Cray and HPCC:
Benchmark Developments and Results from the Past Year

Nathan Wichmann, *Cray Inc.*

ABSTRACT: *The High Performance Computing Challenge (HPCC) benchmark continued to evolve in 2004 and 2005. HPCC developments will be discussed with particular emphasis on what has changed since CUG 2004. Results will be given for a number of different Cray machines.*

KEYWORDS: HPCC, Cray, Cray X1, Cray XD1, Cray X1, Cray X1E

Introduction

The High Performance Computing Challenge (HPCC) benchmark was first introduced in November 2003, and this author first presented results for Cray machines at CUG 2004. At that time HPCC consisted of five to six primary test benchmarks, with each of those benchmark sometimes having multiple sub results. But since the spring of 2004 HPCC has continued to evolve. Several new tests have been added, some stressing global bandwidth, while other test single CPU performance.

At the same time, HPCC has started to show up in procurements of real machines. This is a very exciting development that seems to show HPCC gaining traction, not only as a method of testing one's machine and comparing it to others, but as a tool in deciding how to spend real money. But as was the case one year ago, there is no standard method of using numbers generated by HPCC when comparing machines.

This paper will look at both of these areas. First, we will review each sub benchmark in the HPCC suite. Along the way, we will point out the newest tests, or in some cases how tests have changed since the spring of 2004.

We will also review the different methods of comparison when using HPCC. There is no such thing as a perfect method, so the author will attempt to simply point out the pros and cons of each.

HPC Challenge Suite

Let's look at the eight primary tests in the HPCC suite to see what each test implies about real world application performance, and to look at how individual systems score across this breadth of tests. As Partridge observes, "The HPC Challenge benchmark suite reflects a spectrum of attributes required to attain HPC success. While some systems may score well on individual tests, such as Linpack, balanced platforms, such as Cray's, achieve high marks across a broad range of benchmarks, attesting to designs that offer the requisite balance of compute and communication capabilities."

The HPC Challenge benchmark suite is intended to test multiple attributes that can substantially contribute to the real-world performance of HPC systems. The collaborators' goal was to augment the Linpack benchmark with additional benchmarks tests that would allow HPC users to examine performance of HPC architectures; tests that would provide performance indicators for not only processing power, but also interconnect and memory system performance. To do this, the test suite needed to use kernels that reflect real-world communication patterns for memory access patterns and interprocessor data exchange. The resulting suite includes 23 tests (see Table 1) 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.

Table 1 - HPC Challenge Benchmark Suite

HPC Challenge Benchmark Test		Description
Indicators of Process Performance		
1	HPL:	Linpack TPP benchmark which measures the floating point rate for solving a linear system of equations (not considered here)
2	DGEMM	measures the floating point rate of execution of double precision real matrix-matrix multiplication
Indicators of Communication Performance		
7 & 8	MPI Random Ring latency test (^{b}eff) & bandwidth	(MPI bandwidth & latency test): a set of tests to measure the latency and bandwidth of a number of simultaneous communication patterns
5	FFTE	measures the floating point rate of execution of double precision complex one-dimensional Fast Fourier Transform
2	PTRANS	(parallel matrix transpose) measures global network bandwidth
Indicators of Memory System Performance		
3	Random Access	measures the rate of random updates of memory
4	STREAM SN-STREAM (single node) EP-STREAM (embarrassingly parallel)	measures sustainable memory bandwidth for simple array accesses

High Performance Linpack (HPL)

G-HPL 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. This test involves solving 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). Figure 1 – HPL results for 128 CPUS shows LINPACK numbers for a number of systems with similar numbers of CPUS. One can see that the Cray X1(E) do get considerably higher performance for an equal number of CPUS but they are vector processors and are often purchased in lower CPU counts.

