Early Experiences with the Naval Research Laboratory XD1

Wendell Anderson, *Naval Research Laboratory (Code 5593)*, **Dr. Marco Lanzagorta** *ITT Industries,* **Dr. Robert Rosenberg** *NRL (Code 5593)* and **Dr. Jeanie Osburn** *NRL (Code 5592)*

ABSTRACT *Scientists at the Naval Research Laboratory (NRL) are engaged in a broad spectrum of research. In order to provide the high performance computing resources to support that work NRL has recently obtained a two cabinet XD1 with 288 Opteron 275 dual core CPUs and 144 Vertex II FPGAs. This paper will examine the applicability of the XD1 to these scientific problems.*

KEYWORDS: XD1, FPGA, MTA-2, Dual Cores

1. Introduction

The Center for Computational Sciences of the Naval Research Laboratory (NRL) is a leading edge distributed center of the High Performance Computing Modernization Program Office (HPCMPO) of the United States Department of Defense. As such, the center obtains and evaluates high performance computer systems that are not available at the HPCMPOs major resource centers (MSRCs). For the past several years NRL has been evaluating the Cray Multi-Threaded Architecture MTA-2. Last year the system was replaced with a Cray XD1. The choice of the Cray XD1 was based on a desire to continue the evaluation of multithreading opportunities provided by multiple core architectures and Field Programmable Gate Arrays (FPGAs) while providing additional computational resources to NRL and DoD scientists. The Cray XD1 with 288 Opteron 275 dual core CPUs and 144 Xilinx Vertex II FPGAs was delivered in September 2005 and accepted in February 2006.

2. XD1 System

The Cray XD1 is a modular high performance computing system with the base unit consisting of a chassis with up to six nodes. Each node of the NRL system consists of two Opteron 275 2.2 GHz dual core processors with 8GBs of shared memory, a 73GB 10K rpm 3.5 in. SATA drive and a Xilinx Vertex-II Field Programmable Gate Array. The nodes are interconnected by a 96 GB per second nonblocking switch fabric capable of supporting MPI communications of up to 8GBs per second with a 1.7 microsecond latency.

Each node contains an active manager system interconnected via 100Mb Ethernet. The active manager uses a real time operating system to oversee the operation of each node in the system.

Each node runs a Cray modified version of SuSE Linux (kernel 2.65.5). The XD1 has both the gnu and Portland Group Fortran and C/C++ Compilers available. MPI support is provided

through mpich 1.2.6. In addition, users may improve the performance of their applications by using the tuned AMD Core Math Library (ACML) or the Cray Scientific Library.

In addition to the local disks on each node, the XD1 also has available 15 TBs of fibre channel external disk storage. This space is available from any of the processors via the Lustre disk system.

User access to the XD1 is available through an XD1 node dedicated to logins. One additional node is devoted to the control of the active manager and two more nodes are dedicated to supporting the Lustre disk system. The remaining 140 nodes are compute nodes that are available to users only via the PBSPro batch queuing system.

In May 2006, the system will be expanded by adding another cabinet with 144 dual core processors and 6 Xilinx 4 FPGAs. Also another disk controller will be added doubling the available disk space on the Lustre disk system to 30TBs. The system then will be configured with one node dedicated to the active manager, one to user logins, four to the Lustre disk system, and 210 to the batch system. NRL is also looking at purchasing several software packages that will allow the user to program the FPGAs using a high level language.

3. XD1 Acceptance

The XD1 had to meet seven criteria before it would be accepted by NRL. First, a set of hardware diagnostics that tested all the processors, data paths, memory, and FPGAs had to run for 48 consecutive hours with no failures. Second, a single process (in this case an FFT) had to be run through the batch queue across all 560 processors on the compute nodes. Third, a code (again an FFT) using the FPGAs had to run successfully on the XD1. Fourth, the XD1 had to demonstrate the ability to read and write data to the Lustre disk system at a rate of at least 500 MB per second. Fifth, the three codes that ran most often on the NRL MTA had to run on the XD1 with equal or better wall clock times using 10 nodes (40 processors). Sixth, the XD1 had to run for 24 hours without crashing while executing a mix of applications on the compute nodes. Finally there was a 30-day acceptance where the system was made available to NRL users. During this time, the machine had to be available 85% of the time during normal business hours and could have no more than 5 failures within 24 hours of being returned to NRL for general use.

