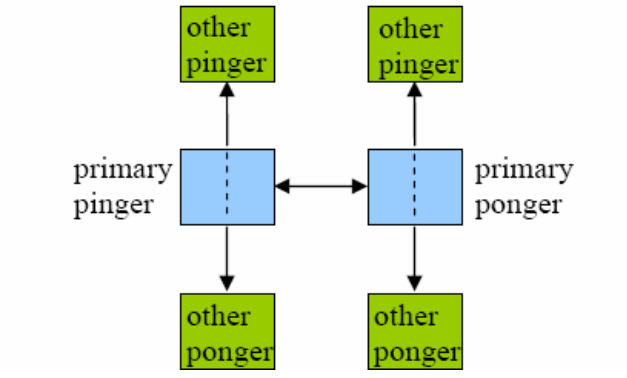

Figure 2.2.1: MPI Latency

Further, every primary node is constrained from executing the receive operation until the inbound message has arrived. To ensure against perturbation of timings, the primary nodes are paired to align in the same topological direction, and every primary node has only one partner. Since a primary node never serves as either a source or destination of other work, the nodes performing the other work (*other nodes*, in green) are constrained to the periphery of the topological domain of the test. The pairs of other nodes are separated by multiple links, but since their roles are ancillary and their performance is not being measured, this is acceptable.

Link Bandwidth (206)

For this test, a component group consists of a pair of compute nodes separated by a single network link. The key metric is the bidirectional bandwidth (gigabytes/second) when exchanging MPI messages of 1 megabyte or less between the nodes. The target is 3.8, and the deviation tolerance is 0.03.

The scaling of this test is constrained to ensure that execution in parallel does not unfairly perturb the timings. Specifically, the nodes are paired to align in the same topological direction.

Bisection Bandwidth (205)

For this test, a component group (see Figure 2.2.2) consists of an even number $2N$ of topologically contiguous and collinear compute nodes (labelled 0 through 2N – 1). Typical values for N range from 2 to 4. The nodes are paired so that each node is N hops away from its partner. Nodes 0 through N – 1 are assigned the role of *reporter* (blue in Figure 2.2.2), computing and storing the bandwidth achieved with its partner, and node 0 is identified as the *group leader*. The key metric is the aggregate bidirectional bandwidth (terabytes/second) across the links between nodes N – 1 and N of all groups when exchanging MPI messages of 1 megabyte or less between the paired nodes. This bandwidth is computed by summing the individual bandwidths observed by the reporters. The target is $0.0062M$, where M is the number of component groups, and the deviation tolerance is 0.05.

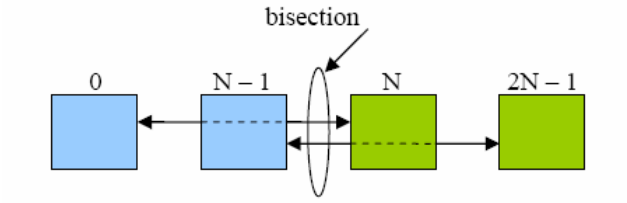

Figure 2.2.2: Bisection Bandwidth

The scaling of this test abides by LOFI scaling constraints. It is further constrained to comply with

detailed requirements, and to ensure that execution in parallel does not unfairly perturb the timings. First, the component groups are formed to align in the same topological direction. Second, to comply with requirements, the test is able to run on a full rectangular "slab" of the compute partition. This is in order to produce an aggregate bandwidth for a full bisection of the system.

2.3 Inter-partition Interconnect Network Hardware Tests

Bisection Bandwidth (211)

For this test, a component group (see Figure 2.3.1) consists of a pair of nodes, one compute node and one service node, separated by a single network link. The two nodes communicate using the Portals interface. Port assignment is negotiated via a shared data file. Each node runs two pieces of code, called *client* and *server* (blue and green, respectively), each in a dedicated MPI process – client sends the messages and server receives them. On the compute node, client and server are part of the same program, launched with the `yod -VN` option: client runs on core 'A' of the processor and server runs on core 'B'. On the service node, client and server are separate Linux programs, launched with the `mpiexec -machinefile` option so that both processes execute on the node. Each client exchanges Portals messages with the server on the other side of the plane. The two processes are termed *partners*. Clients are assigned the role of computing and storing the bandwidth achieved with their partners. Client processes of rank 0 are the *group leaders*, reporting the unidirectional aggregate bandwidth for their respective sides of the bisection plane. The key metric is the bidirectional aggregate bandwidth (gigabytes/second) when exchanging Portals messages of 1 megabyte or less between the nodes. The target is 2.5*M*, where *M* is the number of component groups, and the deviation tolerance is 0.2.

