# HPCC Update and Analysis

Jeffery A. Kuehn
Oak Ridge National Laboratory
Oak Ridge, TN 37831
kuehn@ornl.gov

Nathan L. Wichmann
Cray, Inc
Mendota Heights, MN 55120
wichmann@cray.com

***Abstract:*** The last year has seen significant updates in the programming environment and operating systems on the Cray X1E and Cray XT3 as well as the much anticipated release of version 1.0 of HPCC Benchmark. This paper will provide an update and analysis of the HPCC Benchmark Results for Cray XT3 and X1E as well as a comparison against historical results.

## 1 Introduction

Over the last year, the Cray systems installed at the Center for Computational Sciences (CCS) at Oak Ridge National Laboratory (ORNL) have undergone significant changes. Last year the Cray X1 was a stable but slightly aging system and the Cray XT3 was a new and rapidly changing platform. Over the last 12 months, the ORNL/CCS Cray X1 has been upgraded to a Cray X1E and the Cray XT3 system has undergone a series of firmware and software upgrades (particularly changes to the programming environments), with minor field hardware modifications (component voltage adjustments, etc) which have dramatically improved the stability and performance of the system.

Together these systems form the backbone of the DOE Office of Science (DOE-SC) Leadership Computing Facility (LCF). The true measure of their effectiveness is the rate of scientific discovery, which is related loosely to the performance of the actual simulation workload. However, such full application benchmarks based on well understood simulation cases are problematic from multiple standpoints. The size and complexity of a full application workload suite for a large center such as ORNL/CCS is unwieldy both from the perspective of the volume of code required as well as the total execution time required for characterization. Moreover, the workload suite is not constant, but rather is expected to change with DOE-SC's annual allocation cycles. Additionally, the performance of a full application is not easily

correlated to architectural strengths and weaknesses and differences in system configurations from one site to another (particularly I/O hardware) can make cross-comparisons very difficult. Lower level benchmarks address many of these issues. They are generally compact, portable, and kept constant over a long period of time. In a relatively short execution time, they can provide a level of insight and detail into the behavior of the overall system which is directly attributable to characteristics of the system architecture, and they can exclude issues like configuration differences to foster comparison between systems. The primary weakness of the low level benchmark approach it is one step further removed from the desired metric of scientific progress than the collections of results of application benchmarks.

## 2 Machine Descriptions

### 2.1 Cray XT3

The Cray XT3 supercomputer [2] is Cray's third-generation MPP system, focused on high scalability and sustained application performance. Its principal attributes are its balanced system performance, usability, scalability, and ease with which it is upgraded.

The Cray XT3 can be constructed with up to tens of thousands of single processor nodes, or PEs, configured as either compute nodes or service nodes. Each node includes a 64-bit AMD Opteron processor with dedicated memory and a data routing resource. The Cray XT3 uses a simple memory model in which applications are distributed across any number of nodes, each node with its own processor and local memory, no shared memory. It uses DDR400 memory parts which can deliver up to 6.4 GB/second bandwidth, more than 1 byte per flop per processing element. It has a very low memory latency of near 50 ns, benefiting performance algorithms that require irregular memory access patterns.

Each node communicates with the rest of the system via a high bandwidth, low-latency 3-D torus interconnect network. The processor is connect to the interconnect through the Cray SeaStar communication processor and router chip. The SeaStar is designed to offload communications overhead from the Opteron processor to increase application efficiency and accelerate communications. The Opteron has a bidirectional injection bandwidth of more than 2 GB/s to and from the SeaStar chip, while each SeaStar can sustain a bandwidth of 6.5 Gbytes/s on each of the six links, one in each direction of the 3-D torus.

The Cray XT3, like previous Cray MPP systems, uses a microkernel on its compute PEs call Catamount to minimize system overhead and in turn reduce "jitter". The general rule is that the microkernel only runs when it is working on behalf of the application. Catamount eliminates SMP overhead, paging, and spurious daemons, ensuring high, reproducible performance.

For the benchmarks, we compare last year’s software stack of PGI 6.0 compilers, ACML 2.6 BLAS, and MPT 1.0 message passing libraries versus this year’s stack of PGI 6.1, ACML 3.0, and MPT 1.3.

### 2.2 Cray X1E

The Cray X1E supercomputer [3] is the second in a series of scalable vector systems from Cray. Like its predecessor, the Cray X1, it is built using high-speed custom vector processors high local memory bandwidth, and a very high-bandwidth, low-latency interconnect. However, the Cray X1E is a processor upgrade which doubles the number of processors, each running

at a higher clock frequency, in a system while leaving most of the rest of the components the same. It is meant to combine the performance of vector processors with the scalability of microprocessor-based systems. The Cray X1 is also the first vector supercomputer to be designed to scale to thousands of processors with a single system image.

The Cray X1E is a collection of SMP nodes, each composed of 4 Multistreaming Processors (MSPs). Each node offers 72 Gflops of peak performance in 64-bit precision. There are two logical SMP nodes that share a physical board and a memory system capable of delivering 200 Gbytes/s of DRAM bandwidth. The Cray X1E at ORNL has 32 Gbytes of memory per board, or 2 Gbytes of memory per MSP.

Each MSP is a collection of 4 Single Streaming Processors, or SSPs. Each SSP has its own independent scalar processor and a 2-pipe vector core capable of delivering 2 flops per pipe per cycle. The vector architecture has been modernized to have a large register set, double speed 32-bit capabilities, vector mask registers and a large number of latency hiding features include decoupling of scalar and vector and Load buffers. The 4 SSPs share a 2 Mbytes high bandwidth cache with also allows very fast synchronization between the 4 SSPs.

The Cray X1E nodes are connected via a 16 slice, modified 2-D Torus using direct memory addressing. This means that communications are done with direct memory loads and stores, where the lower bits of the address are used to determine which of the 16 parallel slices are used.

The current software stack on the ORNL/CCS X1E is PrgEnv 5.5 which was used for the current set of benchmarks, whereas the historical results cited are based on PrgEnv 5.4.

### 2.3 IBM BG/L

The IBM BGL supercomputer [4] is a massively parallel supercomputer from IBM. BGL's design has been optimized for a large number (1K to 64K) of low power nodes. Each node consists of two 700 MHz PowerPC 440 processors, with dual floating point units capable of 64 bit floating point multiply–add, for a theoretical peak performance of 5.6 GFLOPS per node. The nodes are connected by 5 separate networks: a 3D torus network (2.1 GB/s aggregate per node), a collective tree network (2.8 Gb/s aggregate per node with 1-3 connections to other nodes per node), a global barrier/interrupt network (1.5 microsecond latency), and two Gigabit Ethernet networks (control and connectivity). The two processors on each node can be used in two different modes: virtual node mode, and communication co-processor mode. In virtual node mode, the node is logically divided into two nodes with half of the node's memory associated to each processor, while in communication co-processor mode the second processor is dedicated to communication while the first is programmed by the user application. The BGL results were produced using the blrts_xlc 7.0 C compiler, MPICH 1.0.1 and 1.1 message passing libraries, and ESSL 4.2 for the BLAS.

## 3 HPCC Benchmark Description

The HPC Challenge benchmark suite [1] tests multiple system attributes which exhibit substantial impact on the real-world performance of applications on HPC systems. The design goal for HPCC was to augment the venerable LINPACK benchmark with additional tests to support a broader analysis of HPC architectures. Additionally, since locality of reference can

be a dominating factor in software performance on modern architectures, the selected tests examine how the architecture performs for high and low degrees of both spatial and temporal locality (see Figure 1). Collectively, the tests provide performance indicators for processing power, interconnect, and memory system performance. To do this, the test suite needed to use kernels that reflect real-world patterns of memory access and interprocessor data exchange. The resulting suite includes 23 tests in eight categories, and stresses not only the processors, but also the memory system and the interconnect maps each test category to the aspects of the HPC system it evaluates.