In February 2006 the XD1 was certified as satisfying all of these requirements. Further information on the MTA applications, 24-hour tests, and I/O rates are given later in this report.

4. MTA-2 Applications

Part of the acceptance criteria for the XD-1 was that three of the NRL applications (STATIC, CAUSAL, and LANCZOS) that were running on NRL's 40 processor MTA would run faster on forty processors of the XD1. As seen from Table 1 in fact all of the applications did achieve this requirement when run on all four processors of ten nodes.

Application	MTA (seconds)	XD1 (seconds)	Speedup
STATIC	9529	378	25.2
CAUSAL	936	293	3.2
LANCZOS	336	284	1.2

Table 1 MTA vs. XD1 Performance

STATIC [1] is a tight-binding code used to calculate the lowest energy configuration of the spins of a set of atoms. The calculation is done for a lattice of a face-centered cubic structure solid with n^3 atoms. For each point in the lattice the eigenvalues of a complex Hermitian matrix of size $9n^3 x 9n^3$ must be calculated, once for an up spin and once for a down spin. Most of the

time running STATIC is spent in zhpgv, the LAPACK routine that calculates all of the eigenvalues of a Hermitian matrix stored in a packed format. In the case benchmarked, n was six and the lattice contained 108 points. On the XD1, the code is embarrassingly parallel and is run by having each processor calculate the eigenvalues of a subset of the matrices. However on the MTA, the 216 matrices did not provide enough parallelization to effectively use all of the threads available on the MTA, so the eigenvalue calculation for a single matrix was spread over multiple processors. This method of parallelization was much less efficient than the XD1 effort as some portions of the zhpgv routine required serial processing. Thus the speedup of the static code is nearly 2.5 times the 10-fold increase of the XD1 clock over the MTA.

CAUSAL [2] solves the modified linear wave equation by using a fourth order in time and space 2-D Finite Difference Time Domain (FDTD) scheme with the addition of a convolutional propagation operator to incorporate attenuation and dispersion. Most of the running time of CAUSAL is spent in one of two kernels – the FDTD kernel that is calculated at every point of the grid and the causal correction that is applied to only those grid points located in an area where the medium is dispersive. Both of these kernels have nearly a one to one ratio between the number of floating-point operations and the number of memory accesses required. There is little reuse of data (especially for the causal calculation) and thus many references to memory result in cache misses. On the MTA, the code was run in parallel by assigning the calculations for each point to a separate thread. This provided the MTA with millions of threads to schedule. On the XD1, parallelization was achieved by dividing the grid into bands with each band containing a set of contiguous depths. Load balancing was attempted by setting the size of each band so that each band had roughly the same amount of work to do (bands containing a significant number of dispersion points had fewer depths than bands containing no dispersion points). The benchmark was run for 5000 time steps on a 2000 by 2000 grid with 118,300 dispersion points. Due to the poor reuse of cache and the difficulty and coarseness of the load balancing among processors, the XD1 code ran only three times faster than the MTA code.

LANCZOS is a code for estimating the largest and smallest eigenvalues of a Hermitian Sparse matrix such as those generated in the analysis of quantum systems [3]. The code generates the elements of a real tridiagonal symmetric matrix whose eigenvalues approximate the eigenvalues of the original matrix. The algorithm spends over 99% of its time in the multiplication of a sparse matrix and a dense vector. The MTA executes the code in parallel by dynamically assigning to a single thread the multiplication of a single row of the matrix and the vector. The XD1 MPI implementation divides the problem into parts that can be executed in parallel by statically breaking the original matrix into submatrices containing complete rows and performing the various submatrix-vector multiplies on separate processors. This method requires that the entire dense vector be available on each processor. The case used for the performance evaluation generated the coefficients of a 100 by 100 tridiagonal matrix for a 12 million by 12 million complex matrix containing about 35 non-zero elements per row. Of the three MTA applications LANCZOS provides the greatest performance challenge to the XD1 as the large size and sparseness of the matrix provides for little cache reuse, accesses to an element of the dense vector often results in the loading of adjacent elements that are not used, and a dense vector must be propagated to each processor at each step. Thus even though the clock rate of the XD1 is 10 times that of the MTA the running time of the program only decreased by 20%.

