HPC Benchmarking and Performance Evaluation With Realistic Applications

Brian Armstrong, Hansang Bae, Rudolf Eigenmann,
Faisal Saied, Mohamed Sayeed, Yili Zheng
Purdue University

1 Intro: Goals of HPC Benchmarking and Performance Evaluation

The goal of benchmarking and performance evaluation, as viewed in this paper, is to assess the performance and understand characteristics of HPC platforms and their important applications. An obvious use of the gained results is the search for machines that are best for a given purpose. Equally important uses are the creation of yardsticks for research and the anticipation of needs of future HPC technology.

The main thesis of this paper is that there is a dire need for basing such assessments on realistic computer applications. While the use of metrics that rely on measuring the behavior of simple program kernels has served us for many purposes in the past, it fails to answer questions that are of decisive importance today and in the future. These questions deal with the directions future HPC development should take, which have a direct impact on the competitiveness of our industry and nation.

Benchmarking and Performance Evaluation allow us to answer two important areas of questions, to which we will refer as the *Relevant Questions*:

1) **How do HPC platforms perform in solving todays important problems?** We may learn that a biology application may take a full year to fold a certain protein on a present-day HPC platform. Beyond such absolute time metrics, we may ask how computer platforms perform relative to others for a certain application or application area. Furthermore, we may be interested in finding out how certain system components perform for these applications. For example, we may ask for absolute communication times, the fraction of IØtaken in the overall execution, or the percentage of peak cpu power exploited.
2) **What are the characteristics of today's computational applications and how do these characteristics change in future problems?** We may want to know the memory required by a realistic weather forecast problem or how much communication will happen per quantity of computation. We may also ask how these properties change as a chemist increases the number of particles in a molecular dynamics simulation 100-fold.

In this paper we focus on the first area of questions. Key to both tasks is the use of today's realistic applications and realistic assessments of what represents future problems. Efforts to base performance evaluation and benchmarking on realistic applications exist, today. However, they are often overlooked, as we are still used to considering metrics that are based on kernel benchmarks or even raw machine performance. A main point of the present paper is to analyze the underlying reasons. We compare metrics supplied by kernel and realistic benchmark measurements and discuss their relationship. We find that in most regards, today's widely used benchmarks and performance evaluation methods do not answer the Relevant Questions. Realistic application benchmarks are able to provide these answers. However, we have to pay the cost that these codes are substantially more difficult to run.

Two properties of benchmarks are of paramount importance. First, by using realistic computer applications as our yardsticks for performance evaluation, we ensure that our observations are of *relevance*. Second, we must be able to *share* our yardsticks with others. For example, it is not meaningful to compare HPC systems using different applications that are proprietary to different organizations. To become useful yardsticks, these applications must be shared openly, so that the same programs can be run on both systems and that the features of the programs can be analyzed and understood in detail by the readership of the evaluation results. Relevance and openness are key requirements for all benchmarking efforts.

In the remainder of this paper we will discuss three main points:

- Although there has been a clear need for benchmarking and performance evaluation with realistic applications, doing so is by far not common practice. Section 2 describes the challenges faced by efforts that try to address this need.
- Efforts to provide realistic application benchmarks do exist. Section 3 describes the current state of these efforts. Section 4 describes the only sustained effort today that satisfies the above key requirements, SPEC HPC.
- Section 5 shows performance evaluation results of the SPEC HPC2002 benchmarks, using the *Relevant Questions* introduced above. We will review kernel and realistic benchmarks in their ability to provide answers.

2 Challenges for HPC Evaluation with Realistic Applications

The need for performance evaluation and benchmarking with realistic applications has been widely acknowledged. Better yardsticks have been called for by communities ranging from customers of HPC systems to computer manufacturers to research funding agencies to scientific teams. This section tries to explain the stark contrast between the clear need and present practice.

2.1 Simple Benchmarks are Overly Easy to Run

One of the greatest challenges for performance evaluation with realistic applications is the simplicity with which kernel benchmarks can be run. With a time investment of minutes to a few hours, one can generate a benchmark report that seemingly puts one's machine "on the map". The simplicity of Kernel benchmarks may overcome portability issues – no changes to the source code is necessary to port and optimize it to the new machine. The simplicity may also overcome software environment issues – the small code is unlikely to engage compiler optimizations that are not yet proven or to break programming tools that are available in a beta release. The simplicity may also overcome hardware issues – the kernel is unlikely to approach numerical instability, to which a new processor's floating point unit may be susceptible.

Unfortunately – for true HPC evaluation – these are the features that we *want* to impact the results. We do not want a machine to show up in the "Best 100" if it takes major code restructuring to port a real application, if porting such an application requires major additional debugging, if the tools and compilers are not yet mature, or if the hardware is still unstable.

Similar to kernel benchmarks, specialized benchmarks have been created for a large number of metrics. There are test suites for message-counting in MPI applications, measuring memory bandwidth of SMP platforms, gauging fork-join overheads of OpenMP implementations, and many more. The SPEC benchmarking organization (www.spec.org) alone distributes twelve major benchmark suites, today. Like kernel benchmarks, these diverse suites have an essential role, as they help us understand a specific aspect of a system in some depth. When it comes to understanding the behavior of a system as a whole, these detailed measures are not adequate, however. Even if there were a specific benchmark measuring each and every aspect of a system, there is no formula that helps us combine the diverse numbers into an overall system metric. For understanding overall behavior and for quantifying the contribution of a specific component to the whole system, we must consider the overall performance of a realistic application.

2.2 Realistic Benchmarks Cannot be Abstracted from Real Applications

A possible approach towards benchmarking with more realistic applications may be to extract important excerpts from real codes. In doing so, we want to preserve the features that are relevant, while omitting the unnecessary. Unfortunately, in this process, we are making decisions of what are the less important parts of an application, and these decisions may be wrong. For example, we may find a loop that executes 100 computationally near-identical iterations and reduce it to just a few iterations. However, this abstraction might render the code useless for evaluating adaptive compiler techniques; the repetitive behavior can be critical for the adaptive algorithm to converge on the best optimization technique. Similarly, data down-sizing techniques [1] may ensure that a smaller problem's cache behavior remains the same. However, the code would be useless for answering the question of what the memory footprint of the realistic problem would be.

The difficulty of defining scaled-down benchmarks is also shown by the many criteria for benchmark selection that have been suggested in past efforts to define new evaluation suites. It has been suggested that such codes be selected so they contain a balance of various degrees of parallelism, they should be scalable, simple yet reflective of real-world problems, they should use a large memory footprint and a large working set, they should be executable on a large number of processors, exhibit a variety of cache hit ratios, flops per cache miss, message densities, I/O activities, and they should be amenable to a variety of programming models, including shared-memory machines and clusters. Last but not least, benchmark codes should represent a balanced set of computer applications from diverse fields.

