Early Evaluation of the Cray XD1

Mark R. Fahey,
Sadaf Alam, Thomas H. Dunigan, Jr.,
Jeffrey S. Vetter, Patrick H. Worley
Oak Ridge National Laboratory

ABSTRACT: Oak Ridge National Laboratory received 12 chassis of early access Cray XD1 nodes in October 2004, where each chassis has 12 AMD Opteron processors. This paper describes our initial experiences with the system, including micro-, kernel, and application benchmarks results.

KEYWORDS: performance evaluation; Cray XD1; Linux Synchronized Scheduler

1 Introduction

Oak Ridge National Laboratory (ORNL) obtained a 144 processor Cray XD1 for evaluation in October 2004. The Cray XD1 product (formerly from Octigabay) is an Opteron-based cluster with a RapidArray interconnect. It is currently under evaluation as part of the DOE Advanced Computing Research Testbed. As part of the evaluation, a primary task is to evaluate application performance and compare with systems from other high-performance computing (HPC) vendors. Standard benchmarks and custom microbenchmarks are used to evaluate the system where applicable, and for comparison with other evaluations. However, the emphasis here is on performance evaluations of full applications of interest to DOE. The applications involved in our current evaluations include codes drawn from climate modeling, fusion, materials, and hydrodynamics.

Section 2 provides an overview of the ORNL Cray XD1. Section 3 describes the evaluation process including methodology and test machines. Section 4 presents performance results from communication and computation microbenchmarks and custom kernels. Section 5 has performance data from applications in global climate, fusion, materials, and hydrodynamics. Section 7 recaps the performance characteristics observed in this study and discusses a few of the outstanding issues.

At the time of writing this report, no substantial testing of the FPGAs has been done and thus discussion of FPGAs will be omitted.

2 XD1 Overview

In collaboration with Cray Inc, Oak Ridge National Laboratory acquired twelve chassis of early access Cray XD1 nodes. Each chassis has 12 AMD Opteron 248 series processors running at 2.2 GHZ, and 4 GB of memory per processor. The total system has 144 processors (633 Peak GFlops), 576 GB of memory, and 18 TB of disk.

The Cray XD1 combines a new interconnect, management tools and reconfigurable computing technologies with the AMD Opteron processor.

2.1 Compute Processors

The ORNL XD1 has 64-bit AMD Opteron 248 series 2-way (single core) processors. These processors have a 64KB L1 instruction cache, a 64KB L1 data cache, and 1MB L2 cache. The processors can issue two floating-point instructions per cycle for a peak rate of 4.4 GFlops per processor.

2.2 Interconnect

The following information is taken from the Cray XD1 Overview webpage [1].

Cray's RapidArray Communications Processor provides the interface from HyperTransport on the Opteron to the RapidArray interconnect fabric. The entire system is interconnected with Cray's RapidArray terabit backplane, providing low-latency, high-bandwidth connections among the pro-

cessors. The RapidArray interconnect is an embedded switching fabric that uses 12 custom communications processors and a 96 GB/s nonblocking switching fabric per chassis to deliver 8 GB/s bandwidth between SMPs with 1.7 microsecond MPI latency. Each chassis presents 24 RapidArray interchassis links with an aggregate 48 GB/s bandwidth. In addition, one chassis contains six Xilinx Virtex-II Pro Field Programmable Gate Arrays (FPGA) modules, each with 5 million gates. All twelve chassis are installed in a single rack, demonstrating excellent computational density. Multirack configurations are available that integrate hundreds of processors into a single system.

As Figure 1 shows, the Cray XD1 system is based

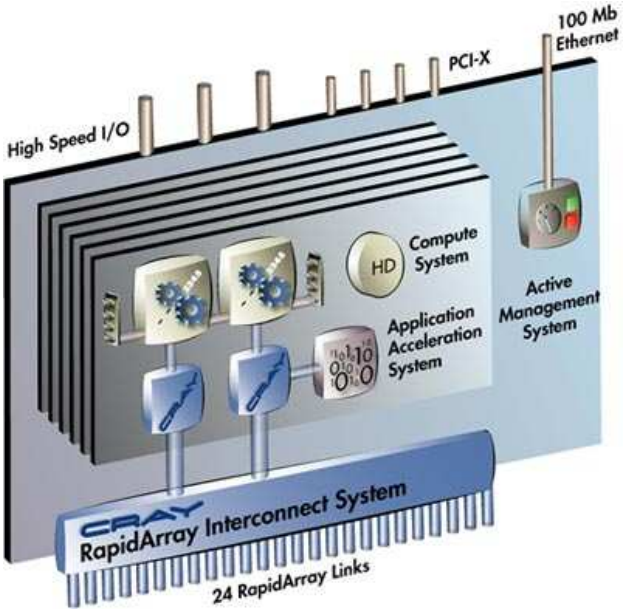

Figure 1: XD1 Direct Connected Processor Architecture. Image courtesy of Cray, Inc.

on the Direct Connected Processor (DCP) architecture, coupling many processors into a single, unified system. Cray's implementation of the DCP architecture optimizes message-passing applications by directly linking processors to each other through a high performance interconnect fabric, eliminating shared memory contention and PCI bus bottlenecks.

2.3 System Software and Synchronization

Standard Linux is optimized for interactive and transaction processing applications that involve short, variable compute cycles and minimal interprocessor dependencies. On the other hand, HPC applications have very different profiles, running many interdependent processes on multiple processors. These applications use (explicit or implicit) barriers to ensure all processors or some subset finish a given step before any continue on. Sitting idle at barriers waiting for other processors to complete housekeeping functions obviously reduces productiviy.

On the XD1, all the nodes run Linux, but the compute nodes have a special kernel containing the Linux Synchronized Scheduler (LSS) that allows them to synchronize with a global clock and coschedule processes to avoid latency in global communication. Therefore, the LSS ensures that the time slots for all nodes in a partition will be synchronized. This means that all nodes in the partition are guaranteed to be executing the same system or user time slots at any give time. By ensuring that an application executes in the same timeslot systemwide, time spent in wait-states can be minimized, substantially improving processor efficiency.