Figure 1 - HPL results for 128 CPUS

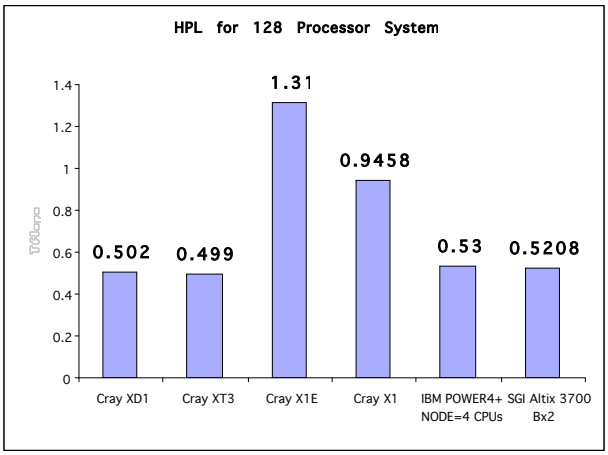
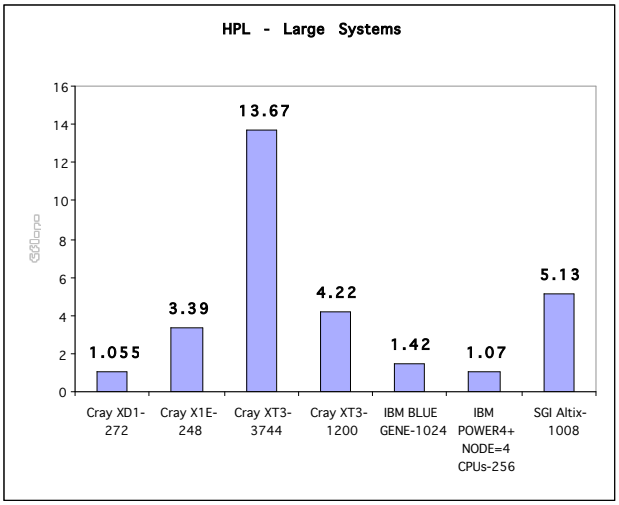


Figure 2 shows the results on systems with larger CPUS counts. Here the XT3 stands out with a combination of large CPU counts, very good peak per CPU, and very good percentage of peak. The IBM Blue Gene machine is much lower, it has a very low peak CPU rate AND a low percentage of peak, even on LINPACK.

Figure 2 - HPL Results for large processor counts



DGEMM

The DGEMM test measures the floating point rate of execution of double precision real matrix-matrix multiplication in gigaflops per second (GFLOPS) using the DGEMM BLAS library routine.

DGEMM is the purest measure of processor floating point performance, as this routine 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.

This is a new test first added in the summer of 2004. While it is considered by many to be one of the 8 primary tests, this author really views it as a sub test of HPL, much like Single Node and Embarrassingly parallel versions of Random Access are subtest of Global Random Access. Result for DGEMM correlate strongly with per CPU HPL results.

One source of confusion surrounding the new DGEMM is that it may be run on a node, as opposed to a single processor. This can favor nodes with more processors, such as the 4-way nodes in the IBM Power4 SMP systems.

PTRANS

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).

Figure 3 shows the performance of PTRANS on similar CPU counts. Again, the Cray X1 shows up as powerful, but the XT3 is also very high.

Figure 3 - GPTRANS Results

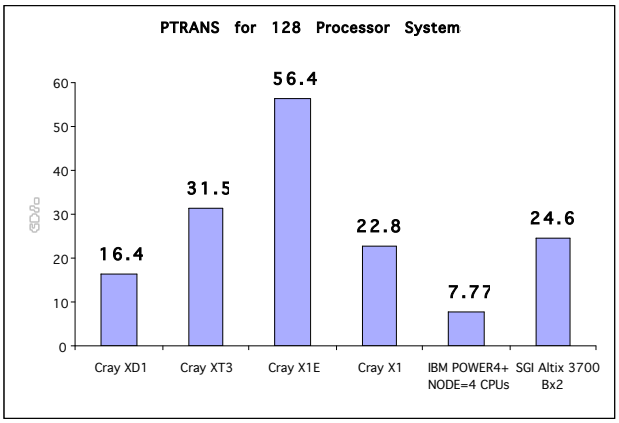
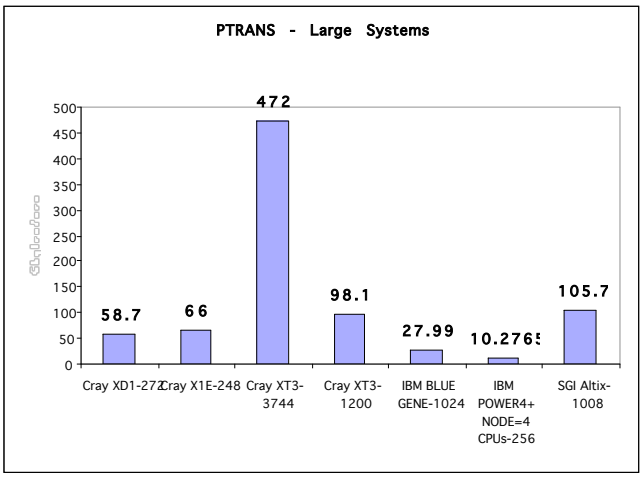