### 3.1 Network Parameterization

There are several different approaches to characterizing interconnect performance for HPC systems. The most thorough approach attempts to map the "performance surface" for several to all of the MPI data exchange routines as the processor count and message size are independently varied. The results are reported variously as either bandwidth or time dependent on whether the concept of bandwidth is well defined for the operation in question. While this approach is the most thorough, performing the required tests to a high degree of accuracy is resource intensive. A simpler method which attempts to characterize the zero-byte message latency and the asymptotic bandwidth runs much faster and provides much of the same information, since more complex MPI operations can be understood in terms of these fundamentals. For instance, we can *approximate* the time for a single communication as:

$$\text{time} = \text{latency} + \text{bandwidth}^{*}(\text{message size})$$

Though this approach fails to show us performance inflections which indicate points at which the MPI libraries transition from one algorithm to another (e.g. Eager vs. Rendevous), it captures enough of the essential character of the interconnect to indicate its impact on latency sensitive or bandwidth sensitive applications. HPCC takes this abbreviated approach of measuring latency and bandwidth, but recognizes that the communication pattern can play a role in determining these factors. Thus, the latency and bandwidth are measured for three different fundamental communication patterns.

#### 3.1.1 PingPong

This is the most basic communication pattern and involves a simple send-receive exchange between two processors. While frequently you will see vendors publish results for the MPI PingPong Latency test, as it is relatively easy to measure, the MPI test measures a transfer of only 1 byte between only 2 processors in a system. Additionally, because latency and bandwidth vary for some interconnect topologies depending on the number of network hops between the sending node and the receiving node, HPCC reports not only an average PingPong time, but also the minimum and maximum PingPong times. The minimum PingPong time will be representative of the latency and bandwidth to an MPI task running on the same node for systems which run multiple MPI tasks per node or to an MPI task running on the nearest network neighbor for systems which run only a single MPI task per node.

#### 3.1.2 Natural Ring

The Natural Ring test uses a more realistic scenario where all processors in the system

communicate with their “neighbors”, at approximately the same time. However, it should be noted that “neighbors” in the context of the benchmark are defined as MPI tasks whose ranks differ by one. Thus, for larger SMP nodes, “neighbors” are likely a task on the same node, whereas for uniprocessor nodes, “neighbors” are at least one network hop away. In systems with SMP nodes, as the processor count per node increases, this benchmark becomes increasingly weighted towards reflecting the latency and bandwidth within a node. Codes parallelized with domain decomposition generally represent the decomposition in one of the dimensions as adjacent MPI ranks with the communication in this dimension mapping well to the Natural Ring communication pattern.

### 3.1.3 Random Ring

The Random Ring test is similar to the Natural Ring, except that the order of the ring is "randomized" so that each processors "ring neighbor" is neither its logical neighbor as ordering in the MPI communicator, nor its physical neighbor in the machine, instead it is a processor chosen at random in the system. While some applications do perform communication between random nodes, even more regular problems exhibit some behavior consistent with the Random Ring pattern, as is the case for a domain decomposition in multiple dimensions. Specifically, only one of the dimensions can be represented by adjacent MPI ranks, and thus the others may be more topologically remote, mapping well to the Random Ring pattern.

Codes that send many small messages are very latency sensitive. For example the widely used ocean modeling code, Parallel Ocean Program (POP) from Los Alamos National Laboratory, and LS-DYNA engineering software from Livermore Software Technology Corp. (LSTC) are highly dependent on global summations, which require the processors to communicate frequently with small messages. The computing system’s latency is a significant factor in how well POP or LS-DYNA will run on that system, and therefore, Random Ring latency is a good predictor of real world performance for applications that place heavy demands on simultaneous communications.

## 3.2 Spatial vs Temporal Locality of Reference

The design of the HPCC benchmarks acknowledges that applications vary in the type and magnitude of the data locality intrinsic to their core algorithms. Two types of locality are recognized: temporal and spatial. Temporal locality refers to the likelihood of a single memory address being referenced several times in a short period, whereas spatial locality refers to the likelihood of adjacent memory addresses being referenced in a short period of time. Since these two types of locality are orthogonal, one can locate any algorithm on a 2D plot of spatial-temporal locality diagram (STLD) (see figure 1). 12 of the HPCC tests were selected because of their correspondence to the corners of this space, and the performance of these 12 tests provides us with a clearer picture of how the full application space will map to the architecture, than would be possible with but a single benchmark. (See figure 1).

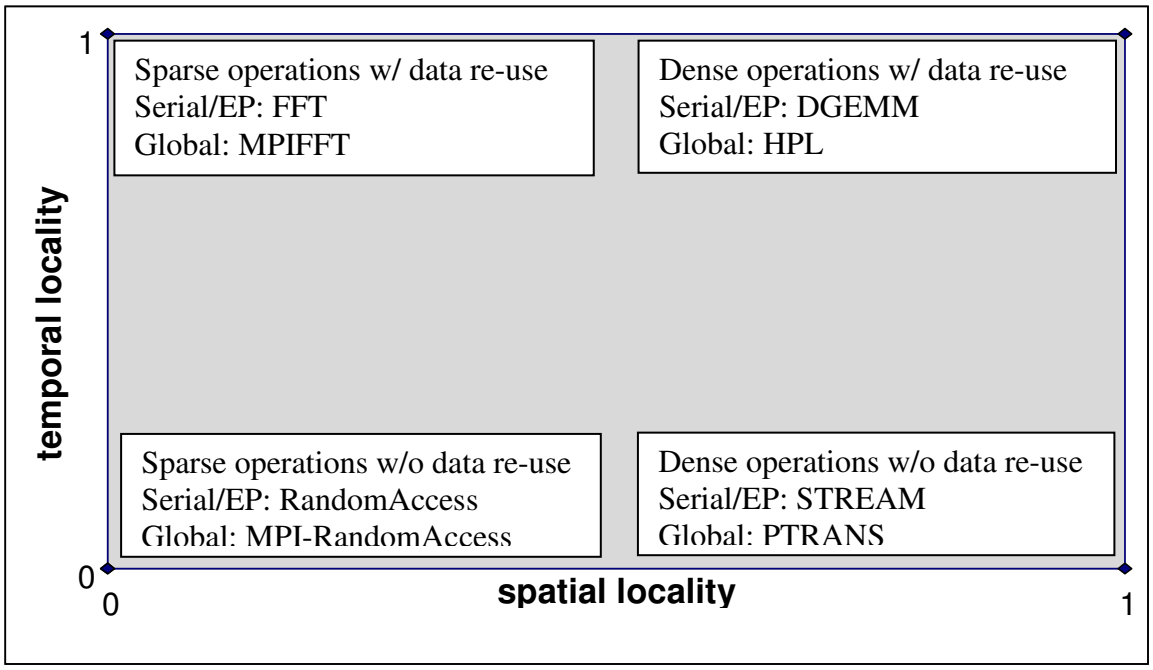


Figure 1: Spatial-Temporal Locality Diagram

### 3.3 Serial vs Embarrassingly Parallel vs Global

In addition to varying the spatial and temporal locality, the HPCC benchmark also recognizes three levels of parallelism. First, a serial (or "single") test involves only one processor running the test. Second, an embarrassingly parallel ("EP" or "*") test is run simultaneously by each of the processors, though without any communication required between the processors. And third, a globally parallel ("G") test is run in which all of the processors collaborate on a single test requiring communication. Thus, for each of the four corners of the STLD (Figure 1), we test the architecture's performance on each of these three levels of parallelism creating a total of 12 application kernels.

### 3.4 High Spatial and High Temporal Locality

Dense linear algebra was chosen to represent those algorithms which exhibit both high spatial locality and high temporal locality. DGEMM measures the floating point execution rate of double precision real matrix-matrix multiplication in GigaFLOPS per second (GFLOPS) using the DGEMM BLAS library routine. DGEMM is used for both the single and EP tests, as it has near maximal spatial and temporal locality. DGEMM is the purest measure of processor floating point performance, as this test is very coarse grained, and places no load on the interconnect and almost none on the memory system. DGEMM is useful as a measure of the very best performance one could ever achieve on highly computational intensive codes.