3 Evaluation Overview

The evaluation is hierarchical and staged. In this hierarchical approach, the low-level functionality of the system is examined first. Results are then used to guide and understand the evaluation of kernels and application codes. Standard benchmarks are used when appropriate, to compare with evaluations of other systems, but the emphasis is on application-relevant studies.

3.1 Methodology

The hierarchical approach described above has been employed in the examination of performance issues. The first step was to establish functional correctness. A system incorporating novel architectural features as the XD1 must be checked for correctness. The second step was to establish performance correctness. While some performance details are unknowable at this time, gross performance, based on hardware and software specifications, can be used to predict performance in a coarse sense. Verification of these performance expectations is important to identify and correct hardware and software implementation errors. The third step was to perform "technology evaluations" of specific subcomponents

of the Cray XD1. Our evaluation covers the following technologies:

- RapidArray fabric. This fabric provides an advertised 2GB/s bandwidth link (aggregate bisection) to each Opteron and MPI latency is 1.7 μsec.
- Cray's RapidArray Communications Processor provides the interface from HyperTransport on the Opteron to the RapidArray interconnect fabric. This specialized processor routes messages between processors, ensures reliable transport, optimizes message transfer, and handles time synchronization across processors.
- AMD Opteron. This platform provides DOE a development system using AMD Opteron processors to begin porting and tuning libraries and applications to these processors.
- Synchronized Scheduler. Cray has modified the SuSE Linux OS scheduler to support the synchronization requirements of parallel applications rather than interactive applications. This enhancement should improve application scalability and overall performance.

Performance activities are staged to produce relevant results throughout the duration of the evaluation. For example, subsystem performance was measured as soon as a system arrived, and measured again following a significant upgrade or system expansion. Fortunately, these experiments can be conducted relatively quickly. In contrast, the porting, tuning, and performance analysis of complex applications, such as those involving unstructured or dynamic data structures, take much longer to complete, and were started as soon as possible to have an impact on the evaluation.

3.2 Test machines

For comparison purposes, performance data are also presented for the following systems:

- Cray XT3 at ORNL: 3748 compute nodes with 46 service nodes connected as 10 x 16 x 24 (X x Y x Z) with a torus topology in X and Z dimensions. The Y dimension is a mesh. Each node has a 2.4 GHz AMD Opteron and 2 GB of memory.
- Cray X1 at ORNL: 512 Multistreaming processors (MSPs), each of which is capable of 12.8 GFlops for 64-bit operations. Each MSP is comprised of four single streaming processors (SSPs). The SSP uses two clock frequencies, 800 MHz for the vector units and 400 MHz for the scalar unit. Each SSP is capable of 3.2 GFlops for 64-bit operations.
- Earth Simulator: 640 8-way vector SMP nodes and a 640x640 single-stage crossbar interconnect. Each processor has 8 64-bit floating point vector units running at 500 MHz, and is capable of a peak performance of 8 GFlops for 64-bit operations.
- SGI Altix at ORNL: 256 Itaninium2 processors and a NUMAlink switch. The processors are 1.5 GHz. The machine has an aggregate of 2 TB of shared memory.
- IBM p5 720 at ORNL: Two 2-way processor cards comprising a 4 processor SMP running Linux. The processors are 1.65 GHz POWER5.
- IBM p690 cluster at ORNL: 27 32-way p690 SMP nodes and an HPS interconnect. Each node has 2 HPS adapters, each with two ports. The processors are 1.3 GHz POWER4.
- IBM SP at the National Energy Research Supercomputer Center (NERSC): 184 Nighthawk(NH) II 16-way SMP nodes and an SP Switch2. Each node has two interconnect interfaces. The processors are the 375MHz POWER3-II.
- HP/Compaq AlphaServer SC at Pittsburgh Supercomputing Center (PSC): 750 ES45 4-way SMP nodes and a Quadrics QsNet interconnect. Each node has two interconnect interfaces. The processors are the 1GHz Alpha 21264 (EV68).

4 Performance Results

The results presented here are from standard microbenchmarks and custom kernels. The results represent snapshots of the machine at various times. All results reported here were obtained while the machine was configured as two 72-processor systems.

4.1 Microbenchmarks

We use a collection of microbenchmarks to characterize the performance of the underlying hardware, compilers, and software libraries. The microbenchmarks measure computational performance, memory hierarchy performance, and interprocessor communication.

For a more in-depth look at microbenchmark performance on the ORNL XD1, see [10] and [2].

4.1.1 Computation

Figure 2 shows a platform intercomparison of the double-precision floating point performance of a matrix-matrix multiply (DGEMM) on a single processor using the vendor's scientific library (ACML on Cray XT3 and XD1, LIBSCI on Cray X1, ESSL on IBMs, and SCSL on SGI Altix.) The performance

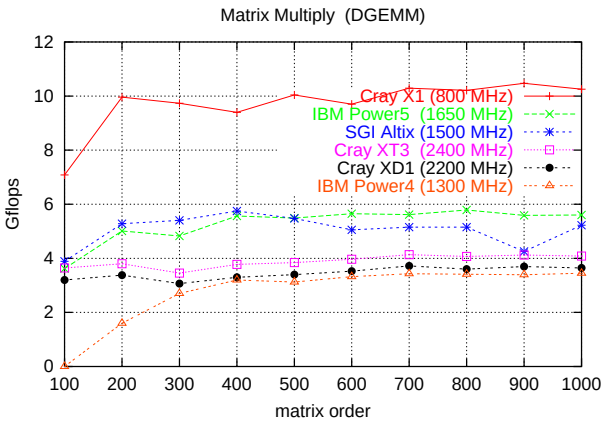

Figure 2: XD1 DGEMM performance.

of DGEMM has long served as a good indicator of a machine's practical peak performance. The XD1 Opteron achieves 3.7 GFlops, about 84% of theoretical peak.

