Red Storm / Cray XT4: A Superior Architecture for Scalability

Mahesh Rajan, Douglas Doerfler, Courtenay Vaughan

***Abstract*—The benefits of the Cray XT4 node and interconnect architecture on scalability of applications are analyzed with the help of micro-benchmarks, mini-applications and production applications. Performance comparisons to a large InfiniBand cluster with multi-socket nodes incorporating a similar AMD processor brings out the importance of architectural balance in the design of HPC systems. This paper attempts to relate application performance improvements to simple architectural balance ratios.**

***Index Terms*—Parallel application, HPC architectures, multi-core processors, affinity control, communication balance, memory contention**

I. Introduction

THIS paper examines application scalability on the Sandia National Laboratories Red Storm/Cray XT4 supercomputer. Red Storm has been in service since late 2004 and has undergone substantial upgrades since the initial deployment [1]. In October 2006 the processor was upgraded from the single core AMD 2.0 GHz Opterons to 2.4 GHz dual core AMD Opterons. A third upgrade completed recently in 2008 brought the Red Storm from a peak performance of 124 TFLOPS to a peak of 284TFLOPS. The center section or 6,240 of the 12,960 compute nodes, were upgraded to 2.2 GHz quad-core Opteron processors. The memory on the entire system was brought up to 2 GB per core. The existing cabinets, backplanes, interconnect, cabling, and service and I/O nodes were reused. The system now has 38,400 compute node cores with 75TB of compute node memory. It is now a mildly heterogeneous system with the "center section" using the quad-core nodes and the rest using the older dual core nodes.

In a recent major acquisition by DOE/NNSA Tri-labs, several Tri-lab Linux Capacity Clusters (TLCC) have been installed at: SNL, LANL and LLNL. The TLCC architecture with ccNUMA, multi-socket, multi-core nodes, and InfiniBand interconnect, is representative of the trend in HPC architectures. TLCC uses a similar quad-core processor as in Red Storm, but differs in the node and interconnect architecture. The processor and memory DIMM similarities between Red Storm and TLCC provide an opportunity to investigate the impact of node and interconnect architecture on applications. Performance and scalability are evaluated by studying these architectures with micro-benchmarks, mini-applications and full applications. An attempt is made to relate the observed performance to architectural balance between memory, CPU, and the high-speed interconnect.

II. Architecture Description

A. Red Storm

To facilitate the comparison of performance to the Red Storm/XT4 at Sandia National Laboratories in Albuquerque, we provide a brief description of it here. Red Storm currently is a heterogeneous system made up of 6,240 2.2 GHz AMD Opteron-Budapest quad-core nodes and 6,720 2.4 GHz AMD Opteron-280 dual-core nodes for a total of 12,960 compute nodes. The Budapest quad-core processor was developed by AMD for single socket servers, and has the same L1 cache (64 KB data and 64 KB instruction per core), L2 cache (512K per core) and a shared 2 MB L3 cache as the AMD Barcelona processor in the TLCC system. The most significant technical difference between the Budapest and Barcelona chips is that Budapest processors use AMD's HyperTransport 3 technology while the Barcelona processors utilize HyperTransport 2. The Budapest Opteron is functionally equivalent to the Barcelona Opteron , except that they have fewer HyperTransport links and the electronics for the NUMA-style shared memory processor clustering removed. The dual core nodes have 4GB of DDR-400 memory and the quad-core node have 8GB of DDR2-800 memory. The DDR2 memory provides a peak memory bandwidth of 12.8GB/s while the older DDR-400 has a bandwidth 6.4GB/s. The Theoretical peak performance of the Red Storm is 284 TeraFLOPS. Nodes are connected with the Cray SeaStar NIC/routers attached to the node processors via HyperTransport.

Manuscript submitted to Cray User Group (CUG) meeting , May 4-9, 2009, Atlanta, Georgia, USA 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.

Author are with the Sandia National Laboratory, P.O.Box 5800, MS0807, Albuquerque, NM 87185-0807; e-mails: mrajan@sandia.gov; dwdoerf@sandia.gov; ctvaugh@sandia.gov.

The network is a 27x20x24 mesh topology (torus in Z), with a peak bidirectional link bandwidth of 9.6 GB/s. The nearest neighbor NIC to NIC latency is specified to be 2 µsec, with 4.8 µsec measured MPI latency. The compute nodes run the Catamount lightweight kernel (LWK). The I/O and administrative nodes run a modified version of SuSE Linux. Additional details of Red Storm can be found in these references [1], [2].

Table 1. Red Storm and TLCC system architectural characteristics

Name	Cores/ Node	Networlk/ Topology	Total nodes(N)	Clock (GHz)	Mem/core & Speed	MPI Inter Node Latency (usec)	MPI Inter Node BW (GB/s)	Stream BW (GB/s/Node)	Memory Latency (clocks)
Red Storm (dual)	2	Mesh/ Z Torus	6,720	2.4	2GB; DDR-400MHz	4.8	2.04	4.576	119
Red Storm (quad)	4	Mesh/ Z Torus	6, 240	2.2	2GB; DDR2-800MHz	4.8	1.82	8.774	90
TLCC	16	Fat-tree	272	2.4	2GB; DDR2-667 MHz	1.0	1.3	15.1	157

Table 1 provides a summary of the key architectural characteristics of Red Storm and TLCC. These values are used to calculate the memory and interconnect balance characteristics tabulated in Table 2.

B. TLCC

The SNL TLCC cluster has 38 teraFLOPS peak performance. It is built with 2 'scalable units' for a total of 272 compute nodes with 16 cores per node for a total of 4352 processor cores. Each scalable unit is nominally constructed with 144 nodes (136 compute, 1 login, 1 management, 6 gateway). The node board is a SuperMicro H8QM8-2 motherboard. The node has four AMD Opteron 8300 Barcelona quad-core processors clocked at 2.2 GHz. The quad-socket node has independent 667 MHz DDR2 DIMMS for each socket. The total node memory is 32 GB giving the compute nodes a total memory of 8.7 TB. The nodes are interconnected in a full bisection bandwidth fat-tree topology using a twelve 24-port InfiniBand 4X DDR switches. The InfiniBand HCAs are the Mellanox ConnectX 4x DDR cards. The cluster runs the Tri-lab standard software stack, TOSS, which includes the Red Hat Enterprise Linux operating system and the MOAB/SLURM resource manager. Intel, PGI, GNU compilers and math libraries together with OpenMPI and MVAPICH MPI libraries form the programming environment.

Table 2 is a collection of memory and interconnect architectural balance ratios. Higher the Bytes-to-FLOPS Memory ratio, less is the loss in performance due to memory contention. Similarly, higher the Bytes-to-FLOPS Interconnect ratio less is the performance degradation due to contention at the high-speed network. For obtaining these ratios STREAMs bandwidth numbers were used for memory and MPI Inter-node Ping-Pong bandwidth numbers for the interconnect as recorded in Table 1. The maximum (MAX) values were computed for a single MPI task on a node. The minimum (MIN) values assumes full contention and all the cores on a node are sharing the memory or interconnect bandwidth.

