Parallel Computing 17 (1991) 1067–1069
North-Holland

Guest editorial

Benchmarking of high performance computers

Jack Dongarra and Wolfgang Gentzsch

Advanced computer architectures, such as RISC, vector and parallel computers, are today highly complex and their modes of operation for some applications may present challenges in extracting maximum efficiency.

Manufacturers often give 'Peak Performance' numbers (MIPS, MFLOPS) that are considerably higher than the performance numbers based on real applications (e.g. Linpack MFLOPS) and, therefore, meaningless for the users of these machines.

The performance of a computer is a complicated issue and a function of many interrelated quantities. These quantities include: the application, the algorithm, the size of the problem, the high-level language, the implementation, the level of human effort used to optimize the program, the compiler's ability to optimize, the age of the compiler, the operating system, the architecture of the computer, and the hardware characteristics.

The task of evaluation in this multifaced domain becomes a balancing act, weighing in all of the qualitative and quantitative issues, understanding the nature and scope of the problem, and navigating through the myriad paths available while avoiding the many pitfalls along the way.

For the end user, only the resulting execution time of a real application is important. Therefore the best way of performance evaluation is to run the user's application program. However, it is not always possible, for obvious reasons, to run the complete application on each machine in question.

The usual method to evaluate the performance of a computer is to compose a benchmark of kernel and application programs which should have one or more of the following characteristics. They

- are written in a high-level language, making it portable across different machines,
- are representative for a certain class of real applications,
- can be measured easily, and
- are widely distributed.

The codes used in scientific benchmarking studies belong to essentially three major categories: kernel, algorithm, and application, each measuring some aspect of a computer system's performance. Unfortunately, however, a benchmark may tell you more about the quality of your software, than it tells about the quality of the tested computer! Therefore, no single finite collection of codes can possibly aspire to being the final solution.

To present a useful overview on benchmarking, we invited over a dozen experts in this field to contribute papers on the three main topics concerning benchmarking advanced scientific

computer systems: taxonomy and performance metrics, well-known standard and application benchmarks, and benchmarks for parallel computers. We have tried to assemble an overview of performance measurement and benchmarking of high-performance computers as it exists today.

The first part contains contributions which consider the problems of taxonomy and performance metrics of computer architectures in theory and practice. The literature is full of taxonomies of computer architectures, but until recently no rigorous taxonomy of performance metrics exists. This rather curious deficiency in the content of computer science contributes to the controversy over performance-evaluation methods.

In *Worlton*'s article 'Toward a taxonomy of performance metrics', a general attempt is made to introduce the subject and to suggest some modest steps in the direction of taxonomy. It introduces to the concepts of taxonomies and dimensional analysis, and explores some of the elements of a taxonomy of performance metrics.

In a second paper on this subject 'Toward a better parallel performance metric', *Sun* and *Gustafson* concentrate on two performance metrics, sizeup and general speedup, and suggest to replace the older parallel speedup. Theoretical and experimental results show that this most commonly used performance metric favors slow processors and poorly coded programs. In particular, the algorithm with the fastest speed does not necessarily execute in the least time.

In order to understand better the behavior of a computer, it is helpful also to define relatively simple performance parameters that highlight particular aspects of performance, or in particular, bottlenecks, and to design relatively simple benchmarks specifically to measure these parameters. *Hockney*, in his paper 'Performance parameters and benchmarking of supercomputers', compares the strengths and weaknesses of the most commonly used benchmarks of supercomputer performance. His comparison is in relation to performance parameters characterizing the peak performance, the degradation of performance from inadequate vector length, inadequate computational intensity, and synchronization overhead.

A global performance formula, which factorizes these degradations, and allows for an a priori estimate for the overall performance, is presented in 'Performance estimates for supercomputers' by *Schönauer* and *Häfner*. To reveal the sources for these degradations, micro-measurements are presented for individual operations.

In the second group, we have papers focussing on several well-known benchmarks, such as Whetstone, Dhrystone, Linpack, Perfect, SPEC, NAS, LANL, LLL, EuroBen, SLALOM, and a compiler benchmark, in detail.