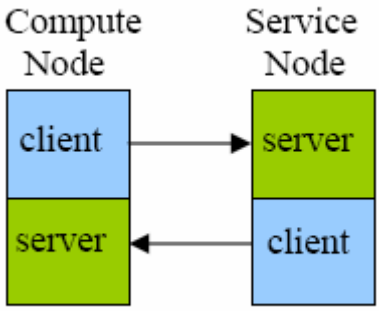

Figure 2.3.1: Bisection Bandwidth

The scaling of this test abides by LOFI scaling constraints. It is further constrained to comply with detailed requirements. Specifically, the test is able to run across a full rectangular "slab" of the system, in order to produce an aggregate bandwidth for a full bisection. The configuration of the system constrains the formation of component groups to align in the same topological direction.

2.4 Service Partition I/O Tests

I/O Bandwidth (208)

This test is a scaled component-group test with summary metrics. For this test, a component group (see Figure 2.4.1) consists of a small number *n* of compute nodes (clients, in blue) performing I/O on *n* files assigned to a single Lustre Object Storage Target (OST, in gold) being serviced by a service node (in green). Typical values for *n* range from 1 to 9. The key metric is the aggregate I/O bandwidth (gigabytes/second) achieved on the OSTs for read and write operations. This bandwidth is computed by summing the bandwidths reported by the OSTs' clients via the iobuf interface. The target is 0.157*M*, where *M* is the number of component groups, and the deviation tolerance is 0.1.

The scaling of this test abides by LOFI scaling constraints. In addition, it is constrained to comply with detailed requirements, and to ensure that execution in parallel does not unfairly perturb the timings. To comply with requirements, the test is able to run on 25% of the system's compute nodes. In addition, the total amount of data written and read in the I/O operations exceeds the total amount of disk cache on the OSTs by a factor of 10. To ensure against perturbation of timings, the test is loaded onto the compute partition so that the clients are aligned with the OSTs on the mesh as closely as possible, and the groups are formed so that each client performs its I/O only on the OST that is topologically nearest to it.

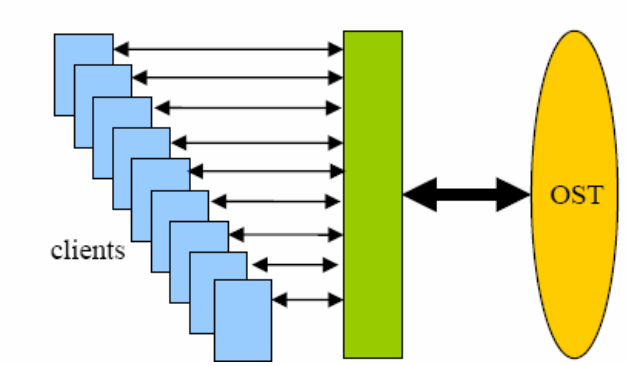

Figure 2.4.1: IO B/W, Compute

Currently, this test runs only one application. This application (CTH) performs its I/O in a one-file-per-client mode. The files typically reside within a single directory on a single file system, and Lustre striping is performed to ensure an even spread of files across the OSTs of the file system. Enhancement of the test to include the execution of another application (SAGE) that performs its I/O in a single-file mode is the target of future work.

