Investigating the balance between capacity and capability workloads across large scale computing platforms

Mahesh Rajan, Courtenay Vaughan, Robert Leland, Douglas Doerfler, Robert Benner
Sandia National Laboratories
P.O. BOX 5800, Albuquerque, NM 87185

Abstract -- The focus of this paper is on the effectiveness of HEC (high-end computing) systems on meeting engineering and scientific analysis needs. Performance measurement and analysis of the applications constituting the work load, on a large commodity InfiniBand cluster, and, on a large custom Cray XT3, is used to assess the merits of the competing HEC architectures. Those applications with communication intensive algorithms show a factor of 2 to 10 better (on 1024 processors) performance on XT3, making XT3 ideal for long, large capability simulations. However, applications with moderate to low communication need have comparable performance on the cluster and these commodity clusters eminently meet the need for higher volume capacity computing cycles. We analyze the reasons for the performance difference seen between the two systems. Since the single cpu wall clock execution time is very close between the two systems, we use parallel efficiency as a measure, to analyze optimal workload mapping on our capability and capacity computing resources.

Introduction:

Understanding the performance of scientific applications on high performance computers is important for setting resource management policies. Application performance can be influenced by the architecture of the computer, software characteristics, and characteristics introduced by the simulation being run. The same application may be used to run very large capability class simulations or used with a smaller number of processors in several runs in a capacity context to cover a range of parameter space for analysis like uncertainty quantification. In the context of current and future major investments in capacity and capability computing systems, it is useful to analyze mapping of workload against the available computing resources. Current HEC systems vary in the node/processor architecture, the inter-connect, and, system software. IDC classification of HEC systems into two broad categories [1], namely, capability and capacity, is widely used. However the demarcation is not strictly defined. Moreover applications and analysis that are targeted for these HEC systems again cross the definition boundaries. Our experience with a number of applications and analysts needs, clearly indicate need for large capacity compute cycles. At the same time capability computing often addresses need for interesting and new science that were often not undertaken previously due to lack of compute power. A broad guide line for classifying capability class simulations currently under vogue, include [2]:

1) Simulations that use a significant fraction of the total nodes installed
2) Simulations that require large memory, I/O, and storage
3) Simulations with stringent time-to-solution and short design cycle times
4) Some combination of the above analysis characteristics making it the only means of achieving the goal

In this context, both from a management concern for providing the correct investment to meet an institutions need as well as from an analyst desire to extract optimal performance, there exists a strong need to understand effectiveness of different classes of HEC systems on meeting the engineering and scientific analysis needs.

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.

Table 1 is the result of a usage survey done few years ago, listing the top few applications and node-hour percentage usage. The current fraction is based on usage logs and estimated future fraction is based on user surveys reflecting programmatic needs. The recent availability of large capability computing systems like ASC Red Storm at Sandia and ASC Purple at LLNL has enabled analysts to conceive new approaches and analysis that were if not impossible, were difficult to undertake on a routine basis. The statistics of node-hours for such large capability class simulations are just beginning to emerge. However, the question of appropriate allocation of compute cycles on capability and capacity computing systems when the demand for total node-hours exceeds available resources is an area of much interest.

In this paper we have measured application scaling characteristics so that efficiency gains of a capability class system for each application guides the selection and allocation of limited capability computing cycles. A large InifiniBand cluster with over 8000 processors and a large Cray XT3 with over 20000 processors are used to measure performance of seven applications of interest. The measured parallel efficiency on both these systems is used to understand impact of architectural balance. Parallel efficiency works as a useful measure because the single cpu performance is very close. In some cases, we used strong scaling with engineering models that do not lend to easy construction of weak scaling inputs. It is recognized that scaling behavior is data set dependent and often bigger models permit scaling to a larger number of processors. However, the performance ratio between the two systems provides broad guidelines on optimal usage of both the systems to meet capability and capacity computing node-hour demands.

Table 1. SNL application node-hour usage and projections

Code	Use	Numerical Method	Current Fraction	Future Fraction
Presto	**Crash/ Solid dynamics**	**FEM, explicit time integration**	**34.4%**	**15%**
Salinas	**Vibration/ Structural dynamics**	**FEM, spectral analysis**	**15.8%**	**10%**
LAMMPS	**Molecular dynamics**	**FFT, sparse matrix methods**	**12.8%**	**10%**
DSMC	**Plasma dynamics**	**Discrete Simulation Monte Carlo**	**10.4%**	**10%**
CTH	**Penetration/ Hydrodynamics**	**Control volume, explicit time integration**	**7.4%**	**10%**
ITS	**Radiation transport**	**Monte Carlo**	**.08%**	**15%**
SAGE	**Hydrodynamics**	**Finite Volume**	**0.0%**	**TBD**
		TOTAL	**81%**	**70%**

In the following sections we first provide a short description of each application and the analysis that was benchmarked on the two systems. The wall clock run time and parallel efficiency plots show the scaling characteristics of the applications. With T denoting wall clock run time, we define parallel efficiency at p processors as: $(T_{ref}/T_p)/(p/ref)$ for strong scaling and as $(T_{ref}/T_{p)}$ for weak scaling. Here, *ref,* denotes the minimum number of processors at which the problem fits in memory and T_{ref} may refer to a parallel implementation run time on a single

processor where appropriate. Our approach is similar to that of Oliker, et.al. [3] in so far as we investigate the performance of full applications constituting most of the workload (Salinas was omitted and SIERRA/Fuego was included in our scaling studies) shown in Table 1. The benefit of Red Storm's Light Weight kernel and fast network performance has been presented by Hoise, et.al [4] considering application performance and performance models.