It is obvious that these demands cannot all be satisfied. In fact, some of them are directly contradictory. Furthermore, selecting our yardsticks by such criteria would dismiss the fact that we want to *learn* about these properties from real applications. For example, we want to learn how scalable a real application is, rather than selecting a scalable one in the first place. Similarly, if the realistic data sets of an application do not have large memory footprints, then this is an important result of our evaluation. Inflating input data parameters to fill some memory footprint benchmark selection criterion would not be meaningful.

We stipulate that the only realistic yardsticks for overall system performance are full, real applications. We need to face the challenge of using them in our evaluation efforts. Techniques need to be developed that measure, within the realistic problem and execution, the quantities that we wish to observe. For example, a computer architect may "sample" the execution of a full-size problem, by periodically turning on a simulator for detailed analyses.

2.3 Today's Realistic Applications May Not be Tomorrows Applications

Large, realistic codes tend to be "legacy codes" with programming practices that do not reflect tomorrow's software engineering principles and algorithms. For evaluating future machines, we need tomorrow's applications.

This is perhaps the strongest argument against using today's realistic applications for determining HPC needs of the future. However, when asking what these future applications should include, the answer is not forthcoming. Should we use selection criteria such as the ones discussed in Section 2.2? Are we sure that the best of today's algorithms and software engineering principles will find themselves in tomorrow's applications? If we choose that path and miss, we risk losing on two fronts: abandoning today's established practices and erring in what tomorrow's technology will be.

It has been argued that the best predictor of tomorrow is today's established practice. Using current, real applications combined with a continuous benchmark update process may be our best option. The update process could be similar to the one used in the SPEC benchmarking organization, where the selection of the next benchmark suite begins immediately after the latest release.

2.4 Benchmarking is not Eligible for Research Funding

Performance evaluation and benchmarking projects are long-term infrastructure efforts. Performance data needs to be gathered and kept for many years[1]. Furthermore, this task does not easily include advanced performance modeling topics, as would be of interest for scientific research in this area. The authors of this paper have been involved in a number of benchmarking projects sponsored by academic funding agencies, all of which lacked continuity.

Establishing programs to create such community services at funding agencies may be possible, and this paper may help motivate such initiatives. One alternative is a combined academic/industrial effort, such as SPEC's High-Performance Group (HPG). In this group, both industrial benchmarking needs and academic interests joined forces to develop suites of realistic high-performance computing applications. To the authors' knowledge, SPEC HPG is the only current and sustained effort at performance evaluation, satisfying the criteria of relevance and open sharing. We will describe the effort further in this paper.

2.5 Maintaining Benchmarking Efforts is Costly

Maintaining a performance evaluation and benchmarking effort entails collecting test applications, ensuring portability, developing self-validation procedures, defining benchmark workloads, creating benchmark run and result submission rules, organizing result review committees, disseminating the suite maintaining result publication sites, and even protecting evaluation metrics from misuse. As explained above, this process needs to re-start periodically – say, every 3–5 years. Dealing with realistic, large-scale applications further necessitates providing assistance to benchmark users and nurturing the involvement of domain experts in the respective application areas.

The high cost of these tasks is obvious. Many benchmarking efforts have covered their costs through initial research grants or volunteer efforts – the support typically covered the first round of benchmark definitions, but failed to sustain subsequent steps or result publication sites. Important representatives of efforts that aimed at real applications were the Perfect Benchmarks [2] and the ParkBench [3] collection, described more thoroughly in Section 3. The NAS parallel benchmarks [4] represent a notable intermediate step. They are a collection of core algorithms extracted from real applications in computational fluid dynamics.

The SPEC organization is the only organization that has maintained full, continuously updating benchmarking efforts. SPEC funding comes primarily from the involved industrial membership, plus a comparably small fee for the actual benchmark suites. SPEC is perhaps best known for its CPU benchmarking suites, which update every 4-6 years (SPEC CPU95, SPEC CPU2000, SPEC CPU2006 – in preparation). SPEC's HPC suite has a longer update cycle (SPEC HPC96, SPEC CPU 2002). As SPEC HPC is the only sustained HPC evaluation effort today, we will use it as the main reference point, in this paper.

2.6 Proprietary Full-Application Benchmarks Cannot Serve as Yardsticks

It has been argued that, if realistic benchmarks are to be used, they must be the exact applications that will run on the target system of interest to the reader. Many such applications are proprietary. The use of proprietary applications for the evaluation of computer systems is in direct conflict with our criterion for open sharing of yardsticks. Clearly, for the prospective customer of a computer system, their own applications seem the best choice for testing the desired functionality of a system. If multiple vendors commit to running these applications in a way that can be compared fairly, this may be best for the customer. However, as the applications are proprietary, the generated performance claims can neither be verified nor scrutinized by the scientific community or the public. Hence their significance outside the customer-vendor relationship is small.

It is worth noting that, even to the above customers, public benchmarks may be of higher value. As described in Section 2.5, generating fair benchmark results is very costly. During a procurement phase, vendors are under high pressure to produce good results in a very short period of time, often competing internally for machine resources that can be used for benchmarking purposes. Unless the customers supervise this process very closely and with significant expertise, they may not obtain results that can be compared in a fair manner. Without any deceptive intentions on the vendor side, the lack of an extremely clear evaluation methodology often allows shortcuts and "optimizations" to be taken that may differ significantly between the involved evaluation groups. By contrast, when using established, public full-application benchmark results, although these applications may not contain computational patterns identical to the customer's codes, the consistency and time-to-availability of the performance numbers may outweigh this drawback.

[1] The 15 year old perfect-benchmarks@csrd.uiuc.edu mailinglist still receives occasional queries

3 State of the Art of Benchmarking with Realistic Applications.

Performance evaluation efforts with real applications have begun to emerge in 1988 with the Perfect Benchmarks [2], [5]. This set of scientific and engineering applications was created with the intent to replace kernel-type benchmarks and the commonly used peak-performance numbers. The Perfect benchmarks were a significant step in the direction of application-level benchmarking. Perhaps more important than their use for overall machine benchmarking was the presence in the research community for evaluating new software and hardware techniques and components. The Perfect Benchmarks introduced a methodology of *performance diaries,* which recorded program modifications and the performance improvements they led to. The Perfect codes continue to be distributed to the research community for current performance evaluation efforts. The Perfect Benchmarks effort was funded by research grants. The "Perfect 2" effort was unable to obtain continued funding and did not result in the dissemination of a new suite. The original Perfect Benchmarks are small compared to today's realistic applications (the largest included some 20,000 lines of Fortran 77) and their data sets execute in the order of seconds on today's machines. No result web site is available for the Perfect Benchmarks.