I/O Bandwidth – Service Partition (405)

This test is a scaled component-group test with summary metrics. For this test, a component group (see Figure 2.4.2) consists of a service node (*client*, in blue) performing I/O on a file striped across a small number *n* of OSTs (*servers*, in gold). The value of *n* typically varies from 1 to 8. The client service node is one that has access to the Lustre file systems, and is not servicing any work other than as client for the test. The key metric is the aggregate I/O bandwidth (gigabytes/second) achieved on the clients for read and write operations to the OSTs. This bandwidth is computed by dividing the aggregate amount of data moved (a predefined quantity) by the elapsed time to execute the region of code performing the I/O operations. The target is 25/*nM*, where *M* is the number of component groups, and the deviation tolerance is 0.2.

The scaling of this test abides by LOFI scaling constraints. In addition, it is constrained to ensure that execution in parallel does not unfairly perturb the timings. Specifically, the test is loaded onto the service partition so that the clients are aligned with the OSTs on the mesh as closely as possible, and the groups are formed so that each client performs its I/O only on the OSTs that are topologically nearest to it.

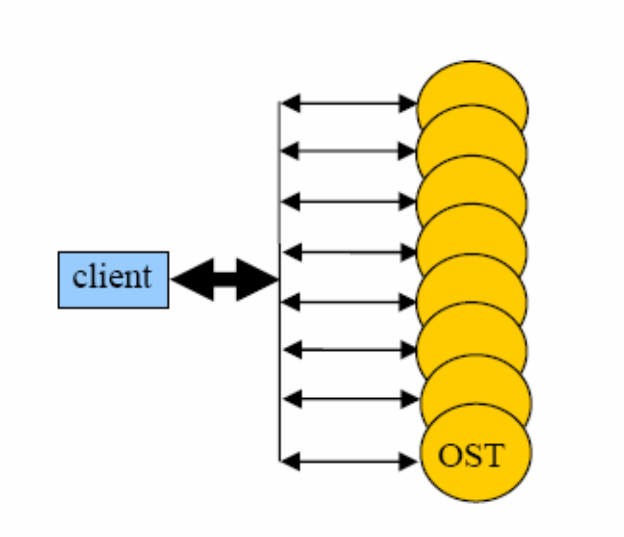

Figure 2.4.2: IO B/W, Service

Currently, this test runs only one application. This application, CTH, performs its I/O in a one-file-per-client mode. The files typically reside within a single directory on a single file system, and Lustre striping is performed to ensure an even spread of files across the OSTs of the file system. Enhancement of the test to include the execution of another application (SAGE) that performs its I/O in a single-file mode is the target of future work.

Single File Size and Accessibility (607)

This test is a single metric test. For this test, a number *n* of compute nodes (clients) performs I/O on a single file striped across a number *M* of OSTs in a file system. A known data pattern is written onto the file, then read back and verified. The key metric is the size of the file generated (terabytes). The target is 50.

The scaling of this test is constrained to comply with the requirements, and is optimized to ensure efficient parallel execution. To comply with requirements, the number *M* of OSTs used must exceed the quotient of the target size and the OST capacity. To ensure efficiency, the number *n* of clients is chosen to maximize the I/O bandwidth to each OST. Typical values for *n* are *M*, 2*M*, and 3*M*. Further, the test is loaded onto the compute partition so that the clients are aligned with the OSTs on the mesh as closely as possible. Finally, each client is assigned an equal I/O workload, and is directed to perform its I/O only on the OST that is topologically nearest to it.

2.5 External Network I/O Tests

Aggregate Network Bandwidth (209)

This test is a scaled component-group test with summary metrics. For this test, a component group (see Figure 2.5.1) consists of a service node with attached 10GigE network riser (*client*, in blue), a remote server (in green) dedicated to handling the riser network traffic, and a number *n* of Lustre OSTs (in gold). The value of *n* typically varies from 1 to 6. The key metric is the aggregate I/O bandwidth (gigabytes/second) through the clients when moving data from files striped across the clients' OSTs to the remote servers using iperf. The target is 0.25*M*, where *M* is the number of component groups, and the deviation tolerance is 0.1.