The High Performance LINPACK test (HPL) is used for the global test in this quadrant. It primarily tests processor performance and is included in this suite to allow comparison with the Top 500, which is based on the very same test. HPL solves a randomly generated dense linear system of equations in double floating-point precision (IEEE 64-bit) arithmetic using MPI and is measured in teraflops, or trillions of calculations per second (TFLOPS). HPL has

the enviably characteristics of being very computationally intensive as well as requiring relatively little local or global memory bandwidth. As a result, many systems can sustain 70 to +80 percent of peak. Results for DGEMM correlate strongly with per CPU HPL results.

### 3.5 High Spatial and Low Temporal Locality

Algorithms dominated by data movement were chosen to represent those algorithms which exhibit high spatial locality but low temporal locality. The STREAM benchmark was chosen for the single and EP tests in this quadrant. STREAM is a simple synthetic benchmark that measures sustainable memory bandwidth to local memory, in Gigabytes per second, and the corresponding computation rate for a simple vector kernel. STREAM provides a good approximation for high memory bandwidth codes. It is worth noting that STREAM assesses bandwidth to memory only, and does not stress interprocessor communications, or I/O communications. For the global benchmark in this quadrant a parallel matrix transposition is used. The PTRANS (parallel matrix transpose) benchmark implements a parallel matrix transpose for two-dimensional block-cyclic storage. This is an important benchmark because it exercises the communications of the computer heavily on a realistic problem where pairs of processors communicate with each other simultaneously. PTRANS is a useful test of the total communications capacity of the network, measured in gigabytes per seconds (GB/s).

Several molecular dynamic codes and most climate models must transpose large arrays to perform multi-dimensional Fast Fourier Transforms (FFTs), relying heavily on the processor's ability to exchange data quickly. Applications such as CPMD (a computational chemistry code that simulates static and dynamical properties of solids, liquids and disordered systems), FPMD, and VASP molecular dynamics simulation codes and climate spectral models, are most likely to perform well on systems that do well on the PTRANS benchmark.

One problem the authors have seen with PTRANS is that performance can vary significantly with small changes in the parameter settings. This behavior has been observed on multiple platforms. To counter this, the HPCC input file provides for separate specification of the PTRANS blocking factors and problem sizes, however, there is no separate specification for the PTRANS decompositions. At runtime the configuration producing the best result is selected and reported. While this helps one get better numbers, the underlying behavior still can make it difficult to interpret PTRANS numbers.

### 3.6 Low Spatial and High Temporal Locality

Fast Fourier Transform algorithms were chosen to represent those algorithms which exhibit low spatial locality but high temporal locality. The FFT test measures the floating point rate of execution of complex one-dimensional Fast Fourier Transforms in double precision. The FFT is performed in each of the three (single, EP, global) modes. The global FFTE exercises the computation and the all-to-all communications capabilities of the computer, providing a useful test of the total communications capacity of the computers network, or interconnect. It is measured in Gigaflops per second.

The FFT implementation supplied in HPCC is a conversion of Fortran to C, as a result, a significant amount of dependence information is lost along the way. Furthermore, it is written in a non-vector friendly format. These two realities significantly hurt performance on vector machines compared to what is achievable. Moreover, on vector machines, the low spatial locality and high temporal locality are often overcome by performing sets of FFT's in parallel on the vector hardware. To overcome this problem, the optimized version run on the Cray X1E actually uses the original FFTE package written in Fortran. Furthermore, the FFTE

package has an option where it might use an algorithm better suited toward vector machines and using this algorithm greatly improves performance.

### 3.7 Low Spatial and Low Temporal Locality

This quadrant is the newest frontier of scientific algorithm research, including graph theory and adaptive mesh refinement (AMR). Excelling in this quadrant is challenging for most modern computers since the locality constraints provide few opportunities to amortize memory latency. A graph theory kernel called RandomAccess (RA) was chosen to represent those algorithms with both low spatial and low temporal locality. The Random Access test measures the rate at which the computer can update pseudo-random locations of a large table stored in its memory, expressed in billions (giga) of updates per second, or GUPS. By testing gather-scatter functions, the Random Access kernel provides an indication of how codes (including graph theory and AMR) that need frequent, random access to data, will perform on a given system. The RA benchmark is run in single (one table on a single node), EP (each node with its own table) and global (one large table spanning all nodes) modes.

## 4 Results and Discussion

A new series of HPCC benchmarks have been run on the Cray XT3 for problem sizes ranging from 16 to 4096 PEs, and new data collected on the Cray X1E. This data will be examined relative to previously published HPCC results from the HPCC public website [7] and other sources [2][3][5][6].

### 4.1 Network Parameterization

The HPCC benchmark measures latency and bandwidth for a two task Ping Pong pattern, and two ring patterns, one ordered naturally as implied by the MPI task ranks and one ordered randomly. For the Ping Pong test, multiple pairs are tested and reported as a minimum, average and maximum for both the latency and bandwidth. For a system built on single processor nodes with a reasonably good mapping of MPI tasks to nodes, one would expect the latency and bandwidth for the Natural Ring to closely track those of the Ping Pong test since the minimum latency and maximum bandwidth would typically be observed between nodes which fall closer together within the network. Whereas the Random Ring test typically generates communications between MPI tasks ranks that are further apart, and hence between nodes that are further apart within the network. The latency and bandwidth results on the Cray XT3, fail to follow this pattern (Figs 2 & 3). The Natural Ring latency and bandwidth track more closely with the Random Ring latency and bandwidth rather than the Ping Pong latency and bandwidth. Thus the task placement, the mapping of MPI task ranks to nodes on the network, behaves less like an expected good placement and more like a random placement, i.e. the placement is far from optimal. One needs not dig too far to explain the problem. The nodes, cabinets, and rows within the Cray XT3 system are numbered sequentially in each of these directions, presumably to simplify locating individual hardware components for maintenance. However, to minimize the maximum cable length, the wiring within the cabinets and between cabinets down a row are cabled in a serpentine fashion – in each of these directions, the odd numbered nodes are linked in a chain and the even nodes are linked in a chain, then the chains are linked at the ends. (Serpentine cabling is not currently used between rows because the cable length required to “skip” rows would have exceeded the original cable length restrictions. Modifications are in process which will allow toroidal connections in this direction as well.) Thus, with the exception of those nodes which are the endpoints linking the odd and even chains, nodes which are physically adjacent to

each other, are topologically more remote, potentially opposite each other on the torus ring. Thus, if nodes are allocated and mapped sequentially when allocating resources for a job, those nodes will be topologically remote from each other in the network, giving rise to the observed behavior. Also worth noting is that the latencies increase and the ring bandwidths decrease as the processor count increases. The latency increase is easily explained by the increasing network hop count required to span the larger job configuration. The bandwidth decrease is less intuitive, but in the case of the Random Ring test, is not difficult to imagine the ring wrapping back on itself due to the random place of tasks, leading to contention for the links. Because of the placement problems noted previously, the Natural Ring pattern is similarly impacted, though to a lesser extent. It should be noted that the benchmarks were not run on a dedicated system and the scattered placement implies that nodes belonging to other jobs may be interspersed within the nodes for the benchmark jobs as an additional source of link contention.