Table 2. Red Storm and TLCC system architectural balance ratios

	MAX Bytes-to-FLOPS Memory	MAX Bytes-to-FLOPS Interconnect	MIN Bytes-to-FLOPS Memory	MIN Bytes-to-FLOPS Interconnect
Red Storm (dual)	0.824	0.379	0.477	0.190
Red Storm (quad)	0.756	0.232	0.249	0.058
TLCC	0.508	0.148	0.107	0.009
Ratio: Quad/TLCC	**1.49**	**1.57**	**2.33**	**6.28**

III. MICRO BENCHMARKS

A. *Memory access latency benchmark:*

The memory access latency benchmark consists of accessing elements of an integer array of size n that contains integers 1 to n, in a random order forcing loads from cache or memory in a non-sequential order. The maximum size of the array used in the test is 32 MB. Performance for this benchmark was measured on Red Storm Dual nodes with 400 MHz DDR DIMMS, on the Red Storm Quad nodes with 800 MHz DDR-2 DIMMS and on TLCC with the 667 MHz DDR-2 DIMMS. The TLCC cluster is operated with a nominal 4 KB page size while the Red Storm's default page size at job launch is 2 MB. Three clock cycles latency for the data on L1 cache for sizes up to 64 KB bytes, twelve cycles for the data in L2 cache up to 512KB and thirty two cycles latency for 1 MB array size for data in the L3 cache were observed on the quad-core processors on both TLCC and Red Storm. . However once the data access occurs from the main memory Red Storm is faster by as much as 100 cycles as shown in Figure 1. The impact of page size is not captured in Figure 1 as we wanted to reflect the default page sized used with the applications. Red Storm provides for use of small pages by specifying 4 KB page size at job launch. TLCC does not have such an option and the operational page size is set at 4 KB. Tests on Red Storm show that the impact of TLB page size is about 30-40 clock cycles with the 2 MB page leading to lower latencies. To investigate possible impact of the slower memory speed on TLCC, the same test was run on a Red Storm Qualification system that fortunately has the same 667 MHz memory on a quad-core node. But the results showed no significant degradation. This suggests that there is a substantial cache coherency overhead of 60 to 70 clocks for the quad socket node on the TLCC.

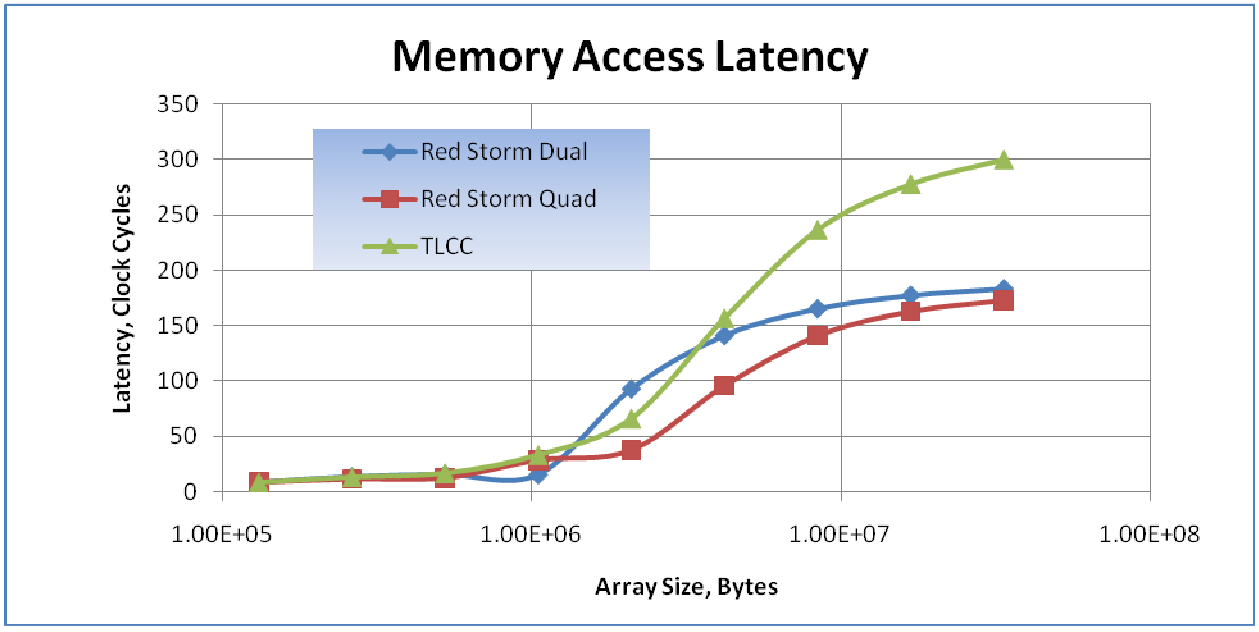


Figure 1. Memory Access Latency

B. *Memory Bandwidth Benchmark – STREAMS*

The MPI version of the STREAMS bandwidth test was used to measure the memory bandwidth on Red Storm dual core nodes, Red Storm quad core nodes and on the TLCC node. The PGI compiler flags chosen were the same (-O3 –fastsse) on these platforms. Although higher bandwidth may result with more optimal compiler flags it does not impact the comparisons. The single MPI task memory bandwidth test shows that the Red Storm quad-core node has about 30% better performance than TLCC as shown in Figure 2. Again to investigate possible impact of the slower memory speed on TLCC, the same test was run on a Red Storm Qualification system that has the 667 MHz memory on a quad-core node. The results showed that the memory speed reduced the bandwidth by 9%, giving a Triad Bandwidth of 6089 MB/sec. The rest of the difference is mostly due to the cache-coherence overhead on the TLCC because the page size was seen to have only a small impact.

On the Red Storm quad-core node two MPI ranks saturate the node. Similarly on TLCC two MPI tasks saturate memory at a single socket. For STREAMS test of two or more MPI ranks the process placement on TLCC was explicitly controlled via "*NUMACTL –physcpubind= -membind=*" flags. At eight MPI tasks the memory access is saturated and no increase in memory bandwidth is observed on the TLCC node with additional processes. This impacts applications that requesting sixteen MPI tasks per node on TLCC, in that the sixteen MPI tasks might experience memory contention.

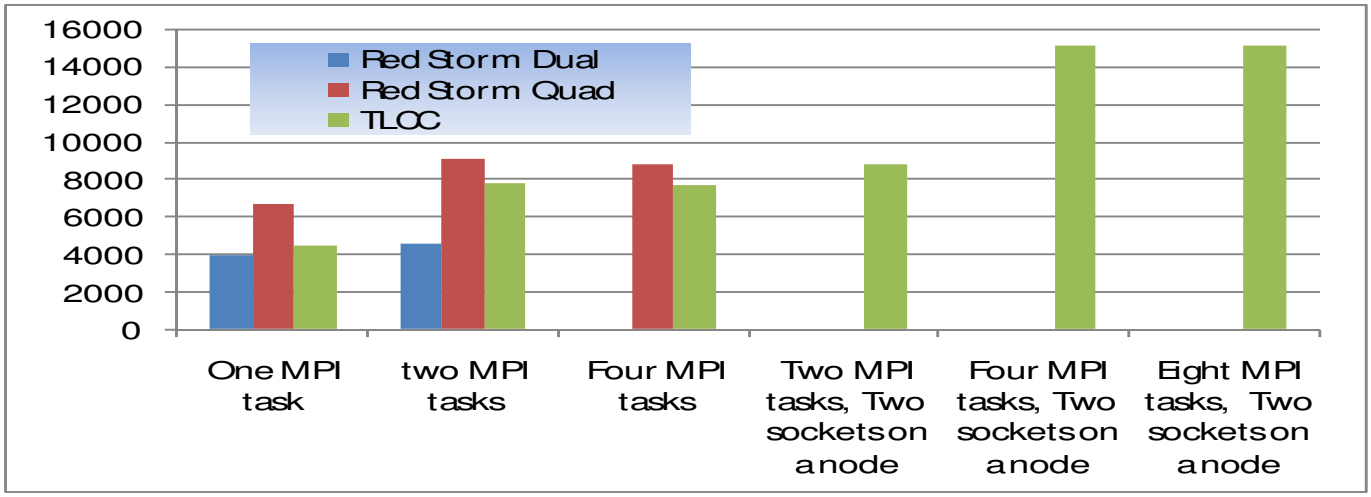


Figure 2. Streams Triad Memory Bandwidth (MBytes/sec)