The STREAM benchmark [12] is a simple synthetic benchmark program that measures sustainable memory bandwidth (in MB/s) and the corresponding computation rate for simple vector kernels. The triad memory bandwidth of the STREAM benchmark is presented in the Table 1.

The CacheBench benchmark [14] was run to evaluate the performance of the XD1 memory hierarchy. CacheBench calculates raw bandwidth and establishes a peak computation rate for optimal cache reuse, thereby enabling an application developer to

Table 1: TRIAD memory bandwidth from STREAM benchmmark.

Processor	Triad (GBs)	Compiler
Cray X1 MSP	23.8	PE 5.3
Cray XT3	5.1	Pathscale 2.1
Cray XD1	4.1	Pathscale 2.1
Cray XD1	3.2	PGI 5.2-4
IBM p690	2.1	xlf 8.1
IBM power5	4.0	xlf 9
SGI Altix	3.8	Intel 8.1

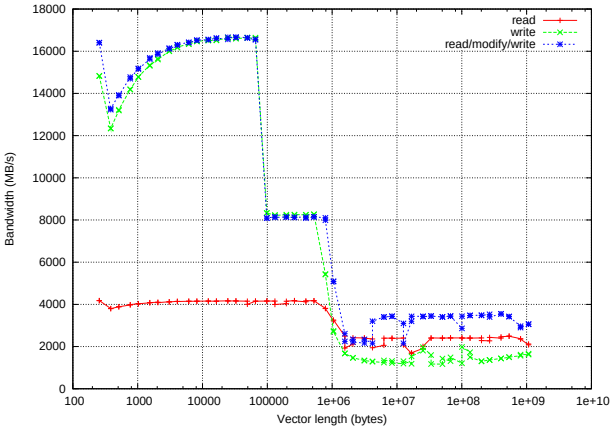

Figure 3: XD1 Cachebench performance with Portland Group 5.2-4 compiler.

understand the ability of a cache sub-system to sustain large, unit-stride, floating-point workloads.

Figures 3 and 4 show the results of compiler-optimized read, write and read/write/modify bandwidth tests with varying vector lengths on an XD1 processor. The significant variations in the bandwidth at different vector lengths show the capacity of the three different cache levels. For instance, the XD1 processor has a 64KB level 1 data cache, therefore, the read bandwidth drops at a vector length of 524288 bytes. The write latencies on the other hand depend on a number of architectural features including replacement policy and write buffering schemes.

4.1.2 Communication

Message-passing on the XD1 utilizes the Opteron HyperTransport interface to RapidArray. Our tests results show that the MPI latency is 1.7 μsec and bandwidth is 1.3 GB/s between nodes [10].

Interestingly at this time, MPI bandwidth is

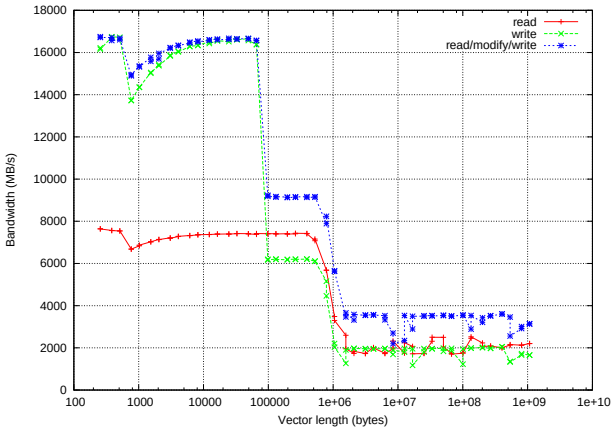

Figure 4: XD1 Cachebench performance with Pathscale 2.1 compiler.

higher between nodes than between the two processors of an SMP node, see Figure 5. Latency and bandwidth show little degradation as one communicates with more distant processors. Note that we only tested 64-processor configurations.

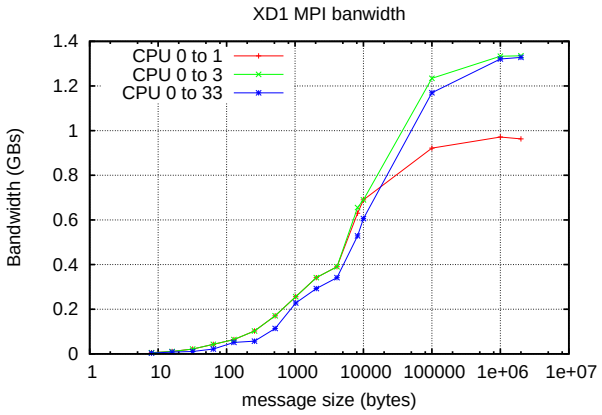

Figure 5: XD1 MPI Bandwidth.

The HALO benchmark is a synthetic benchmark that simulates the nearest neighbor exchange of a 1-2 row/column "halo" from a 2-D array. This is a common operation when using domain decomposition to parallelize (say) a finite difference ocean model. There are no actual 2-D arrays used, but instead the copying of data from an array to a local buffer is simulated and this buffer is transfered between nodes. Figure 6 compares the HALO latency for MPI when using 16 processors. The XD1's low latency makes it the best performer for small messages.

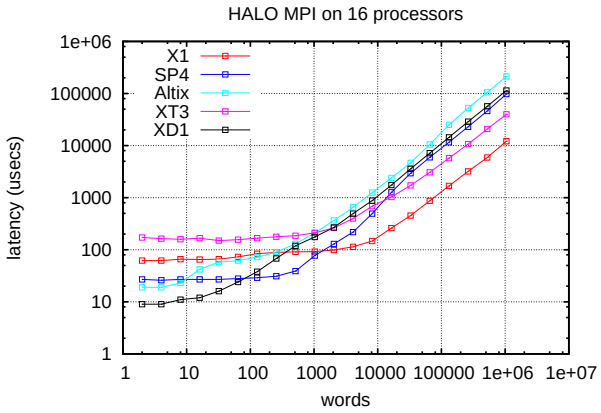

Figure 6: XD1 Halo.

