Experiences with the use of CrayPat in Performance Analysis

Mahesh Rajan
Sandia National Laboratories
P.O. BOX 5800, Albuquerque, NM 87185

Abstract -- *Performance analysis, tuning, and, modeling, of applications running on thousands of processors on the Sandia Red Storm/XT3 is facilitated by the use of CrayPat tool. This investigation describes the successful use of the tool with a variety of applications and also discuses some of the challenges encountered in its use. Performance data is compared against other tools and measurements on other HPC systems to fully understand serial bottlenecks and parallel scaling limitations.*

Introduction:

The Red Storm/Cray XT3 at Sandia National Laboratories (SNL) is one of the fastest computers in the world. It is used to support High-End Computing (HEC) needs at the DOE national labs. HEC systems like the Red Storm are specifically designed to enable large capability class simulations running on thousands of processors. Parallel performance of applications is influenced by a number of hardware and software characteristics. Applications may also vary a great deal in their algorithmic characteristics and in the nature of their use by the analysts. As large capability simulations on Red Storm evolve to consume most of the compute cycles, it becomes imperative that we understand the scaling and performance characteristics of all the applications targeted for this system. Understanding of application performance is greatly facilitated by tools such as CrayPat [1].

Our need and use of the tool may be broadly classified into three categories; scaling analysis, performance tuning and performance modeling. In scaling analysis we wish to understand application scaling data obtained from a weak or strong scaling runs and relate it to fraction of execution time spent in MPI. Deeper understanding of different MPI call overheads is useful to identify scaling limitations and opportunities for performance improvement. Applications that have independent parallel computations followed by some global MPI operations may sometimes exhibit non-linear growth in the fraction of MPI time with increasing processor counts. This increase in MPI time, attributed by the tool to result from global MPI operation, may indeed be caused by load imbalance in the preceding compute phase. Performance tuning consists of improving serial performance and I/O and communication overhead minimization. For serial performance, CrayPat and Apprentice2 [1], provide a wealth of information, particularly when used in conjunction with PAPI for obtaining hardware counter data. I/O overheads can also be obtained from CrayPat by using appropriate switches at the instrumentation step. For performance modeling, our focus is mostly on understanding the communication time model as this has the biggest impact on scalability. Communication time model is typically obtained from a thorough understanding of the communication pattern and message sizes [3]. Often times performance modelers are not necessarily the same analysts who have developed the application code. While the modelers may have some idea of the MPI communications from the characteristics of the application and the algorithms, the details needed for a communication model are often extracted from the use of a tool such as CrayPat that have a tracing capability.

In the following sections we first provide a short description of four applications, followed by CrayPat / Apprentice2 usage for our analysis. We provide wall clock run time and parallel efficiency plots on Red Storm and other HEC systems to facilitate discussion of the application characteristics and the impact of architectural balance on parallel performance. For each application we have also identified some of the challenges encountered in use of the CrayPat / Apprentice2 tools. Alternate approaches to overcome these limitations and to generate useful information, is discussed. We also identify opportunities for performance improvement that the use of the tool has revealed. Most of this work was accomplished in the

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. Cray User Group Meeting; Seattle, WA; May 7-10, 2007

last two week of April 2007 following a hands-on tutorial by Cray held on site at Sandia National Labs on April 16-18, 2007.

Red Storm 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 a minimum 2 GB of main memory and a Cray SeaStar NIC/router attached via HyperTransport. The network is a 27x20x24 mesh topology, with 2.0 GB/s bidirectional link bandwidth and 1.5 GB/s bidirectional node bandwidth. 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 of-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. Key architectural highlights are listed in Table 1.

Table 1. Red Storm architectural highlights

Name	Arch	Network	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	Opteron	Custom	Mesh / Z-torus	25,920	2	2.4	4.8	2.5	5.4	2.1

Applications, CrayPat Results and Analysis:

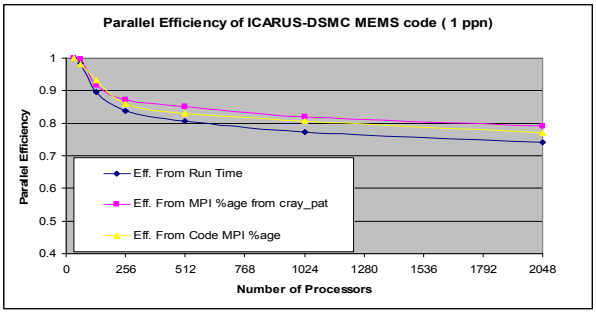
a) ICARUS/DSMC:

The first application we discuss is ICARUS/DSMC. It is frequently used at SNL for neutron generator design and for MEMS design. 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. In addition based on the analyst request to periodically dump particle, surface, and chemistry states at desired intervals, I/O overheads can impact scalability in large parallel simulations.