Target Architecture Description:

The Red Storm machine at Sandia National Laboratories in Albuquerque, New Mexico currently consists of 12,960 dual-core nodes with a 2.4GHz Opteron CPU with 2, 3, or 4 GB of main memory and a Cray SeaStar NIC/router attached via HyperTransport. The network is a 27x20x24 mesh topology, 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 5.4 µsec measured MPI latency. The compute nodes run the Catamount lightweight kernel, a follow-on to the Cougar/Puma design used on ASCI Red. The I/O and administrative nodes run a modified version of SuSE Linux. The Cray-designed SeaStar communication processor / router is designed to off-load network communication from the main processor. It provides both send and receive DMA engines, a 500MHz PowerPC 440 processor, and 384 KB of scratch memory. Combined with the Catamount lightweight kernel, the SeaStar is capable of providing true OS-bypass communication. The Red Storm platform utilizes the Portals 3.3 communication interface, developed by Sandia National Laboratory and the University of New Mexico for enabling scalable communication in a high performance computing environment. The Portals interface provides true one-sided communication semantics. Unlike traditional one-sided interfaces, the remote memory address for an operation is determined by the target, not the origin. This allows Portals to act as a building block for high performance implementations of both one-sided semantics (Cray SHMEM) and two-sided semantics (MPI-1 send/receive). The Cray XT3 commercial offering was nearly identical to the Red Storm machine installed at Sandia, before the recent upgrade to dual core nodes and newer SeaStar NIC. The notable difference is that while the Red Storm communication topology is a 3-D mesh, the XT3 utilizes a 3-D torus configuration. The difference is to allow a significant portion of the Red Storm machine to switch between classified and unclassified operation.

The Thunderbird system was purchased for coordinated use as a production capacity computing cluster in a technical collaboration with Dell Computer Corporation (Computational nodes), with Cisco Systems (high-speed message passing interconnect), with Force10 Networking (Ethernet interconnect), and with the Technology Integration Group (vendor/integrator). Thunderbird is comprised of 4480 Dell PowerEdge 1850 commodity servers with 3.6GHz Intel EM64T dual-processors, with 6GB per node memory, linked with an InfiniBand message passing interconnect. The interconnect is a dual layer hierarchical fat tree InfiniBand network. There are 140 Compute racks, each with two 24 port InfiniBand 4x switches and 32 compute nodes. There are 6 Ethernet racks with a single Force10 E1200 switch and Eight IB racks with a single 288 port IB 4x switch. All MPI traffic is conducted across the InfiniBand network and all I/O is done across the Ethernet network. Each 24 port IB switch has 16 compute nodes connected to it and a single connection to each of the eight 288 port IB switches producing a 2-to-1 over subscription. There is a core E1200 switch that is connected via 4 channel bonded 10GigE ports to the remaining 5 E1200s. 4 of the 5 lower level ethernet switches have 1024 compute nodes connecting at half GigE bandwidth and the remaining switch has 384 compute nodes also at half bandwidth. Thunderbird's software was recently upgraded to

OpenFabric Enterprise Distribution (OFED) and OpenMPI - Linux-based open source software stack qualified by the OpenFabrics Alliance to operate with multi-vendor InfiniBand hardware and implement open source Message Passing Interface (MPI) protocol. Table 2 summarizes the important architectural characteristics of Red Storm and Thunderbird. All the applications were compiled on both systems with PGI 6.2.3, except for the SIERRA/Fuego for which the Intel compiler 9.1 was used on Thunderbird.

Table 2. Red Storm and Thunderbird architectural highlights

Name	Arch	Network Topology	Total P	P/ Node	Clock (GHz)	Peak (GF/s/P)	Streams BW(GB/s/P)	MPI Lat (µsec)	MPI BW (GB/s/P)
Red Storm	**AMD Opteron**	**Mesh / Z-torus**	**25,920**	**2**	**2.4**	**4.8**	**2.5**	**5.4**	**2.1**
Thunderbird	**Intel EM64T**	**Fat tree**	**8960**	**2**	**3.6**	**7.2**	**3.8**	**6**	**0.468**

Applications and Benchmarks:

a) SIERRA/Fuego:

This application is an integral part of the SIERRA [5] 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.[6] describe the details of the governing equations, discretization, decomposition and solution procedures. The general coupling strategy for the suite of abnormal-thermal environments is provided in Figure 1. SIERRA/Fuego, SIERRA/Syrinx, SIERRA/Calore depend heavily on the core architecture developments provided by SIERRA for massively parallel computing, solution adaptivity, and mechanics coupling on unstructured grids.

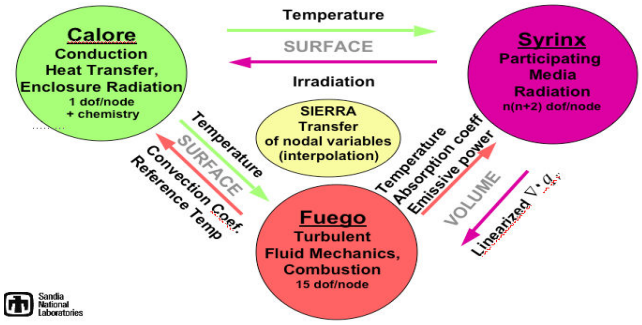

Figure 1. Abnormal-thermal coupling analysis with SIERRA/Fuego

In the application chosen for this paper, coupled fire/thermal response predictions for a weapon-like calorimeter is validated for a quiescent fire representative of a transportation accident scenario. The model constructed was used to compare numerical predictions against experimental data. Temperature measurements were used to validate the coupled Fuego/Syrinx/Calore predictions. The model consists of fluids (Fuego), radiation (Syrinx) and object heat transfer (Calore) meshes along with an output mesh. The main Fuego fluid mesh for the scaling study was constructed with a 1M element model fluid mesh. Similar mesh sizes were used in the Syrinx radiation calculations. The Calore mesh size is much smaller as it contains