The XD1 supports a generic version of SHMEM (GPShmem), but it's performance is worse than MPI at this time; latency is 6.4 μsec and bandwidth is 981 MB/s.

We also show some data from the Pallas MPI-1 Benchmark (PMB) Suite V2.2 [3]. Figures 7 and 8 show the performance of MPI Allreduce and Alltoall on the XD1 when the Linux synchronized scheduler

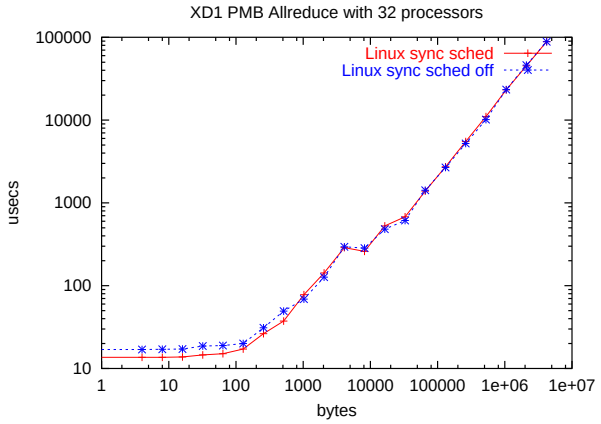

Figure 7: PMB MPI_Allreduce.

(LSS) is both on and off.

The data suggests that the LSS helps but only for message sizes up to somewhere between 1000 MB and 10000 MB. Also in this range, both Figures 7 and 8 show a curious bump going from 4096 MB to 8192 MB. This bump is a result of a switch in the communications protocol.

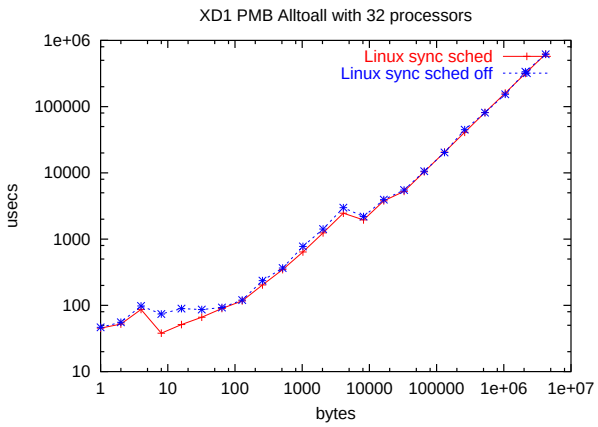

Figure 8: PMB MPI_Alltoall.

4.2 PSTSWM Kernel

The Parallel Spectral Transform Shallow Water Model (PSTSWM) [16] represents an important computational kernel in spectral global atmospheric models. As 99% of the floating-point operations are multiply or add, it runs well on systems optimized for these operations. PSTSWM exhibits little reuse of operands as it sweeps through the field arrays; thus it exercises the memory subsystem as the problem size is scaled and can be used to evaluate the impact of memory contention in SMP nodes. PSTWM is also a parallel algorithm testbed, and all array sizes and loop bounds are determined at runtime.

On the XD1 we used PSTSWM to analyze compiler optimizations, evaluate performance of the memory subsystem, and compare performance with other supercomputers.

Figure 9 shows comparisons of optimization options. The comparisons are presented as computation rate versus horizontal resolution for two vertical resolutions. The problem sizes T5, T10, T21, T42, T85, and T170 are horizontal resolutions. Each computational grid in this sequence is approximately 4 times smaller than the next larger size. Although the two figures are for 1 and 18 vertical levels, this aspect of the problem size does not change the compiler option comparison. PSTSWM manages its own heap, and all loop bounds and array sizes are defined at runtime. From the figure, we see that this seems to limit some of the possible performance optimizations.

Figures 10 and 11 compare performance between all problem sizes, showing impact of memory hierarchy. Most of the work is coupled most tightly hor-

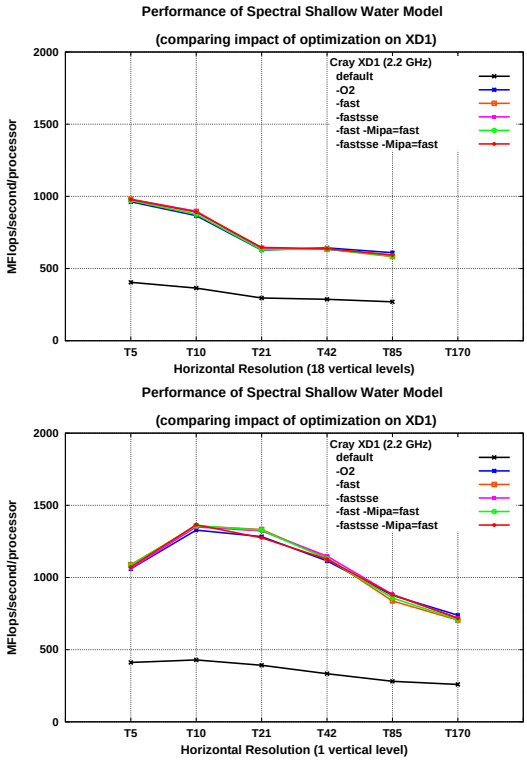

Figure 9: PSTSWM: comparing optimization options.

izontally, so additional vertical levels spreads things out, increasing memory access. Large problems drop from more than one GFlops to 600 MFlops quickly as a function of number of vertical levels, but stay at the 600 MFlops rate from then on. Smaller problems drop more slowly, but to a lower asymptotic rate. The one and two processor per node comparison shows that this behavior is unaffected by both processors accessing memory simultaneously. Thus using both processors does not decrease memory performance or, in other words, increase memory contention.

Figure 12 compares single processor performance for various horizontal resolutions and a fixed 18 vertical levels. The X1's superior scaling with respect to problem size is due to the much higher processor/memory bandwidth and increased efficiency as the vector lengths increase.