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. Cray User Group Meeting; Seattle, WA; May 7-10, 2007

Unsteady DSMC simulations for a two-dimensional microbeam investigated by Gallis and Torczynski [4] is used to set up a weak scaling study, fixing the number of simulators per processor. 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, f) compute monte-carlo collisions, g) solve EM field, h) output cell, surf data at requested frequency. Depending on the problem, some of the stages, such as the electromagenetic-field-solve, 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 thousands of time steps.



	32	64	128	256	512	1024	2048
%MPI time from Cray_pat	25.80%	25.90%	32.00%	35.30%	36.80%	39.20%	41.30%
%MPI Time from Code output	14.57%	16.13%	20.41%	26.52%	29.13%	31.03%	34.15%

Figure 1. Parallel Efficiency and MPI Overhead for ICARUS/DSMC

Figure 1 shows use of CrayPat for performance analysis. In this benchmark, execution time for thousand time steps are recorded, to compute parallel efficiency. One MPI process per dual-core node (1 ppn) was used. Parallel efficiency for this weak scaling run is first calculated directly from the measured run time. The code has a nice feature in that it reports fraction of time spent in major computational stages including the time taken for interprocessor communication. However the CrayPat tool was needed to understand which MPI routines account for the increasing fraction of the communication time as seen in Figure 1. At first use, CrayPat instrumentation was attempted with *"pat_build –u –g mpi"*. In terms of the size of this application characterized by number of functions or procedures ICARUS is very small compared to many other SNL applications. It has close to 200 subroutines. While instrumenting with pat_build was successful, the execution of the instrumented code, resulted in very high execution overhead, leading to run times that were as high as 20 times the un-instrumented code run times. This also completely distorted the fraction of time spent in major compute stages. In view of this difficulty the initial use of this tool was targeted for generating the kind of information shown in Figure 1 obtained easily with *"pat_build –g mpi"*. Although, the code reported communication time, is seen to be different from the CrayPat measured percentage of MPI time, the efficiency plot shows that they both capture the efficiency trend well. Looking through the code showed that, while CrayPat accounts for all MPI time like initialization and output, the code reports only the sum of the communication time between the compute time steps.

To overcome the large run time distortion on instrumenting all the functions, selective instrumentation of key functions was attempted with "*pat_build –w –T move_, collide_, communicate_ -g mpi*. This was successful and run times with very little distortion to the normally measured run times permitted further analysis with this tool. Also observed was a big improvement in the integrity of the .xf results-file in the version 3.2 compared to earlier versions. However, the environment variable *PAT_RT_FILE_PER_PROCESS=1* was needed to generate uncorrupted .xf files for 512 processors and above. There are certainly problems with the generation of the results .xf files for large processor counts. At 2048 and above processor counts, even the above environment setting or even adding the companion environment setting *PAT_RT_RECORD_PE=4* (groups of 4 PES write a file) did not overcome the file corruption problem. The file corruption is suspected to be related to I/O timing issues in the Lustre file system. This seems to indicate that for future use with 20,000 cores on Red Storm this file corruption issue must be resolved, as problems have been seen with order of magnitude smaller processor counts, with

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. Cray User Group Meeting; Seattle, WA; May 7-10, 2007

relatively simple applications. However the successful runs lead to the generation of .ap2 files with the pat_report utility and analysis with apprentice2 revealed some important application characteristics that needs to be further investigated.