only the outer shell of the object. The output mesh is a vertical slice through the centerline of the fire that is only one cell thick. The simulations solve the governing set of complex coupled equations whose solution over a broad range of time and length scales is sought. This complexity in the model and the long run times to resolve the fire for 60-90 seconds could only be carried out on massively-parallel capability class supercomputers. Figure 2 presents side-by-side the execution time plot and the parallel efficiency plot. The most dominant computation, namely the fluid region solve is plotted. The reason that Red Storm scales better at 256 and 512 processor counts is because of the better communication to computation balance, that is required for the implicit ML solver used for the fluid solve. As this is a strong scaling run, the work per processor decreases and therefore it stresses the communication fabric over the several iterations required for the implicit solution.

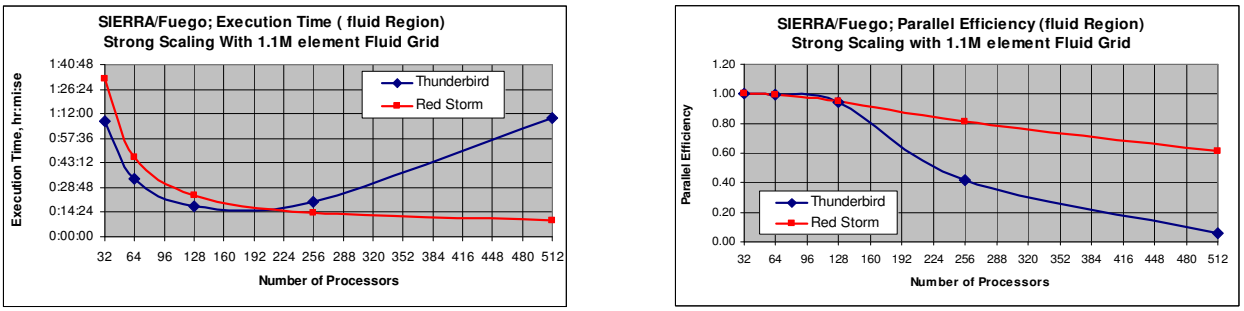

Figure 2. SIERRA/Fuego Performance on Red Storm and Thunderbird

b) ITS Monte Carlo radiation transport:

The INTEGRATED TIGER SERIES (ITS) code is an evolving Monte Carlo radiation transport code that has been used extensively in weapon-effect simulator design and analysis, radiation dosimetry, radiation effect studies and medical physics research. Many individuals from the DOE labs and NIST have been involved over the years in the development and enhancement of ITS. Physical rigor for the analysis 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. The ITS code is capable of analyzing particle transport through both combinatorial geometry models and CAD models. It also has been significantly enhanced to permit adjoint transport calculations.

For the purposes of this paper we have analyzed the performance using as input, data from a real satellite model. The physical problem solved takes advantage of the MITS mutli-group/continuous energy electron-photon Monte Carlo transport code's capability to address realistic three-dimensional adjoint computations. The run times for simulations for a complex combinatorial geometry model using conventional, or forward, transport are prohibitive and hence the adjoint calculations used in our satellite model. Figure 3 presents side-by-side the execution time plot and the parallel efficiency plot for ITS. The weak scaling runs were set up with 1.6 Million histories per processor. The difference in parallel efficiency for this application can be directly related to the MPI bandwidth, as we have developed a performance model [7] that easily explains the increased overhead for the master/slave communications at the end of each batch of history computations. As noted in Ref. [7] the algorithm for gathering the statistics after each batch has been modified in newer version of ITS to improve parallel scaling even on systems with lower communication performance. However, for this exercise we present the results from the older algorithm as it exaggerates the difference between Thunderbird and Red

Storm, helping to point out the importance of communication bandwidth on scalability that may not be apparent when using a few hundred processors. For Megabyte size messages that are sent from the slave processor to the master processor in a serial fashion, the factor of 3X better bandwidth on Red Storm explains the differences in parallel efficiency observed.

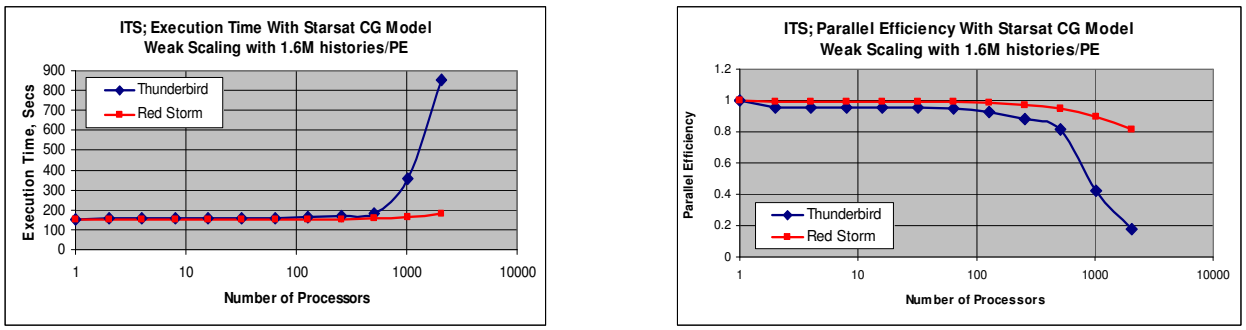

Figure 3. ITS Performance on Red Storm and Thunderbird

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 weak scaling analysis with the Lennard-Jones liquid benchmark. The dynamics of the atomic fluid with 864,000 atoms per processor for 100 time steps is timed. The execution time and parallel efficiency is shown in Figure 4.