4.3 SMG2000 Kernel

SMG2000 [6] is a parallel semicoarsening multigrid solver for the linear systems arising from finite dif-

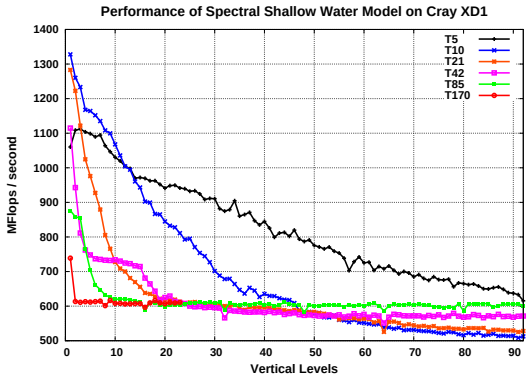

Figure 10: PSTSWM: one processor per node.

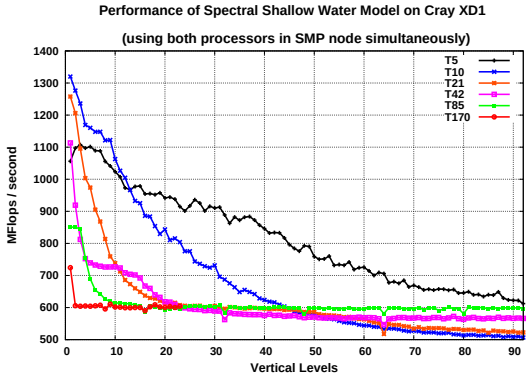

Figure 11: PSTSWM: two processors per node showing performance as seen by single processor.

ference, finite volume, or finite element discretizations of the diffusion equation on logically rectangular grids. The code solves both 2-D and 3-D problems with discretization stencils of up to 9-points in 2-D and up to 27-points in 3-D. Applications where such a solver is needed include radiation diffusion and flow in porous media. Our examination includes both the setup of the linear system and the solve itself. Note that this setup phase can often be done just once, thus amortizing the cost of the setup phase over many timesteps. This trait is relatively common in implicit timestepping codes.

SMG2000 is an SPMD code that uses MPI, and parallelism is achieved by data decomposition. SMG2000 is a highly synchronous code. The communications and computations patterns exhibit the surface-to-volume relationship common to many parallel scientific codes. Hence, parallel efficiency is largely determined by the size of the data "chunks" mentioned above, and the speed of communications

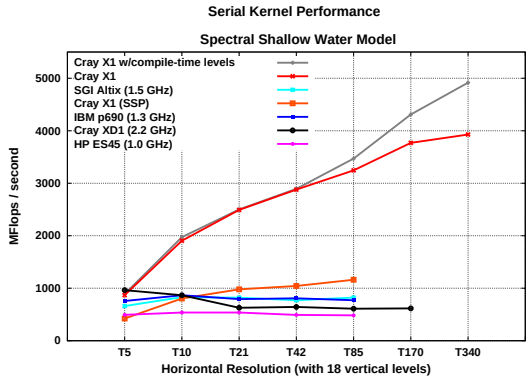

Figure 12: PSTSWM: 18 vertical levels, all horizontal resolutions.

and computations on the machine. SMG2000 is also memory-access bound, doing only about 1-2 computations per memory access, so memory-access speeds will also have a large impact on performance.

As Figure 13 shows the performance of SMG2000 on the XD1, XT3, SP4 and SP3. The problem size per processor stays fixed as the number of processors increases (weak scaling) and so a "flat" line in the figure is good. Thus, the performance of SMG2000 shows the best performance on the Cray XD1, and then the XT3, even though the XD1 has a 2.2Ghz Opteron (when compared to the 2.4Ghz Opteron in the XT3). In a typical solve, SMG2000 sends thousands of messages per processor, so its performance benefits from the very low latency interconnect provided by the XD1.

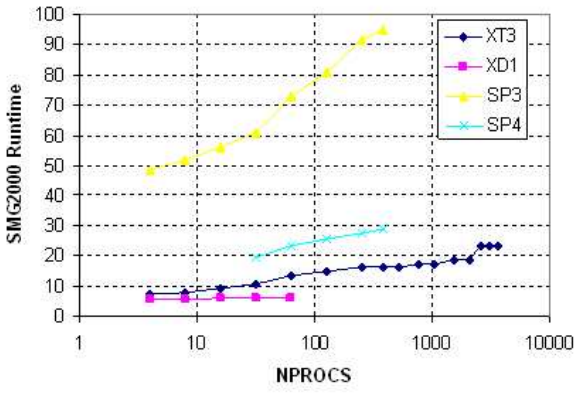

Figure 13: SMG2000.

5 Applications

The performance and efficiency of applications in the areas of global climate, fusion, materials, and hydrodynamics have been evaluated. These full applications (described below) have been ported to the machine with the initial emphasis on correct functioning of the code, and, subsequently, upon sequential and then parallel performance and scalability.

The following sections describe the initial application targets for evaluation. Future evaluation will be open to other domains.

5.1 POP

The Parallel Ocean Program (POP) [11] is the ocean component of CCSM [5] and is being developed and maintained at Los Alamos National Laboratory (LANL). The code is based on a finite-difference formulation of the three-dimensional flow equations on a shifted polar grid. In its high-resolution configuration, 1/10-degree horizontal resolution, the code resolves eddies for effective heat transport and the locations of ocean currents.

We used a benchmark configuration (called x1) representing a relatively coarse resolution similar to that currently used in coupled climate models. The horizontal resolution is roughly one degree (320x384) and uses a displaced-pole grid with the pole of the grid shifted into Greenland and enhanced resolution in the equatorial regions. The vertical coordinate uses 40 vertical levels with a smaller grid spacing near the surface to better resolve the surface mixed layer. Because this configuration does not resolve eddies, it requires the use of computationally intensive subgrid parameterizations. This configuration is set up to be identical to the actual production configuration of the Community Climate System Model with the exception that the coupling to full atmosphere, ice and land models has been replaced by analytic surface forcing.