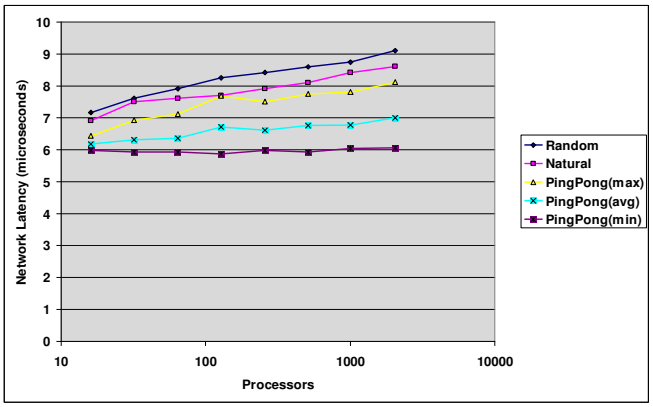


Figure 2: XT3 Latency Summary

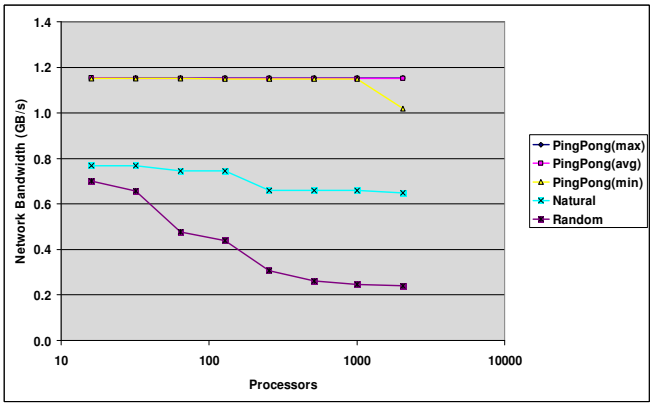


Figure 3: XT3 Bandwidth Summary

Figures 4 and 5 compare latency and bandwidth between systems and over the last year. The first characteristic to note on this chart (as well as the succeeding charts) is that with the exception of the new XT3 dataset, the lines connecting the data points are not intended to imply strong connection between the data points for those machines, but rather only suggest the general shape of the performance envelop. The three data points for the Cray XT3 above 20 microseconds of latency represented the state of Cray XT3 last year, the new Cray XT3 trend line runs in the 6-9 microsecond range – a significant improvement. The upgrade of the Cray X1 to a Cray X1E improved both latency and bandwidth for the system, and while its latency is still slightly worse than XT3's current latency, X1E's reputation as a "bandwidth" machine is clearly well deserved. Also interesting to note is the inverted relationship between latency and bandwidth on the XT3 and BGL – while BGL wins the latency competition, XT3 wins the bandwidth competition.

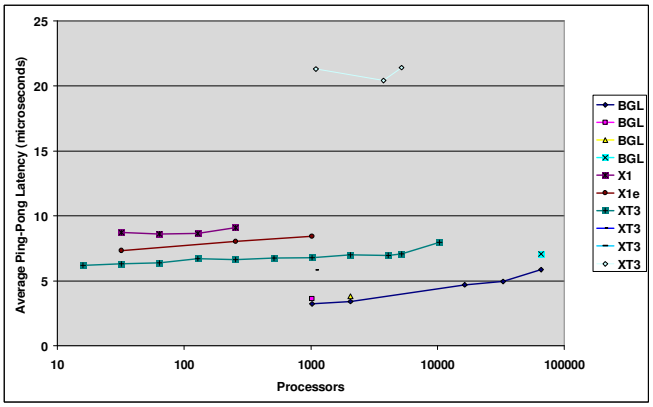


Figure 4: Ping Pong Latency Comparison

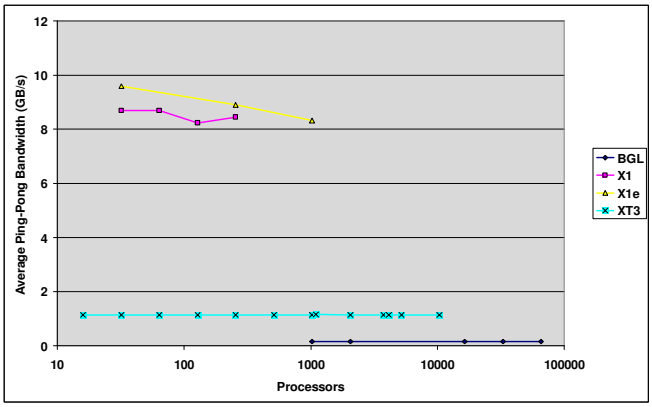


Figure 5: Ping Pong Bandwidth Comparison

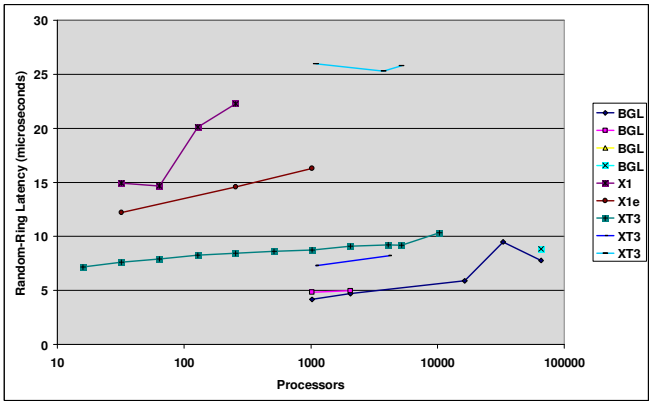


Figure 6: Random Ring Latency Comparison

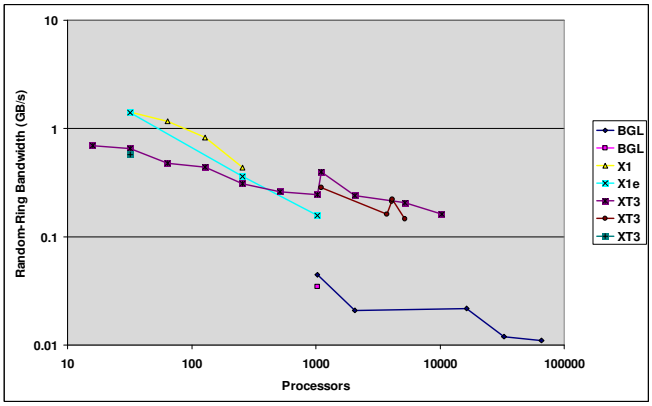


Figure 7: Random Ring Bandwidth Comparison

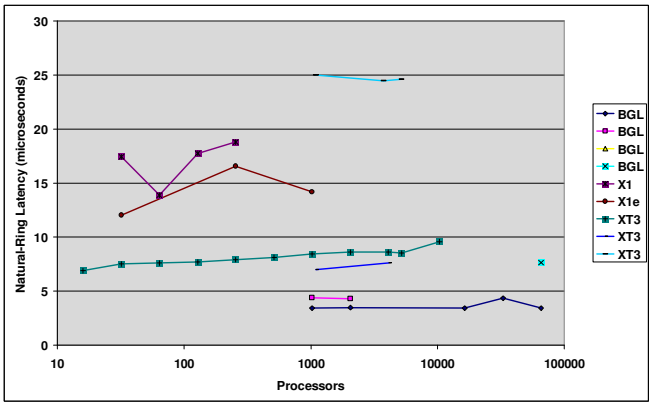


Figure 8: Natural Ring Latency Comparison

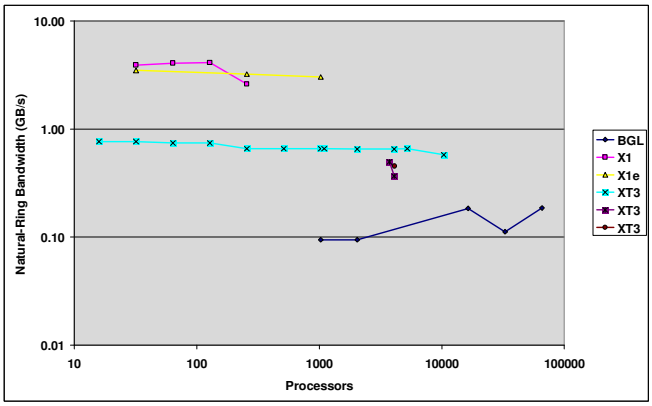


Figure 9: Natural Ring Bandwidth Comparison

In figures 6-9, additional points are included for XT3 which fall slightly below the May 2006 latency trend line. These data points were collected near the end of 2005 on an intermediate release of MPT which had slightly better latency, but still lacked some critical reliability features which when added, caused a slight increase in latency.