C. MPI Ping-Pong benchmark

Figure 3 shows the performance of a simple inter-node ping-pong test. Red Storm achieves a peak uni-directional bandwidth of 2.04 GB/s with dual-core nodes, 1.82 GB/s with quad-core nodes. The TLCC maximum inter-node bandwidth was measured at 1.3GB/s. The InfiniBand inter-node latency on TLCC is 1 µsec while the Red Storm latency is 4.8 µsec. The lower latency on TLCC also leads to better bandwidth for data sizes less than 5 KBytes seen in Figure 3. On TLCC this inter-node bandwidth is shared between a maximum of 16 MPI tasks, while on the Red Storm it is only shared between a maximum of 4 MPI tasks.

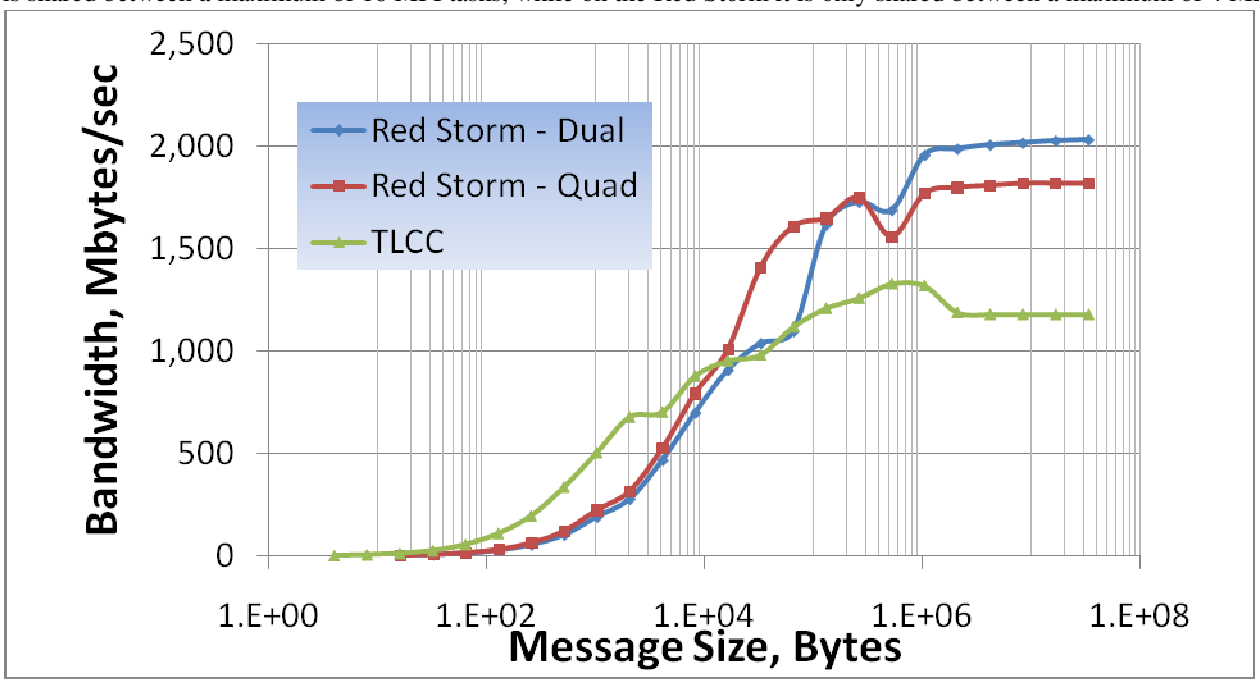


Figure 3. MPI Ping-Pong Unidirectional Bandwidth (MBytes/sec)

D. MPI Global Operation (Allreduce) benchmark

Figure 4 shows the performance of MPI_Allreduce with an eight byte data transfer. The TLCC measurements shown were taken with the MVAPICH (version 1.0.1) MPI library and as it was significantly better than the measurements with OpenMPI (version 1.2.7) library. The TLCC performance compares well with the Red Storm performance. The lower inter-node latency on TLCC leads to slightly lower run times. In both cases all cores on each node were used for this benchmark.

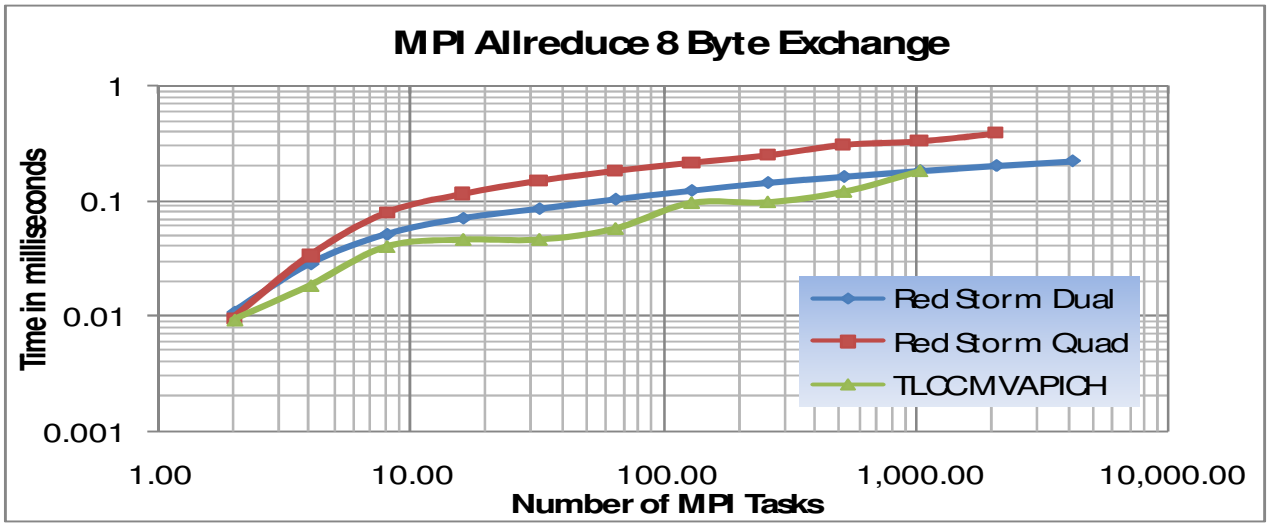


Figure 4. MPI Global Operation (Allreduce) time in milliseconds

E. MPI Random and Bucket-Brigade benchmarks

A couple of other simple MPI benchmarks that approximates messaging pattern for applications discussed in section V were run on Red Storm and TLCC. These results are helpful in understanding the causes for significant differences in performance seen between the two platforms. In the Random Messaging (RM) benchmark, thousands of message sizes (varying from 100 to 1KB) are sent to random MPI rank destinations. The messaging rate from each process and the average messaging rate are computed. In the Bucket-Brigade benchmark with small messages (BBS) 8 byte messages are exchanged in a ring pattern. In the second Bucket-Brigade benchmark with large messages (BBL) 1 MB messages are exchanged. The results are shown in Table 3. Observe the significantly lower performance of TLCC for RM and the degradation at the maximum scale.