5. Dual Cores

After years of chip manufacturers speeding up the basic clock rate of their processors according to Moore's law (a factor of two every 18 months), the last few years have seen a much slower rate of growth in clock rates. In order to provide more processing power in the same space, both Intel and AMD have produced dual core memory chips and are planning on going to quad cores in the near future. The XD1 has the earliest of the AMD dual cores – the Opteron 275. While the 275 has the same L1 and L2 cache for each core as the single core version of the chip, the dual cores share the same DDR memory controller as the single chip processor. This sharing of memory bandwidth can lead to a degradation of the performance of codes running on the dual core chips.

In order to further investigate and quantify the effect of sharing the memory bandwidth between cores, several applications were run with three different scenarios: n nodes using all 4 cores on the node, 2*n nodes using only one core of each dual core processor, and 4*n nodes using only one processor per node. In all cases, the applications were run using the PBSPro #excl option to ensure that no other jobs were run on these nodes at the same time. The first three applications analysed were the three MTA applications mentioned in Section 4. In addition, the NRL applications NRLMOL and ARMS, an Air Force application NOZZLE, and the AVUS, HYCOM, OOCORE, and RFCTH2 programs from the TI05 benchmarks were evaluated.

NRLMOL [4] implements the Density-Functional formalism for clusters and molecules by using MPI to parallelize the problem by using a master process to distribute work to slaves. All parallelism in NRLMOL is carried out through this master/slave relationship. The problem chosen for the benchmark was the light-emitting molecule described in [5], the largest problem ever solved using NRLMOL.

The Adaptive Refined Magneto-hydrodynamic Solver (ARMS) performs three-dimensional, time-dependent simulations of solar magnetic storms. The ARMS is a massively parallel, flux-corrected transport based code built upon Message Passing Interface communications and NASA Goddard's PARAMESH parallel adaptive meshing toolkit.

The NOZZLE program performs a numerical simulation of a Coanda wall jet experiment with or without turbulence modeling. This MPI code solves the compressible Navier-Stokes equation on structured multi-block domains using a domain decomposition model for parallel processing.

The AVUS program performs anisotropic 3D CFD calculations over unstructured grids to model turbulent viscous flow over geometries. The standard test case is a wind tunnel model geometry for a constant cross-section wing with a partial span flap and end plates on either end of the wing.

The Hybrid Coordinate Ocean Model (HYCOM) encodes a primitive equation for the numerical simulation of a general ocean circulation model. This code can be used as a global ocean data assimilation system or as the ocean component of a coupled ocean-atmosphere model.

The Out-of-Core OOCORE program uses the ScaLAPACK, BLACS, and BLAS routines from the Cray Scientific Library to factor and invert large matrices and solve large linear systems. For large matrices OOCORE uses efficient I/O routines to write and read parts of the matrix to scratch files on disk.

The Reduced Functionality CTH2 (RFCTH2) code is, as its name suggests, an incomplete distribution of the cth2 code that is used to study the effects of strong shock waves on a variety of materials using many different models. RFCTH2 retains the basic shock hydrodynamics equations of CTH2, but removes the most advanced equations describing the behavior of

the material, replacing them with simpler models.

In all cases running times for a particular application were the same for both the 1-processor per node and 2-processor per node runs. The timing results comparing the two processors per node and the four processors per node are given in Table 2

Application	One Core	Both Cores	Efficiency %
STATIC	313	450	56
CAUSAL	275	293	93
LANCZOS	771	1371	22
NRLMOL	14283	16260	90
ARMS	2090	2524	79
NOZZLE	27498	27286	101
AVUS	1197	963	120
HYCOM	823	849	97
OOCORE	5274	7716	54
RFCTH2	279	448	39

Table 2 Dual Core Efficiency

Since the one core and two core cases were run over the same number of processors, the running times assuming no memory contention should have been the same. In order to determine how close we came to obtaining this ideal, the efficiency is given by

$$100*(1-(T_4-T_2)/T_2)$$

where T_2 is the wall clock time for running the code on N nodes using one core per processor and T_4 is the wall time running it on N/2 nodes using all the cores on the nodes. Since the same number of processors are used in both cases, if the times T_2 and T_4 are the same, we have perfect efficient (100%). If T_4 is twice T_2 then there is no advantage as running two copies of the application, each on N/2 nodes would finish at the same time as running the application consecutively on N nodes using only one-half of the cores and the efficiency is 0%.