Relative to the earlier discussion of XT3 job layout, note the shape and trend of the BGL's

Natural Ring latency (and bandwidth) which tracks well with BGL's (minimum) Ping Pong latency (and Ping Pong bandwidth) and note the disparity between BGL's Natural Ring and Random Ring. Taken collectively, they show MPI natural ring behavior tracking closely to the optimal Ping Pong behavior and qualitatively differently from the Random Ring behavior, implying reasonably good MPI job layout on BGL.

### 4.1.1 High Spatial and High Temporal Locality

#### 4.1.1.1 DGEMM

Figures 10 & 11 show historical and current DGEMM in both serial and embarrassingly parallel modes, on each of the machines. Neither the X1E nor the XT3 shows a significant improvement from last year, primarily an indication of the maturity of the BLAS implementations on both platforms. (While there does appear to be a slight degradation on X1E, this is likely attributable to the new benchmarks having been run on a non dedicated system, whereas the old benchmarks were performed on a dedicated system.) Note that the BGL's lower frequency processor underperforms the other systems.

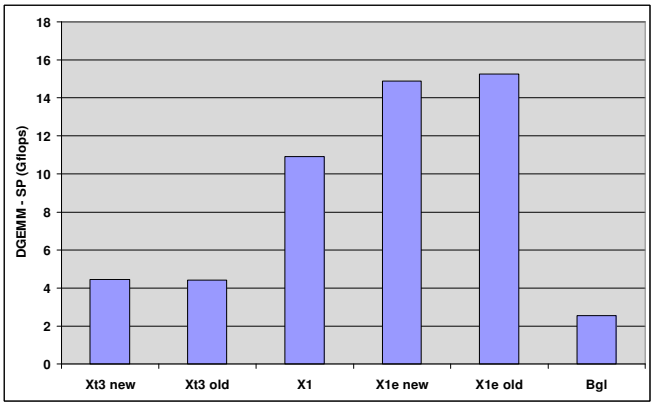


Figure 10: Serial DGEMM Comparison

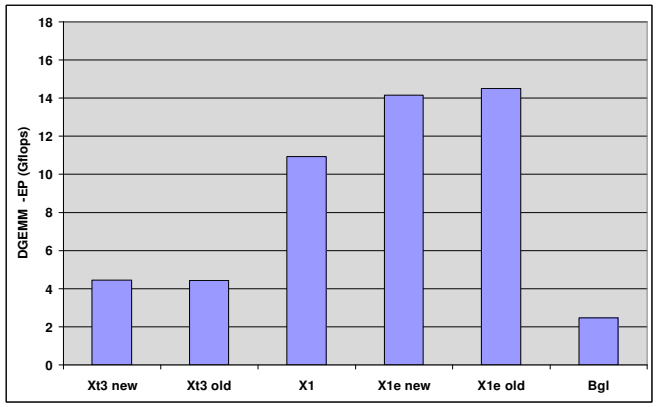


Figure 11: EP DGEMM Comparison

#### 4.1.1.2 HPL

Figures 12 shows HPL results for the compared systems. Three trend lines are shown for BGL for comparison, representing a mix of HPCC "base" and "optimized" (source code changed) results. The top BGL performance line compares well with the XT3 trend line, indicating that if both FPUs on both processors are used, BGL can be competitive on a node for node basis with XT3, though accomplishing this, without doubling the MPI task count and halving the memory available to each task, does require change to the benchmark source code. The lower BGL trend lines compares more directly with the others as they represent the case of no source code changes.

There are strong arguments both for and against "normalized" results for the HPCC benchmarks. On the "pro" side, normalizing the results removes "wallet size" from the performance equation, allowing comparison of results on machines of varying architecture and scale. On the "con" side, normalizing results removes "wallet size" from the performance equation. Several normalizations are possible relative the peak floating point rate, peak memory bandwidth, number of nodes, etc. but in the case of HPL, normalizing is essentially the same as quoting the percentage of peak floating point performance achieved. This is not

particularly new or controversial. For figure 13, the BGL normalizations need to be explained. Since BGL's nodes have two processors, the aggregate floating point speed of both is used to define the peak capability consistently across all the data points. While conventional wisdom often suggests that it is easier to obtain a larger percentage of peak speed with vector processors, the data tells a different story for the HPL benchmark: XT3's Opteron processors sustain the highest percentage of peak. Also worth noting is that while X1E achieved better absolute performance than X1, the percentage of peak sustained suffered from increasing the performance and number of processors without a commensurate increase in memory bandwidth.

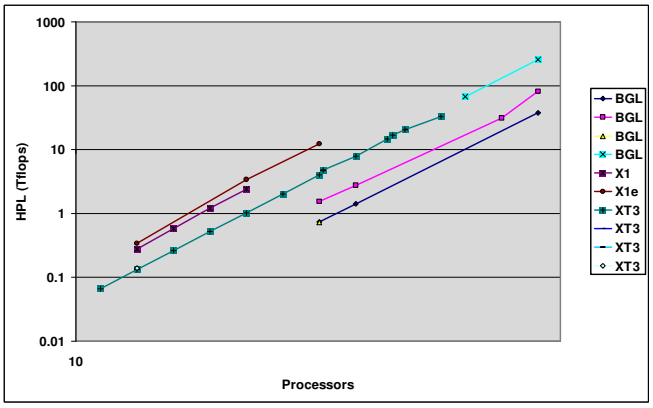


Figure 12: HPL Comparision

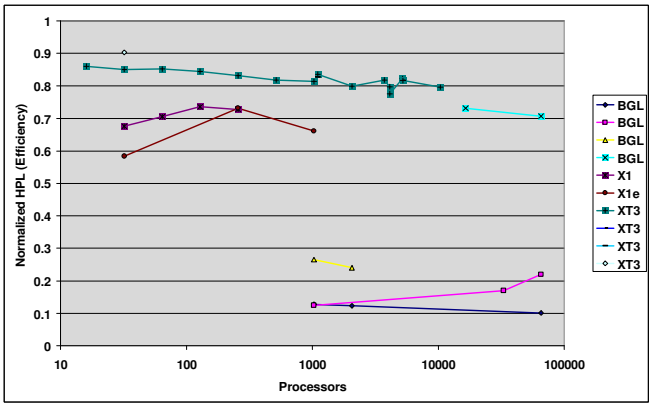


Figure 13: Normalized HPL comparison

### 4.1.2 High Spatial and Low Temporal Locality

#### 4.1.2.1 STREAM

The STREAM benchmark is a straightforward memory bandwidth test. The results are summarized in Figures 14 &15. The X1 to X1E upgrade increased the processor performance and doubled the number of processors, but did not double the memory bandwidth as can be seen in the charts. The serial version of the benchmark shows that an X1E processor can drive memory at higher bandwidth (closer to the peak memory bandwidth), than a slower X1 processor, which could not fully utilize the full memory bandwidth alone. The embarrassingly parallel benchmark however shows the impact of doubling the number of processors. While the X1E processors are able to derive a higher aggregate bandwidth from the entire memory system, the per processor bandwidth on the X1E is less than that of the X1. The slight drop from last year's (old) X1E STREAM results versus the new results is still under investigation, though likely attributable to the original benchmarks having been run on a dedicated machine, whereas the new results were run on a machine in normal batch production. Relative the XT3 and BGL, X1E still maintains its reputation as a "bandwidth machine", at least on a per processor basis, with XT3 falling just below 5GB/s on each of the benchmarks and BGL falling between 0.8-1.6 GB/s. Note that comparing the STREAM copy rate against the DGEMM rate (serial) for each of the machines also tells a story with X1 and X1E at 1.3 Bytes/FLOP, XT3 at 1.1 Bytes/FLOP, and BGL at 0.7 Bytes/FLOP. It may seem unusual that X1 and X1E both weigh in at 1.3 Bytes/FLOP, but consider the raw STREAM results, which suggest that for the X1, the ratio is limited by a single processor not being able to saturate the memory bandwidth, while X1E's performance

is processor limited with X1E showing a slightly lower percentage of peak on DGEMM.