Table 3: MPI Random and Bucket Brigade messaging rate (MBytes/sec) with 1024, 256 and 64 MPI ranks

	RM -1024	RM-256	RM-64	BBS-1024	BBS-256	BBS-64	BBL-1024	BBL-256	BBL-64
Red Storm Dual	67.7	71.9	75.3	1.19	1.19	1.20	1100.2	1116.1	1132.4
Red Storm Quad	41.6	45.0	46.9	0.86	0.86	0.86	654.7	647.6	632.2
TLCC	0.43	1.59	3.64	1.77	3.37	3.42	275.47	314.3	344.1

IV. MINI-APPLICATION PERFORMANCE AND SCALING

A. Mantevo-HPCCG:

HPCCG is a Sandia Mantevo Mini-Application[3] that captures the key computational kernel in the implicit solvers of the Trillinos [6] solver package. HPCCG implements a Conjugate Gradient (CG) algorithm in which the coefficient matrix is stored using a sparse matrix format. Most of the compute time is dominated by sparse matrix-vector multiplication. The interprocessor communication is minimal, requiring exchange of nearest neighbor boundary information, in addition to global MPI_Allreduce operations required for the scalar computations in the CG algorithm. Weak scaling studies, assigning identical computational load to each MPI task, have been carried out. Figure 5 shows the total wall time as a function of the number of MPI tasks for each system. This mini-application clearly brings out the impact of memory architecture on application scaling. We can see that for the quad-core Red Storm node, the jump from two to four MPI tasks degrades performance by 41% due to memory contention. Once the best performance within a node is achieved, the weak scaling curve is near perfect (flat). For the quad-socket, quad-core TLCC node, we see that from 1 to 2 MPI tasks we get perfect scaling (no contention for memory), 37% degradation in performance from 2 to 4 MPI tasks and additional 44% loss in going from 8 to 16 MPI tasks on a node. Using multiple nodes on TLCC, assigning an MPI task to every core on each node, shows subsequent perfect weak scaling, when memory and processor affinity are forced with the *NUMACTL* utility.

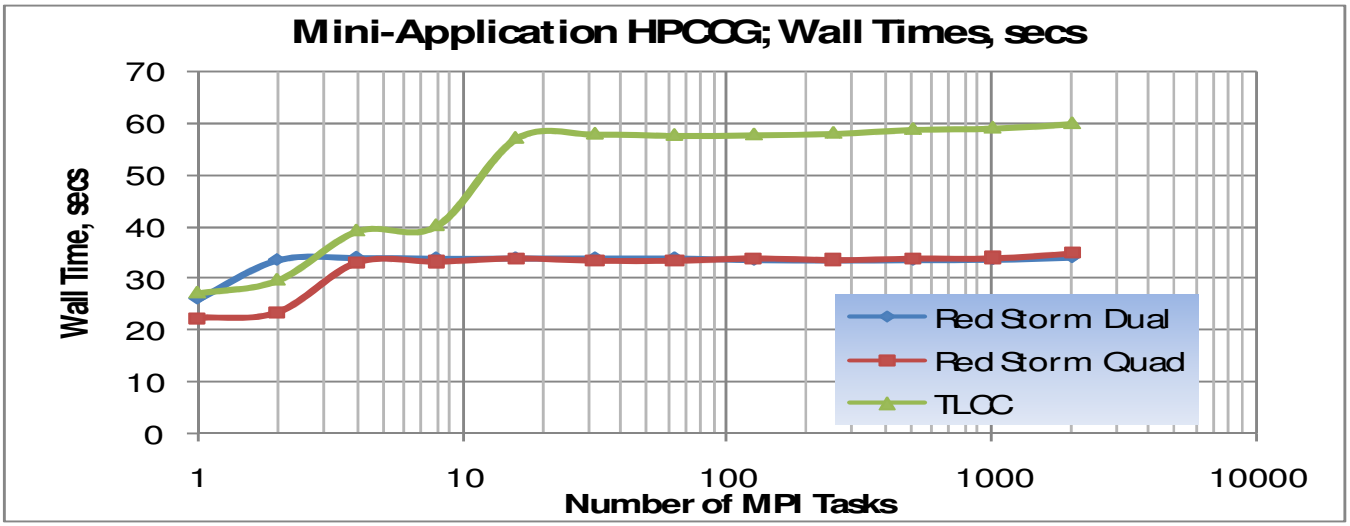
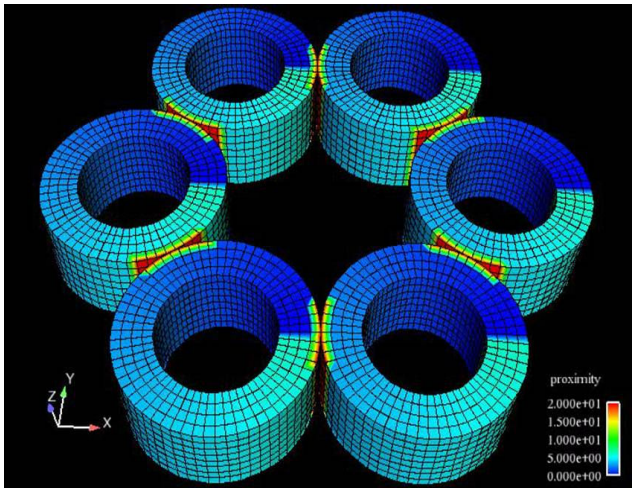


Figure 5. HPCCG performance, wall time in seconds

B. Mantevo-phdMesh:

Contact detection has been a performance-critical algorithm for parallel explicit dynamics simulation codes for over a decade. The parallel heterogeneous dynamic mesh (phdMesh) is a library in Trilinos [4] that provides an in-memory mesh and field database for parallel, heterogeneous, dynamic, unstructured meshes. This library includes a parallel implementation of an oct-tree geometric proximity detection algorithm with state-of-the-practice *N*log(N)* complexity. The phdMesh library and oct-tree geometric search algorithm are integrated to form a parallel geometric proximity search mini-application.



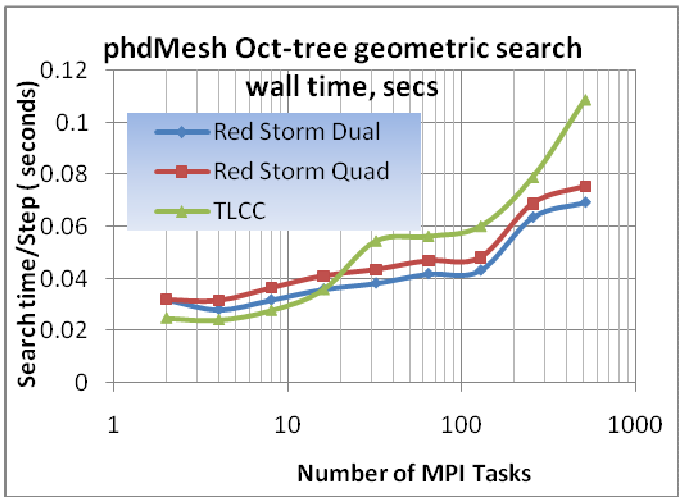


Figure 6. phdMesh test case: grid of counter-rotating gears and Oct-tree search performance

The weak scaling study for this mini-application uses a grid of simple counter-rotating gears, as shown in Figure 6. Gear configurations of 4x3x1, 4x3x2, 4x3x4, 4x6x4, etc., were set up to run on 2,4,8,16, etc., processors respectively. In other words, the number of gears along the X,Y,and Z axes were scaled so as to create an approximate weak scaling input, while maintaining a reasonable aspect ratio. The basic 4x3x1 configuration has 5952 surface-facets (N) and the surface-facets for the others scale accordingly. When the application is executed, the run times for different execution regimes, such as Meshing, Rebalance, and Search/step, etc., are tabulated on output. The performance characteristic of principal interest is the oct-tree geometric search proximity detection execution time for the rolling surface-facets of these gears. Figure 6 presents the weak scaling characteristics observed by plotting the Search/step wall time in seconds, against the number of MPI Tasks.