6. I/O

The running time of applications on high performance computers can be significantly influenced by the time it takes to move data between memory and disk. Users may need to read input data, write out calculated results, store and retrieve data on temporary scratch space, and save restart files. A program running on the XD1 has two disk systems available – a high-speed parallel file system Lustre available to all the nodes on the XD1 and the local low speed SATA disk drives that are available only to the local node. Generally input, output, and restart files will be stored only on the Lustre disk system, as they need to be available to the user from any node including the login node. Temporary scratch files however may be written to the local disk and deleted when the run is completed. Although the data rate to the local disk is slower, the application does avoid competing with other nodes in the system that may also be writing to the Lustre system.

Part of the acceptance criteria for the XD1 was that users had to be able to read and write to the XD1 of rates of at least 500 MBs per second. To measure the I/O rate, an MPI program was developed by Cray and modified by NRL to allow multiple nodes to read and write data to/from any disk system available to the compute nodes. Because Linux caches disk data in local memory and the Lustre file system also provides a caching mechanism, the tests were run with parameters to ensure that the data was actually read/written to the disk and did not only reside in a cache.

For the acceptance test, an MPI program was run that read and wrote 8192 records of length one Megabyte from 12 different processors to the Lustre disk system The I/O between the XD1 and the Lustre disk system was measured at just under 900 megabytes per second in each

direction. In order to measure the scaling of writing data to the local disks and to Lustre, the program was rerun on 1, 2, 4, 8, 16, and 32 nodes. The I/O to the Lustre disk system saturated at a little less than 900 Megabytes by the time 16 nodes were used. Since the read and writes to the local disks for any node are independent of the read and writes for any other node, the I/O rates to the local disks continued to grow linearly with the number of nodes.

NODES	Read (MB/sec)	Write (MB/sec)
1	206	165
2	325	324
4	629	646
8	794	709
16	892	862
32	859	893

Table 3 Lustre I/O rates

NODES	Read (MB/sec)	Write (MB/sec)
1	46	58
2	98	105
4	114	209
8	273	406
16	374	804
32	720	1565

Table 4 Local Disk I/O rates

7. 24 Hour Software Test

The 24-hour software test was designed to keep nearly all of the compute nodes busy by insuring that jobs were always available on the PBSPro queue for execution. The mix of jobs included STATIC (on 16, 30, and 108 processors), CAUSAL (40 and 100 processors) a single processor application FLUX, an adaptive mesh code ALLA (40 and 100 processors), and the I/O test program (12 processors) writing data to the Lustre disks. Table 5 provides a synopsis of the application runs. Note that for any given application and number of nodes, times for most of the runs fall within fairly narrow range of times, although for all the applications the maximum time is much larger than the median. Table 6 provides the same measures for the writing and reading of data by the I/O test program to the Lustre disk file system.

Application	Nodes	Median	Min	Max
ALLA	25	878	851	1441
CAUSAL	25	88	83	191
FLUX	1	1872	1700	2127
STATIC	26	133	122	251

Table 5 Application Running Times

Operation	Median	Minimum	Maximum
Read	953	652	988
Write	821	511	982

Table 6 I/O rate (MB/sec)

8. Scaling

The usefulness of high performance computers is highly dependent on the ability of the system to scale as the number of processors used to run the application increases. To evaluate the scaling of the XD1, five of the applications evaluated in Section 4 were run on the XD1 using all 4 cores on each node for different number of nodes to determine how the applications scaled. The results are given in the five tables below. In each case, the wall clock time decreased considerable as the number of processors applied to the problem increased. In fact, in three cases, AVUS, NOZZLE and OOCORE super-linear scaling was obtained.