Similar to the Perfect Benchmarks, the ParkBench [3] effort was created with initial research funding, but did not update its initial suites in response to newer generations of HPC systems. The effort was very ambitious in its goal of delivering a set of benchmarks that range from kernels to full applications. The largest, full-application suite was never created, however.

The Standard Performance Evaluation Corporation (SPEC) was also founded in 1988. SPEC is largely vendor-based, although the organization includes a range of academic affiliates. Initially, SPEC focused on benchmarking uniprocessor machines and, in this area, has become the leader in providing performance numbers to workstation customers. SPEC suites have also increasingly been used by the research community for evaluating the performance of new architecture concepts and software prototypes. Today, SPEC offers a large number of benchmarks that evaluate workstations, graphics capabilities, and high-performance systems. Most notably for HPC evaluation, in 1994, SPEC's High-Performance Group was formed, out of an initiative to merge the expertise in high-performance computer evaluation of the Perfect Benchmarks effort with SPEC's capability to sustain such efforts long-term. Since its foundation, this group has produced a number of benchmarks for HPC, including the HPC suite [6], [7], [8], the OMP suite (OpenMP applications) [9], [10], [11] and the MPI suite (MPI applications, under development). The SPEC HPC suite satisfies the criteria of relevance and open sharing, provides a result submission and review process, a result repository, and a continuous benchmark update process. The remaining sections of this paper will describe this suite in more detail.

A range of other attempts to provide HPC benchmarks have been made, over the past decade. A notable example is the Euroben [12] effort (www.euroben.nl). Also, the NAS [4] parallel benchmarks aim at HPC benchmarking with application codes. The suite includes several small codes derived from computational fluid dynamics applications, also referred to as the NAS kernels.

A notable, recent effort is the benchmarking project of DARPA's[2] High-Productivity Computing Systems (HPCS) [13] program. Currently, this effort has defined suites of kernel benchmarks and a number of synthetic compact applications. Also, the National Science Foundation's High Performance Computing System Acquisition program defines a set of benchmarks that include realistic applications satisfying the criteria of relevance and openness [14]. Currently, there is no associated effort to create and maintain a benchmark result repository that could be viewed by the public, however.

3.1 Metrics for HPC Evaluation

There is general agreement that overall performance must be evaluated using wallclock time measurements. One open and often controversial issue is how to combine such measurements for multiple benchmarks into one number. SPEC HPC's approach is to leave this decision up to the reader of the benchmark reports. That is, each code is reported separately. Other suites, such as SPEC CPU and SPEC OMP, report the geometric mean of the individual program performance results. Similarly, the TAP list, which ranks the SPEC HPC results (see Section 4.2) defines an aggregate performance metric.

Kernel benchmarks evaluate a wide variety of system components. Accordingly, the metrics vary widely. This is also true for metrics that characterize computational applications. Examples are working set sizes, hardware counter values (cache hit rates, instruction counts), and software metrics (code statistics, compiler results).

3.2 Tools for Gathering Metrics

Obtaining overall timing metrics is relatively straightforward. By contrast, tools for gathering detailed execution characteristics are often platform-specific. Therefore, it can be difficult to obtain the metrics of interest on a given platform; it is even more difficult to conduct comparative evaluations that gather a certain metric across a number of platforms. Among the tools we used in our projects are *mpiP*, *hpmcount* and *strace*. mpiP is a lightweight profiling library for MPI applications, that reports the percentage of time spent in MPI [15]. It includes the times used by each MPI function. IBM's Hardware Performance Monitor [16] suite includes a simplified interface, hpmcount, which summarizes data from selected hardware counters, and computes some useful derived metrics, such as instructions per cycle and flops per cycle. We measured I/O behavior by recording I/O system call activities, using the strace command; from this output we extracted statistics for file I/O using a script.

These tools are but a small sample of a large set of instruments available on the myriad of today's platforms. An

[2] Defense Advanced Research Project Agency

important goal is to achieve uniformity. Tools and interfaces need to be developed that allow benchmarkers to gather relevant performance data consistently across the range of available platforms. Ideally, these tools will not just report volumes of performance counter results; they will be able to abstract these volumes, creating the end-metrics that are of interest to the reader.

4 SPEC HPC Benchmarks

This section describes the SPEC HPC benchmarks, which is the result of an ongoing effort to make available as a benchmark suite relevant and openly shared, full applications. The section also describes a project to use these benchmarks in a rank list of HPC platforms based on these applications.

4.1 The SPEC HPC2002 suite

The SPEC HPC benchmarks are based on computational applications that are in wide use and that can be openly distributed to the community. The codes are implemented using the MPI and OpenMP standards for parallel processing. The intended consumers of the benchmark results include end-users, system vendors, software vendors, and researchers.

The SPEC HPC2002 suite improves upon and replaces SPEC HPC96. It comprises three benchmarks. SPECseis, SPECchem, and SPECenv. Each code has a small- and a medium-sized data set (with increasingly larger sets under development).

SPECseis: is also known as *Seismic*. It is a suite of codes typical of seismic processing applications used in industry for the search of oil and gas. The code consists of four configurable application phases: data generation, stacking of data, time migration, and depth migration. The first phase generates synthetic seismic traces from a configuration of simple subsurface structures provided by the data set. Stacking of data is used to reduce the large volume of data that has to be processed by summing seismic traces that have a common midpoint between the source and receiver of the trace. Time and depth migration image the subsurface structures. Seismologists could then use the images to locate cavities where it is most likely to find an oil reservoir.

Two migration (imaging) techniques are included since both are prevalent in today's industry. Time migration transforms the traces into the Fourier domain and solves for the intersection of the wave f source to the point of reflection and the wave from the point of reflection to the receiver. On the other hand, depth migration uses a finite difference scheme to propagate waves down in depth. Time migration is significantly faster than depth migration but assumes that velocity is horizontally static whereas depth migration accounts for laterally varying velocity, and can therefore handle surfaces at grades as steep as 45 degrees.

SPECseis includes approximately 25,000 lines of Fortran and C code. It can run in OpenMP or MPI mode.

SPECchem: is also known as *GAMESS*. This code is often used to exhibit performance of high-performance systems among computer vendors. Portions of GAMESS codes date back to 1984. It comes with many built-in functionalities, such as various field molecular wave-functions, certain energy corrections for some of the wave-functions, and simulation of several different phenomena. Depending on what wave-functions are chosen, GAMESS has the option to output energy gradients of these functions, find saddle points of the potential energy, compute the vibrational frequencies and IR intensities, and more.

