CTWatch QUARTERLY
November 2006 A

Design and Implementation of the HPC Challenge Benchmark Suite

Piotr Luszczek,
University of Tennessee
Jack Dongarra,
University of Tennessee,
Oak Ridge National Laboratory
Jeremy Kepner, MIT Lincoln Lab

Introduction

The HPCC benchmark suite was initially developed for the DARPA's HPCS program [1] to provide a set of standardized hardware probes based on commonly occurring computational software kernels. The HPCS program has initiated a fundamental reassessment of how we define and measure performance, programmability, portability, robustness and, ultimately, productivity in the high-end domain. Consequently, the suite was aimed to both provide conceptual expression of the underlying computation as well as be applicable to a broad spectrum of computational science fields. Clearly, a number of compromises must have lead to the current form of the suite given such a broad scope of design requirements. HPCC was designed to approximately bound computations of high and low spatial and temporal locality (see Figure 1 which gives the conceptual design space for the HPCC component tests). In addition, because the HPCC tests consist of simple mathematical operations, this provides a unique opportunity to look at language and parallel programming model issues. As such, the benchmark is to serve both the system user and designer communities [2].

Finally, Figure 2 shows a generic memory subsystem and how each level of the hierarchy is tested by the HPCC software and what are the design goals of the future HPCS system - these are the projected target performance numbers that are to come out of the wining HPCS vendor designs.

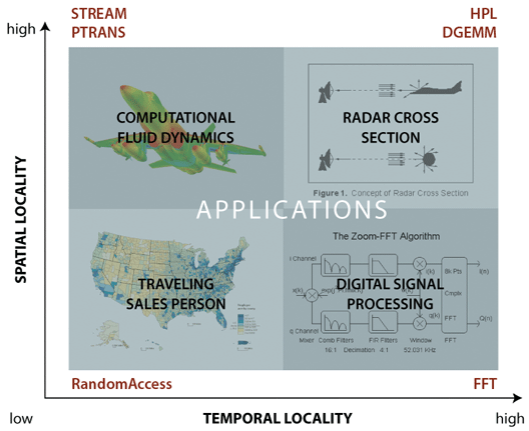

Figure 1. The application areas targeted by the HPCS Program are bound by the HPCC tests in the memory access locality space.

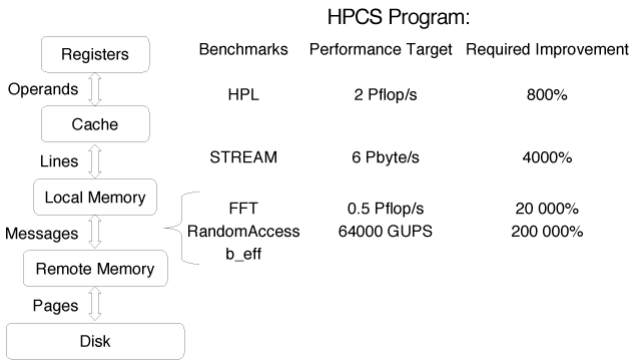

Figure 2. HPCS program benchmarks and performance targets.

The TOP500 Influence

Table 1. All of the top-10 entries of the 27th TOP500 list that have results in the HPCC database.

Rank	Name	Rmax	HPL	PTRANS	STREAM	FFT	RANDA	Latency	Bandwidth
1	BlueGene/L	280.6	259.2	4665.9	160	2311	35.47	5.92	0.16
2	BlueGene W	91.3	83.9	171.5	50	1235	21.61	4.70	0.16
3	ASC Purple	75.8	57.9	553.0	44	842	1.03	5.11	3.22
4	Columbia	51.9	46.8	91.3	21	230	0.25	4.23	1.39
9	Red Storm	36.2	33.0	1813.1	44	1118	1.02	7.97	1.15

The most commonly known ranking of supercomputer installations around the world is the TOP500 list [3]. It uses the equally famous LINPACK Benchmark [4] as a single figure of merit to rank 500 of the worlds most powerful supercomputers. The often raised issue of the relation between TOP500 and HPCC can simply be addressed by recognizing all the positive aspects of the former. In particular, the longevity of TOP500 gives an unprecedented view of the high-end arena across the turbulent times of Moore's law [5] rule and the process of emerging of today's prevalent computing paradigms. The predictive power of TOP500 will have a lasting influence in the future as it did in the past. While building on the legacy information, HPCC extends the context of the HPCS goals and can serve as a valuable tool for performance analysis. Table 1 shows an example of how the data from the HPCC database can augment the TOP500 results.