Nodes	Time	Speedup (actual)	Speedup (% of ideal)
8	4670		
16	1992	2.34	117
24	1276	3.66	122
32	963	4.85	121

Table 7 AVUS scaling

Table 11 RFCTH scaling

Nodes	Time	Speedup (actual)	Speedup (% of ideal)
6	4154		
12	2459	1.69	84
15	2358	1.76	70
20	1569	2.65	79
24	1152	3.61	90
28	1105	3.76	81

Table 8 HYCOM scaling

Nodes	Time	Speedup (actual)	Speedup (% of ideal)
4	60495		
8	27286	2.22	110
16	13930	4.34	109
32	6701	9.03	113
64	3032	19.95	125

Table 9 NOZZLE scaling

Nodes	Time	Speedup (actual)	Speedup (% of ideal)
4	5953		
8	3102	1.92	96
12	2005	2.97	99
16	1427	4.17	104

Table 10 OOCORE scaling

Nodes	Time	Speedup (actual)	Speedup (% of ideal)
8	4670		
16	1992	1.40	70
24	1276	1.87	62
32	963	2.30	57

8. Field Programmable Gate Arrays

Field Programmable Gate Arrays offer the potential for speeding up by an order of magnitude or more the execution of specific kernels. Recent applications of FPGAs in the HPC world include modelling of electro-magnetic phenomena using FDTD methods [6,7], DNA matching [8], molecular dynamics [9], astrophysical n-body problems [10], encryption [11], and lattice gas problems [12].

Until recently FPGAs could only be programmed at very low levels using VHDL or Verilog. Recently several software products including those of Celoxica, Impulse, and Mitrionics have developed C-like Higher Level languages for programming FPGAs. Another company DPSLogic has developed software based on Matlab and Simulink to ease the burden of programming these devices. Cray has provided I/O core libraries and XD1 API's to allow users to more easily move data between the Opteron memories and the memories on the FPGA boards.

9. FPGA Example

In order to evaluate the application of FPGAs to applications of interest to NRL scientists, we have started a preliminary examination of the Landmarks ISOMAP algorithm [13-14] used to generate images from hyperspectral data of dimensionality between 64 and 224. This algorithm is used to model the non-linear effects in hyperspectral imagery. A typical image is of the order of N= 10^6 pixels with the color at each point determined by the values of the multi-dimensional hyperctral data. The ISOMOP algorithm is based on selecting a

subset of Landmark points of size L << N, determining the proper image value for these points and then processing all N points as the interpolated values from the three or four L points closest to the point. The algorithm used by NRL to find the nearest landmark neighbors to a point requires answering the following question millions of times: Does the point $p=(p_1, p_2, \ldots, p_n)$ lie within a hyperspherical shell centered at $x=(x_0, x_1, \ldots, x_n)$ and radius r?

When the algorithm is run on a standard CPU, the points and radii are floating point numbers and the measure of the distance is the Euclidean norm. For each point and hypershere, the following loop was originally used to determine whether or not a point p_k lies within the hypershere with center x_i and radius r_i.

```
flag=FALSE;
sum=0.0;
For (j=0;  j< NUMDIM; j++) {
  diff=p[k][j]-x[i][j];
  sum=sum+diff*diff;
 }
  if (sqrt(sum) < r[i]) flag=TRUE ;
```

where r[i], x[i][j] and p[k][j] are floating point numbers. The code can be speeded up significantly by using the square of the Euclidean distance and testing after each partial sum

```
flag=TRUE;
sum=0.0;
j=0;
while (j< NUMDIM && flag ){
  diff=p[k][j]-x[i][j];
  sum=sum+diff*diff;
   if (sum >= r[i]*r[i]  ) flag=FALSE ;
 }
```

Since the data was originally collected as 16-bit fixed-point numbers, the FPGA code can be written to use 16 bit unsigned numbers for the input data. Also instead of a hypersphere, the choice of a hypercube would avoid the use of the multiple that produces a 32-bit result:

```
flag=TRUE;
j=0;
while (flag==0 && j< NUMDIM) {
  testlo=x[i][j]-r[i];
  testhi=x[i][j]+r[i];
  if (!((testlo < p[k][j]) && (p[k][j] < testhi)))
   flag=FALSE;
  j++;
}
```

The if statement within the while still presents a challenge to coding the FPGA. One possible solution is to test only a few dimensions at a time and then continue the testing on those hypercubes that may contain the point until all dimensions have been exhausted. Over the next few months, we will test various forms of these algorithms on the XD1 to determine those that do indeed provide the best performance gains.