Figure 4 shows the performance of PTRANS on large system. Here the Cray XT3 clearly stands out and is much better than the IBM Blue Gene. In fact, a 272 XD1 system is twice and good as a 1024 Blue Gene machine.

Figure 4 - PTRANS results on large processor counts



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 this author has seen with PTRANS is that performance can vary significantly with a small change in the parameter settings. This behavior has been observed on multiple platforms. To counter this, in the HPCC input file there is a way to specify running the PTRANS test with several different blocking factors. The best results is selected and reported in the final summary. While this helps one get better numbers, the underlying behavior still can make it difficult to interpret PTRANS numbers.

STREAM EP–Triad

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. EP-STREAM is run in an embarrassingly parallel manner, with each node performing the calculation at the same time.

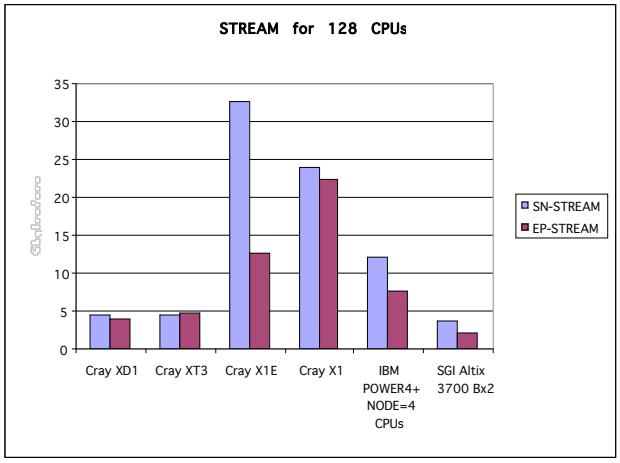
STREAM provides a good approximation for high memory bandwidth codes or codes that need sections of high memory bandwidth. It is worth noting that STREAM assesses bandwidth to memory only, and does not stress interprocessor communications, or I/O communications.

Figure 5 shows the results per processor for EP-STREAM and SN-STREAM. Comparing the SN Triad/ CPU

number to the EP Triad/ CPU number for a given system indicates how the system performance will degrade as the load approaches capacity. The Cray X1E show significant degradation between SN and EP. This is not surprising as it was a major processor upgrade to the Cray X1 with little to no change in the memory subsystem. On the other hand, the Cray XT3 and the Cray XD1 show little to no degradation going from SN to EP results. One can expect per CPU performance to be constant, even as you load up the system.

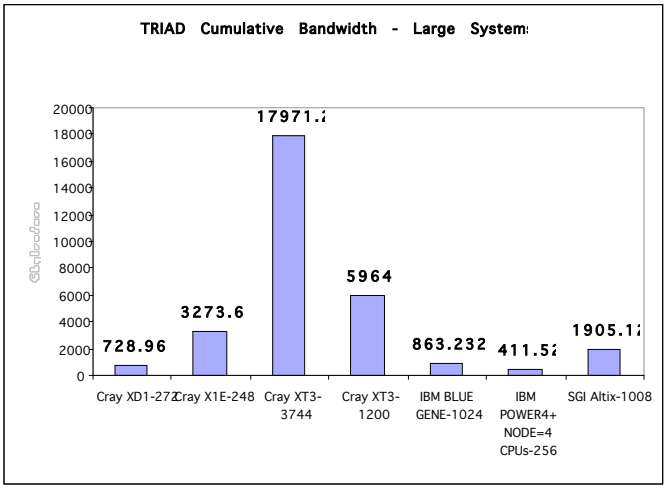
One thing to note here is the IBM result is actually for a run where a "node" equals 4 CPUS. It is not clear from the material on the web site if this means that STREAM was run using shared memory parallelism across 4 CPUS or if this means that only 1 CPU was running but still had zero contention access to the bandwidth normally used by 4 CPUS? Either way, this makes the IBM results look much large than it really is.