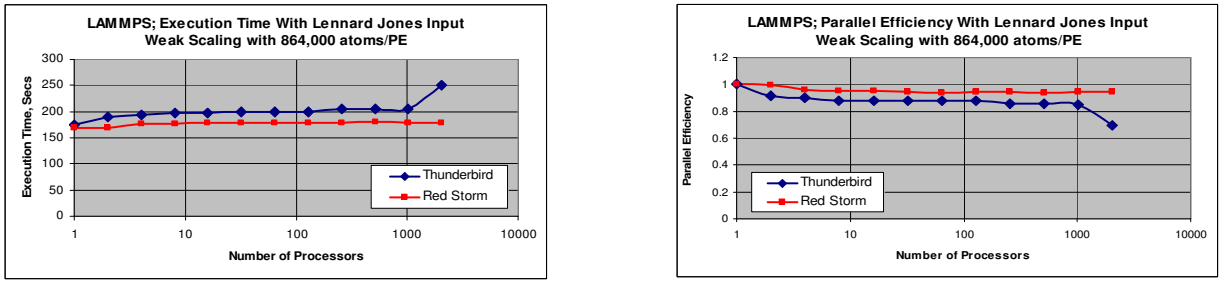

Figure 4. LAMMPS Performance on Red Storm and Thunderbird

The reason that Red Storm and Thunderbird show similar performance is because LAMMPS has a high computation to communication time ratio and the message exchanges are implemented very efficiently. LAMMPS divides the computational three dimensional space into three dimensional sub-volumes, and makes the sub-volumes as cubic as possible. The amount of data exchanged is proportional to the surface area of the sub-volume. This favorable volume to surface ratio leads to less than 3% MPI overhead for all the processor counts as shown in Table 3 for this weak scaling analysis. This Red Storm data was obtained using CrayPat instrumentation tool. The data for Thunderbird would be similar and is not shown as there is insignificant difference in performance between the two systems except at 2048 processors.

Table 3. LAMMPS Red Storm MPI overhead

Num PEs	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**	**2048**
% time in MPI	**0.4**	**1.5**	**0.9**	**1.5**	**2.1**	**1.5**	**2.1**	**1.8**	**2**	**2.4**

d) SIERRA/Presto:

Presto is a Lagrangian, three-dimensional explicit, transient dynamics code for the analysis of solids subjected to large, suddenly applied loads [9]. 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 [10]. 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 analysis used in this investigation is the Brick Walls problem consists 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 timestep. As the problem grows with the number of processors, the contact problem also grows. Figure 5 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 rather rapid increase in run time after 256 processors on Thunderbird is suspected to be a consequence of the contact algorithm's sensitivity to latency and due to the increase in the maximum latency with processor count as discussed by Leininger and Seager [11].

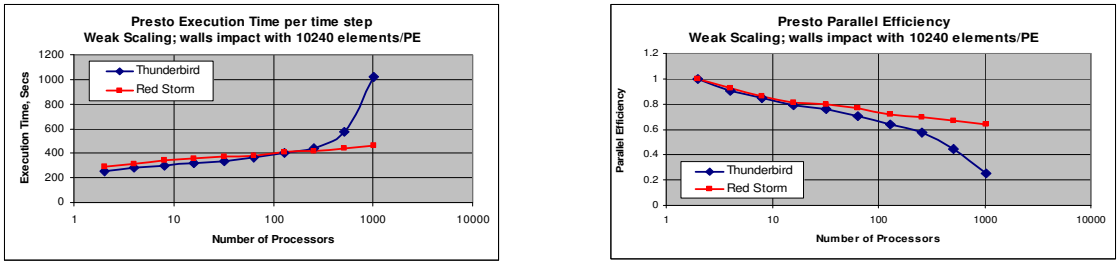

Figure 5. SIERRA/Presto Performance on Red Storm and Thunderbird

e) SAGE:

SAGE is a LANL/SAIC multi-dimensional multi-material Eulerian hydrodynamics code with adaptive mesh refinement. The code uses second order accurate numerical techniques. SAGE was tested extensively on Red Storm with simple inputs and complex asteroid impact input decks in the early days of bringing up Red Storm. SAGE performance has been studied extensively by Kerbyson, et.al.,[12] and is frequently used by LANL to predict performance of new HPC architectures, using their application performance model[4]. We have used SAGE (version 20030505) to investigate scaling characteristics of Thunderbird and Red Storm. The code was executed in a weak-scaling mode with a constant sub-grid per processor, thereby increasing the global problem with increasing processor count. The input deck used is called timing_c and the problem was set up with approximately 80,000 cells per process, and it performs only hydro calculations. This input deck imposes a high communication time to computation time ratio. Figure 6 shows the wall time and parallel efficiency with this input deck. The parallel efficiency is calculated using the 2 processor timing as the reference.