9. Acknowledgments

The NRL XD1 was purchased with funds provided through the DOD High Performance Computing Modernization Program Office. All benchmarks in this report were run on that machine. The authors also wish to acknowledge the many scientists including Dr. Spiro Antiochos, Dr. Chris Bachman, Dr. Tunna Baruah, Dr. Richard Devore, Dr. Andreas Gross, Dr. Stephen Hellberg, Dr. Michael Mehl, Dr. Guy Norton, Dr. Mark Pederson, and Dr. Alan Walcraft who willing shared their codes and expertise.

References

[1] **W. Anderson R. Rosenberg, and M. Lanzagorta,** "Experiences Using the Cray Multi-Threaded Architecture MTA-2", *DoD HPCMP Users Group Meeting*, Belleview, WA. May, 2005

[2] **G. Norton, W. Anderson R. Rosenberg, and M. Lanzagorta,** "Modeling Pulse Propagation and Scattering in a dispersive

Medium Using the Cray MTA-2", *DoD HPCMP Users Group Meeting*, Albuquerque, NM. June, 2005

[3] **W. Anderson, M. Lanzagorta, and C. Stephen Hellberg,** "Analyzing Quantum Systems Using the Cray MTA-2", *DoD HPCMP Users Group Meeting*, Knoxville, TN. May, 2004

[4] **M. Pederson, D. Porezag. J. Kortus and D. Patten,** "Strategies for massively parallel local-orbital-based electronic calculations", *Physica status Solidi,* KnB217 197 (2000)

[5] **T. Baruah, M. Pederson. and W. Anderson,** "Massively Parallel Simulation of Light Harvesting in an ", *DOD HPCMP Users Group Meeting*, Nashville, TN. June, 2005

[6] **R.N.Schneider, L. E. Turner and M.M. Okoniewski,** "Application of FPGA Technology to Accelerate the Finite-Difference-Time-Domain (FDTD) Method ", *Tenth ACM International symposium on Field Programmable Gate Arrays"*, Monterey, CA. February, 2002

[7] **R. Bodner,** "FPGA Accelerated Finite-Difference-Time-Domain Simulation on the Cray XD1 Using Impulse C", *Cray Users Group Meeting*, Lugano, Switzerland. May, 2006

[8] **B. Smith and E. Staler,** "Implementation of SMART for Fuzzy Mapping in Bioinformatics Applications", *Cray Users Group Meeting*, Lugano, Switzerland. May, 2006

[9] **B. Tsakam-Sotch,** "Molecular Dynamics Acceleration with Reconfigurable Hardware on Cray XD1: A System Level Approach", *Cray Users Group Meeting*, Lugano, Switzerland. May, 2006

[10] **R. Spurzem, A. Ernst, T.Hamada, and R. Nakasota,** "Astrophysical Particle Simulations and Reconfigurable Computing on the Cray XD1"", *Cray Users Group Meeting*, Lugano, Switzerland. May, 2006

[11] **J. Maltby, J. Chow, and S. Margerm,** "Accelerated FPGA Based Encryption", *Cray Users Group Meeting*, Albuquerque, NM. May, 2005

[12] **R. Minnich and M. Scottie,** "Early results from a lattice-Gas Simulation on the Cray XD1 FPGA", *Cray Users Group Meeting*, Albuquerque, NM. May, 2005

[13] **C. Bachmann, T. Ainsworth. And R. Fusina,** "Exploiting Manifold Geometry in Hyperspectral Imagery", IEEE Transactions on Geoscience and Remote Sensing, Vol 43, No 5, March, 2006

[14] **C. Bachmann, T. Ainsworth. and R. Fusina,** "Improved Manifold Coordinate Representations of Large Scale Hyper-spectral Scenes", to be published Fall, 2006