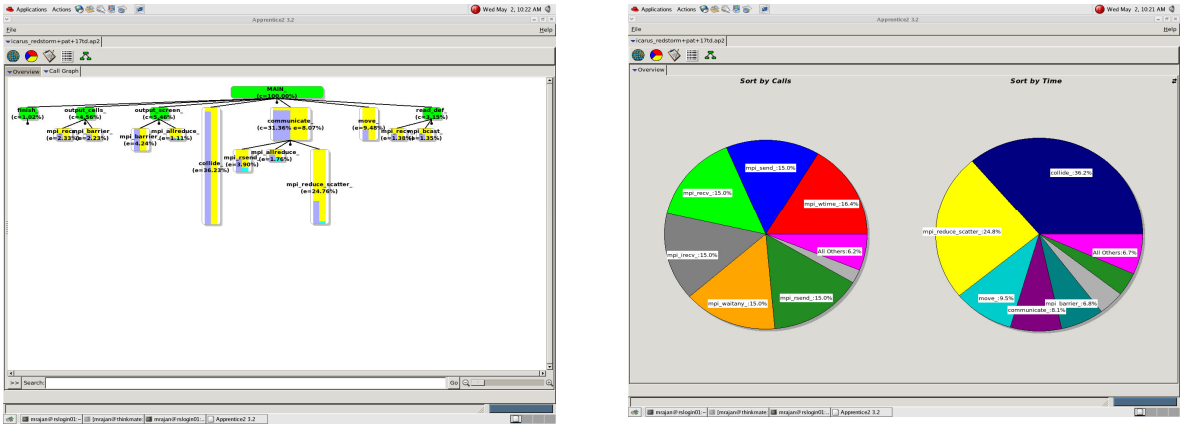


Figure 2. Apprentice2 Call Graph and Overall Profile for ICARUS/DSMC

From Figure 2 it is seen that a large fraction of the execution time (25%) is in the call to *MPI_Reduce_scatter*. The call graph with its use of color shading to show load imbalance, suggests that the problem may lie elsewhere. In fact the other major function with poor load balance in the call graph is the function *move_*. As described earlier in the stages of computation, following the movement of the particles, a communication stage updates the particle in each cell before proceeding to the next time step. This communication is first facilitated by a call to the *MPI_Reduce_scatter* to setup the point-to-point communication to transfer all the particle related information from the source processor to the destination processor. Since *MPI_Reduce_scatter* is a synchronous call, that follows *move_*, the large percentage of time attributed to it is really a consequence of load imbalance in *move_*. This is seen by diving into the load imbalance plots from the call graph and it is shown for both these functions in Figure 3.

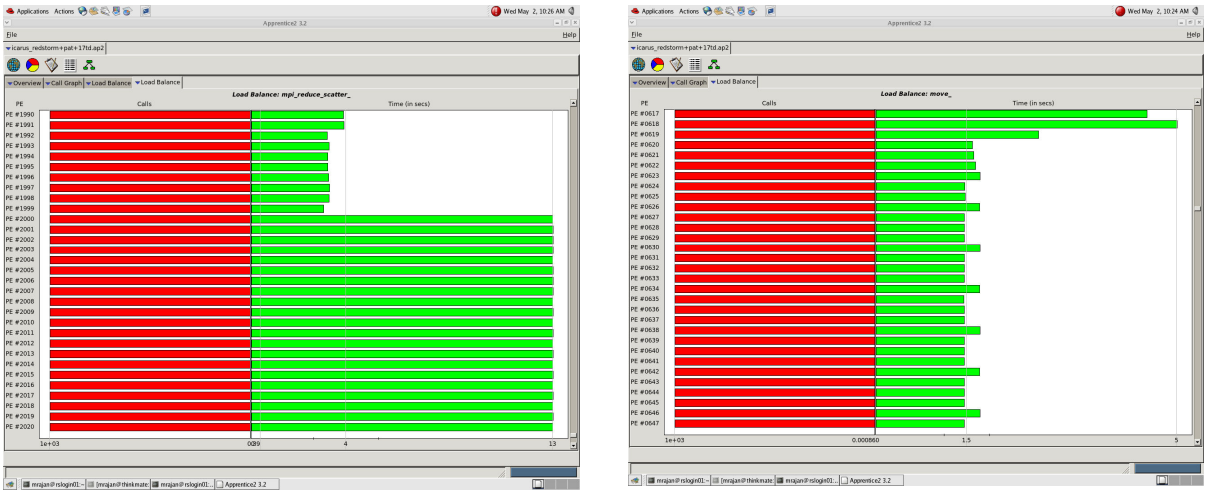


Figure 3. Apprentice2 plots of the *MPI_Reduce_scatter* and *move_* shows load imbalance

Attempts to use *PAT_RT_SUMMARY=0* to generate trace files led to very large trace files and difficulties with long delays generating the .ap2 files and in loading the files to Apprentice2. However limiting the number of processors to 32 we were able to generate a trace file that was compared to trace

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. Cray User Group Meeting; Seattle, WA; May 7-10, 2007

files obtained from using the VAMPIR tool. Figure 4 shows, a VAMPIR trace for the same problem. The ability with the VAMPIR tool to filter the data and zoom into focus on the communication messages between compute stages was particularly useful. The message length statistics plot for the zoomed region in VAMPIR is very useful for developing performance model. To our knowledge no equivalent functionality is available with CrayPat/Apprentice2. However it should be mentioned that the text report with message size bins and message counts was very helpful.