Figure 5 - STREAM Results



A very informative way of looking at these results is to multiple the per process EP-TRIAD result by the number of processes, creating a Cumulative Bandwidth number. Figure 6 shows the results of that calculation. Here one can clearly see that the combination of large processor counts, very good bandwidth per CPU and zero contention between CPUs makes the Cray XT3 stand out. The Cray XT3-1200 has over 6 times the cumulative bandwidth of an IBM Blue Gene using only 10% more CPUs.

Figure 6 – Cumulative STREAM TRIAD bandwidth for large processor counts

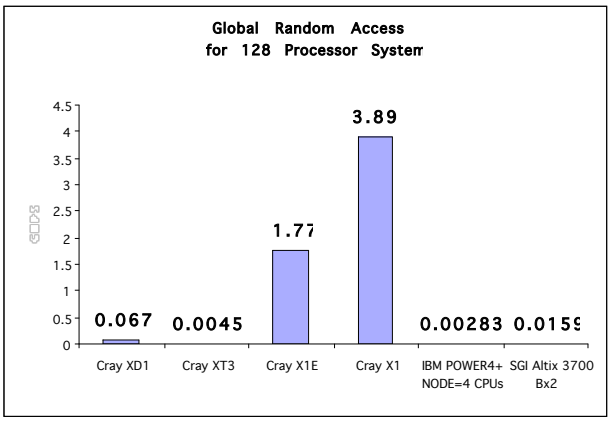


Random Access

The Random Access test measures the rate at which the computer can update pseudo-random locations of 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, such as those that use adaptive mesh refinement (AMR), that need frequent, random access to data, will perform on a given system. In adaptive mesh calculations, the decision to refine the grid is made cell-by-cell and cycle-by-cycle continuously throughout the problem, thus placing frequent, random demands on memory access.

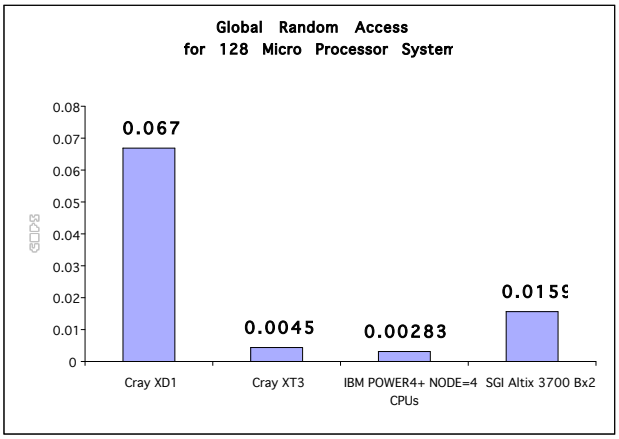
Figure 7 shows the results of solving the Global Random Access problem on a number of machines with approximately 128 CPUS. Here we see that the Cray X1(E) gets a much higher GUPS number compared to the other machines. This is due to two reasons. First, the version run on the Cray X1(E) is written using Unified Parallel C, or UPC. UPC allows one to set up a distributed array called TABLE, and then, using simple C syntax, directly access that TABLE. This not only greatly simplifies the algorithm, but also eliminates the overhead associated with using MPI. The second reason for the very high results is that the Cray X1 can then vectorize the main loop containing all of the updates to TABLE. This means that the Cray X1 can maintain thousands of outstanding memory references per CPU to locations across the machine.