The selection starts with *Weicker*, 'A detailed look at some popular benchmarks', describing and analyzing the industry standard benchmark programs Whetstone, Dhrystone, and Linpack. He names a number of pitfalls that users have to be aware of. Other benchmarks are also mentioned, and the relative merits of small vs. large benchmarks are discussed.

Besides the standard industry benchmarks, there are many well-known scientific application benchmarks such as Perfect, SPEC, NAS, LANL and LLL described in 'Scientific benchmark characterizations' by *Berry, Cybenko* and *Larson*. They compare 69 of the programs based on data gathered by the CRAY X-MP hardware performance monitor.

Dixit presents the benchmark of the Systems Performance Evaluation Cooperative (SPEC) which represents an effort of major computer companies to create a yardstick to measure the performance of computer systems. In the paper, he discusses strengths and weaknesses of SPEC and warns users about the danger of single number performance characterization, such as SPECmark, Linpack-MFLOPS or LLL-mean values.

While the SPEC benchmark is heavily used in the workstations market, the EuroBen and the Perfect benchmarks are mainly used for high performance systems.

A problem with almost all of the supercomputers is the enormous performance range,

sometimes potential a factor of thousand or more, depending on the suitability of the code for the underlying architecture. To measure this wide range, the hierarchical EuroBen benchmark program has been developed. *Van der Steen* describes the structure of 'The benchmark of the EuroBen Group', its rationale, and the supporting activities of the EuroBen Group.

Machine characteristics such as processors, memory, I/O, influence the overall performance of a program. However, as pointed out above, performance is also greatly affected by the structure of the user's program and its data, and the ability of the compiler. For the latter, *Levine, Callahan*, and *Dongarra* present a test suite of Fortran loops for 'A comparative study of automatic vectorizing compilers'. This collection of loops tests the analysis capabilities of such compilers.

Finally, the last part contains five papers on benchmarking shared- and distributed-memory parallel computers.

An approach very similar to the one just described, but for parallelizing compilers, is discussed by *Dongarra, Furtney, Reinhardt*, and *Russel* in their paper 'Parallel Loops – A test suite for parallelizing compilers'. They developed a test suite of Parallel Loops to measure the ability of parallelizing compilers to convert code to run in parallel, and to determine how effectively parallel hardware and software work together to achieve high performance across a range of problem sizes.

Parallel processing results for a few shared-memory parallel machines are presented in *Grassl*'s paper on 'Parallel performance of applications on supercomputers'. He implemented the Perfect Benchmark suite consisting of thirteen programs from fluid dynamics, chemistry and physics modeling, engineering design and signal processing on a CRAY Y-MP. These programs are designed primarily to evaluate supercomputer performance on application programs.

Hey's contribution about 'The Genesis distributed memory benchmarks' reports on a benchmark suite for highly parallel distributed memory MIMD systems, developed in the European Esprit-2 project Genesis. The suite mainly contains synthetic code fragments from the areas FFT, partial differential equations, quantumchromodynamics, molecular dynamics, and linear algebra.

Two of these MIMD systems are compared in 'Performance of the Intel iPSC/860 and Ncube 6400 hypercubes' by *Dunigan*, with earlier hypercubes from Intel and Ncube, to better understand the basic structure of the architecture and the relative performance and capacities of the CPU, memory, communication, and I/O.

The final paper by *Nagel* and *Linn* discusses 'Benchmarking parallel programs in a multiprogramming environment: The PAR-Bench system', which enables measurements of effects introduced by parallel programs running in a multiprogramming mode. According to user-supplied parameters, the system generates synthetic benchmark programs by using the hardware performance monitor of the CRAY. These programs can be used to simulate a given site's workload, and to provide information about mutual influences of parallel programs and background load in an actual multiprogramming production environment of a shared-memory system.

We would like to thank all of the authors of this Special Issue as well as the editors of *Parallel Computing* for their contributions and valuable support to the realization of this volume.

J. Dongarra and W. Gentzsch
Guest Editors