Figure 4. VAMPIR trace plots help identify messaging parameters for performance model

b) ITS Monte Carlo radiation transport:

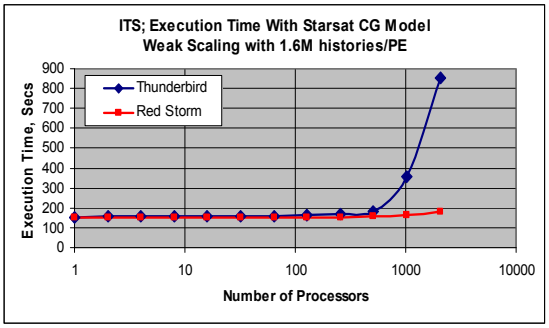
The INTEGRATED TIGER SERIES (ITS) code is an evolving Monte Carlo radiation transport code that has been used extensively in weapon-effect simulator design and analysis, radiation dosimetry, radiation effect studies and medical physics research. Many individuals from the DOE labs and NIST have been involved over the years in the development and enhancement of ITS. The different features/sections of the code in ITS: TIGER, MITS, CEPXS, XGEN etc., are applied to an analysis under investigation through the selection of appropriate pre-processor directives when the code is built. 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 adjoint transport method is a powerful technique for simulating applications where the knowledge of the particle flux is only required for a restricted region of the phase space, but where this knowledge is required for source parameters spanning a large region of phase space. 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. Although the code has been recently updated to improve parallel scaling, we have used the older version of the code as it amplifies the difference between a commodity cluster and a tightly integrated MPP and the difference in scaling performance related to a performance model we had developed [5].

Figures 5 presents side-by-side the execution time plot and the parallel efficiency plot for ITS, comparing Red Storm and a large infini-band commodity cluster, called Thunderbird [6], with Intel EM64T processor. 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 [5] that easily explains the increased overhead for the master/slave communications at the end of each batch of history computations. As noted in Ref. [5] 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 chose to use the older algorithm as exaggerates the difference between Thunderbird and Red Storm, helping us understand the impact of

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. Cray User Group Meeting; Seattle, WA; May 7-10, 2007

architectural balance on scalability.



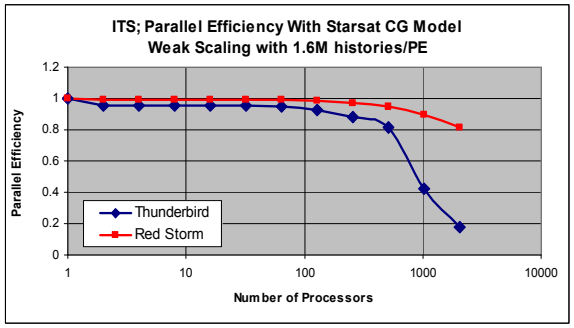


Figure 5. ITS Performance on Red Storm and Thunderbird

As in the case of the application ICARUS discussed above, this application on instrumentation of all the functions with "*pat_build –u –g mpi*" lead to large run time distortions due to very large number of function calls associated with computing particle distance and velocity. In this application also the corruption of the .xf file for large processor count was observed. Due to our previous experience with the use of VAMPIR tool for building a communication performance model, we attempted to use the *PAT_RT_SUMMARY=0* with instrumentation of only the MPI calls. Shown in Figure 6 are the VAMPIR plots that were very helpful in developing the communication performance model. As of writing this report the reason for the lack of details in the CrayPat/Apprentice2 trace plot was not understood. Some of the trace .xf files produced was so large that pat_report was unable to process them in reasonable amount of time even for 32 processor runs. However, we were able to use CrayPat with hardware counters to obtain information on instruction mix as shown in Table 2. Table 2 also shows the results of using PAPI on a IA-64 system and on a IBM Power3 system. The reason for looking at instruction mix on these other systems is to understand what instructions dominate this application. This table shows that this application has a high fraction of integer and branch instructions. The lack of counters on Opteron for integer, load, store instructions is a serious shortcoming. CrayPat facilitated collection of hardware events through simple modifications to environment variables, as opposed to our earlier time-consuming efforts, using PAPI API.

Table 2 ITS instruction mix on IA 64, Power3 and Opteron

PAPI DATA	**IA-64, 1.4GHz**	**Power3, 375MHz**	**Opteron, 2.0GHz**
TOTAL CYCLES	5,471,391,792	2,524,426,100	3,841,925,011
TOTAL INSTRUCTIONS	8,348,552,835	3,022,782,250	4,627,544,804
% Floating point ins or ops	0.026	0.052	0.040
% Load instructions	0.305	0.312	N/A
% Store Instructions	0.251	0.235	N/A
% Branch Instructions	0.084	0.137	0.199
% Integer Instructions	N/A	0.376	N/A
% Unaccounted ins	0.334	-0.112	0.761

c) LAMMPS:

LAMMPS [7] 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

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. Cray User Group Meeting; Seattle, WA; May 7-10, 2007

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. See lammps.sandia.gov for information on LAMMPS performance and scalability, and the Benchmarks.