V. APPLICATION PERFORMANCE AND SCALING

This effort was undertaken to meet management's request to find middle and upper range scalability of SNL's applications on ASC HPC systems. The goal of this effort is to guide the users on targeting their simulations on suitable platform and to initiate efforts towards optimizing performance on the systems. Application performance was measured on TLCC for a variety of

applications and compared to Red Storm. The Red Storm achieves excellent application scalability through three important architectural design features: 1) maximize node memory performance by using single socket nodes minimizing contention when all the cores are assigned an MPI task 2) maximize inter-node messaging performance through custom designed network interface, and 3) minimize operating system interference by using a light weight operating system kernel. Applications vary in how they benefit from these architectural features in minimizing wall-clock run time and on parallel scalability. Contrastingly, the TLCC architecture takes advantage of commodity multi-socket node boards, interconnect and operating system to provide a cost-effective solution for large group of SNL users needing capacity computing cycles that typically require 64 to 256-way parallelism. The TLCC architecture with 16 cores, 4sockets per node and a cache coherent NUMA memory sub-system, imposes additional memory access overhead if a process on a core seeks data from memory associated with a processor on a different socket. SNL applications are predominantly based on the message passing programming model. Application scaling studies were conducted assigning one MPI task per core on TLCC and Red Storm. TLCC runs benefited by enforcing processor and memory affinity to prevent remote memory access penalty and process migration overhead. A quick comparison of the performance on the two platforms for a variety of applications (for many of which details are provided in this paper) is shown in Figure 7. The ratio of run times for 64, 256 and 1024 way parallel jobs are provided taking the best run time on TLCC, which typically required runs with the *NUMACTL* utility and repeating runs that may have an inflated run time due to load on the system. Red Strom run times rarely varied from run to run.

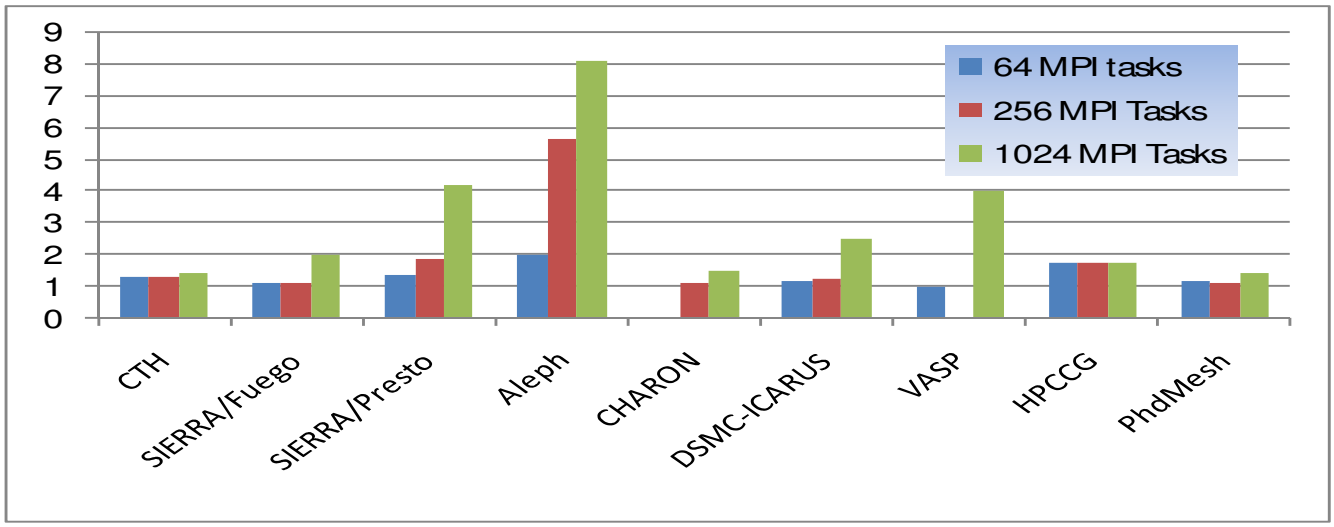


Figure 7. Wall clock run time ratio: TLCC/Red Storm for 64, 256 and 1024 way parallel jobs

In Figure 7, if there run time ratio does not increase in going from 64 tasks to 1024 tasks, the application does not stress the communication network and the communication to computation balance of the architecture is not an important metric for that code. An application like HPCCG which shows a 1.7X better performance on Red Storm for all the three cases is entirely due to Red Storm's superior balance ratio of memory access bandwidth to FLOPS (bytes-to-FLOPS). The sharp increase in the ratio at 1024 tasks for some of the applications is thought to result from poorer interconnect performance relative to the computation at the node and OS noise effects.

A. CTH:

CTH is an explicit, three-dimensional, multi-material shock hydrodynamics code developed at Sandia for serial and parallel computers. It is designed to model a large variety of two- and three-dimensional problems involving high-speed hydrodynamic flow and the dynamic deformation of solid materials, and includes several equations of state and material strength models [5]. The numerical algorithms used in CTH solve the equations of mass, momentum, and energy in an Eulerian finite difference formulation on a three-dimensional Cartesian mesh. CTH can be used in either a flat mesh mode where the faces of adjacent cells are coincident or in a mode with Automatic Mesh Refinement (AMR) where the mesh can be finer in areas of the problem where there is more activity. We will be using the code in a flat mesh mode for this study.

The shaped-charge model consists of a cylindrical container filled with high explosive and capped with a copper liner. When the explosive is detonated from the center of the back of the container, the liner collapses and forms a jet. The problem is run in quarter symmetry and includes a target material. By using the code in flat mesh mode, the communication patterns are fairly simple and fixed for the entire calculation. The problem space is a rectilinear grid of cells where each processor has a rectilinear sub grid of cells. The processors' domains are also arranged in a grid so that if two processors' domains meet at a face, they share the entire face. Quantities are exchanged at regular intervals across these faces, so each processor exchanges information with up to six other processors in the domain. These messages occur several times per time step and are fairly large since a face can consist of several thousand cells with each cell containing forty quantities. For this simulation, there are processors that

communicate with six other processors once the number of processors in the simulation reaches 128. There are also a few global communications to determine quantities such as the length of the next time step.