Figure 7 – Global Random Access Results



It is still valuable to directly compare the results for the microprocessors using MPI. Figure 8 shows the Cray XD1 having very good performance in this area, over 4 times the performance of a similar sized SGI and 30 times the performance of IBM. The Cray XT3's performance, while better than the IBM, is not as good as the XD1. This is not surprising given the early state of the XT3 and the fact its MPI is not yet optimized.

Figure 8 – Global Random Access Results for Microprocessors



MPI and Random Ring Latency and Bandwidth

These tests compare latency and bandwidth for MPI and Random Ring communication patterns. While frequently you will see vendors publish results for the MPI Ping-Pong 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. The Random Ring latency test uses a more realistic scenario where all processors in the system communicate with their logical, but not necessarily physical, neighbors, at approximately the same time. As this communication pattern is very common in codes that break the problem up into sections of a grid, the Random Ring tests are better predictors of real world performance.

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.

Figure 9 – Random Ring Latency Results shows the Cray XD1 system in the lead with a significantly lower latency than any other system, followed by the SGI Altix. Again we see the Cray XT3 with long latency, but we expect that to improve dramatically as the MPI library is optimized.

Figure 2 - Random Ring Latency Results

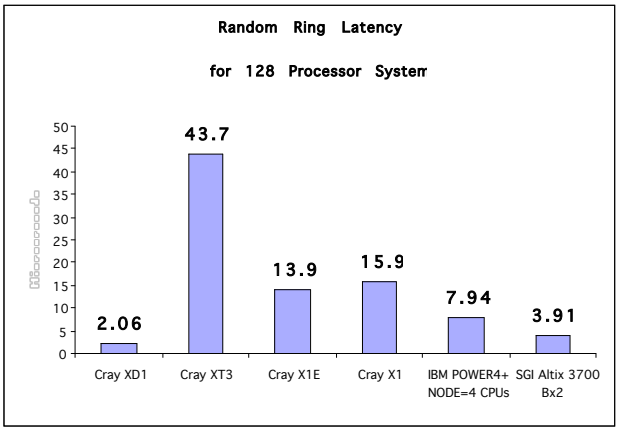
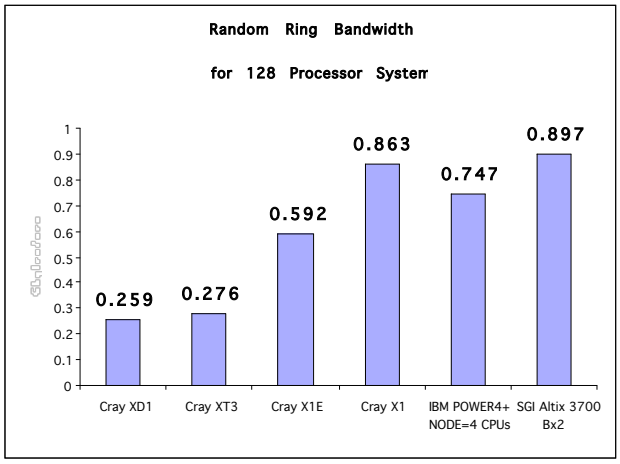


Figure 10 shows the results for Random Ring Bandwidth test on a per CPU bases. Here both the Cray XD1 and the Cray XT3 are virtually equal and much better than any other system.

Figure 3 - Random Ring Bandwidth Results



FFTE

The FFTE test measures the floating point rate of execution of complex one-dimensional Fast Fourier Transforms, when using double precision. 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 version supplied 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. There have been some optimizations done to regain the dependence information and allow some vectorization, but in the future I would like to explore the use of other FFTE packages that are more vector friendly.