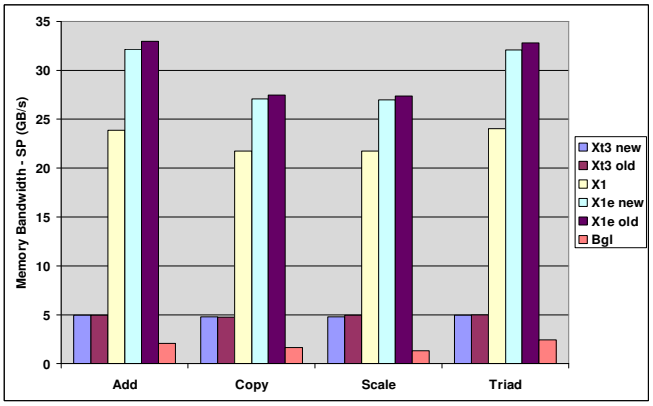


Figure 14: Serial STREAM comparison

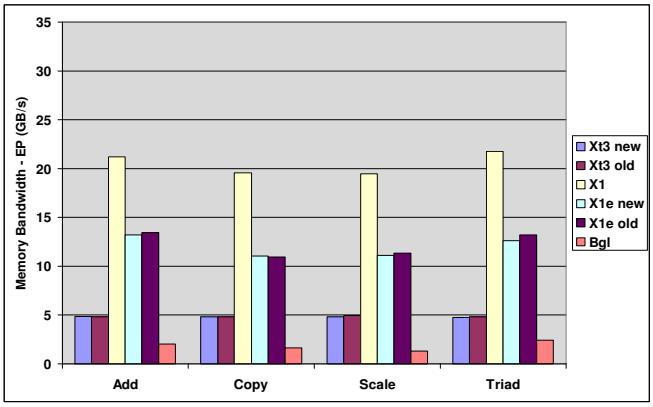


Figure 15: EP STREAM Comparison

#### 4.1.2.2 PTRANS

PTRANS results are summarized in Figure 16 & 17. The most unusual feature in the summary is the erratic behavior of XT3's PTRANS scaling curve. The "sawtooth" characteristic of this curve again implicates a topology problem, largely attributable to the job layout problem identified in the Network Parameterization section identified above. Also noteworthy is a data point for BGL, which seems unusually high for that system. The data point corresponds to an unusually small array size for the PTRANS problem, likely allowing the result to be dominated by cache and small message performance, but at the cost of a corresponding HPL result, which is a factor of 7 worse than a larger (more appropriate) problem size. Also noteworthy is that, if the results are normalized relative to the peak floating point rate (and hence number of processors), most of the machines see a sharp decrease in normalized PTRANS performance, though XT3's decline is much less severe, largely due to its high network bandwidth.

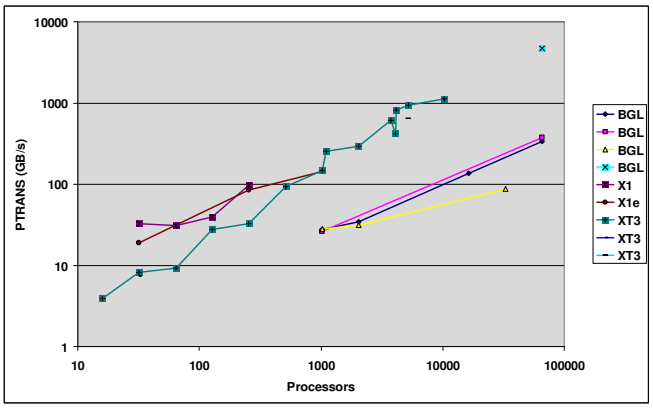


Figure 16: PTRANS Comparison

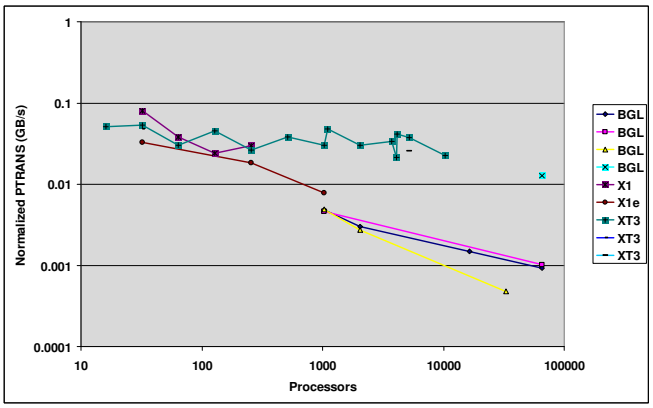


Figure 17: Normalized PTRANS Comparison

### 4.1.3 Low Spatial and High Temporal Locality

#### 4.1.3.1 FFT

The most striking feature of the summarized serial and embarrassingly parallel FFT results in Figures 18 & 19 is the dramatic performance improvement on the X1 / X1E which improved by a factor of almost 4 from X1 (PrgEnv 5.2) to X1E (PrgEnv 5.4) and almost doubled again from PrgEnv 5.4 to 5.5. While hardware improvements can account for a portion of this improvement, the majority of the credit goes to changes in the source to use the vectorized version of the original FFTE package, written in Fortran, rather than the same package translated into C. The advantages of this are two-fold. First, the language and coding of the original Fortran allow the compiler to perform more sophisticated dependence analysis, and ultimately more effective transformations and optimizations. Second, the vector version is written to take full advantage of the vector hardware on the Cray X1E. It is interesting to note here for comparison, that prior to this optimization the Cray X1 and BGL performed at roughly the same per processor rate on FFT, which will turn out to be important in understanding the MPI-FFT results.

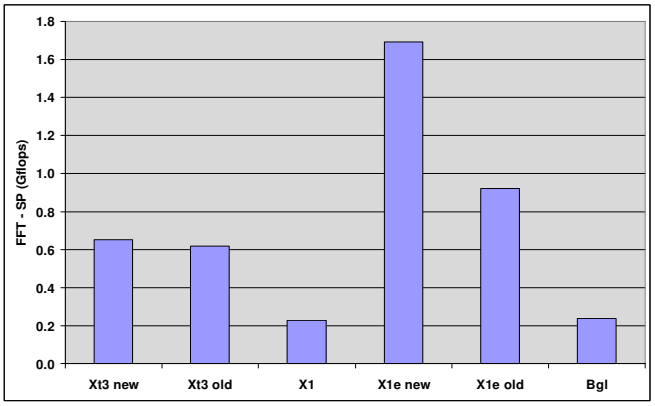


Figure 18: Serial FFT Comparison

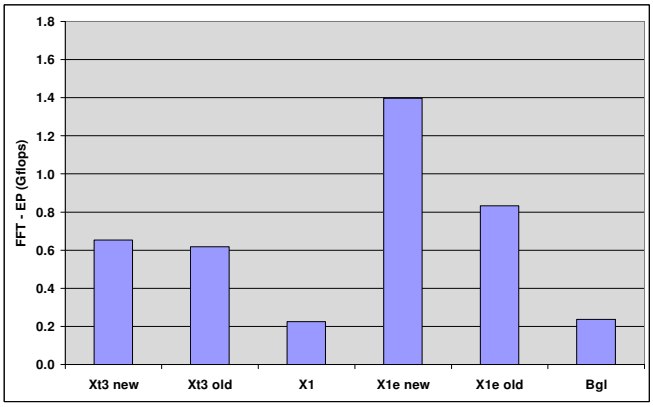


Figure 19: EP FFT Comparison

#### 4.1.3.2 MPI-FFT

Figures 20 & 21 summarize the results of the MPI-FFT benchmarks. Additional data points have been included at 3744 and 5200 processors from the previously published results on XT3 to illustrate the step-like behavior in the MPI-FFT benchmark which is currently restricted to using a “power of 2” number of processors. Thus this benchmark does not achieve the smooth scaling that such charts imply, but rather they have a characteristic stepping at each power of two. As noted above, the single processor FFT performance on X1 and BGL are similar, but as seen in Figure 21, the global FFT performance is rather different, with X1 showing quite well. Looking back to the network parameterization however, X1 has a much better network bandwidth (though poorer latency), providing ample reason to attribute the differences on global FFT to network bandwidth differences. The impact of BGL’s network bandwidth becomes most apparent in Figure 21, which shows the results normalized against the theoretical peak floating point rate of the partition of the machine used. That XT3 fairs much better in the comparison of normalized performance is attributable again to its higher