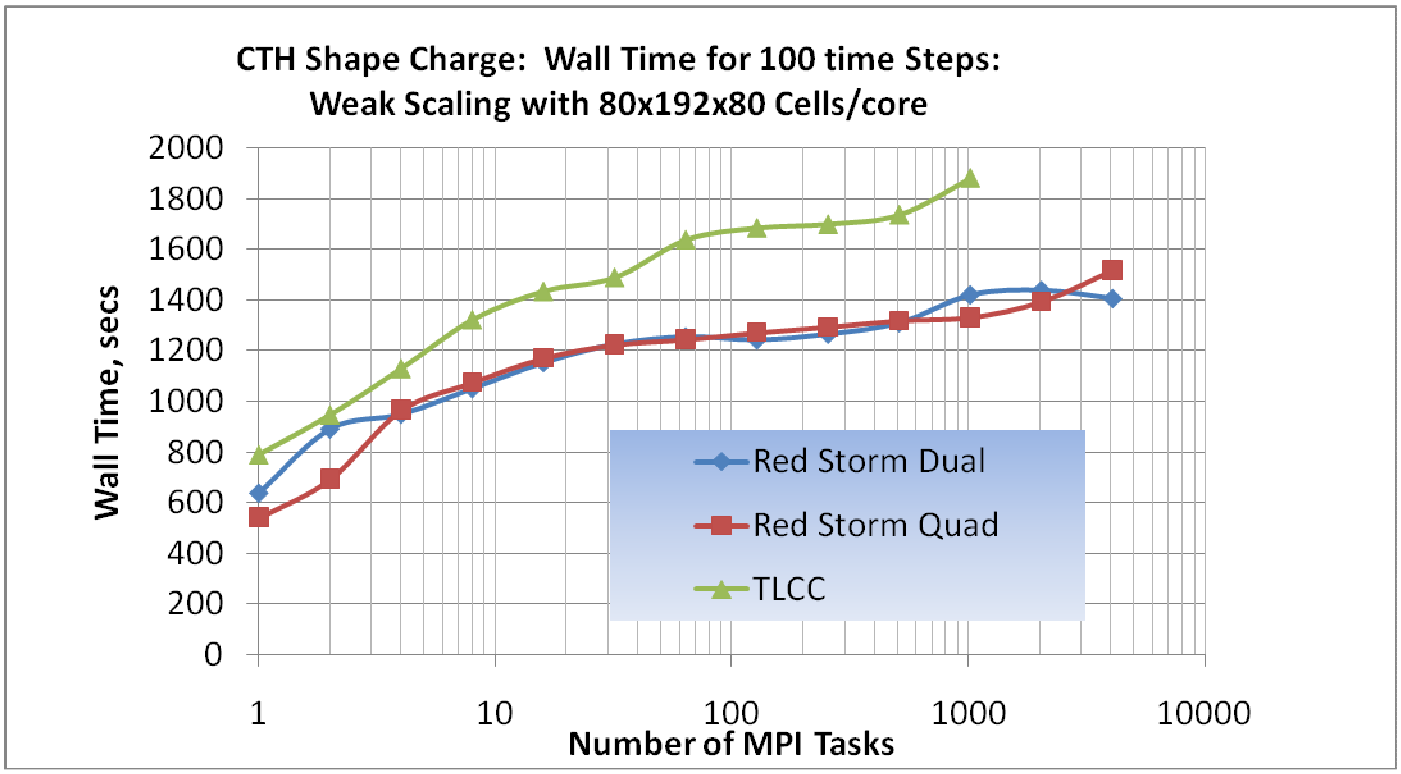


Figure 8. CTH Shape Charge weak scaling performance, wall time in seconds

Figure 8 shows the weak scaling performance with a large number (80x192x80) cells per MPI task. The TLCC performance is quite good. The Mosaic plot from the cray_pat performance analysis tool shown in Figure 9 supports the conclusion that CTH, with structured nearest-neighbor type communications and fairly large message sizes, performs quite adequately on TLCC. The data from the Bucket Brigade benchmark (BBL) shown in Table 3 provides further evidence for this conclusion. Also observable from Figure 8 is the increased run time for less than 16 MPI tasks when the application runs within a single TLCC node. This seems to be related to the additional memory latency overhead discussed in section III A. The cray_pat Time-Line plot displays one time step and shows the high ratio of computation-time to communication-time. This again brings about good scaling characteristics for CTH.

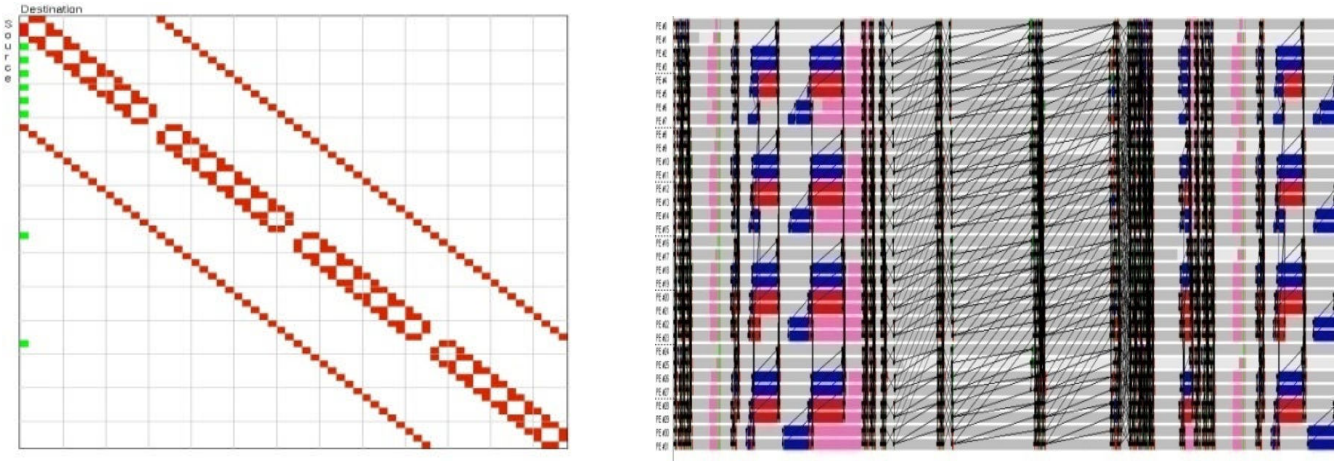


Figure 9. Cray_pat Mosaic and Time-Line plots for CTH

B. *SIERRA/Presto*

Presto is a Lagrangian, three-dimensional explicit, transient dynamics code for the analysis of solids subjected to large, suddenly applied loads [6]. 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 [7]. 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.

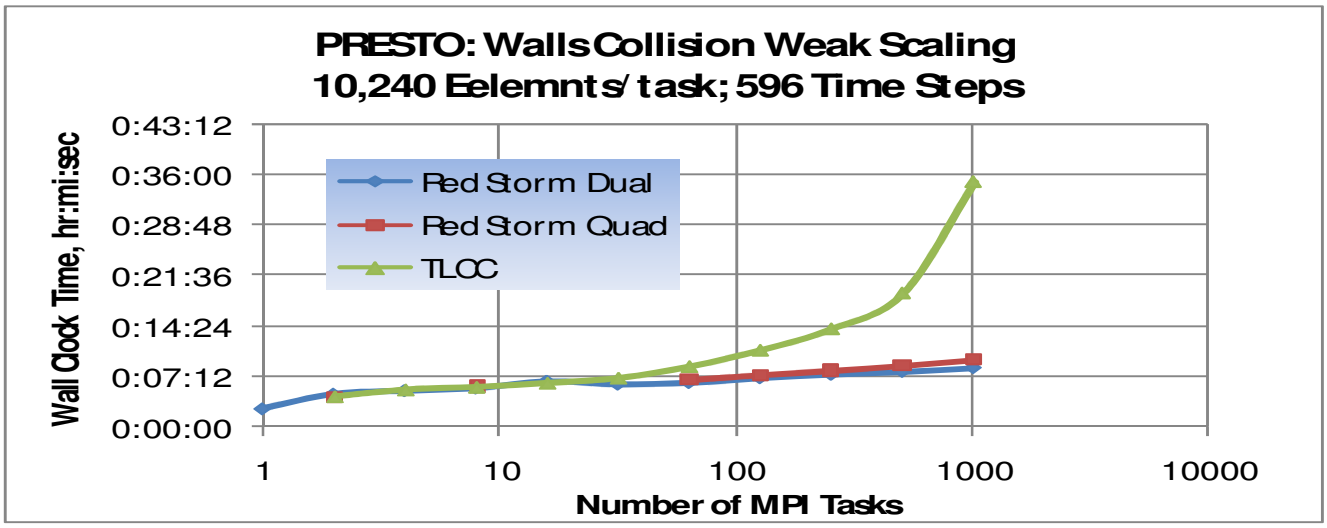


Figure 10. SIERRA/Presto Walls Collision weak scaling performance, wall time in hr:min:sec