Figure 14 shows a platform comparison of POP throughput for the "x1" benchmark problem. The XD1 performance is similar to that of the IBM Power4 and SGI Altix.

Figure 15 shows the performance of the barotropic portion of POP. This component is dominated by solution of 2D implicit system using conjugate gradient solves and is known to scale poorly. Figure 15 clearly shows that the XD1 scales better than several competing platforms.

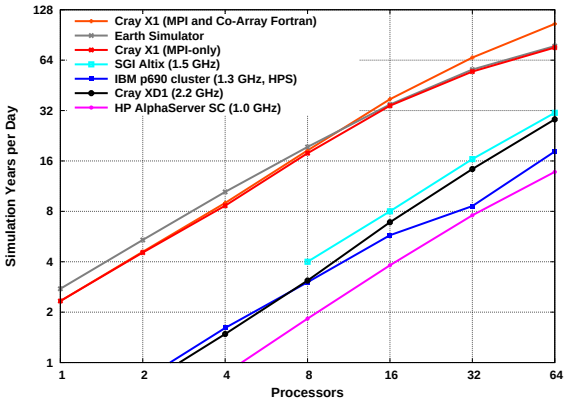

Figure 14: POP "x1" benchmark showing simulated years per day.

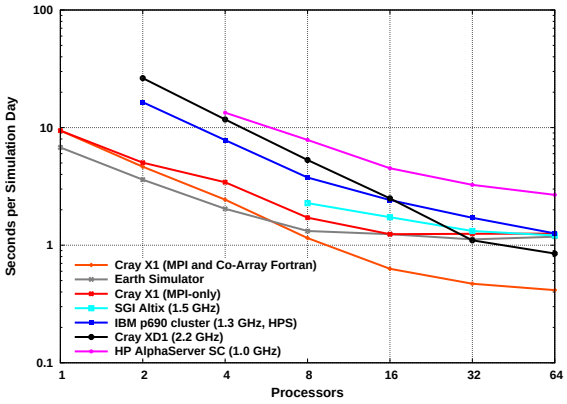

Figure 15: POP "x1" benchmark showing the barotropic contribution, which typically does not scale well.

5.2 CAM

The Community Atmospheric Model (CAM) [9] is the atmospheric component of the Community Climate System Model (CCSM), the primary model for global climate simulation in the U.S. and the target of the climate SciDAC project, "Collaborative Design and Development of the Community Climate System Model for Terascale Computers." The prominent dynamics algorithm of the CAM is a semi-Lagrangian transport method in combination with a semi-implicit Eulerian spectral method. This code grew out of the spectral models developed at the National Center for Atmospheric Research (NCAR).

At this time we do not have performance numbers for CAM, but we nonetheless report the cur-

rent status. We have noticed that results from CAM are not deterministic. Running two identical experiments with same processor count resulted in different answers. We found that it is already known [8] that the "CAM pergro" [15] tests fail on Opterons compiled with PGI 5.2-4. In contrast, CAM has been shown to work correctly on Opteron clusters at NCAR with the Pathscale compilers [8]. We have not had time to verify this on the XD1 yet.

5.3 GYRO

GYRO [7] is a code that simulates tokamak turbulence by solving the time-dependent, nonlinear gyrokinetic-Maxwell equations for both ions and electrons. The coupled gyrokinetic-Maxwell (GKM) equations provide a foundation for the first-principles calculation of turbulent tokamak transport. GYRO uses a five-dimensional grid and advances the system in time using a second-order, implicit-explicit (IMEX) Runge-Kutta (RK) integrator. GYRO is a GKM code that has both global and electromagnetic operational capabilities. GYRO has been ported to a wide variety of modern MPP platforms.

In the following, we show performance results for the B1-std benchmark problem. This problem is otherwise known as the Waltz Standard Case, which is a flux-tube electrostatic simulation with kinetic electrons and collisions on a $140 \times 8 \times 8 \times 16 \times 20 \times 2$ grid for a total of 5,734,400 gridpoints. It uses multiples of 16 processors.

Figure 16 shows the walltime to completion for each processor count when running in different modes on the XD1. These modes are: one processor per node with main (interconnect) fabric only, one processor per node with main and expansion fabric, two processors per node with main fabric only, and two processors per node with main and expansion fabric. Figure 17 shows the time attributed to MPI from Figure 16. Although the data presented earlier in Figure 5 indicated the communication between processors in the same node is more expensive than between nodes, the data in Figure 17 do not correspond to that theory knowing that GYRO uses a lot of communication bandwidth when performing its `MPI_Alltoall`s. In fact, using both processors is more efficient than using one processor per node for the same total number of processors. This may be because the one processor per node tests are "spread out" more (running on more chassis), leading to higher interchassis contention.

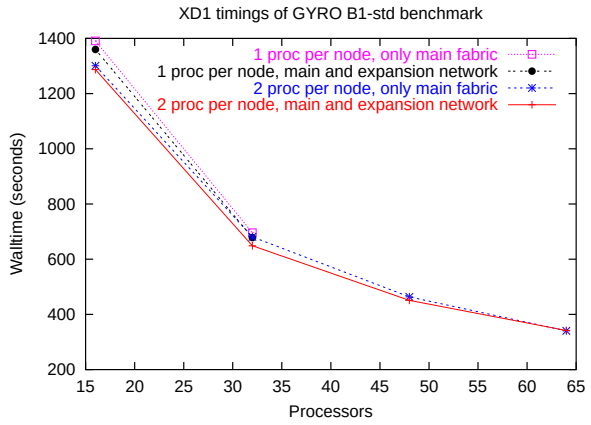

Figure 16: GYRO B1-std benchmark run in various modes on the XD1.