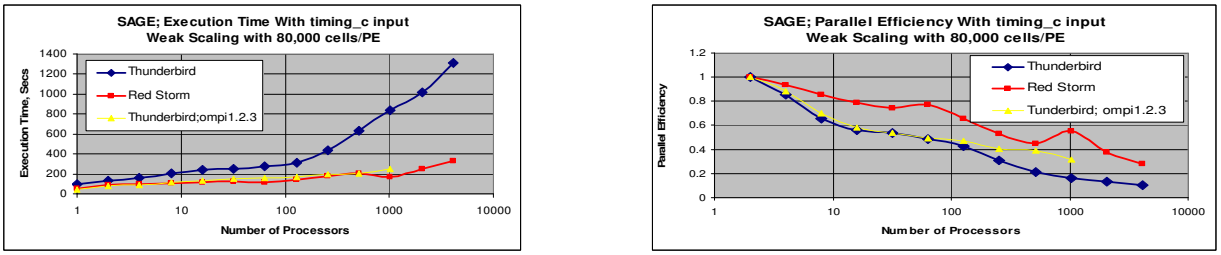

Figure 6. SAGE Performance on Red Storm and Thunderbird

MPI profiles of the runs at 64, 256, and 1024 processors appear in Table 4, comparing the MPI overhead. From Kerbyson's [10] performance model we know that the communication is dominated by gather/scatter operations particularly in the z-direction exchanging boundary cell information and also by hundreds of MPI_Allreduce operations 4 bytes long at each time step.
The large increase in MPI_Allreduce time in Thunderbird was associated with the release 1.1.2 of OpenMPI used in most of our performance measurements. This was confirmed by allreduce timings shown in Figure 10 in the last section on performance scaling analysis. The linear increase in allreduce time with 1.1.2 is corrected in the later release 1.2.3. The allreduce curve for Thunderbird measured more recently with 1.2.3, in fact follows very close to the performance curve for Red Storm showing a logarithmic growth with number of processors. On observing this improvement the sage performance was measured again using the OpenMPI 1.2.3 release and clearly the execution time is significantly better and the trend is quite similar to Red Storm as shown in Figure 6. The parallel efficiency plot still shows a 10% to 20% performance

advantage for Red Storm.

Table 4. Sage MPI Profile comparison

Num Procs	% Total MPI time; Red Storm	% MPI Time in gather/scatter; Red Storm	% MPI time in Allreduce; Red Storm	% Total MPI time; Thunderbird	% MPI Time in gather/scatter; Thunderbird	% MPI time in Allreduce; Thunderbird
64	27.7	59	27.4	45.79	64.36	28.43
256	33.1	46.7	32.1	64.55	37.77	48.27
1024	40.4	57.2	30.7	82.10	30.2	55.5

f) ICARUS/DSMC:

The Direct Simulation Monte Carlo (DSMC) method is the only proven method for simulating noncontinuum gas flows because continuum methods break down where particles move in ballistic trajectories with mean free path larger than cell dimensions, often because the device is small (micro-or nano technology) or the fluid is very low pressure as in plasma or upper atmosphere. Unlike most flow-simulation methods, DSMC uses computational molecules ("simulators") that mimic real molecules by moving through space, reflecting from solid boundaries, and colliding with one another. By sampling the velocities of large numbers of computational molecules, the gas flow is determined.

Since DSMC is a Monte Carlo technique using computational molecules, the phases of computation corresponding to movement, reflection and collision of the molecules parallelizes easily. However, based on the density distribution and the decomposition of the particle grid, between stages of computations, there could be significant messaging overhead as particles migrate among the cells. Unsteady DSMC simulations for a two-dimensional microbeam investigated by Gallis and Torczynski [13] is used to set up a weak scaling study, fixing the number of simulators per processor. Figure 7 shows the wall clock time for thousand time steps and the corresponding parallel efficiency.

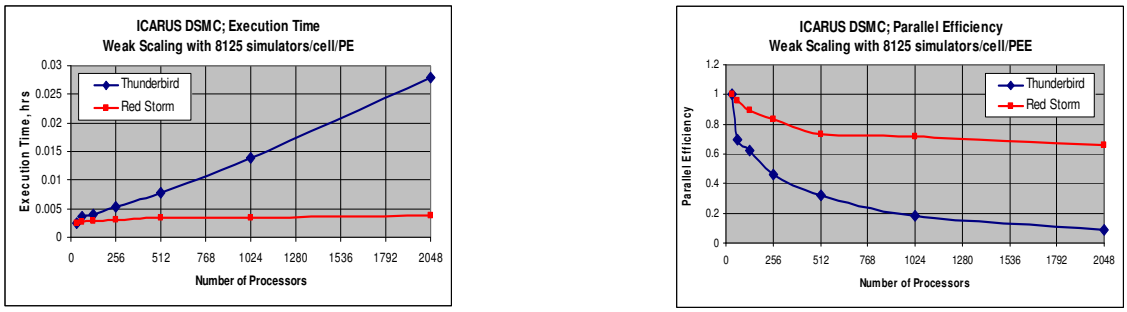

Figure 7. DSMC/ICARUS Performance on Red Storm and Thunderbird

Towards understanding the performance seen in Figure 7, the MPI overhead of runs at 64, 256, and 1024 processors is shown in Table 5. The major computational stages at each time step are: a) create particles, b) move particles, c) communicate particles that have moved to cell

owned by another processor, d) compute electron / particle chemistry, e) compute monte-carlo collisions, f) solve EM field, and h) output cell, surface data at requested frequency. Depending on the problem, some of the stages such as the electromagenetic field solve in this microdevices example are not invoked. Outside the key computational loop are data input and results output whose computational overhead is negligible in comparison to the cost of resolving the flow over typical thousands of time steps. The principal communication operation at each computational stage is the communication of the simulator/particle (position, velocity, etc.) information to the target processor that now has these new particles within its cells.