The current version of LAMMPS is written in C++. Earlier versions were written in F77 and F90. 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. LAMMPS is most efficient (in a parallel sense) for systems whose particles fill a 3d rectangular box with roughly uniform density.

lj.inp 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. Other parameters used are: reduced density = 0.8442 (liquid), force cutoff = 2.5 sigma, neighbor skin = 0.3 sigma, neighbors/atom = 55 (within force cutoff), with NVE time integration. The execution time and parallel efficiency is shown in Figure 6 comparing the performance of Red Storm with Thunderbird, an infini-band Intel cluster.

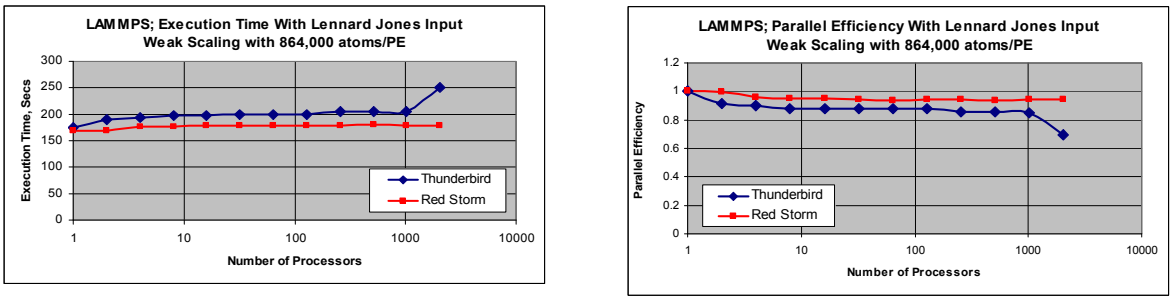


Figure 6. LAMMPS Performance on Red Storm and Thunderbird

The reason that two systems show similar performance in Figure 6 is because this application spends very little time in communication. This was confirmed with CrayPat instrumented runs on Red Storm, which showed that even at 2048 processor the fraction of time in MPI was less than 3%. LAMMPS was successfully instrumented with "*pat_build –u –g mpi -D trace-max=2000*" after using the switch to increase the default number of functions to be traced. The resulting instrumented code was run successfully to generate a .xf and .ap2 files. Figure 7 shows the Apprentice2 call-graph and overall profile. The large fraction of time spent in the function Pair_LjCut, should help with a focus for deeper analysis for optimization. The use of small pages resulted in a 2X improvement in the run time. Further investigations on code tuning are yet to be undertaken. The successful instrumentation of this application with over 1500 function was very encouraging and promises to meet our need for performance analysis with CaryPat for large applications.

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. Cray User Group Meeting; Seattle, WA; May 7-10, 2007