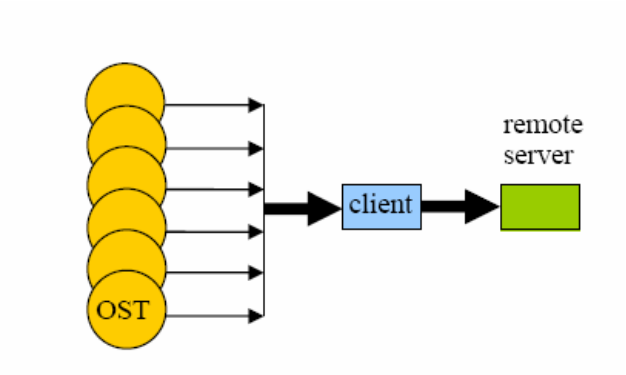

Figure 2.5.1: Ext. NW BW

The scaling of this test abides by LOFI scaling constraints. It is further constrained to comply with detailed requirements. Specifically, the test is able to run on as many as 50 clients in parallel.

2.6 Full System and Software Tests

IEEE-754 Compatibility (302)

This test is a software test. The components being tested include the compute node processor and the compilers. The test consists of over 30 separate programs from the Nelson Beebe IEEE-754 Compliance Suite. The programs are executed in a serial round-robin fashion on randomly selected compute nodes for a user-specified time period. Because the programs are functionality tests rather than performance tests, the key metric is the program output, the target is a predefined baseline output file, and the key assessment involves a simple textual comparison.

CPU Performance Counter Accessibility (303)

This test is a software test. The components being tested include the compute node processor and the PAPI library. The test consists of programs that exercise the counters for floating-point add and multiply, level-1 data cache hit and miss, and level-2 data cache hit and miss. The programs are executed in a serial round-robin fashion on randomly selected compute nodes for a user-specified time period. Each program compares the number of counted events against a preset expected value, and unfavorable comparisons are reported as non-compliance. The key metric is the sum of all detected non-compliances, and the target is 0.

High-Performance LINPACK (202)

This test is both a full system test and a software test. As a full system test, it stresses the interconnect network with traffic patterns of higher complexity than the other tests in the suite. It also can be used to test the system's environmental monitoring and control features, because of its ability to induce the Opteron processor to draw power and generate heat at near-maximum levels. As a software test, it is used to track the performance of the linear algebra components of the ACML library. The test is based on the Netlib distribution. Execution scripts have been developed to run an optimally sized problem given a specified node count and run time, and to run multiple instances of the test on separate "slices" of the system concurrently. The key metric is the performance of the matrix solver in teraflops/second. The target is $0.0036M$, where M is the number of processor cores specified for the run.

Job Load/Launch Time (615)

This test is a full system test. The key metric is the elapsed time, in seconds, to load and launch a heterogeneous application onto the full system. The time measurements for "load and launch" are taken when the launch command (yod) is issued and when the application completes the execution of the `MPI_Init` function, and the elapsed time is taken to be the difference in these two measurements. "Heterogeneous" means an application consisting of at least three executables, each at least 1 megabyte in size, and all sharing a single communicator space. "Full system" means all available compute nodes and all available service nodes configured to run applications. The target is 60.

MPI-2 Functionality (605)

This test is a software test, designed to verify the functionality of the MPI-2 capabilities of the MPI library. The test consists of programs from the MPICH2 1.0.3 distribution. The programs are executed serially on a subset of about 100 compute nodes. In order to optimize execution time, some of the longer-running programs are executed concurrently. Because the programs are functionality tests rather than performance tests, the key metric is a pass/fail message reported by the test.

TotalView™ Capability (617)

This test is a software test, designed to verify the functionality of the TotalView debugger under various conditions of special interest. First, TotalView is presented with a heterogeneous application (at least three executables, each compiled from a different source language) and is directed to execute the application through a predefined command sequence. Second, TotalView is presented with a single large application (1024 processes), and is directed to execute the application, from both initial launch and live attach, through a predefined command sequence. Because the test is a functionality test rather than a performance test, the key metric is the output of the test sequence, the target is a predefined baseline output file, and the key assessment involves a simple textual comparison. The

execution scripts for this test use the expect tool to drive the TotalView command-line interface.