Table 5. ICARUS MPI Profile comparison

Num Procs	% Total time in MPI; Red Storm	% Total time in MPI; Thunderbird
64	14.6	37.9
256	26.6	56.0
1024	31.0	75.6

This is implemented in the code using an MPI_Reduce_scatter call that sets up the send/receive pairs between processors for all the particles that need to be exchanged. This is a global synchronous operation and the time registered under MPI profile for this operation dominates the communication time. However looking at the details of the profile shows that load imbalance in the move phase impacts this global operation as it does not begin to do the actual reduce_scatter till last slowest processor has completed its move phase. Because of the physical geometry of the MEMS device and variations in the number of simulators per cell this load imbalance has a significant effect on parallel performance. This is evident from the MPI time increasing for both Red Storm and Thunderbird in Table 5. However the compounding effect slower message transfer, slower global operation in OpenMPI 1.1.2, together with influence of operating system noise interference in global operations [15] results in the much higher MPI overhead observed for thunderbird. Work is in progress to understand quantitatively the impact of each of these and improve the performance on Thunderbird.

g) CTH:

CTH is an explicit, three-dimensional, multimaterial shock hydrodynamics code which has been 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 [14]. 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 consists of a cylindrical container filled with high explosive 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. The weak scaling analysis with CTH was setup with 90x216x90 computational cells per processor. Figure 8 shows the wall clock time per time step and the corresponding parallel efficiency.

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 timestep and are fairly large since a face can consist of several thousand cells which have forty quantities in this simulation which are exchanged. 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 timestep.

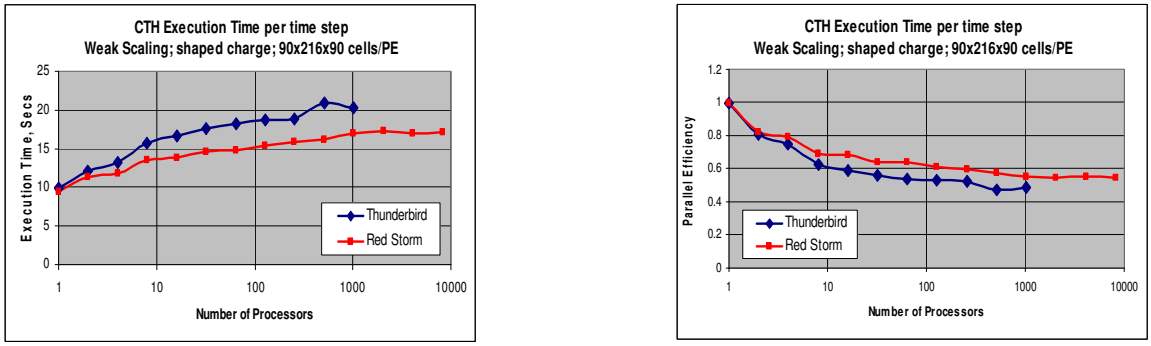

Figure 8. CTH Performance on Red Storm and Thunderbird

Workload Performance Scaling Analysis:

As stated in the introduction we have analyzed the performance of various applications that constitute our workload with a view to understanding why we see differences in performance between the two compute systems considered. In this section we present a simple analysis of the efficacy of the two systems against the work load. It is recognized that this analysis may not correctly represent the current and future workload that would be undertaken in these two systems. Application scaling behavior is strongly dependent on the amount of computation assigned per processor, which in turn is a function of the model size (or such similar parameter) that influences the compute time to communication time balance. However, we hope to understand through this analysis, computer architectural balance issues that have big impact on matching the workload to the system. One thing that we observed is despite Thunderbird's Intel processors having a clock speed 50% larger than Red Storm's AMD processors, one processor performance is very similar between these two machines. No easy explanation is perhaps appropriate without further instrumentation and analysis considering the widely varying nature of these applications, other than the observation that the better bandwidth of the Opteron seems to compensate for the lower clock speed. The first obvious conclusion that can be drawn from these application performance charts is that for many of our usual analysis needs that fall in 64 to 256 processor range, the performance of the capacity cluster is good. This is further evident from the efficiency ratio between Red Storm and Thunderbird at a few discrete processor

configurations listed in Table 6.

Table 6. Efficiency ratio, Red storm to Thunderbird

Apps.\PEs	64	256	1024
ITS	1.048	1.101	2.121
SAGE	1.590	1.692	3.413
Fuego	0.999	1.933	10.133
DSMC	1.385	1.800	3.943
LAMMPS	1.074	1.109	1.108
CTH	1.183	1.135	1.136
Presto	1.091	1.214	2.563

To analyze this further it is instructive to use a simple model of parallel efficiency as, $E = 1 / (1 + f)$, Where f is the ratio of communication time to compute time. One way to investigate the impact of the parameter, f, is to plot parallel efficiency as function of communication load to computation load.

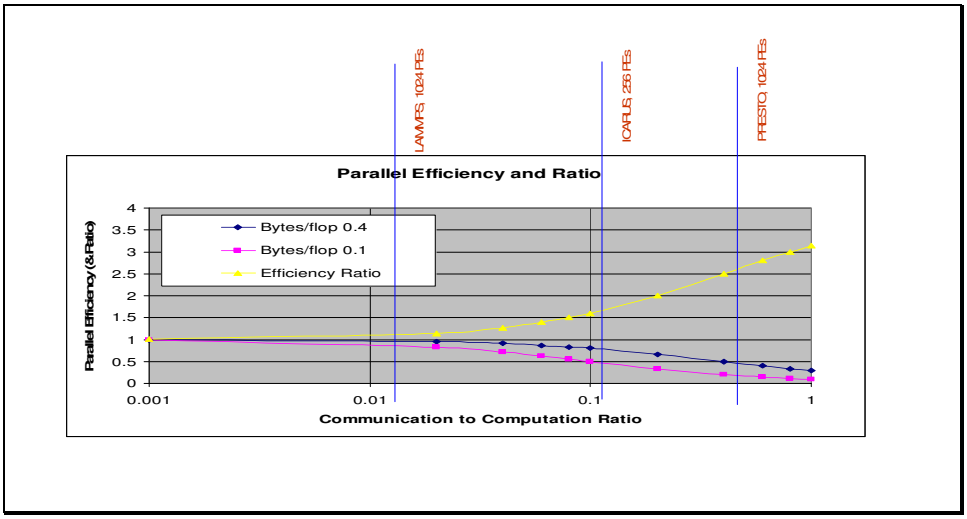

Figure 9. Simple parallel efficiency model and impact of communication to computation ratio of different applications