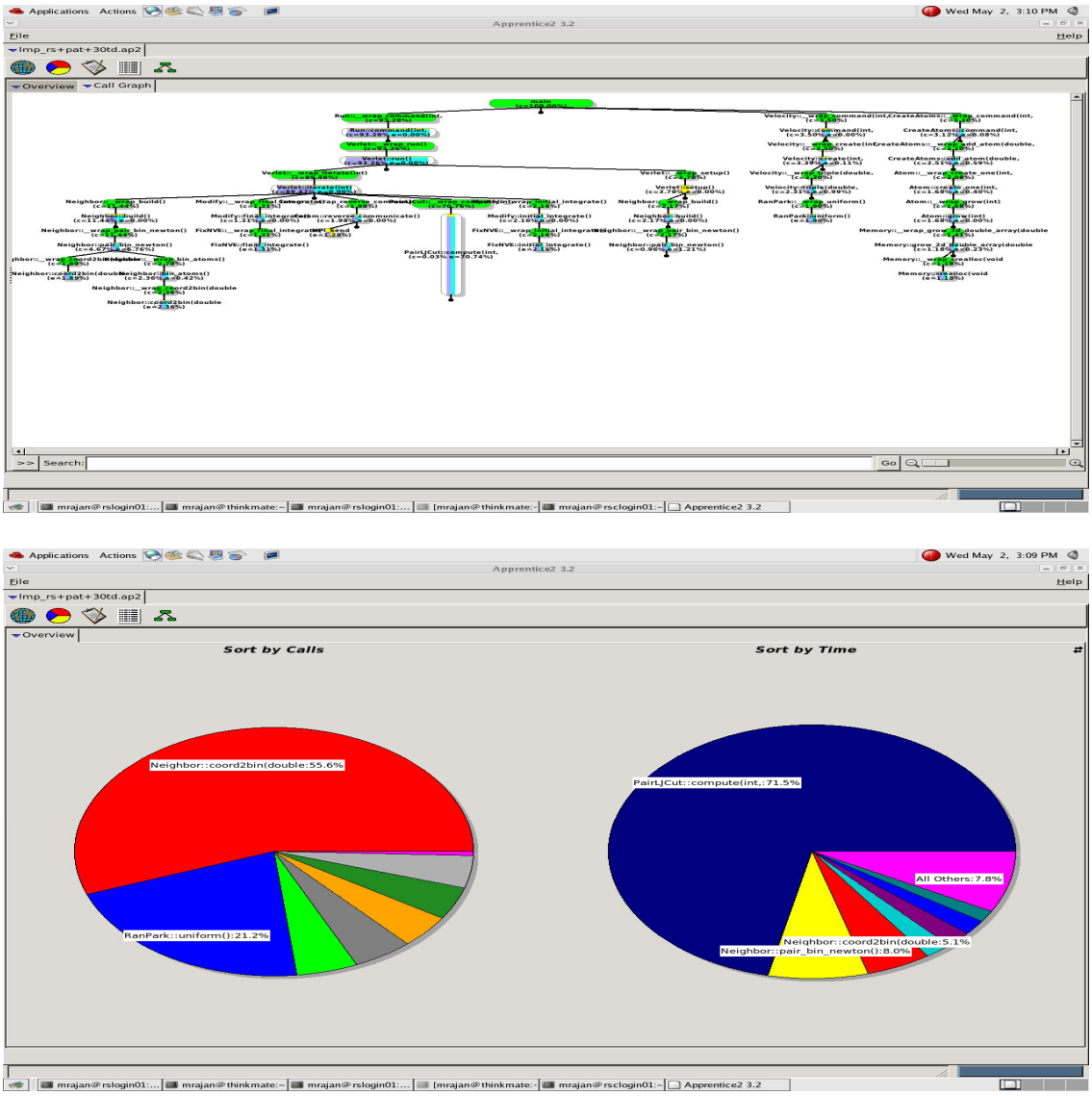


Figure 7. LAMMPS CrayPat/Apprentice2 plots with Call Graph and Overall Profile

d) POP:

The LANL Parallel Ocean Program, POP, is a widely used code to investigate ocean circulation models. It has been studied extensively from a performance point of view, with the recent paper by Kerbyson and Jones [8] providing a detailed performance model. The interest in POP at Sandia comes from a group working on atmospheric modeling and also from our use of the code recently to compare HEC systems. Recent scaling studies comparing performance of POP between the two large ASC capability systems, Red Storm and Purple, revealed that POP is sensitive to operating system jitter/noise for large processor counts. Figure 8 illustrates this observation. An environment variable MP_POLLING_INTERVAL was modified on recommendation of IBM and LANL staff to obtain the better scaling plot for Purple shown in the Figure 8. A large polling interval was used to remedy the high cost of interrupts. These interrupts nominally ensure the progress of the MPI communicators. It is well known among the POP research community that in the *Barotropic* solve portion of the compute cycle, the pre-conditioned conjugate gradient solver used invokes many *MPI_Allreduce* global operations. Our interest in the use of CrayPat was to quantify the

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. Cray User Group Meeting; Seattle, WA; May 7-10, 2007

effect of noise on such global operations. We were able to successfully instrument and run POP using selective instrumentation of the key routines with, "*pat_build –w –T baroclinic_ , barotropic_ , solvers_ -g mpi*" . Initial attempts at simply instrumenting all the functions with the –u switch led to core dumps upon trying to run the instrumented code.