Figure 11 – Global FFT Results

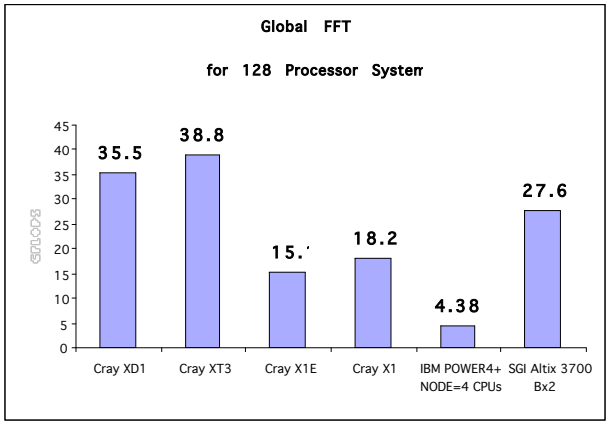


Figure 11 shows the performance of G-FFT on machines of approximately 128 CPUs. We see that both the Cray XT3 and the Cray XD1 are clearly ahead of everyone else. In particular, that XT3 has over 8 times the performance of the POWER4+ based IBM.

Cray has experience problems running the Global FFT test on all three of our platforms. HPCC seems to run small G-FFT problems, both in terms of memory size and number of CPUS, without difficulties, but errors occur as problem sizes grow. The problem seems to occur in initialization and has nothing to do with the actually computation of the FFT. Cray has contacted the authors who were already aware of the problem on other platforms, so it is not unique to Cray. We are working internally as well as with the authors to try and debug the problem.

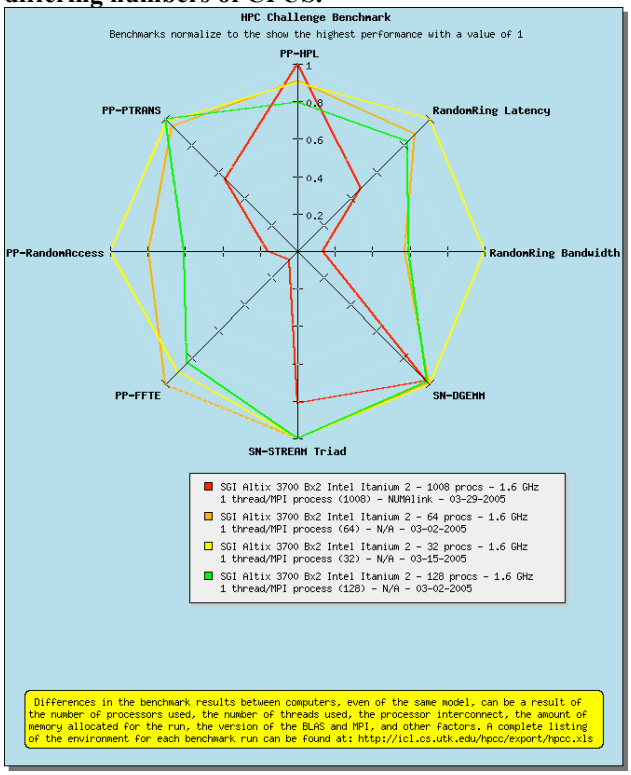
Machine Comparisons

With so many numbers being produced by HPCC, it is sometimes useful to try and make high-level comparisons between different machines using the data produced. There are many differing opinions on how this should be done, we will discuss two methods here.

There is one comparison technique new to the HPCC web site, and that is the ability to create Kiviat diagrams. A Kiviat diagram is a two dimensional graph of radially extending lines where each line corresponds to a different metric. To create the diagram, the web site first turns each of the main 8 test results into a per process number. The scores are then normalized so that the machine with the highest per process score in each test is given a value of 1 (one) and all other scores are given a value between zero and one, where zero would be zero performance and one would be equal to the best performance. Finally, these point are graph on the Kiviat diagram and a perimeter is drawn, connecting all of the points for any given machine. Figure 12 shows an example of a Kiviat diagram comparing SGI machines of different sizes.