network bandwidth. One might also note the slight bump between the 1024 and 1100 PE XT3 results, this is caused by differences in network latency between these two runs, the 1100 PE run having been performed near the end of 2005 with an intermediate version of MPT which exhibited lower latency, though at a reliability deficit subsequently fixed at the cost of a small latency increase. The large inflection in the X1E result at 1008 processors is the result of an optimization: the C version of the benchmark (which was produced by passing the FFTE Fortran code though the f2c converter) was replaced by the original Fortran code resulting is a significant improvement in performance.

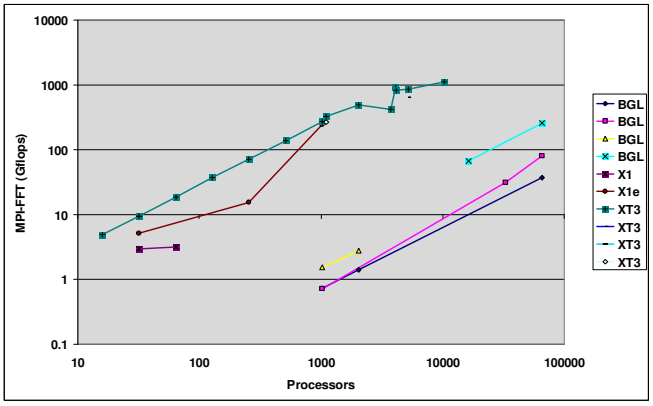


Figure 20: MPI-FFT Comparison

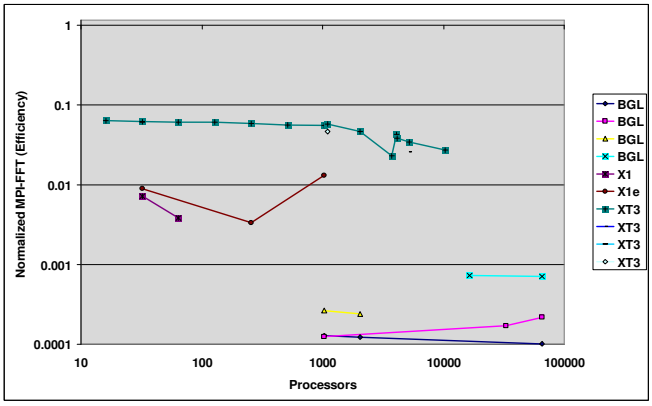


Figure 21: Normalized MPI-FFT Comparision

### 4.1.4 Low Spatial and Low Temporal Locality

#### 4.1.4.1 RandomAccess

Figures 22 & 23 summarize the results of the serial and embarrassingly parallel RandomAccess benchmark. Despite an essentially unchanged memory system, the upgrade from X1 to X1E resulted in a significant improvement in the serial RandomAccess results, however the unexplained drop in the EP version of RandomAccess for X1E (old vs new) is still under investigation, though it is believed to again be attributable to dedicated vs non-dedicated systems for the respective benchmarks. The serial and EP RandomAccess results on the XT3 also appear to have lost a little ground over the last year, however, the drop is within the range of repeatability for the experiments. The X1 (0.21 GUPs) and X1E (0.33 GUPs) perform well on this benchmark due to their ability to reference memory via gather/scatter in a truly random fashion at high performance, whereas the XT3 (0.020 GUPs) and BGL (0.007 GUPs) systems, whose caches rely on high data locality to improve memory performance, perform quite poorly when locality is not present.

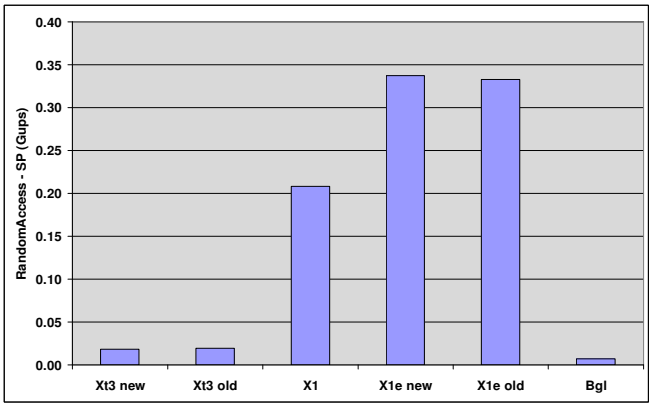


Figure 22: Serial RandomAccess Comparison

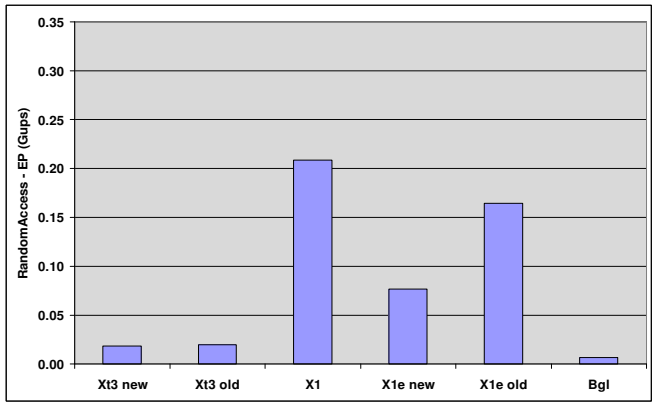


Figure 23: EP RandomAccess Comparison

#### 4.1.4.2 MPI-RandomAccess

Figures 24 & 25 summarize the results of the MPI RandomAccess benchmarks. Several of the data points on these charts represent optimized results. The BGL results in the upper right hand portion of the chart (yellow) were computed with a new algorithm. This algorithm creates and exploits locality in the benchmark's communications, arguably moving it from the low spatial and temporal locality quadrant of the Spatial-Temporal Locality Diagram (fig. 1). Because of this, the results from the new algorithm are not directly comparable to results from the old algorithm. The two data points for larger configurations of X1E are also optimized results, though in this case, the benchmarkers used the original algorithm, but re-implemented it in UPC to take advantage of X1E's globally addressable memory, resulting in a roughly 80x speed-up. Turning attention to the non-optimized results, we see that BGL scaling appears to have acquired negative slope somewhere between 2K and 32K processors, though without intermediate points we can only guess where. XT3 on the other hand appears to still be scaling reasonably to roughly 10K processors. This benchmark is not computationally intensive, and in fact, at lower processor counts (1-2K) BGL seems on par or slightly faster than XT3. Moreover, though RandomAccess is sensitive to latency (note how lower random ring latency allows XT3 to outperform the unoptimized X1 and X1E results despite the X1 and X1E serial results being roughly 10x and 15x faster, respectively), BGL's lower random ring latency does not appear to carry through to larger processor counts. The answer lies in network bandwidth at high processor counts: BGL lacks sufficient bandwidth to avoid contention problems at very large processor counts, whereas XT3 seems to be surviving contention better at least to ~10K processors. Moreover, the increased latency due to poor job layout by the scheduler on XT3 and the associated link contention may be suppressing the XT3 results -- were this issue fixed, improvement should be expected. Lastly note the historical XT3 data points falling well below the current trend line. As we saw in the serial and EP results, there has not been a significant change in the node performance for RandomAccess, thus the noted improvements are wholly attributable to the improvements in network latency.