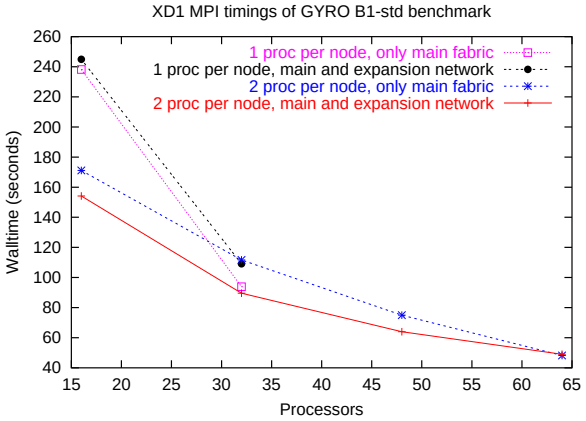

Figure 17: GYRO B1-std benchmark run in various modes on the XD1.

We also use this test case to compare the XD1 to the other test platforms. Figure 18 shows performance of GYRO across the test platforms. The data are plotted as timesteps per seconds against the number of processors. The XD1 compares quite favorably to its competitors.

5.4 VASP

VASP is the Vienna Ab-initio simulation package [4]. VASP is a package for performing ab-initio quantum-mechanical molecular dynamics (MD) using pseudopotentials and a plane wave basis set. The approach implemented in VASP is based on a finite-temperature local-density approximation (with the free energy as variational quantity) and an exact

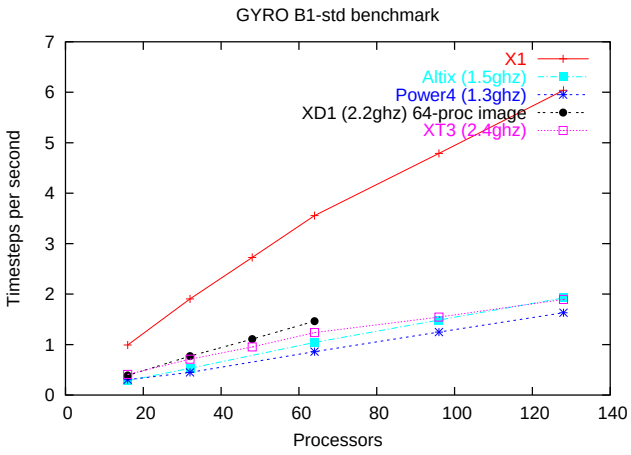

Figure 18: GYRO B1-std benchmark case showing timesteps per second versus processor count.

evaluation of the instantaneous electronic ground state at each MD-step using efficient matrix diagonalization schemes and an efficient Pulay mixing.

In Figure 19, the performance of VASP on a test case with 216 atoms is plotted across the various test platforms. The scaling is quite similar among the microprocessor based architectures with only the

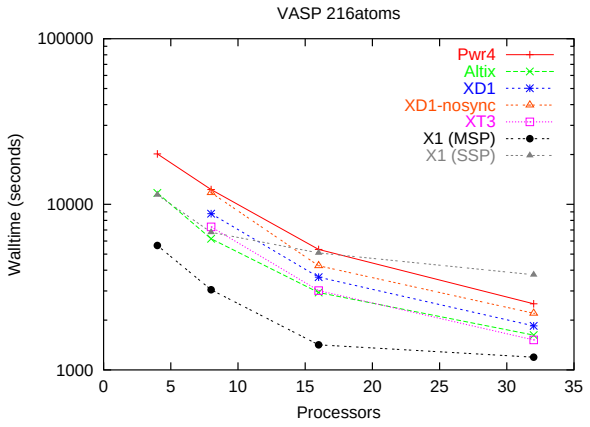

Figure 19: VASP walltime for 216 atoms test case.

vector architecture showing some odd scaling characteristics especially in SSP mode.

Figure 19 includes timing data on the XD1 with the Linux synchronized scheduler off. The data show for this application that the synchronized scheduler provides approximately a 20% reduction in walltime.

5.5 sPPM

sPPM [13] solves a 3-D gas dynamics problem on a uniform Cartesian mesh, using a simplified version of the Piecewise Parabolic Method. The algorithm makes use of a split scheme of X, Y, and Z Lagrangian and remap steps, which are computed as three separate sweeps through the mesh per timestep. Message passing provides updates to ghost cells from neighboring domains three times per timestep.

The standard sPPM benchmark case [13] involves a shock passing through a gas with a density discontinuity. The interaction of the shock and the discontinuity leads to the Richtmyer-Meshkov instability. It is decomposed on a 2304 x 2304 x 4608 grid totalling over 24 billion zones.

sPPM has been tested on numerous computer systems, and it is easy to scale the problem (weak scaling) to any number of processors. As we see in Figure 20, sPPM scales very well across four platforms. On the XD1, scaling from 4 processors to 64 processors has 98.3% parallel efficiency. Because sPPM sends very large messages infrequently, MPI latency impacts performance less than bandwidth.

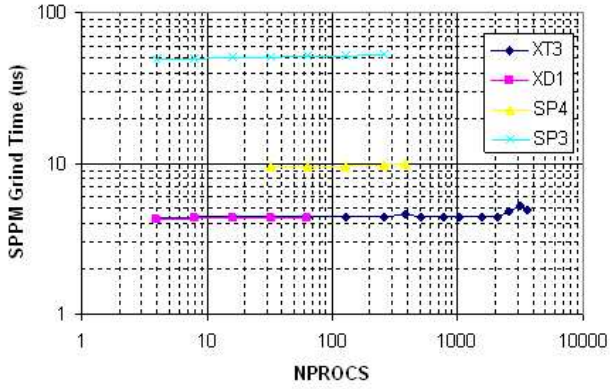

Figure 20: sPPM.

6 Conclusions

We have observed most notably that the XD1 has a very low MPI latency of 1.7 μsec. The Linux synchronized scheduler (LSS) results showed that this has a significant impact on performance; up to 20% improvement for the VASP application. The XD1 has outperformed competitor platforms that have a

higher peak Megaflop rating on several applications. Finally, the XD1 scales well for the tests we have run up to 64 processors.

The machine was upgraded one week before the conference to a 144 processor system. Several tests were run in this larger configuration, but there were anomalies in the data that were not understood and therefore not reported.