GAMESS can compute SCF wavefunctions using methods ranging from RHF, ROHF, UHF, GVB, and MCSCF. Correlation corrections to these SCF wavefunctions include Configuration Interaction, second order Perturbation Theory, and Coupled-Cluster approaches, as well as the Density Functional Theory approximation. Nuclear gradients are available, for automatic geometry optimization, transition state searches, or reaction path following. Computation of the energy hessian permits prediction of vibrational frequencies, with IR or Raman intensities. Solvent effects may be modeled by the discrete Effective Fragment Potentials, or continuum models such as the Polarizable Continuum Model. Numerous relativistic computations are available, including third order Douglas-Kroll scalar corrections, and various spin-orbit coupling options. The Fragment Molecular Orbital method permits use of many of these sophisticated treatments to be used on very large systems, by dividing the computation into small fragments.

SPECchem includes 120,000 lines of Fortran and C code. It can run in OpenMP, MPI or mixed MPIOpenMP mode (hybrid).

SPECenv: which is also known as WRF, is developed within the Weather Research and Forecasting Modeling System development project. It is a next-generation mesocale numerical weather prediction system designed to serve both operational forecasting and atmospheric research needs. The project is being undertaken by several agencies. Members of the WRF Scientific Board include representatives from EPA, FAA, NASA, NCAR, NOAA, NRL, USAF and several universities. SPEC HPG integrated version 1.2.1 of the WRF weather model into the SPEC tools for building, running and verifying results. This means that the benchmark runs on more systems than WRF has officially been ported to. The benchmark runs use restart files that are created after the model has run for several simulated hours. This ensures that cumulus and microphysics schemes are fully developed during the benchmark runs. The code features multiple dynamical cores, a 3-dimensional variational (3DVAR) data assimilation system, and a software architecture allowing for computational parallelism and system extensibility. WRF is suitable for a broad spectrum of weather modeling applications across scales ranging from meters to thousands of kilometers.

SPECenv includes 180,000 lines of Fortran90 and C code. It can run in OpenMP, MPI or mixed MPIOpenMP mode (hybrid).

4.2 TAPList: Ranking HPC Systems with Real Applications

The benchmark reports of the SPEC HPC applications have been used to create a rank list of HPC platforms based

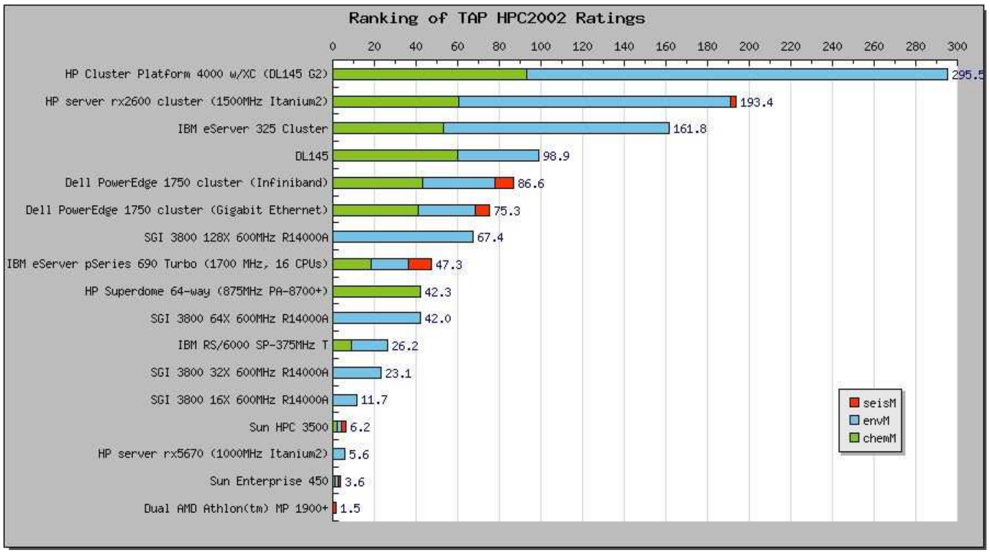

Fig. 1. TAPlist: Rank List of Top Application Performers. In this aggregate view, the SPEC HPC2002 results of three benchmarks are combined into an overall rank. The bars are subdivided into the contributions of each benchmark to the rank.

on realistic applications (www.purdue.edu/TAPlist). Figure 1 shows the list as of January 2006. The TAP list defines an aggregate metric with which the results of the three benchmark applications can be combined into an overall rank. The metric weights the individual application performance results according to their average runtime across the different platforms. The list also allows a single benchmark to be used for ranking.

Using the aggregate metric, the system currently ranking on top is an HP server DL145 G2. The TAP list contains links to the SPEC HPC 2002 reports. The GAMESS (SPECchem) result of the top performer was generated with a 128-processor run and the WRF (SPECenv) result used a 192-processor system. No Seismic (SPECseis) results were submitted for this machine. An interesting observation is that none of the platforms ranking high on kernel-based rank lists, such as the Top 500 Supercomputer Sites (www.top500.org), have reported numbers for SPEC HPC20002. Some of the reasons were discussed in Section 3.

5 Performance Results

This section presents performance results of the SPEC HPC2002 benchmarks in terms of the *Relevant Question #1* introduced in Section 1. Where appropriate for comparison, results obtained from kernel benchmarks are shown. For each type of result we discuss the degree to which kernels and real application benchmarks are able to provide answers.

5.1 Overall Performance

Absolute execution times: Table 1 shows overall problem solving times of the SPEC HPC applications. All measurements have been taken using the *medium* datasets. The used machines include an IBM P690 (www.ccs.ornl.gov/Cheetah/Cheetah.html), an SGI Altix (www.ccs.ornl.gov/Ram/Ram.html) and an Intel Xeon cluster (www.itap.purdue.edu/rcac/news/news.cfm?NewsID=178). The table describes the problems solved by these data sets. The run times for the medium data sets of SPEC HPC2002 on an IBM P690 platform range from approximately 10 minutes (for Seismic) to over an hour (for GAMESS).

Relative Application Performance: Figure 2 shows the relative performance of the individual benchmark applications on three machines, an Intel Xeon cluster, an SGI Altix, and an IBM P690 platform. For comparison, the figure also shows the performance of the HPL [17] benchmarks. The measurements were taken up to 64 processors, except for Seismic (up to 32; the 64-processor runs did not validate).