Short History of the Benchmark

The first reference implementation of the code was released to the public in 2003. The first optimized submission came in April 2004 from Cray using their recent X1 installation at Oak Ridge National Lab. Every since then Cray has championed the list of optimized submissions. By the time of the first HPCC birds-of-feather at the 2004 Supercomputing conference in Pittsburgh, the public database of results already featured major supercomputer makers - a sign

that vendors noticed the benchmark. At the same time, a bit behind the scenes, the code was also tried by government and private institutions for procurement and marketing purposes. The highlight of 2005 was the announcement of a contest: the HPCC Awards. The two complementary categories of the competition emphasized performance and productivity - the very goals of the sponsoring HPCS program. The performance-emphasizing Class 1 award drew attention to the biggest players in the supercomputing industry, which resulted in populating the HPCC database with most of the top-10 entries of TOP500 (some of which even exceeding performance reported in the TOP500 - a tribute to HPCC's continuous results' update policy). The contestants competed to achieve highest raw performance in one of the four tests: HPL, STREAM, RANDA, and FFT. The Class 2 award, by solely focusing on productivity, introduced subjectivity factor to the judging but also to the submitter criteria of what is appropriate for the contest. As a result, a wide range of solutions were submitted spanning various programming languages (interpreted and compiled) and paradigms (with explicit and implicit parallelism). It featured openly available as well as proprietary technologies, some of which were arguably confined to niche markets and some that are widely used. The financial incentives for entering turned out to be all that was needed, as the HPCC seemed to have enjoyed enough recognition among the high-end community. Nevertheless, HPCwire kindly provided both press coverage as well as cash rewards for four winning contestants of Class 1 and the winner of Class 2. At the HPCC's second birds-of-feather session during the SC|05 conference in Seattle, the former class was dominated by IBM's BlueGene/L from Lawrence Livermore National Lab while the latter was split among MTA pragma-decorated C and UPC codes from Cray and IBM, respectively.

The Benchmark Tests' Details

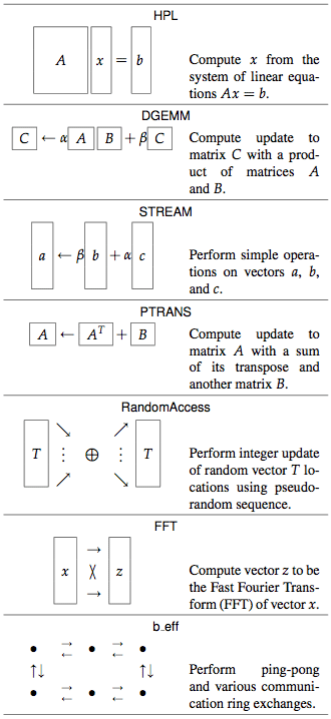

Figure 3. Detail description of the HPCC component tests (*A, B, C* - matrices, *a, b, c, x, z* – vectors, *α, β* - scalars, *T* - array of 64-bit integers).

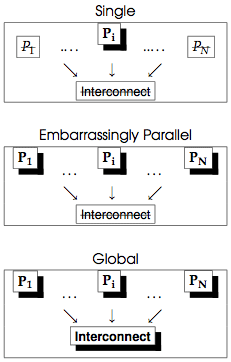

Figure 4. Testing scenarios of the HPCC components.

Extensive discussion and various implementations of the HPCC tests are given elsewhere [6] [7] [8]. However, for the sake of completeness, this section lists the most important facts pertaining to the HPCC tests' definitions.

All calculations use *double precision* floating-point numbers as described by the IEEE 754 standard [9] and no mixed precision calculations [10] are allowed. All the tests are designed so that they will run on an arbitrary number of processors (usually denoted as *p*). Figure 3 shows a more detailed definition of each of the seven tests included in HPCC. In addition, it is possible to run the tests in one of three testing scenarios to stress various hardware components of the system. The scenarios are shown in Figure 4.

Benchmark Submission Procedures and Results

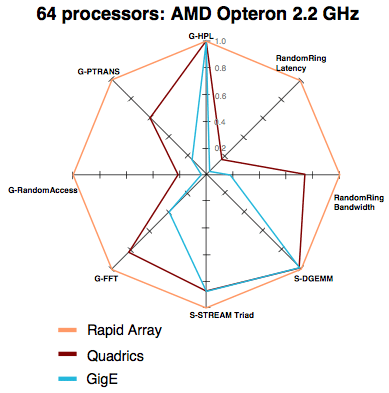

Figure 5. Sample kiviat diagram of results for three different interconnects that connect the same processors.