As with any method of comparison, there are pros and cons with using Kiviat Diagrams to examine HPCC results. On the positive side, it is visually simple. The color-coordinated perimeters are easy to identify and follow. The diagram strongly implies that the larger the area inside the perimeter, the “better” the machine

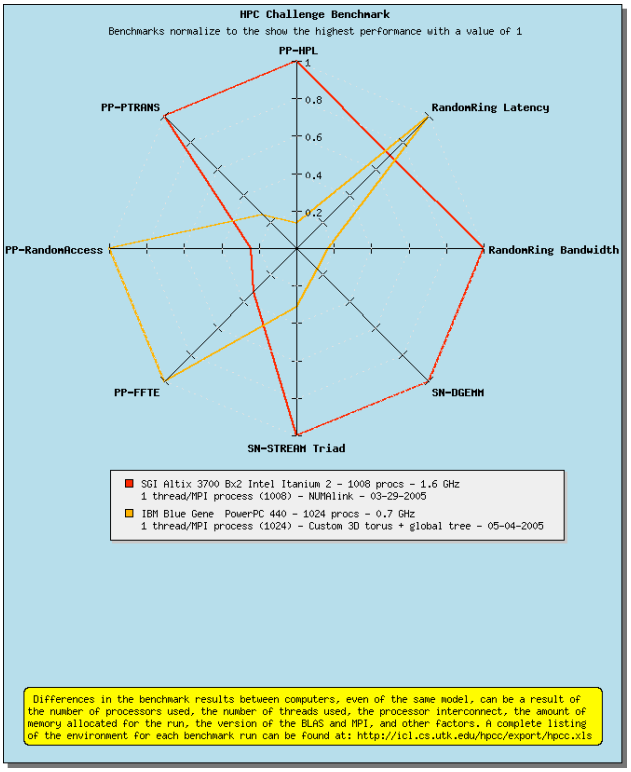
Figure 4 – Kiviat diagram for SGI systems with differing numbers of CPUS.



However the Kiviat diagram also has some negatives. The most obvious negative is that by plotting the results on a per processor basis the end result makes smaller machines look better! In Figure 12 above, 4 machines are displayed, all are SGI Altix systems but with different number of CPUs. One can see that the SGI system with only 32 CPUs consistently has the best results in each category. The SGI system with 1008 CPUS has much smaller per process values and thus has a much smaller area inclosed within the perimeter. Any person looking at this graph, but without being told what machines the lines represented, would conclude that the machine corresponding to the yellow is much, much better, and more powerful than the machine corresponding to the red line. This would obviously be incorrect. Thus, one cannot use the Kiviat diagram with per process scores to determine which of two systems is the more powerful machine. Other problems are that the diagram effectively shows all 8 tests weighted equally where people may not agree with that weighting. Finally, I was not able to plot optimized results on the same graph as base results, although this is probably something that can be easily corrected.

The Kiviat Diagram may be useful when comparing 2 systems of similar number of CPUs. Figure 13 shows the results for a 1008 processor SGI system vs. a 1024 processor Blue Gene system from IBM.

Figure 13 – Kiviat diagram for two systems with similar numbers of CPUS.



By examining figure 13 it is obvious that each machine has different strengths. The SGI system is obviously better at CPU intensive test like TRIAD, DGEMM, and HPL, and is also better at PTRANS and Random Ring Bandwidth. The IBM has good Random Ring Latency and Random Access, and surprisingly, good Global FFT performance. It appears that the SGI has the superior performance overall because the area inside it's perimeter is larger than the IBM's, but this might just be due to how the different tests are arranged.

An alternative way to compare machines has been developed inside of Cray by the author called the App Kernel Power Rating. The power rating attempts to combine the scores for the 5 main kernels; HPL, PTRANS, Cumulative TRIAD Bandwidth, Global Random Access, and Global FFT, and combine them into a single number. To do this, the result for each machine in each test is divided by the total power of all the machines in that category to create a unitless number. The unitless number is actually equal to the percentage of the total power in the category. These 5 numbers are then combined into a single number. The numbers generated for each machine is that machines Power Rating and can be directly compared to any other machine on the list.

Again, each method of comparison has its pros and cons, and the App Kernel Power Rating is no different. In this case, the pros are that a single number is created which

can be directly compared to other machines. This results in an obvious "winner". Obviously that winner can be debated, but at least this defined method of comparison comes up with a single answer. The method is also very simple to understand. The normalizations are straightforward and the weighting is clear, and could easily be changed to accommodate other's opinions. Another positive is that it focuses on kernels that actually do real work, and thus are better predictors of end performance. While the MPI tests are very useful, it is difficult to determine how they effect performance or real applications, and even more difficult to combine with the other numbers into a Power Rating. This author thinks that it is best to simply examine those numbers in isolation, and perhaps to use them to better understand the performance of the other kernels. Finally, the most important positive feature about the App Kernel Power Rating is that larger machines are considered more powerful, and this is obvious from the score. This is both intuitive and useful for eventually making decisions about which machines are better for customers.