The three applications lead to different rankings of the executing machines. The Altix machine performs the best except for the Seismic benchmark. The Xeon cluster performs best for the Seismic benchmark. The WRF benchmark does not scale beyond 32 processors on the Altix and Xeon cluster systems, but still scales up to 64 processors on the P690 platform. The Seismic benchmark shows poor scaling on Altix and shows slightly better scaling behavior on the other two platforms, up to 16 processors.

In terms of the scaling behavior, shown in Figure 3, the P690 platform performs best on both Seismic and WRF, but worst on GAMESS. The Altix machine is worst on Seismic, whereas the Xeon cluster is worst on WRF. For GAMESS, the Altix machine performs best. Both the Altix and Xeon cluster platforms show superlinear behavior up to 32 processors for GAMESS.

The kernel benchmark results (HPL with N=9900) are most similar to those of WRF.

5.2 Component Performance

Measurements of system components allow us to gain insight into the behavior of individual machine features. Their relative performance shows the contribution of a feature to

TABLE 1. Wallclock execution times and problems solved by the SPEC HPC applications on the IBM P690 platform, using 32 processors.

Application	Execution time	Problem description for *medium* data set
SPECseis (Seismic)	625s	The data set processes seismic traces of $512 \times 48 \times 128 \times 128$. (samples per trace×traces per group×groups per line×# of lines, where a trace corresponds to a sensor that has a sampling frequency, the sensors are strung out on multiple cables behind a ship.) The total data set size in the Phase 1 of SPECseis is 1.5 GB. It is reduced to 70 MB in Phase 2.
SPECchem (GAMESS)	3849s	The data set Computes SCF (Self-Consistent Field) wavefunctions (RHF type) for *thymine* (C5H7N3O – 16 atoms), one of the bases found in nucleic acids.
SPECenv (WRF)	742s	The data set simulates the weather over the Continental United States for a 24 hour period starting from Saturday, November 3nd, 2001 at 12:00 A.M. The grid is of size 260x164x35 with a 22km resolution.

the overall solution of a computational problem. Component performance relative to some upper bound shows us how efficiently the machine feature is exploited, compared to theoretical limits. The following figures show selected measurements of the communication, computation, and I/O components.

Figure 4 shows times taken by the communication operations, on a per-processor basis. For Seismic, the communication is shown separately for the four phases of the execution. For HPL, two different data sets are shown. Communication takes from 5% to 25% of the overall execution time, with Seismic doing the least amount of communication, followed by WRF and then GAMESS. HPL communicates less than the application codes; with increasing data set size, communication reduces significantly. This feature of the kernel benchmark leads to the generally good scaling behavior on very large machines; it contributes to the difference in rank lists of realistic versus kernel benchmarks, mentioned in Section 4.2.

The most apparent feature in all graphs of Figure 4 is the significant communication load imbalance. In Phase 4 of Seismic, communication appears to increase linearly and vary significantly with the processor number. In GAMESS, a single processor communicates 100% of the time. WRF and HPL are the most balanced; however, the difference between the least and most communicating processors is still in the 50% range.

Parallel I/O in Seismic: Figure 5 shows disk I/O volume and time for Seismic, on a 32-processor run. (In GAMESS and WRF, I/O is performed on one processor only; HPL has no disk I/O). The graphs show the four phases of Seismic in sequence. The I/O in Phase 1 dominates and the volume of read operations is an order of magnitude less than that of write operations. While these I/O volumes are balanced, the I/O times taken on the 32 processors exhibit significant load imbalance. Also, read and write times are similar, despite the differences in volumes.

Effective I/O Bandwidth: Figure 6 shows the overall I/O volume and percent time taken. From that data, the effective bandwidth is computed. For Seismic, the four phases are measured separately. The figures are in logarithmic scale.

The I/O volumes in the six codes (four Seismic phases, WRF, and GAMESS) range form 61 MB to 5545 MB. The fraction of execution time taken by the I/O is small in both GAMESS and WRF, but significant in Seismic. The computed effective I/O bandwidth numbers show significantly less efficient use of I/O in Seismic than in the other two codes.

Memory Footprints: Figure 7 shows the memory footprints of the benchmarks, as a function of the number of processors. Again, Seismic is split into its four execution phases. There are two different types of behavior. WRF and Seismic Phase 3 and 4 exhibit the commonly expected behavior: the memory footprint decreases steadily with increasing processor numbers. By contrast, in Seismic Phase 1 and 2 and GAMESS, the memory footprint is independent of the number of processors. This finding is important, as it refutes the common assumption that larger systems will be able to accommodate larger data sets. This assumption is the basis for a benchmark methodology that allows data sets to “scale” and thus reduce communication (as discussed in Figure 4 for HPL), leading to seemingly improved performance numbers on large systems. Our results show that this path to scalability may not be correct.

5.3 Discussion of Kernels versus Full Applications

TABLE 2. Comparison of kernel versus full-application metrics in their ability to answer the *Relevant Questions*. (✓=good answer; –=limited answer; n/a=no answer)

Question	Ability to answer	
	Kernels	Full Apps.
What time is required to solve important computational problems on todays HPC platforms?	n/a	✓
What is the relative, overall performance of HPC platforms?	–	✓
How do system components perform?	✓	–
What is the importance of system components relative to each other?	n/a	✓
What is the importance of system components relative to upper bounds?	–	✓
What are the characteristics of important computational problems?	–	✓
What are the characteristics of important future problems?	–	(✓)

Table 2 compares kernel versus full application benchmarks in their ability to answer the *Relevant Questions*. For obvious reasons, kernel benchmarks do not provide answers to the first question; execution times of real problems cannot be inferred from kernel execution times. By contrast, as SPEC HPC2002