The analysis used in this investigation is the Brick Walls problem consisting of two sets of two brick walls colliding with each other. It is a weak scaling investigation where each processor is assigned 80 bricks. Each brick is discretized with 4 x 4 x 8 elements, for a total of 10240 elements per processor. 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. Figure 10 shows the parallel performance of Presto on this problem. Since each brick is assigned to one processor, the communication for the finite element portion of the simulation is reduced to a few global communications to determine the length of the next time step. The contact portion of the calculation, however, involves communication in several phases. First, a small amount of information is communicated to allow for the calculation of the new decomposition. Then the face information for the surface elements needs to be redistributed to the new decomposition. After contact detection is performed, then a smaller amount of information representing the forces on the nodes is communicated back to the original decomposition. The resulting communication pattern is not well structured and can involve the sending of a large number of small messages to processors that may not be nearby. The rapid increase in run time on TLCC is suspected to be a consequence of the contact algorithm's sensitivity to message injection rate and demand on unstructured communication in the network. This application is most demanding among all tested on the communication infrastructure.

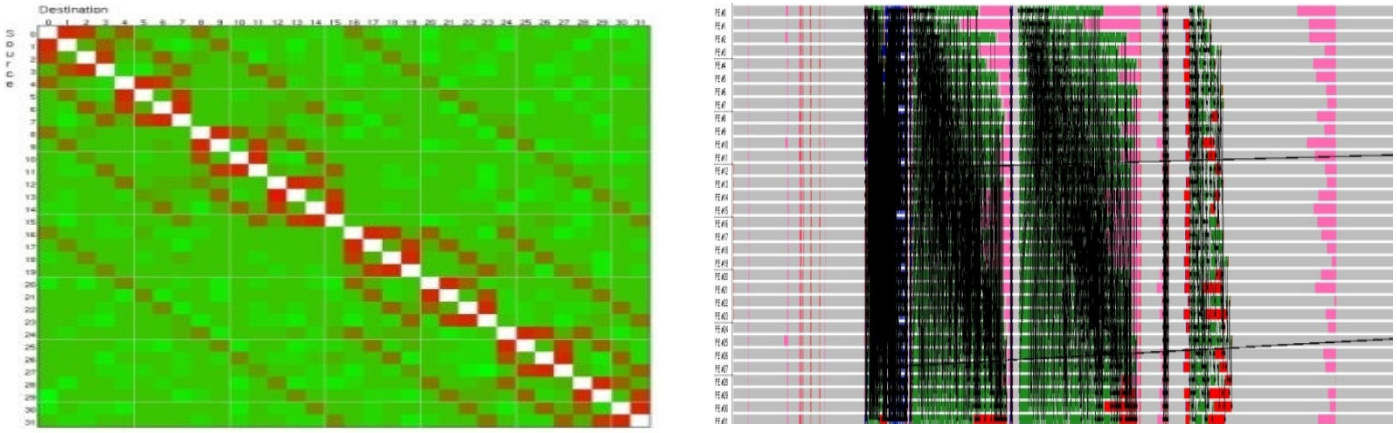


Figure 11. Cray_pat Mosaic and Time-Line plots for Pronto

The Mosaic plot from the cray_pat performance analysis tool shown in Figure 11 provides evidence of the unstructured

communication for this application obtained with Pronto. Pronto uses a similar algorithm to Presto and was easier to profile. Although there is some pattern for the larger message volumes (colored red), the fact that the entire processor grid registers message exchanges, shows the stress on the communication infrastructure. The data from the Random Message benchmark (RM) shown in Table 3 shows that TLCC could be a factor of 10 (at 64 Tasks) to a factor of close to 100 (at 1024 tasks) slower than Red Storm. The cray_pat time-line plot shows one time step and shows the low ratio of computation-time to communication-time which leads to poor scaling unless the architectures communication infrastructure is really good.

C. LAMMPS:

LAMMPS[8] is a classical molecular dynamics code that models an ensemble of particles in a liquid, solid, or gaseous state. It can model atomic, polymeric, biological, metallic, granular, and coarse-grained systems using a variety of force fields and boundary conditions. LAMMPS runs efficiently on single-processor desktop or laptop machines, but is designed for parallel computers. It will run on any parallel machine that compiles C++ and supports the MPI message-passing library. This includes distributed- or shared-memory parallel machines and Beowulf-style clusters. LAMMPS can model systems with only a few particles up to millions or billions.

The current version of LAMMPS is written in C++. In the most general sense, LAMMPS integrates Newton's equations of motion for collections of atoms, molecules, or macroscopic particles that interact via short- or long-range forces with a variety of initial and/or boundary conditions. For computational efficiency LAMMPS uses neighbor lists to keep track of nearby particles. The lists are optimized for systems with particles that are repulsive at short distances, so that the local density of particles never becomes too large. On parallel machines, LAMMPS uses spatial-decomposition techniques to partition the simulation domain into small 3d sub-domains, one of which is assigned to each processor. Processors communicate and store "ghost" atom information for atoms that border their sub-domain. The simulation used in this study is a strong scaling analysis with the RhodoSpin benchmark. The run time to compute the dynamics of the atomic fluid with 32,000 atoms for 100 time steps is measured. The execution time is shown in Figure 12. This LAMMPS benchmark is not memory intensive and does not show significant difference in performance when memory and processor affinity are forced on the TLCC. Red Storm scales well even beyond 64 tasks although the balance of computation to communication is steadily decreased for this strong scaling test.