One negative of the App Kernel Power Rating is that all of the tests are considered equal. While this can be easily changed, there is no other obvious weighting. Another negative is that the MPI tests are not included. This is largely the result of finding it difficult to include those results in a meaningful way, even after normalization.

FIGURE 14 App Kernel Power Rating for 128 CPU machines

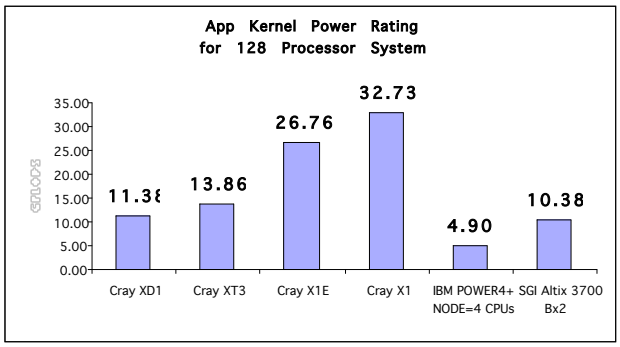
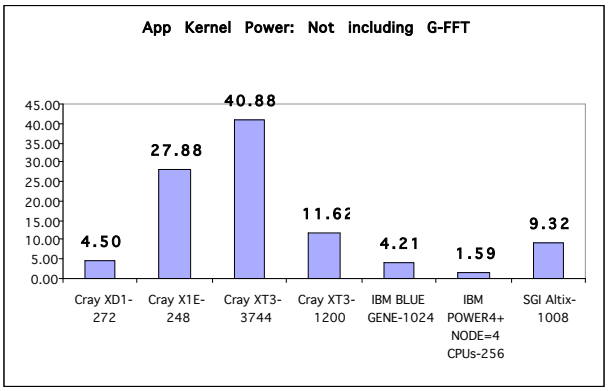


Figure 14 shows the App Kernel Power Rating for machines with approximately 128 CPUS. From this graph the Cray X1 and the X1E have very high Power Ratings. This is the result of their exceptionally high Global Random Access scores. We can still examine the other machines, ignoring the vector machines for the moment. We see that the Cray XT3 is the most powerful with the XD1 close behind. Both are more powerful than the SGI Altix despite the Altix having a higher theoretical peak performance. The IBM is a distant last, with less than half the performance of the XD1 and about one third the performance of the Cray XT3.
The definition of the Power Rating had to be adjusted to accommodate the problems with large Global FFTs mentioned earlier. For simplicity, the Global FFT was simply left out, combining the remaining 4 kernels.

FIGURE 15 App Kernel Power Rating for large systems



There are two things I would like to point out in figure 15. First, the 3744 processor Cray XT3 is clearly the most powerful machine on the list, more than 4 times as powerful than the SGI Altix despite having fewer than 4 times the number of CPUS. Second, notice that the 272 processor Cray XD1 is actually more powerful than the IBM Blue Gene, even though the IBM has 4 times the number of CPUs and the difficulties of trying to scale to those higher CPU counts.

Conclusions

It is obvious from the events of the last year that HPCC continues to grow in both its usefulness and adoption. It has appeared in several recent benchmarks and it is likely to be included in more in the future. As people begin to use it more, people will try and make high-level comparisons of different machines. This author will continue to watch how those results are used and interpreted in the coming year.

Authors and Acknowledgments

Nathan Wichmann is Benchmark Engineer at Cray Inc. He has been involved in many different benchmarks on all of Cray machines, both for HPCC and several other "real world" applications. He can be reached Cray Inc, 1340 Mendota Heights Road, Mendota Heights, MN 55120. E-Mail: wichmann@cray.com

The author would like to thank colleagues, users and vendor staff for assistance in writing and editing this paper and collecting numbers from all of the different machines.