When this ratio is multiplied by the key platform balance characteristic, Bytes/Flop, a plot such as shown in Figure 9 may be constructed. In this figure possible approximate Bytes/Flop balance ratio of 0.4 and 0.1 is taken to represent Red Storm and Thunderbird, respectively. The ratios result from using a measured MPI ping-pong bandwidth of 1.9 GB/s for Red Storm and 700 MB/s for Thunderbird, (see Figure 10 below), while using their peak flop rate from Table 2. Also shown in the plot is the efficiency ratio between these two cases. This chart in conjunction with the Table 6 above and knowledge of the application and associated algorithms sheds much light on the impact of balance on scalability. An application like

LAMMPS with a couple of percent MPI overhead yields similar performance, while for ICARUS/DSMC with 20-30% MPI overhead we begin to observe factor of two efficiency ratio.

Another probable cause for the lower parallel efficiency of Thunderbird is the cost of global operations as typified by the allreduce time shown in Figure 10. At the time of writing this paper, the almost order of magnitude increase (after 128 processors) in time for an eight byte allreduce on Thunderbird when compared to Red Storm, is suspected to result from non-optimized global operations after the recent upgrade to OpenMPI 1.1.2. But it is certainly a major source for the poor efficiency in an application like ICARUS requiring global operations between fine grained particle movement computations.

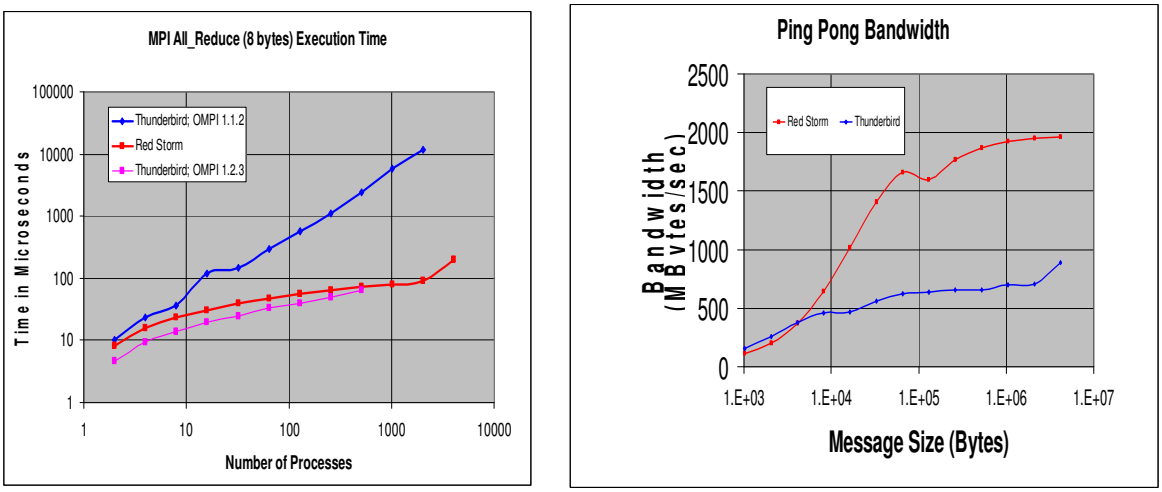

Figure 10. MPI Allreduce and ping-pong performance comparison

Our measurements on Thunderbird showed up to 30% variation in run times, whereas variations on Red Storm were less than 2-3%. For the results presented in this paper, we used the best times that we observed. Thunderbird run time variations were observed in as few as 64 processor jobs. The cause for the variation is suspected to be OS noise sources, similar to the observations by other investigators [14], although job placement on the mesh leading to network contention is also likely to play a part. These jobs were run while the machines were in dedicated mode, so we did not have interference from other jobs being on the machines. As simple test, parallel independent computations for 100 seconds (a matmul loop was used) on 100 processors shows a maximum variation of 0.4% on Red Storm while variations on Thunderbird were as high as 2.5%. Since no network activity is involved this variation is suspected to be caused by OS interrupts. A similar simple test, to measure impact of variations in communication operations was constructed by 50 pairs of nodes exchanging 2GB messages for a nominal total run time of 100 seconds. Red Storm tests showed a maximum difference in time of 3% between any pair of nodes, while Thunderbird tests showed maximum difference of 42% in the run time between any pair of nodes. This implies that applications that spend significant fraction of their compute cycle time in messaging are likely to see degraded performance, especially if there are frequent global operations or barriers requiring all the processors to synch up.

If we take the work load percentages as defined in Table 1 and construct a weighted

efficiency ratio between Red Storm and Thunderbird using the parallel efficiency charts that have been identified for the applications constituting most of the work load, a chart as shown in Figure 11 emerges. It provides a broad picture of the efficiency benefits a system like Red Storm affords on account of its better architectural balance albeit at higher investment costs with a set of measured or estimated workload. We used the best data we had for comparing Red Storm to Thunderbird with the weights shown on the caption in Figure 11. Interestingly a similar analysis conducted in the context of the JASONs review [16] using parallel efficiency ratio between ASCI-RED and Sandia's Cplant, benchmarking the applications constituting the workload, lead to a similar conclusion. Such a chart may be used in a management context to gauge the return on investment between a capability and capacity system. On the other hand, as noted in the introduction, the justification for large capability system may transcend monetary ROI considerations.