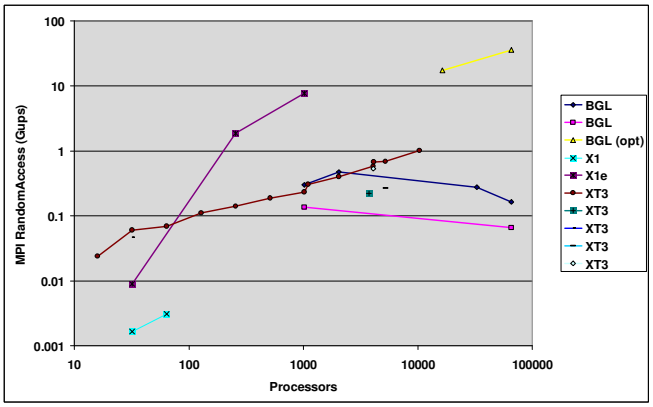


Figure 24 MPI-RandomAccess Comparison

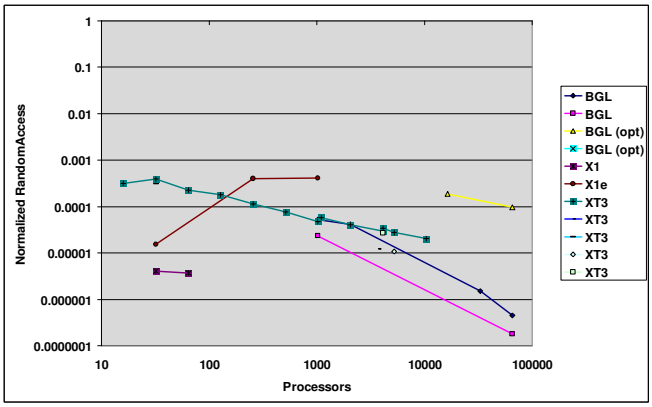


Figure 25: Normalized MPI-RandomAccess Comparision

## 5 Conclusions

In this paper we have identified and attributed several improvements over the last year including algorithm improvements on X1E which resulted in significantly improved FFT results, and XT3 network latency improvements which have impacted the MPI-RandomAccess results (and to a lesser extent other benchmarks). Poor job layout was also noted by several of the benchmarks, most clearly impacting the natural ring latency measurements, and having ripple effects on PTRANS and MPI-RandomAccess.

Overall, X1E shows significant improvement over X1 and the comparison of benchmark results suggests that X1 was sufficiently "over-provisioned" with memory bandwidth that codes with low spatial locality such as FFT and RandomAccess could still draw significant benefit from the processor upgrade to X1E without a memory bandwidth upgrade, however codes with higher spatial locality such as linear algebra and streaming data would see a smaller though still positive benefit. However, based on the results of MPI-FFT and to some extent PTRANS (at larger scales), the cache-based XT3 appears to be very well balanced for some algorithms.

## 6 Future Plans

ORNL CCS intends to continue collecting periodic samples of the HPCC performance results to track the impact of system software changes and hardware upgrades over the life of the CCS machines. This paper left the questions of X1E EP-RandomAccess and SP/EP STREAM performance open for further investigation. We also look forward to evaluating the new IBM RandomAccess algorithm on XT3. Finally we hope to pursue resolution of the job layout issue on XT3.

## Acknowledgements

This research was sponsored by the Office of Mathematical, Information, and Computational Sciences, 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. Also, we would like to thank the Innovative Computing Laboratory at the University of Tennessee for development of the HPCC Benchmarks and public results repository, Argonne National Laboratory for access

to the IBM BG/L, Cray, Inc., and the ORNL Center for Computational Sciences (CCS) for access to the Cray X1, Cray X1E, and Cray XT3. The CCS is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

## References

[1] **"Introduction to the HPCChallenge Benchmark Suite"**, ICL Technical Report, ICL-UT-05-01, (Also appears as CS Dept. Tech Report UT-CS-05-544), 2005, Dongarra, J., Luszczek, P.

[2] **"Early Evaluation of the Cray XT3 at ORNL"**, *Proc. Cray User Group Meeting (CUG 2005), 2005*, J. S. Vetter, S. R. Alam, T. H. Dunigan, Jr., M. R. Fahey, P. C. Roth, and P. H. Worley.

[3] **"Cray X1 Evaluation Status Report"**, *ORNL, Oak Ridge, TN, Technical Report ORNL-TM-2004-13, 2004*, P. A. Agarwal, R. A. Alexander, E. Apra, S. Balay, A. S. Bland, J. Colgan, E. F. D'Azevedo, J. Dongarra, J. B. Drake, T. H. Dunigan, T. H. Dunning, M. R. Fahey, R. A. Fahey, A. Geist, B. Gorda, M. Gordon, W. D. Gropp, R. J. Harrison, R. Kendall, D. Keyes, D. Kaushik, M. Krishnakumar, P. Luszczek, A. Mezzacappa, J. A. Nichols, J. Nieplocha, L. Oliker, T. Packwood, M. S. Pindzola, T. C. Schulthess, H. Simon, R. Stevens, J. S. Vetter, J. B. White, T. L. Windus, P. H. Worley, and T. Zacharia. (Also presented at 46th Cray Users Group Meeting, Knoxville, May 2004).

[4] **"An Overview of the BlueGene/L Supercomputer"**, *SC02: International Conference for High Performance Computing, Networking, and Storage, Baltimore, MD USA, 2002*, N. Adiga, G. Almasi, G. Almasi, R. Barik, D. Beece, R. Bellofatto, G. Bhanot, R. Bickford, M. Blumrich, A. Bright, J. Brunheroto, C. Cascaval, J. Castaos, L. Ceze, P. Coteus, S. Chatterjee, D. Chen, G. Chiu, T. Cipolla, P. Crumley, K. Desai, A. Deutsch, M. Dombrowa, W. Donath, M. Eleftheriou, C. Erway, J. Esch, B. Fitch, J. Gagliano, A. Gara, R. Germain, M. Giampapa, B. Gopalsamy, M. Gupta, F. Gustavson, S. Hall, R. Haring, D. Heidel, P. Heidelberger, L. Herger, D. Hoenicke, R. Jackson, T. Jamal-Eddine, G. Kopcsay, E. Krevat, M. Kurhekar, A. Lanzetta, D. Lieber, L. Liu, M. Lu, M. Mendell, A. Misra, L. Mok, J. Moreira, B. Nathanson, M. Newton, M. Ohmacht, A. Oliner, R. Pudota, R. Rand, R. Regan, B. Rubin, A. Ruehli, S. Rus, R. Sahoo, A. Sanomiya, E. Schenfeld, S. Singh, P. Song, V. Srinivasan, B. Steinmacher-Burow, K. Strauss, C. Surovic, R. Swetz, T. Takken, R. Tremaine, M. Tsao, A. Umamaheshwaran, P. Verma, P. Vranas, T. Ward, M. Wazlowski, W. Barrett, J. Brown, D. Krolak, T. Liebsch, J. Marcella, A. Schram, G. Ulsh, C. Wait, K. Dockser, M. Seager, L. Kissel, J. S. Vetter, and K. Yates.

[5] **"Cray and HPCC: Benchmark Developments and Results from the Past Year"**, *Proc. Cray User Group Meeting (CUG 2005), 2005*, http://www.cug.org, Wichmann, N. L.

[6] **"Leadership Computing at Oak Ridge National Laboratory"**, *Proc. Cray User Group Meeting (CUG 2005), 2005*, http://www.cug.org, Studham, R.S., Kuehn, J.A., White, J.B., Fahey, M.R., Carter, S., and Nichols, J.A.

[7] “HPCChallenge Website”, http://icl.cs.utk.edu/hpcc, Dongarra, J., Luszczezk, P., Bailey, D., Koester, D. McCalpin, J., Rabenseifer, R., Whaley, R., Kepner, J., Lucas, R., Petitet, A., Takahashi, D.