The table below lists the tests in order by test number. (Where prefixes are used to denote multiples or fractions of a unit, a base-ten multiple or fraction is assumed.)

Test	Description	Type	Units	Target	Dev. Tol.
104	CPU ID, freq	SC	GHz	2.4	.0001
105	Mem size	SC	GB	1.9	0.005
202	HPL	SM	TF	0.0036M	N/A
204	MPI latency	CG	us	11.5	0.01
205	Bisection BW	CG	TB/s	0.0062M	0.05
206	Link BW	CG	GB/s	3.8M	0.03
208	Agg I/O BW	CG	GB/s	0.157M	0.1
209	Agg NW BW	CG	GB/s	0.25M	0.1
211	Bisection BW	CG	GB/s	2.5M	0.2
302	IEEE	SM	N/A	N/A	N/A
303	Perf Counters	SM	Events +/-	0	N/A
305	Mem latency	SC	ns	80	0.005
307	Mem BW	SC	GB/s	4.0	0.005
405	Agg I/O BW	CG	GB/s	25.0/nM	0.2
605	MPI-2 func	SM	N/A	N/A	N/A
607	Single file	SM	TB	50	N/A
615	Load/launch	SM	s	60	N/A
617	TotalView	SM	N/A	N/A	N/A

Table 2.1: Tests Comprising the CTS

3. CTS in Action

During Initial Operations (January – May 2005), Test 104 was used to identify Opteron processors whose signature or clock frequency was out of compliance. Test 105 was used to identify node memory components that were of incorrect size or error-prone. Test 206 was not finished yet, so Test 202 was used to identify problem areas in the interconnect network. Test 208 was used to identify Lustre OSTs with improper configuration parameter settings. Test 209 was used to identify 10GigE network nodes with improper configuration parameter settings.

During May – July 2005, the nodes were upgraded with larger memories. Tests 105 and 307 were used to exercise newly installed memory parts. Test 307 identified discrepancies in memory bandwidth between nodes that had Micron™ memory parts and those that had Samsung™ memory parts.

During August – September 2005, engineers were tuning the voltage on the Seastar processors. Tests 202, 205 and 206 were used to test link behavior at various voltages.

During July – September 2006, the system underwent a major upgrade. The Opteron and Seastar processors were all replaced, a fifth row of cabinets was added, and the scratch file systems were reconfigured to improve performance. Tests 104, 105, 305, 307, 205, 206, 208 and 605 were used to exercise the new parts, verify compliance, and verify performance expectations on upgraded sections of the system as they were returned to service. Test 307 identified discrepancies in memory bandwidth between nodes that had mixed-memory parts (Micron and Samsung) and those that had homogeneous memory parts. The new, faster Opteron CPU played a role in bringing out this discrepancy. In the network area, a modified, temporary version of Test 206 (internally named *iorsim*) was used to verify a problem in the aging of network packets as they converged in the Z direction en route to the service partition (which has no modules in the center cages).

From July 2002 to the present, the CTS has been used to test Red Storm and many prior generations of development/prototype systems. The table below describes the Software Problem Reports (SPRs) submitted as a result of this testing. The entries in the table are ordered according to the date they were submitted, with most recent at the top.

Test	SPR	Product	Description
204	737596	Catamount	Bump counter in user pcb whenever app is interrupted
617	737345	Catamount	Fatal error in get_icd_node when trying to terminate session (320p app)
605	737327	MPICH2	Need a way to control buffering of mpi-io operations a la iobuf