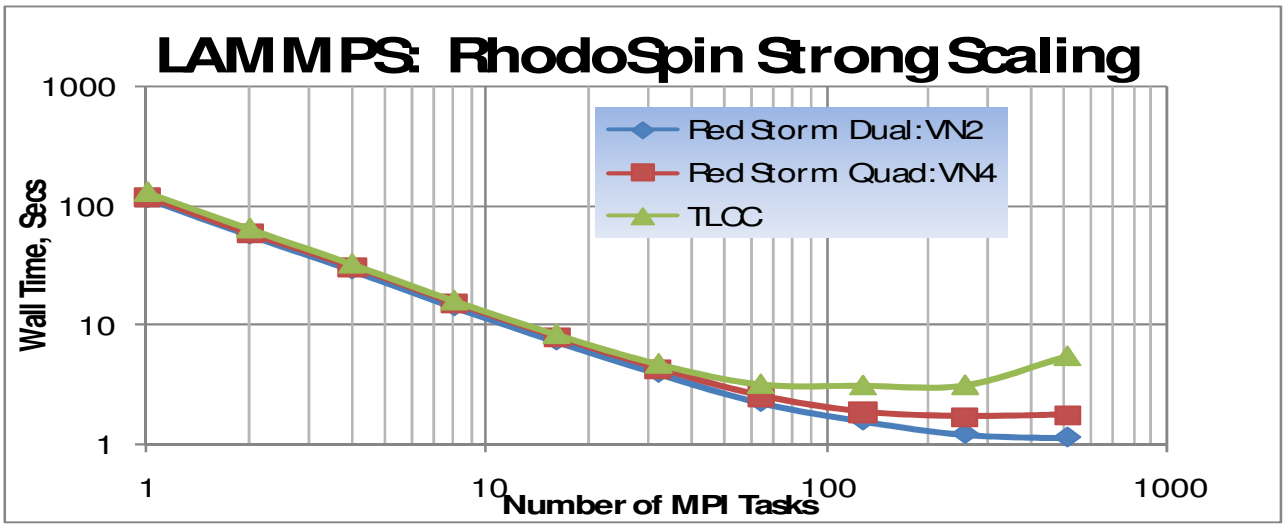


Figure 12. LAMMPS RhodoSpin Protein benchmark strong scaling performance, wall time in seconds

D. SIERRA/Fuego

This application is an integral part of the SIERRA multi-mechanics software development project at Sandia. Fuego represents the turbulent, buoyantly driven incompressible flow, heat transfer, mass transfer, combustion, soot, and absorption coefficient model portion of the simulation software. Syrinx represents the participating-media thermal radiation mechanics. Calore represents the heat transfer within an object. Domino, et.al.[9] describe the details of the governing equations, discretization, decomposition and solution procedures.

In the application chosen for this paper a Fuego simulation using a single fluids mesh for a Methanol EDC model with 2.56 million node FE mesh is used.

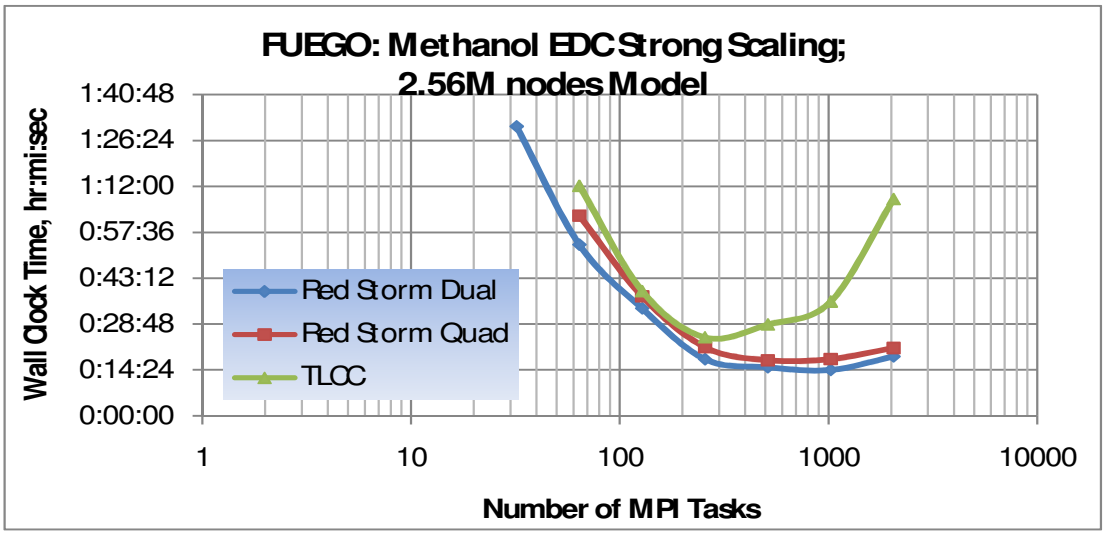


Figure 13. SIERRA/Fuego Methanol EDC model strong scaling performance, wall time in hr:min:sec

Figure 13 presents the results from a strong scaling study, showing the wall clock time for a fixed number of time steps. The most dominant computational kernel in this implicit Eulerian code is the Multi-level (ML) solver. The reason that Red Storm scales better than TLCC is because of the better communication to computation balance ratio, which is required for the implicit ML solver. As this is a strong scaling run, the work per processor decreases with increasing processor count and if the communication time grows disproportionately to the decrease in computation time, poor scaling results.

VI. Conclusion

The superior architecture of the Red Storm is evident from the variety of benchmarks and applications presented. The Red Storm's node architecture (minimizing memory contention), the interconnect architecture (maximizing bandwidth for even unstructured, random messages), and the LWK at the compute nodes (minimizing Operating System overheads) are all absolutely necessary to achieve scalability. However the data presented in this paper also supports the suitability of an architecture like TLCC for 256 or fewer MPI processor jobs. For these cases similar performance was observed on TLCC and Red Storm even when all the cores on a node were assigned an MPI task. But for some applications like presto, there is a need for architecture like the Red Storm even if capability class simulation is not the goal as the scaling results shown here demonstrate. Although the four socket node on the TLCC has cost advantages, our MPI applications show potential 2X performance penalty due to memory contention. Our study also showed that although the Red Storm quad node core has close to 2X peak FLOPS compared to the Red Storm dual node core, the run time are close because most applications could not take advantage of the four FLOPS per clock. This study hopefully points out that the inevitable cost pressures to increase core counts on a node might be detrimental in that no improvements in run time may result.

References

[1] W.J. Camp and J.L. Tomkins, "Thor'sHammer: The First Version of the Red StormMPP Architecture," Proc. Supercomputing 2002 Conf. High Performance Networking and Computing, Nov. 2002;

[2] Ron Brightwell, Trammell Hudson, Kevin Pedretti, Keith D. Underwood. "SeaStar Interconnect: Balanced Bandwidth for Scalable Performance"_. IEEE Micro, Volume 26, Number 2, May/June 2006.

[3] M. Heroux. "Mantevo Home Page," 2009; http://software.sandia.gov/mantevo.

[4] M. A. Heroux, R. A. Bartlett, V. E. Howle et al., "An overview of the Trilinos project," ACM Trans. Math. Softw., vol. 31, no. 3, pp. 397-423, 2005.

[5] Hertel, E.S. Jr., Bell, R.L., Elrick, M. G., Farnsworth, A.V., Kerley, G. I., McGlaun, J. M., Petney, S. V., Silling, S. A ., Taylor, L., Yarrington, P.A., "CTH: A Software Family for Multi-Dimensional Shock Physics Analysis,", Proceedings, 19th International Symposium on Shock Waves 1, 274ff (Université de Provence, Provence, France), 1993

[6] Koteras, Richard. and Gullerud, Arne. S., Presto User's Guide Version 1.05, JSand Report SAND2003-1089, April 2003

[7] Brown, K.H., Summers, R.M., Glass, M.W., Gullerud, A.S., Heinstein, M.W., and Jones, R.E., "ACME Algorithms for Contact in a Multiphysics Environment API Version 1.0," Sand Report SAND2001-3318, October 2001

[8] "LAMMPS Molecular Dynamics Simulator," http://lammps.sandia.gov/index.html.

[9] "SIERRA/Fuego: A Multi-Mechanics Fire Environment Simulation Tool," Domino, S. P., Moen, C. D., Burns, S. P., and Evans, G. H., AIAA Paper 2003-0149, 41st AIAA Aerospace Sciences Meeting, Reno, NV, January 2003