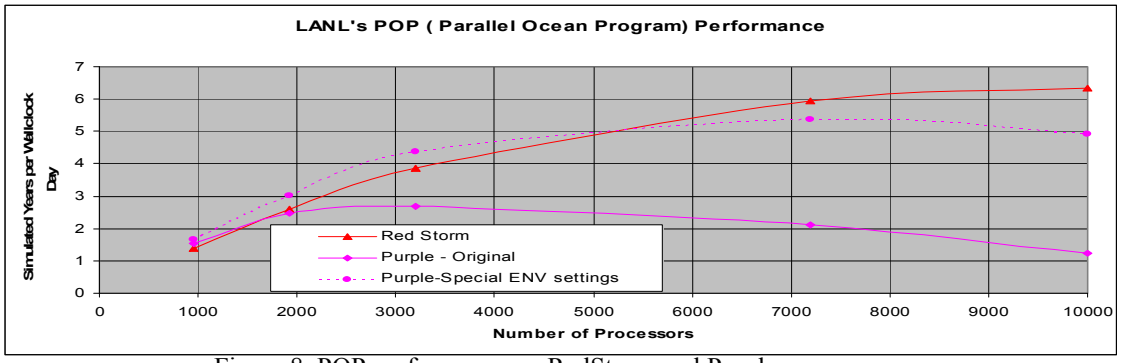


Figure 8. POP performance on RedStorm and Purple

Figure 9 shows the Apprentice 2 plot showing the large load imbalance in *MPI_Allreduce* under the barotropic solve branch in the call graph. While this has helped us understand the application better, further investigations on how operating system noise impacts this application significantly is yet to be undertaken. Red Storm has inherently very low OS noise because of its use of Light Weight Kernel, Catamount, on the compute nodes. By artificially injecting noise and using CrayPat to analyze its impact we hope to better understand this application.

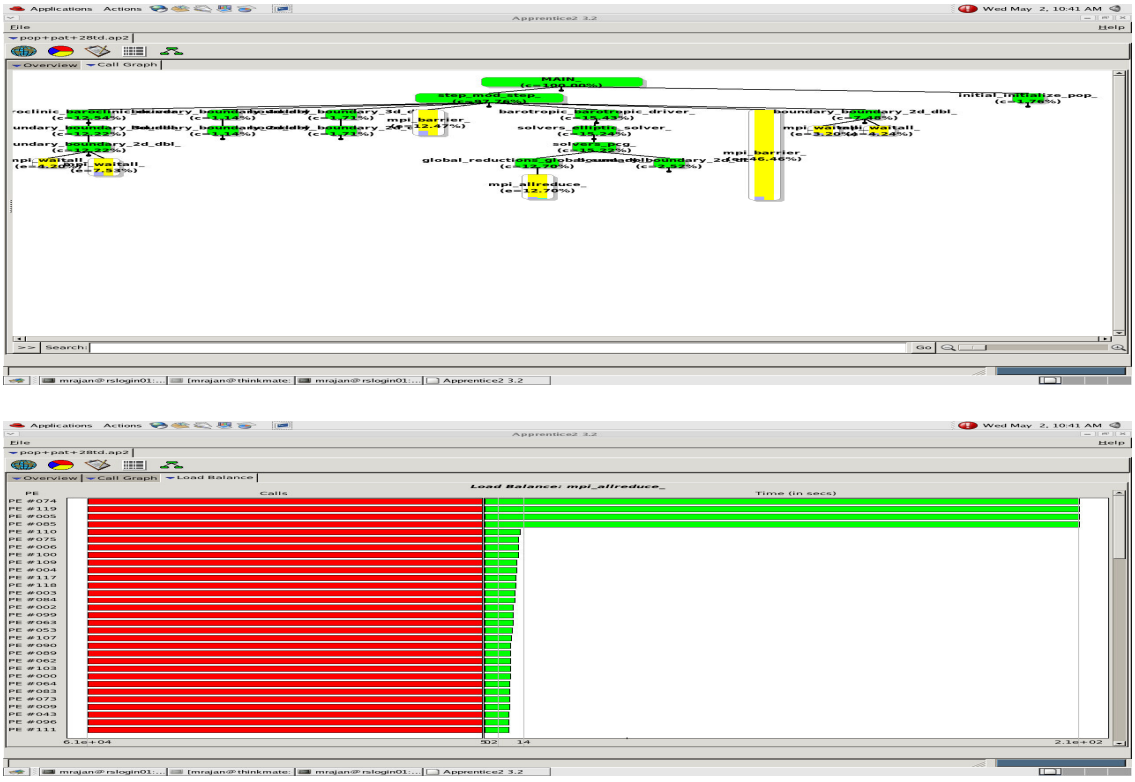


Figure 9. POP Apprentice2 plots showing load imbalance on MPI_allreduce in Baratropic solve

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. Cray User Group Meeting; Seattle, WA; May 7-10, 2007

Conclusions:

CrayPat and Apprentice2 have been very helpful in performance analysis, tuning and modeling. Much of the results generated for this paper were obtained in a short two week period following an on-site training by Cray experts. This hands-on training in conjunction with the use of CrayPat and Apprentice version 3.2 helped overcome most of the difficulties encountered previously. Earlier efforts with older version of these products were frustrated because of large distortions in run time, albeit because of the naive use of the pat_build with the –u option. Additional difficulties with the .xf file corruption with 512 or above MPI processes limited its usability for our use with large capability class simulations. While the file corruption process is still observed for 2048 and above processor count, writing one file per processor in a directory as opposed to a single file, improves the situation. Another important lesson learnt is to selectively instrument only the key functions to avoid explosive increase in the run time. We understand that 'sampling' feature to be available with CrayPat in a future release will help with picking the functions that should be selectively included for profiling, as it promises to preserve the true run time percentages of the functions in the application. The CrayPat hardware counter use with a simple setting of an environment variable greatly facilitates performance tuning. This saves much time compared to our previous use of counters with PAPI API. Sandia applications, such as ITS, dominated by non-floating point operations could benefit from, future AMD Opteron support for load, store, integer operation counters. Our preliminary experiments with generating and using trace files was not very successful, especially when compared to our experiences with VAMPIR. Some mechanism for filtering the data so as to reduce the size of the trace files and filters to restrict processors and compute iterations for repeated operations may be very helpful. However we as users need to further experiment with trace to see how best we could use it to improve our understanding of the application behavior. One addition to the trace plots that would be very helpful for us is a VAMPIR like message statistics plot. This plot should show data for only zoomed region in the trace plot. In summary the feature rich and easy to use CrayPat and Apprentice2 are invaluable in our efforts to extract the optimal performance from Red Storm. Improvements in CrayPat to ensure uncorrupted .xf files for large processor count runs, up to the available 26000 cores on Red Storm, is our most pressing need in view of targeted capability computing jobs. A question that is often asked of applications running on the Red Storm is, "what is the percentage of peak performance observed". To answer that question we used CrayPat with three applications and the results are shown in Table 3.

Table 3. CrayPat hardware counter data with three applications

	LAMMPS – 1PE	ICARUS – 32 PE	SAGE – 1PE
Total Cycles	438334149027	23022223454	146844868231
Total Instructions	322843999218	855901630778	70445132839
Floating point ins.	159193963401	154415317565	12306664167
Branch Instructions	18035055818	76266692411	6753460052
Run time	182.63 secs	9.592 secs	61.18 secs
MFLOPS	871 (18.2%of peak)	16097 (10.5%of peak)	201.13 (4.2% of peak)
%Floating point Ins.	49%	18.04%	17.47%
%branch Ins.	5.6%	8.9%	9.58%
Computational intensity	0.92 ops/ref	0.51 ops/ref	0.42 ops/ref

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. Cray User Group Meeting; Seattle, WA; May 7-10, 2007

Often times quantities such as computational intensity, cache usage are difficult to interpret in an absolute sense. Frequently these are used as measures to improve performance comparing them before and after code modifications. It is also useful for an analyst to have at hand measures of these quantities for most common computational kernels such at matrix multiply, FFT, etc. Table 4 is our attempt to have at our disposal common measures such as computation intensity, ops per cycle, floating point operations per TLB or cache misses, using matmul, FFT, QR factorization, and, a simple Sparse Matrix-vector operation. These will be useful in understanding similar data obtained from applications.

Table 4. CrayPat hardware counter data with simple math kernels

Single CPU reference measures with PAT_RT_HWPC=1,2,3,4				
code	3dFFT; 256x256x256	matmul 500x500	QR Fact. N=2350	HPCCG; sparseMV;100x100x100
Comp. Inten;ops/ref	1.33	1.71	1.68	0.64
MFLOPS/pat	952	4159	3738	352
MFLOPS code	1370	4187	4000	276
percent peak	19.8	86.7	77.9	7.3
fpOps/TLB miss	841.6515146	9040759.488	697703.9649	14.05636016
fpOps/D1 cache miss	25.5290058	167.9364898	144.9081716	10.24364227
fpOps/DC_MISS	29.42427018	170.5178224	149.9578195	11.1702481
ops/cycle	0.4	1.75	1.56	0.15

References:

1) http://docs.cray.com/books/S-2396-15/S-2396-15.pdf
2) http://icl.cs.utk.edu/projects/papi/files/html_man/papi.html
3) "Predictive Performance and Scalability Modeling of a Large-Scale Application", Kerbyson, D.J., et.al., SC 2001, November 2001, Denver, CO; ACM 1-58113-293-X/01/0011
4) " 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
5) "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
6) http://www.sandia.gov/news/resources/releases/2006/thunderbird.html
7) http://lammps.sandia.gov/docs
8) "A Performance Model for the Parallel Ocean Program," Kerbyson, D.J., Jones, P.W., The International Journal of High Performance Computing Applications, Vol. 19, No.3., Summer 2005, pp. 261-276

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. Cray User Group Meeting; Seattle, WA; May 7-10, 2007