Future work includes investigating the affects of affinity provided by the `xd1launcher` command. In particular, the `xd1launcher` commands provides the user with a way to set the CPU affinity with or without the LSS. Each user process is bound to a single CPU. This was not available for most of the early tests. In addition, future work will continue efforts to achieve functional correctness with the CAM application. At this time, this appears to be a PGI 5.2-4 compiler issue. We will continue investigations with the PGI 6.0 compiler and the 2.1 Pathscale compilers.

Acknowledgments

This research was sponsored by the Office of Mathematical, Information, and Computational Sciences Division, Office of Science, U.S. Department of Energy under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

We thank the ORNL CCS XD1 system administrators, Nina Hathaway of Oak Ridge National Laboratory and Dave Londo of Cray, Inc., for all the hard work in setting up, configuring, and upgrading the machine in support of these tests.

About the Authors

Mark R. Fahey is a senior Scientific Application Analyst in the Center for Computational Sciences at Oak Ridge National Laboratory. He is the current CUG X1-Users SIG chair. Mark has a PhD in mathematics from the University of Kentucky. He can be reached at Oak Ridge National Laboratory, P.O. Box 2008 MS6008, Oak Ridge, TN 37831-6008, E-Mail: faheymr@ornl.gov.

Sadaf Alam is a post-doctoral research associate in the Future Technologies group, Computer Science and Mathematics Division at the Oak Ridge National Laboratory. She can be reached at the Oak Ridge National Laboratory, P.O.Box 2008 MS6173, Oak Ridge, TN 37381-6173, Email: alamsr@ornl.gov.

Tom Dunigan is a Senior Research Scientist in the Computer Science and Mathematics Division at Oak Ridge National Laboratory. He has been conducting early evaluations of advanced computer architectures since the mid 80's. When not testing new computer systems, he is investigating protocols for high-speed networks. He can be reached at Oak Ridge National Laboratory, P.O. Box 2008 MS6016, Oak Ridge, TN 37831-6016, E-Mail: thd@ornl.gov.

Jeffrey S. Vetter is a senior R&D staff member in the Computer Science and Mathematics Division of Oak Ridge National Laboratory, where he leads the Future Technologies Group. His research interests include experimental software systems and architectures for high-end computing. Vetter has a PhD in computer science from the Georgia Institute of Technology. He is a member of IEEE and the Association for Computing Machinery. He can be reached at the Oak Ridge National Laboratory, P.O.Box 2008 MS6173, Oak Ridge, TN 37381-6173, Email: vetterjs@ornl.gov.

Patrick H. Worley is a Senior Research Scientist in the Computer Science and Mathematics Division at Oak Ridge National Laboratory. He has been conducting early evaluations of advanced computer architectures since the early 90s. He also does research in performance evaluation tools and methodologies, and designs and implements parallel algorithms in climate and weather models. Worley has a Ph.D. in computer science from Stanford University. He is a member of SIAM and the ACM. He can be reached at Ridge National Laboratory, P.O.Box 2008 MS6016, Oak Ridge, TN 37831-6016, Email: worleyph@ornl.gov.

References

[1] Cray XD1 overview. http://www.cray.com/products/xd1/.

[2] Evaluation of early systems. http://www.csm.ornl.gov/evaluation.

[3] Pallas MPI benchmarks. http://www.pallas.com/e/products/pmb/.

[4] Vienna ab-initio simulation package. http://cms.mpi.univie.ac.at/vasp/.

[5] M.B. Blackmon, B. Boville, F. Bryan, R. Dickinson, P. Gent, J. Kiehl, R. Moritz, D. Randall, J. Shukla, S. Solomon, G. Bonan, S. Doney, I. Fung, J. Hack, E. Hunke, , and J. Hurrel. The community climate system model. *BAMS*, 82:2357–2376, 2001.

[6] P.N. Brown, R.D. Falgout, and J.E. Jones. Semicoarsening multigrid on distributed memory machines. *SIAM J. Sci. Comput.*, 2000.

[7] J. Candy and R.E. Waltz. An Eulerian gyrokinetic-Maxwell solver. *J. Comput. Phys.*, 186:545, 2003.

[8] George Carr. Private communication.

[9] W.D. Collins and et. al. P.J. Rasch. Description of the NCAR community atmospheric model (CAM 3.0). Tech Note NCAR/TN-464+STR, National Center for Atmospheric Research, Boulder, CO, 2004.

[10] T.H. Dunigan. XD1 evaluation. http://www.csm.ornl.gov/∼dunigan/xd1.

[11] P.W. Jones. The Los Alamos Parallel Ocean Program (POP) and coupled model on MPP and clustered SMP computers. In G.R. Hoffman and N. Kreitz, editors, *Making its Mark - The Use of Parallel Processors in Meteorology: Proceedings of the Seventh ECMWF Proceedings Workshop on Use of Parallel Processors in Meteorology*, Singapore, 1999. World Scientific Publishing Co. Pte. Ltd.

[12] John McAlpin. STREAM: sustainable memory bandwidth in High Performance Computers. http://www.cs.virginia.edu/stream.

[13] A.A. Mirin, R.H. Cohen, B.C. Curtis, W.P. Dannevik, A.M. Dimits, M.A. Duchaineau, D.E. Eliason, D.R. Schikore, S.E. Anderson, D.H. Porter, P.R. Woodward, L.J. Shieh, and S.W. White. Very high resolution simulation of compressible turbulence on the IBM-SP system. Proc. SC99: High Performance Networking and Computing Conf. (electronic publication), 1999.

[14] P.J. Mucci, K. London, and J. Thurman. The cachebench report. Technical report, University of Tennessee, Knoxville, TN, 1998.

[15] J.M. Rosinski and D.L. Williamson. The accumulation of rounding errors and port validation for global atmospheric models. *SIAM J. Sci. Comput.*, 1997.

[16] P.H. Worley and I.T. Foster. Parallel spectral transform shallow water model: A runtime-tunable parallel benchmark code. In *Proceedings of SHPCC '94*, pages 207–214. IEEE Computer Society, 1994.