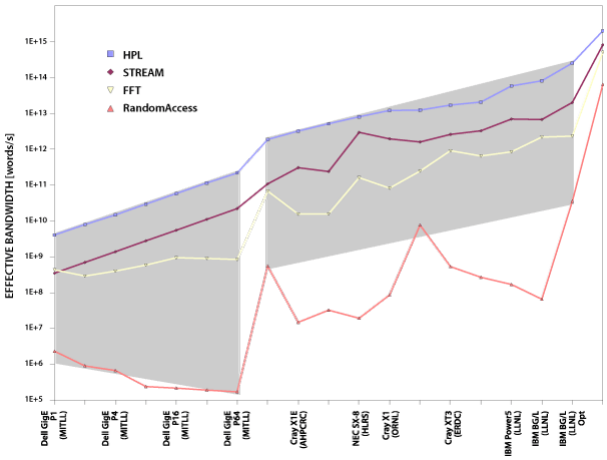

Figure 6. Sample interpretation of the HPCC results.

The reference implementation of the benchmark may be obtained free of charge at the benchmark's web site [11]. The reference implementation should be used for the base run: it is written in portable subset of ANSI C [12] using hybrid programming model that mixes OpenMP [13] [14] threading with MPI [15] [16] [17] messaging. The installation of the software requires creating a script file for Unix's `make(1)` utility. The distribution archive comes with script files for many common computer architectures. Usually, few changes to one of these files will produce the script file for a given platform. The HPCC rules allow only standard system compilers and libraries to be used through their supported and documented interface and the build procedure should be described at submission time. This ensures repeatability of the results and serves as an educational tool for end users that wish to use the similar build process for their applications.

After, a successful compilation the benchmark is ready to run. However, it is recommended that changes be made to the benchmark's input file that describes the sizes of data to use during the run. The sizes should reflect the available memory on the system and the number of processors available for computations.
There must be one baseline run submitted for each computer system entered in the archive. There may also exist an optimized run for each computer system. The baseline run should use the reference implementation of HPCC and, in a sense, it represents the scenario when an application requires use of legacy code - a code that can not be changed. The optimized run allows

developers to perform more aggressive optimizations and use system-specific programming techniques (e.g., languages, messaging libraries, etc.) but at the same time still gives the verification process enjoyed by the base run.

All of the submitted results are publicly available after they have been confirmed by email. In addition to the various displays of results and raw data export the HPCC website also offers a kiviat chart display to visually compare systems using multiple performance numbers at once. A sample chart that uses actual HPCC results' data is shown in Figure 5.

Figure 6 show performance results of currently operating clusters and supercomputer installations. Most of the results come from the HPCC public database.

Scalability Considerations

Table 2. Time complexity formulas for various phases of the HPCC tests (*m* and *n* correspond to the appropriate vector and matrix sizes, *p* is the number of processors.

Name	Generation	Computation	Communication	Verification	Per-processor data
HPL	n^2	n^3	n^2	n^2	p^{-1}
DGEMM	n^2	n^3	n^2	1	p^{-1}
STREAM	m	m	1	m	p^{-1}
PTRANS	n^2	n^2	n^2	n^2	p^{-1}
RandomAccess	m	m	m	m	p^{-1}
FFT	m	$m\log_2 m$	m	$m\log_2 m$	p^{-1}
b_eff	1	1	p^2	1	1

There are a number of issues to be considered for benchmarks such as HPCC that have scalable input data to allow for an arbitrary sized system to be properly stressed by the benchmark run. Time to run the entire suite is a major concern for institutions with limited resource allocation budgets. Each component of HPCC has been analyzed from the scalability standpoint and Table 2 shows the major time complexity results. In following, it is assumed that:

- *M* is the total size of memory,
- *m* is the size of the test vector,
- *n* is the size of the test matrix,
- *p* is the number of processors,
- *t* is the time to run the test.

Clearly any complexity formula that grows faster than linearly with respect to any of the system sizes is a cause of potential problem time scalability issue. Consequently, the following tests have to be addressed:

- HPL because it has computational complexity $O(n^3)$.
- DGEMM because it has computational complexity $O(n^3)$.
- b_eff because it has communication complexity $O(p^2)$.

The computational complexity of HPL of order $O(n^3)$ may cause excessive running time because the time will grow proportionately to a high power of total memory size:

$$t_{HPL} \sim n^3 = (n^2)^{3/2} \sim M^{3/2} = \sqrt{M^3} \qquad (1)$$

To resolve this problem, we have turned to the past TOP500 data and analyzed the ratio of *Rpeak* to the number of bytes for the factorized matrix for the first entry on all the lists. It turns out that there are on average 6±3 Gflop/s for each matrix byte. We can thus conclude that the performance rate of HPL remains constant over time ($r_{HPL} \sim M$), which leads to:

$$t_{HPL} \sim \frac{n^3}{r_{HPL}} \sim \frac{\sqrt{M^3}}{M} = \sqrt{M} \qquad (2)$$

that is much better than (1).