Test	SPR	Product	Description
605	737291	MPICH2	MPI_long_double not supported, even though compiler supports long double
617	737283	Catamount	Connection errors when trying to debug > 64 processes
307	736932	Catamount	Memory bandwidth on mixed-memory dual-core nodes unexpectedly slow
302	736359	Catamount	MMX control and status register (mxcsr) not being reset across yod launches
303	736029	CC	Modify qk-pgcc, qk-pgf90 to support integer operation counter facility
607	735856	Lustre	160-stripe limit on single Lustre file precludes 100 tb file demonstration
607	735834	Lustre	Readx and writex from multiple processes does only entry 0 of vector
617	735673	TotalView	CLI debugging of hetero-geneous applications does not work
607	735639	Linux	C programs that include aio.h fail to compile
605	735568	MPICH2	MPI_Get_processor_nam e return value of hostname not sufficiently unique
607	735288	IOBUF	Add parameter to specify a preallocation for a file spec on create
611	735287	MPICH2	App hangs while issuing mpi error messages when 10000 processes send to rank -2
GE N	735205	Install	Add xt-tools products to /etc/motd rpm name/version table
607	734934	Lustre	Need a way to preallocate a file of a specified size at create time
GE N	734721	C++	CC -v exhibits new and undesirable behavior
617	734603	TotalView	TotalView on xt3 takes approx 2 seconds per PE to attach to running process
617	734568	PUBS_PE	Add examples of using TotalView on apps that read stdin, and on running apps
208	734407	Lustre	Need a command like iostat to report Lustre utilization statistics
302	734093	Catamount	IEEE test produces different results on catamount, Linux
208	732894	LIBC	Need a way to control IO buffer size for apps from shell environment
208	732885	PGF90	pgf90 setvbuf3f doesn't alter buffer size
303	732842	Catamount	Catamount libc function sysconf returns bad value for _sc_open_max parameter
105	732629	PCT	PCT heap fragmentation leads to loss of available user memory on catamount
302	732616	PGF90	min or max intrinsic should never return nan when one argument is a number
303	731972	LIBPAPI	PAPI_get_executable_inf o returns nil pointers for segment addresses
303	731925	LIBPAPI	fpapi.h not available from papi/3.0.8.1 module
302	731757	PGF90	PGI Fortran stop state-ment performs standard IO that is not scalable
GE N	731756	RCA	Provide user app access to topology information
615	731612	Lustre	Lustre mount and init messages negatively impact job startup time
617	731505	Lustre	stat(2) call returns incor-rect, inconsistent results
617	731486	C++	CC command doesn't know where to find liblustre
GE N	731309	CC	cc -v should not try to link
202	731295	CC	Catamount cc compiler driver does not provide linkage to acml library
208	731273	Lustre	Need a method to turn off Lustre messages
303	730771	LIBPAPI	PAPI reports L2 cache hits even on L2 cache misses
303	730707	LIBPAPI	Include file fpapi.h is missing
302	730392	LIBC	Provide an environment variable to control buffer-ing of stdout, stderr

Test	SPR	Product	Description
303	730145	LIBC	PAPI_read_counters resets counters - should just read them
105	729222	QK	Add capability to tune size of portals buffer
302	729005	PGF90	pgf90 outputs -nan for 0/0, while pgcc outputs nan
302	728961	LIBC	Fortran deps routine computes a different epsilon for qk vs Linux node
208	728642	PGF90	Provide pgif90 library routine to get message for iostat error code
302	728376	PGCC	pgcc %le format descriptor on printf produces incorrect results
302	728375	PGCC	pgcc computes unexpected zero result for subnormal underflow limit
105	728235	LIBC	Enhance libc to report peak node memory usage
302	728193	PGCC	pgicc computes quantities involving zeros in a non-standard way
302	728161	PGF90	PGI f90 handling of (arithmetic involving) signed zeros nonstandard
302	728159	PGF90	pgf90-built executable fails on formatted read of inf and nan data
204	728083	MPICH2	MPI ping-pong latency increases logarithmically with number of ping-pong trials
208	727799	PGF90	Provide FORTRAN-callable setvbuf routine
307	727538	PGF90	Stream code fails checksum intermittently if -M nontemporal and arrays in common
208	726981	MPICH2	mpif.h does not define mpi_wtime as external

Table 3.1: SPRs Submitted Based on CTS Results

4. Future

The Red Storm system is expected, at UNICOS/lc 2.0, to be running an accelerated version of Portals, which will execute more of the messaging code on the Seastar processor. This will result in improved MPI latency performance. Test 204 will be run on the system before and after the introduction of Accelerated Portals, to measure the extent of the improvement.