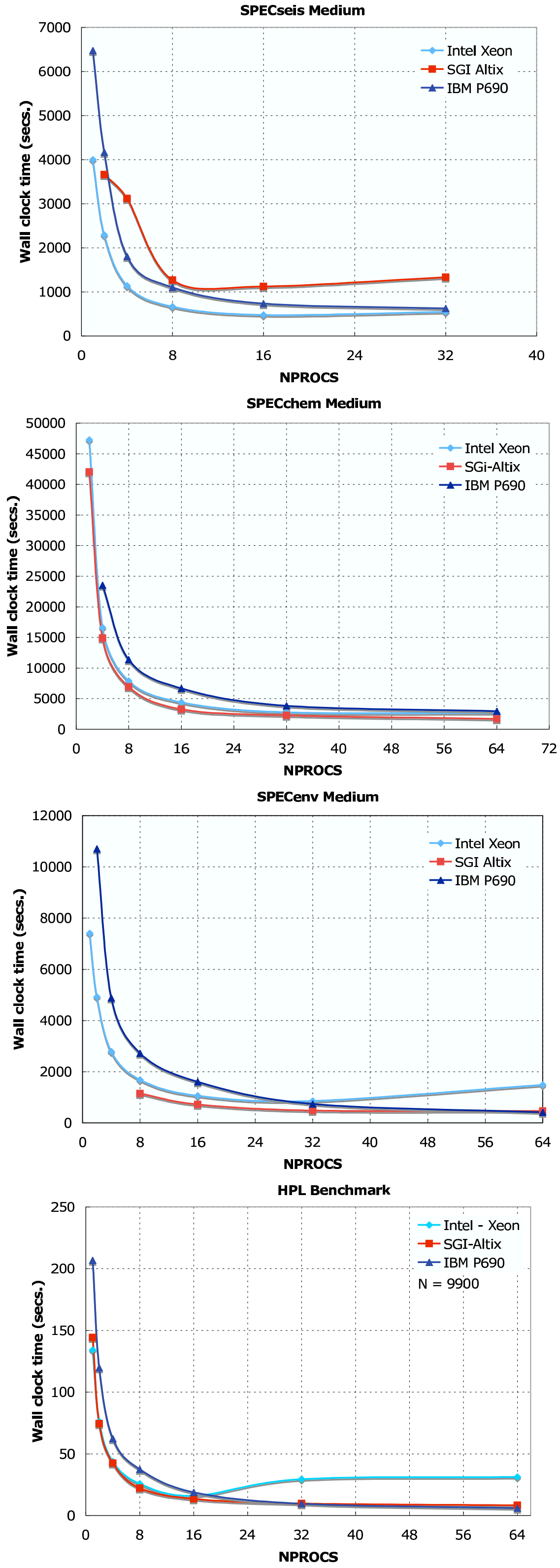

Fig. 2. Execution times of the SPEC HPC applications on three platforms. For comparison, the execution times of the same machines using the HPL kernel benchmarks are shown.

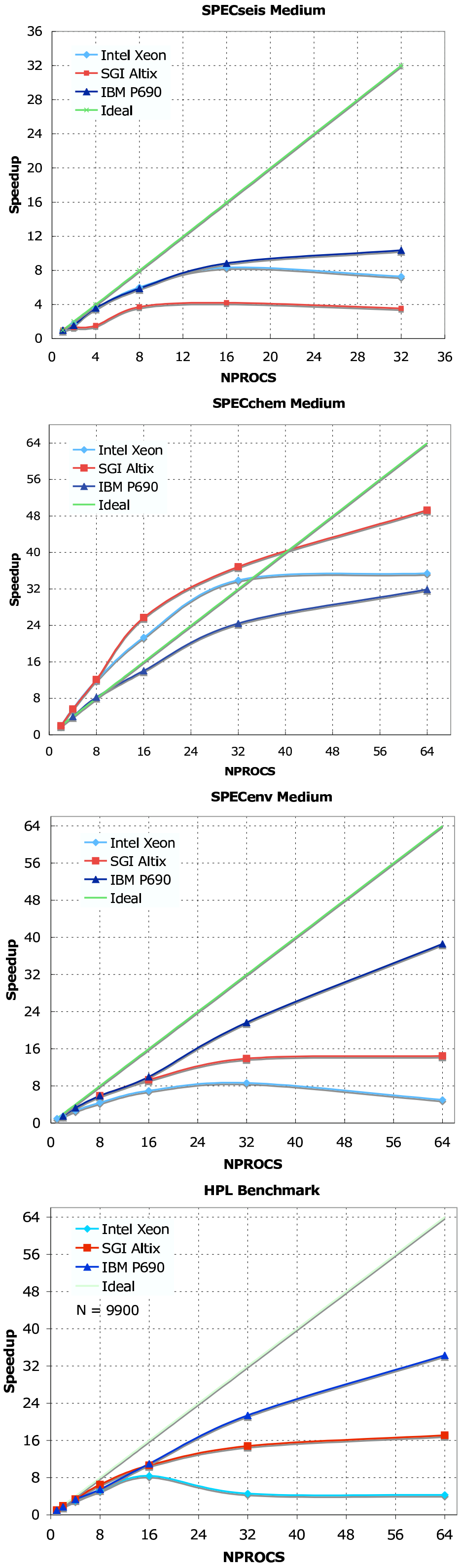

Fig. 3. Scaling behavior of the SPEC HPC applications on three platforms. For comparison, the scaling behavior of the same machines using the HPL kernel benchmarks are shown.