There seems to be a similar problem with the DGEMM as it has the same computational complexity as HPL but fortunately, the *n* in the formula related to a single process memory size rather than the global one and thus there is no scaling problem.

Lastly, the b_eff test has a different type of problem: its communication complexity is $O(p^2)$ which is already prohibitive today as the number of processes of the largest system in the HPCC database is 131072. This complexity comes from the ping-pong component of b_eff that attempts to find the weakest link between all nodes and thus, theoretically, needs to look at the possible process pairs. The problem was remedied in the reference implementation by adapting the runtime of the test to the size of the system tested.

Conclusions

No single test can accurately compare the performance of any of today's high-end systems let alone any of those envisioned by the HPCS program in the future. Thus, the HPCC suite stresses not only the processors, but the memory system and the interconnect. It is a better indicator of how a supercomputing system will perform across a spectrum of real-world applications. Now that the more comprehensive HPCC suite is available, it could be used in preference to comparisons and rankings based on single tests. The real utility of the HPCC benchmarks are that architectures can be described with a wider range of metrics than just flop/s from HPL. When looking only at HPL performance and the TOP500 list, inexpensive build-your-own clusters appear to be much more cost effective than more sophisticated parallel architectures. But the tests indicate that even a small percentage of random memory accesses in real applications can significantly affect the overall performance of that application on architectures not designed to minimize or hide memory latency. The HPCC tests provide users with additional information to justify policy and purchasing decisions. We expect to expand and perhaps remove some existing benchmark components as we learn more about the collection.

This work was supported in part by the DARPA, NSF, and DOE through the DARPA HPCS program under grant FA8750-04-1-0219 and SCI-0527260.

[1] Kepner, J. “HPC productivity: An overarching view,” *International Journal of High Performance Computing Applications*, 18(4), November 2004.
[2] Kahan, W. “The baleful effect of computer benchmarks upon applied mathematics, physics and chemistry,” *The John von Neumann Lecture at the 45th Annual Meeting of SIAM*, Stanford University, 1997.
[3] Meuer, H. W., Strohmaier, E., Dongarra, J. J., Simon, H. D. *TOP500 Supercomputer Sites*, 28th edition, November 2006. (The report can be downloaded from http://www.netlib.org/benchmark/top500.html).
[4] Dongarra, J. J., Luszczek, P., Petitet, A. “The LINPACK benchmark: Past, present, and future,” *Concurrency and Computation: Practice and Experience*, 15:1-8, 2003.
[5] Moore, G. E. “Cramming more components onto integrated circuits,” *Electronics*, 38(8), April 19 1965.
[6] Dongarra, J., Luszczek, P. “Introduction to the HPC Challenge benchmark suite,” *Technical Report UT-CS-05-544*, University of Tennessee, 2005.
[7] Luszczek, P., Dongarra, J. “High performance development for high end computing with Python Language Wrapper (PLW),” *International Journal of High Perfomance Computing Applications*, 2006. Accepted to Special Issue on High Productivity Languages and Models.
[8] Travinin, N., Kepner, J. “pMatlab parallel Matlab library,” *International Journal of High Perfomance Computing Applications*, 2006. Submitted to Special Issue on High Productivity Languages and Models.
[9] ANSI/IEEE Standard 754-1985. “Standard for binary floating point arithmetic,” *Technical Report*, Institute of Electrical and Electronics Engineers, 1985.
[10] Langou, J., Langou, J., Luszczek, P., Kurzak, J., Buttari, A., Dongarra, J. “Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy,” In *Proceedings of SC06*, Tampa, Florida, Nomveber 11-17 2006. See http://icl.cs.utk.edu/iter-ref .
[11] HPCC - http://icl.cs.utk.edu/hpcc/
[12] Kernighan, B. W., Ritchie, D. M.. *The C Programming Language*. Prentice-Hall, 1978.
[13] OpenMP: Simple, portable, scalable SMP programming. http://www.openmp.org/ .
[14] Chandra, R., Dagum, L., Kohr, D., Maydan, D., McDonald, J., Menon, R. *Parallel Programming in OpenMP*. Morgan Kaufmann Publishers, 2001.
[15] Message Passing Interface Forum. “MPI: A Message-Passing Interface Standard,” *The International Journal of Supercomputer Applications and High Performance Computing*, 8, 1994.
[16] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard (version 1.1), 1995. Available at: http://www.mpi-forum.org/ .
[17] Message Passing Interface Forum. MPI-2: Extensions to the Message-Passing Interface, 18 July 1997. Available at http://www.mpi-forum.org/docs/mpi-20.ps .

URL to article: http://www.ctwatch.org/quarterly/articles/2006/11/design-and-implementation-of-the-hpc-challenge-benchmark-suite/