The current version of Lustre running on Red Storm limits the number of OSTs across which a single file can be striped. This condition in turn limits the I/O bandwidth that can be achieved when doing I/O on a single shared file. It also limits the size of a single file to about 50 terabytes. Once the software is upgraded to Lustre 2.6, this limit will be removed, and Tests 208 and 617 will be run in single-file mode with targets of 0.157*M* and 100, respectively.

The current version of UNICOS/lc does not support an SMP kernel for dual-core service nodes. This condition limits the bandwidth between compute and service nodes observed in Test 211. Once the software is upgraded to UNICOS/lc 1.5, this limit will be relaxed, and Test 211 will be run with a target of 3.2*M* and a lower deviation tolerance.

The current version of Lustre running on the Red Storm service nodes has a bug that causes the Linux Lustre client to use an inordinate amount of CPU time when servicing Lustre I/O on the service nodes. This condition, along with the lack of SMP kernel support mentioned above, limits the I/O bandwidth between service nodes running applications and Lustre OSTs. Once the software is upgraded to Lustre 2.6 and UNICOS/lc 1.5, these limits will be relaxed, and Tests 209 and 405 will be run with higher targets (0.5*M* and 50/nM, respectively) and lower deviation tolerances.

The Cray XT3 performance tools are being enhanced to include performance counters for integer math operations, in much the same manner that floating-point operations are currently counted with PAPI. This feature has been referred to internally as *ICP*. Once these enhancements are available, Test 303 will be modified and run to test the new counters.

The current version of UNICOS/lc does not support the execution of heterogeneous applications that span both the compute and service partitions. This capability is targeted for a future release of UNICOS/lc. Once this capability is available, Tests 615 and 617 will be modified to run in this heterogeneous mode, using performance targets comparable to those currently in effect.

Work is ongoing to make the CTS amenable to packaging and distribution to other Cray XT3 systems. This work is focusing on three major areas: flexible build and install system, generalized methods for determining system topology, and streamlined user interface.

5. Acknowledgements

The following people have made contributions to the Red Storm Compliance Test Suite, with advice, review comments, code contributions, test results from related tests, and in many other countless ways. The tests they helped influence are listed after their names. Full responsibility for the content and behavior of the tests, however, rests with the author (Davis).

Cray, Inc.: Bob Alverson (104, 206); Howard Pritchard, Gail Alverson (204); Kevin Welton (206); Sarah Anderson (202, 307); Luiz DeRose, Tom Gardiner (303); Mark Pagel (605); Kevin Thomas, Dennis Dinge (208).

Sandia National Laboratories: Courtenay Vaughan (202, 208, 307); Keith Underwood (206); John Vandyke (204); Kevin Pedretti (211); Sue Goudy (305).

About the Author

Mike Davis is a Cray applications analyst on-site at Sandia National Laboratories. He can be reached at u3186@cray.com.

References

Cray Inc.	http://www.cray.com
Sandia National Laboratories	http://www.sandia.gov
Cray XT3	http://www.cray.com/products/xt3/index.html
workshop	http://www.c3.lanl.gov/lasci-wpp
Iyer et al.	http://parasol-www.cs.tamu.edu/publications/download.php?file_id=183
STREAM	http://www.cs.virginia.edu/stream
Portals	http://www.cs.sandia.gov/Portals
Lustre	http://www.clusterfs.com
iobuf	http://docs.cray.com/books/S-2396-15/S-2396-15.pdf
CTH	http://citeseer.ist.psu.edu/
	416369.html
SAGE	http://ieeexplore.ieee.org/iel5/9584/30302/01392661.pdf
iperf	http://dast.nlanr.net/Projects/Iperf
Nelson Beebe	http://www.math.utah.edu/~beebe
PAPI	http://icl.cs.utk.edu/projects/papi
ACML	http://developer.amd.com/acml.jsp
Netlib	http://www.netlib.org/benchmark/hpl
yod	http://docs.cray.com/books/S-2396-15/S-2396-15.pdf
MPICH2	http://www-unix.mcs.anl.gov/mpi/mpich2
TotalView	http://www.etnus.com
expect	http://expect.nist.gov