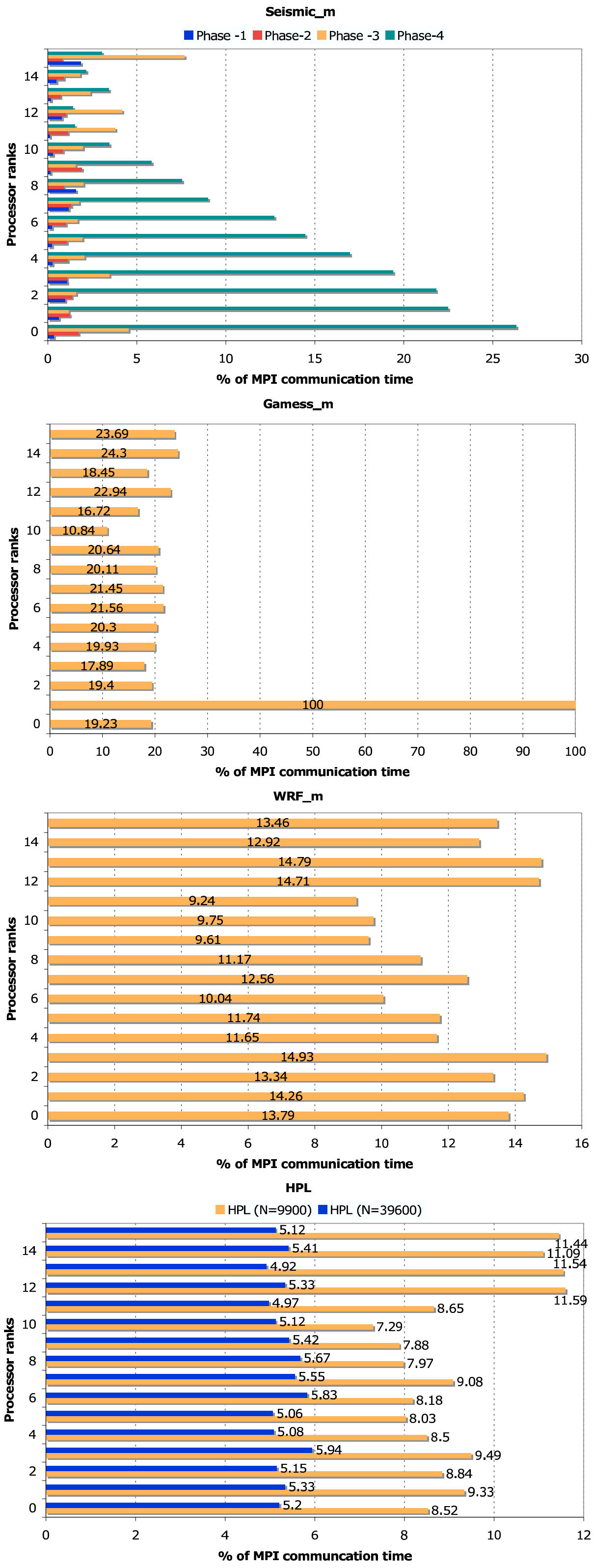

Fig. 4. Communication times of the SPEC HPC 2002 and the HPL benchmark, on 16 processors of an IBM P690 system, as percent of overall execution time.

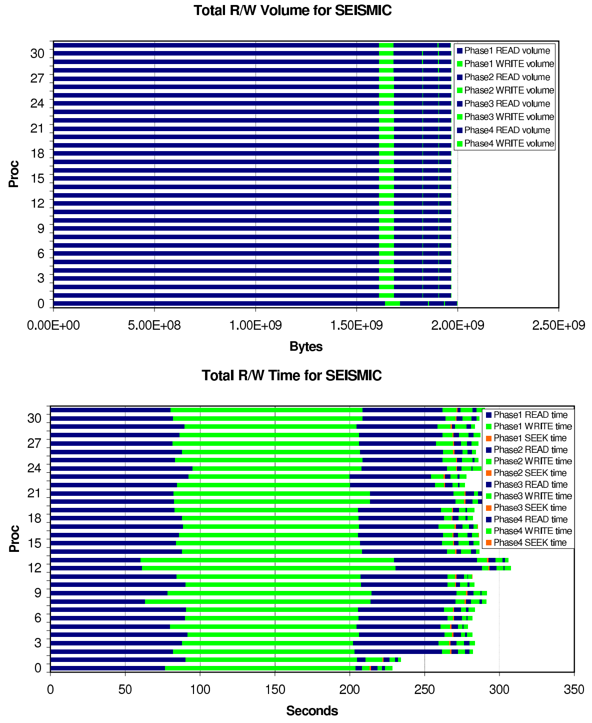

Fig. 5. Disk I/O volume and time spent in Seismic, on 32 processors of an Intel Xeon cluster. In both WRF and GAMESS, a single processor performs all I/O.

benchmarks and their data sets represent real applications that are in significant use today, they provide direct answers.

For the question of relative performance, an answer can be given by both types of benchmarks. As Figure 2 has shown, the ranking depends on the application. While the HPL kernel performance behavior has similarities to the WRF application, the behaviors of GAMESS and Seismic are significantly different.

While application benchmarks give us a realistic picture of the absolute and relative performance of system components, kernels give limited answers. The strength of specialized, kernel benchmarks is to measure individual machine features. Obtaining detailed diagnoses requires us to focus on a particular system aspect, for which kernel programs are most adequate. Comparison across machines may also be useful; however, the results do not answer the question of the relative importance of components. For example, combining Figure 4 and 6 would break down an execution into computation, communication and I/O. Obviously, a kernel metric cannot do the same.

Kernel metrics may give an answer to the question of what percentage of peak (cpu power, communication or I/O bandwidth) can be achieved. They are generally useful as idealized bounds under a given code pattern. These result cannot be interpreted to represent the percentage of peak reached by a real application.

Figure 7 gave a good example of answers that differ drastically between real application and kernel benchmarks. Even though the HPL benchmark is often used as a yardstick

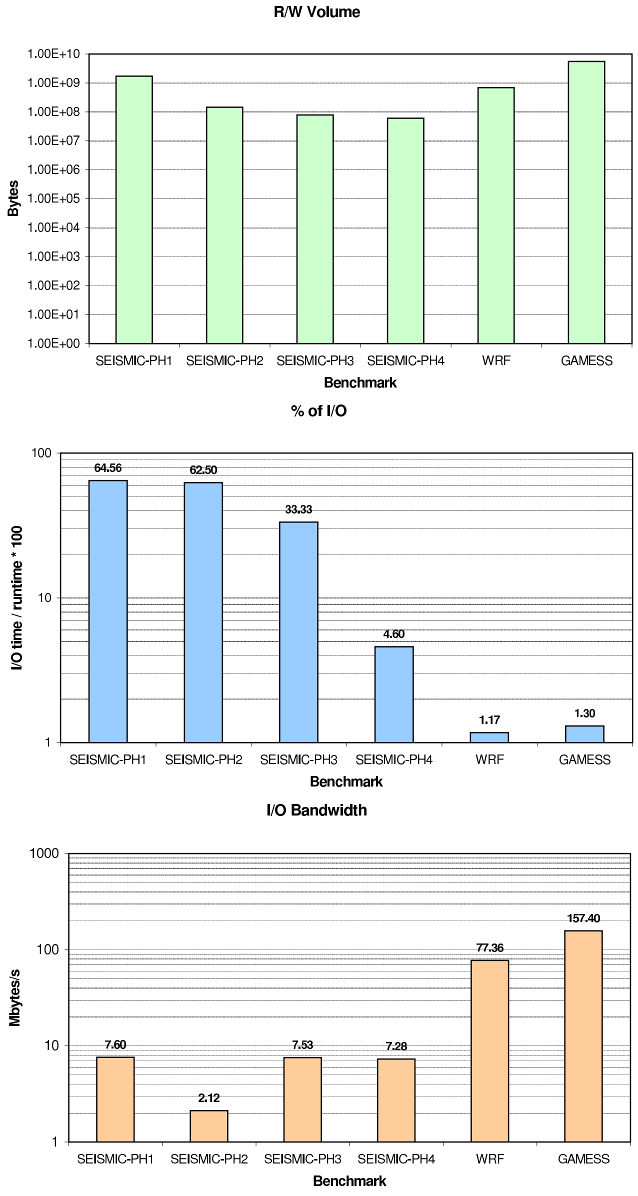

Fig. 6. Overall I/O volume, time, and effective I/O bandwidth used on 32 processors of an Intel Xeon cluster

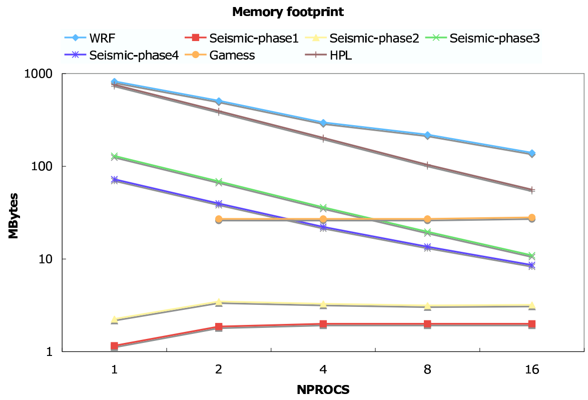

Fig. 7. Benchmark memory footprints (total memory size per processor) on an Intel Xeon cluster

for communication-oriented tasks, it does not represent the behavior of half of the measured applications.