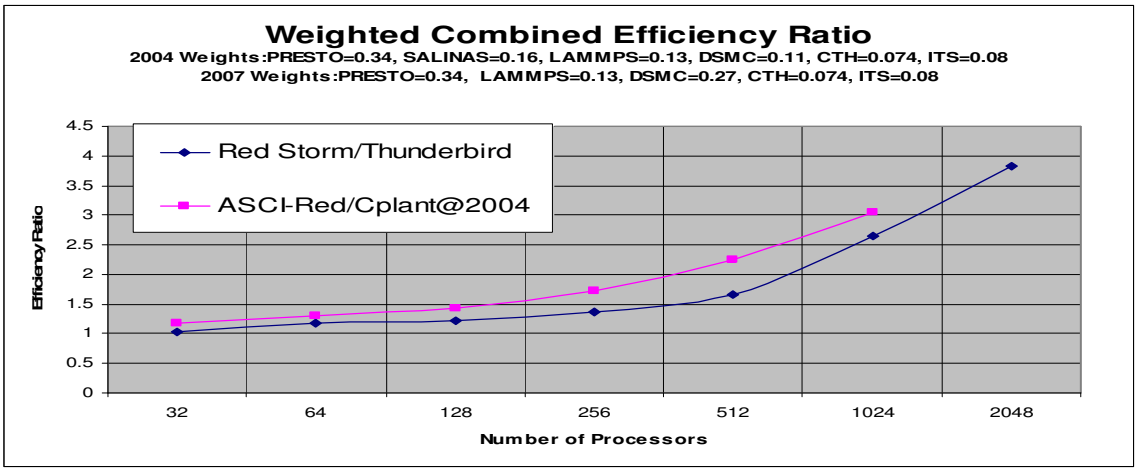

Figure 11. Workload percentage weighted parallel efficiency ratio for Red Storm/Thunderbird and ASCI-Red/Cplant

Conclusions:

From performance analysis of application workload encompassing several applications to thousands of processors, we have measured parallel efficiency ratio between a tightly integrated HEC system, Red Storm, and a large InfiniBand cluster, Thunderbird. Applications whose communication time to computation time ratio grows as a consequence of the inherent algorithm or as a consequence of poorer bytes/flop ratio at large processor counts, lead to less than desired parallel efficiency. Such applications reveal a factor of 2 to 10 better performance on a tightly integrated HEC system like Red Storm. This analysis also investigates the non-linear increase observed bytes/flop ratio on commodity clusters and postulates that OS noise and/or network contention and/or lack of maturity of the interconnect network software layers may be source of the differences seen between Red Storm and Thunderbird. While this analysis exposes the symptoms, further work remains in finding its root cause and remedying the deficiencies. Peak bytes to flop ratio between the two systems is quite reasonable, but does not explain the differences in parallel efficiencies at large processor counts.

References:

1) "2004 Technical Computing Market Results by New Application Workload Segments," IDC Report #IDC00735, November 2005
2) "A Platform Strategy for the Advanced Simulation and Computing Program," Meisner, R., NA-ASC-113R-07-Vol.1-Rev.0, SAND 2007-4343P, August 2007
3) "Scientific Application Performance on Candidate PetaScale Platforms," Oliker, L., et.al., IPDPS, March 24-30, 2007, Long Beach, CA
4) "A Performance Comparison through Benchmarking and Modeling of Three Leading Supercomputers: Blue Gene/L, Red Storm, and Purple." Hoisie, A., et. al., Proceedings of SC06, Tampa, FL, November 2006
5) "SIERRA: A Software Environment for Developing Complex Multi-Physics Applications," Edwards, H.C., and Stewart, J.R., First MIT Conference on Computational Fluid and Solid Mechanics, Bathe, K.J., editor, Elsevier Scientific, 2001
6) "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
7) "Performance Analysis, Modeling, and Enhancement of Sandia's Integrated TIGER series (ITS) Coupled Electron/Photon Monte Carlo Transport Code," Rajan, M., et.al., LACSI Symposium, Santa Fe, NM Oct. 11-13, 2005
8) http://lammps.sandia.gov
9) *Presto User's Guide Version 1.05*, J. Koteras, Richard. and Gullerud, Arne. S., Sand Report SAND2003-1089, April 2003
10) "ACME Algorithms for Contact in a Multiphysics Environment API Version 1.0," Brown, K.H., Summers, R.M., Glass, M.W., Gullerud, A.S., Heinstein, M.W., and Jones, R.E., Sand Report SAND2001-3318, October 2001
11) "To Adapt, or Not to Adapt, That is THE Question," Leininger, M., and Seager, M., Open Fabrics Developers Workshop, Sonoma, CA, April 2007.
12) "Predictive Performance and Scalability Modeling of a Large-Scale Application," Kerbyson, D.J., et.al., SC 2001, , Denver, CO; ACM 1-58113-293-X/01/0011, November 2001
13) " An improved Reynolds-Equation Model for Gas Damping of Microbeam Motion," Gallis, M.A., and Torczynski, J.R., Journal of Microelectromechanical Systems, Vol. 13, No. 4, August 2004
14) "CTH: A Software Family for Multi-Dimensional Shock Physics Analysis," 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., Proceedings, 19th International Symposium on Shock Waves 1, 274ff (Université de Provence, Provence, France), 1993
15) "The Case of the Missing Supercomputer Performance: Achieving Optimal performance on the 8,192 Processors of ASCI Q," Petrini, F., Kerbyson, D., and Pakin, S., IEEE/ACM SC2003, November 2003
16) "Effectiveness of Platforms on Engineering Codes," Leland, R., JASONS Review Report, Sandia National Laboratory OUO document, September 2004.