Table 2 also lists two issues belonging to Relevant Question #2, which deals with the characteristics of applications. Evidently, this is an area where kernel benchmarks cannot help. How to predict future application behavior is controversial. It has been argued [18] that kernels may be better predictors, as they are more flexible in extrapolating into various dimensions. In Section 2.3 we have argued that such extrapolations are significantly based on assumptions about the important versus less important parts of future applications; measuring today's realistic applications may be the better basis for extrapolations into the future.

6 Conclusions

There is a dire need for basing performance evaluation and benchmarking results on realistic applications. We have discussed challenges in doing so and reviewed the state of the art in this field. The SPEC HPC effort is a unique effort that satisfies the main criteria for real-application benchmarking: relevance and openness. We have presented measurements of the SPEC HPC 2002 codes, giving answers to the *Relevant Questions* that benchmarking aims to answer. We have compared our results with those obtained from kernel benchmarks (HPL). In comparing kernel versus full-application benchmarks we find that kernel benchmarks are the best choices for measuring individual system components. While this is an important aspect of performance evaluation, there is a large range of questions that can only be answered satisfactorily using real-application benchmarks.

Acknowledgment

This work was supported, in part, by the National Science Foundation under Grants No. 9974976-EIA, 0103582-EIA, and 0429535-CCF. The machines used in our experiments were made available by Purdue University's Rosen Center for Advanced Computing (RCAC) and the Oak Ridge National Laboratory (ORNL).

References

[1] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," in *Proceedings of the 22nd International Symposium on Computer Architecture*, 1995, pp. 24–36.

[2] M. Berry, D. Chen, P. Koss, D. Kuck, L. Pointer, S. Lo, Y. Pang, R. Roloff, A. Sameh, E. Clementi, S. Chin, D. Schneider, G. Fox, P. Messina, D. Walker, C. Hsiung, J. Schwarzmeier, K. Lue, S. Orszag, F. Seidl, O. Johnson, G. Swanson, R. Goodrum, and J. Martin, "The Perfect Club Benchmarks: Effective Performance Evaluation of Supercomputers," *International Journal of Supercomputer Applications*, vol. 3, no. 3, pp. 5–40, Fall 1989.

[3] R. W. Hockney and M. Berry (Editors), "PARKBENCH report: Public international benchmarking for parallel computers," *Scientific Programming*, vol. 3, no. 2, pp. 101–146, 1994.

[4] David Bailey, Tim Harris, William Saphir, Rob van der Wijngaart, Alex Woo, and Maurice Yarrow, "The nas parallel benchmarks 2.0," Tech. Rep. NAS-95-020, NASA Ames Research Center, dec 95.

[5] George Cybenko, "Supercomputer Performance Trends and the Perfect Benchmarks," *Supercomputing Review*, pp. 53–60, April 1991.

[6] Rudolf Eigenmann and Siamak Hassanzadeh, “Benchmarking with real industrial applications: The SPEC High-Performance Group,” *IEEE Computational Science & Engineering*, vol. 3, no. 1, pp. 18–23, Spring 1996.

[7] Rudolf Eigenmann, Greg Gaertner, Faisal Saied, and Mark Straka, *Performance Evaluation and Benchmarking with Realistic Applications*, chapter SPEC HPG Benchmarks: Performance Evaluation with Large-Scale Science and Engineering Applications, pp. 40–48, MIT Press, Cambridge, Mass., 2001.

[8] Matthias S. Müller, Kumaran Kalyanasundaram, Greg Gaertner, Wesley Jones, Rudolf Eigenmann, Ron Lieberman, Matthijs van Waveren, and Brian Whitney, “Spec hpg benchmarks for high performance systems,” *International Journal of High-Performance Computing and Networking*, vol. 1, no. 4, pp. 162–170, 2004.

[9] Vishal Aslot and Rudolf Eigenmann, “Performance characteristics of the spec omp2001 benchmarks,” in *Proceedings of the 3rd European Workshop on OpenMP (EWOMP’2001)*, Barcelona, Spain, September 2001.

[10] Vishal Aslot, Max Domeika, Rudolf Eigenmann, Greg Gaertner, Wesley B. Jones, and Bodo Parady, “SPEComp: A New Benchmark Suite for Measuring Parallel Computer Performance,” in *Workshop on OpenMP Applications and Tools, WOMPAT 2001, Lecture Notes in Computer Science, 2104*, 2001, pp. 1–10.

[11] Vishal Aslot and Rudolf Eigenmann, “Performance characteristics of the spec omp2001 benchmarks,” in *Proc. of the European Third European Workshop on OpenMP (EWOMP’2001)*, Barcelona, Spain, September 2001.

[12] A. J. van der Steen, “The benchmark of the EuroBen group,” *Parallel Computing*, vol. 17, pp. 1211–1221, 1991.

[13] “High productivity computer systems, benchmark links,” http://www.highproductivity.org/Benchmarks.htm.

[14] National Science Foundation, “Benchmarking information referenced in the nsf 05-625 ”high performance computing system acquisition: Towards a petascale computing environment for science and engineering”,” http://www.nsf.gov/pubs/2006/nsf0605/nsf0605.jsp.

[15] Jeffrey Vetter and Chris Chambreau, “mpip: Lightweight, scalable mpi profiling,” www.llnl.gov/CASC/mpip/, 2005.

[16] Advanced Computing Technology Center IBM, “Hardware performance monitor,” www.research.ibm.com/actc/projects/hardwareperf.shtml.

[17] A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary, “Hpl - a portable implementation of the high-performance linpack benchmark for distributed-memory computers,” http://www.netlib.org/benchmark/hpl/, Jan. 2004.

[18] David H. Bailey and Allan Snavely, “Performance modeling: Understanding the past and predicting the future,” in *Euro-Par 2005 Parallel Processing: 11th International Euro-Par Conference*, 2005, p. 185.