PARKBENCH: Methodology, Relations and Results

Jack J. Dongarra[1,2]*, Tony Hey[3]** and Erich Strohmaier[1]***

[1] Computer Science Department, University of Tennessee, Knoxville, TN 37996
[2] Mathematical Science Section, Oak Ridge National Lab., Oak Ridge, TN 37831
[3] University of Southampton, Department Electronics and Computer Science, Faraday Bldg 816A-1, SO9 5NH Southampton, Southampton, UK

Abstract. The PARKBENCH (PARallel Kernels and BENCHmarks) committee has establish a comprehensive set of parallel benchmarks. The PARKBENCH committee was founded at Supercomputing'92 in Minneapolis, when a group of about 50 people interested in computer benchmarking met under the joint initiative of Tony Hey (University of Southampton, UK) and Jack Dongarra (University of Tennessee/Oak Ridge National Laboratory). Most of the key players were present, from the Universities, Laboratories and industries, representing both computer manufacturers and computer users from both sides of the Atlantic.
In this paper we describes the content of the version 2.0 of the PARKBENCH suite. This version includes MPI and PVM versions for all benchmark software. This allows performance comparison for these different message passing standards.

1 Introduction

The PARKBENCH (PARallel Kernels and BENCHmarks) committee was founded at Supercomputing'92 in Minneapolis [1]. The goals of the group were fourfold:

1. Establish a comprehensive set of parallel benchmarks that is generally accepted by both users and vendors of parallel system.
2. Provide a focus for parallel benchmark activities, and avoid unnecessary duplication of effort and proliferation of benchmarks.
3. Set standards for benchmarking methodology and result-reporting, together with a control database/repository for both the benchmarks and the results.
4. Make the benchmarks and results freely available in the public domain.

Considerable progress has been made toward achieving these goals. The PARKBENCH suite (version 2.0) currently includes part of the Genesis benchmark suite [2] [3] and NPB (NAS Parallel Benchmark) [4]. The initial focus of the PARKBENCH suite is the new generation of scalable distributed-memory

* e-mail: dongarra@cs.utk.edu
** e-mail: ajgh@ecs.soton.ac.uk
*** e-mail: strohmaier@cs.utk.edu

message-passing architectures, for which there is a notable lack of existing benchmarks. In addition, the PARKBENCH committee has developed a graphical interface, called PDS (performance database server) for gaining access to the database of stored benchmark results. This interface is in operation at netlib[4] and at Southampton[5].

2 The ParkBench Suite Version 2.0

The current release of the PARKBENCH suite, version 2.0, has the same hierarchical structure as version 1 [5]. Such a structure is not new to benchmarking; indeed, it is considered a necessary starting point for in-depth interpretations of benchmarking results.

The PARKBENCH structure is as follows:

- The section LowLevel contains single-processor benchmarks as well as simple parallel benchmarks for measuring the communication parameters and the synchronization costs on a parallel system. These benchmarks are based on codes from the Genesis benchmark suite.
- The section Kernels contains five linear algebra problems from the library ScaLAPACK and the kernels from the new NPB suite 2.0.
- The section Compact Applications contains the three compact applications from the NPB version 2.0 [6] as well as the code PSTSWM (a shallow water model) [7].

Table 1 shows the parameters measured by the LowLevel benchmarks in PARKBENCH version 2.0. All multiprocessor LowLevel codes are available as native PVM and MPI implementations. The linear algebra kernels are based on the freely available ScaLAPACK library [8]. ScaLAPACK in turn relies upon the BLAS [9] and BLACS [10] packages. Since the BLACS are implemented in PVM and MPI (as well as some proprietary message-passing interfaces), the linear algebra kernels can run on PVM or MPI just by linking to the appropriate implementation of the BLACS. Use of these packages enables highly efficient computation and communication.

In version 1 of the PARKBENCH benchmark suite, some PVM-based implementations of NPB version 1 had been used. In the new version 2 of the PARKBENCH benchmark suite, however, the new NPB version 2 implementations are included. Based on implementations of the original problems in Fortran 77 and MPI [6], these codes are implemented much more efficiently than were the PVM codes used in version 1; hence, the original PVM codes are no longer included. Since only approximately 30 MPI functions are used for the new NPB codes, a project is under way to provide an efficient interface from MPI to PVM. Currently, the shallow water model application code, PSTSWM, is available in both the PVM and MPI message-passing libraries. When this project is finished, both PVM and MPI versions of NPB will be available.

[4] http://netlib2.cs.utk.edu/performance/html/PDStop.html
[5] http://www.soton.ac.uk

Table 1. Benchmarks in Parkbench 2.0, showing parameters measured by the LowLevel benchmarks (perf. = performance, arith. = arithmetic, comms. = communication, ops. = operations)

Module	Benchmark	Measures	Parameters
LowLevel			
SINGLE PROCESSOR:			
	TICK1	Timer resolution	tick interval
	TICK2	Timer value	wall-clock check
	RINF1	Basic arith. ops.	$(r_\infty, N_{1/2})$
	POLY1	Cache bottleneck	$(\hat{r}_\infty, f_{\frac{1}{2}})$
	POLY2	Memory bottleneck	$(\hat{r}_\infty, f_{\frac{1}{2}})$
MULTIPROCESSOR:			
	COMMS1	Basic message perf.	$(r_\infty, N_{1/2})$
	COMMS2	Message exch. perf.	$(r_\infty, N_{1/2})$
	COMMS3	Saturation bandwidth	$(r_\infty, N_{1/2})$
	POLY3	Comms. bottleneck	$(\hat{r}_\infty, f_{\frac{1}{2}})$
	SYNCH1	Barrier time and rate	barr/s
Kernels			
LINALG:			
	Matrix Multiply		
	Transpose		
	Dense LU factorization		
	QR decomposition		
	Matrix tridiagonalization		
NPB:			
	EP: Embarrassingly parallel (nya)		
	IS: Integer sort (nya)		
	CG: Conjugate gradient (nya)		
	MG: Multigrid solver		
	FT: 3D FFT		
Comact Applications			
NPB:			
	LU: CFD code - SSOR solver		
	SP: CFD code - scalar pentadiag. system		
	BT: CFD code - block-tridiagonal		
	PSTSWM: Shallow water model		

3 Relations to other Standard Benchmarks

One of the major goals of the PARKBENCH committee is to avoid the unnecessary duplication of work. For this, during the last years, several cooperations were undertaken with other groups performing benchmarks. The most important ones are the Genesis group and the NAS PB group. As result, the benchmark suites of both groups are part of the PARKBENCH distribution. However, further

cooperation is certainly necessary to get a unified set of run rules. Currently, there is a wide range of different rules, whose divergency makes it difficult to compare benchmark results. At one end of this range are the run rules of the SPEC consortium, which allow no modification of the benchmark software. At the other end is NPB version 1—a "pencil and paper" benchmark that makes no restrictions to the implementation at all. These different run rules reflect the different areas of interest of the individual groups. SPEC, for instance, is intended for workstations with similar architectures. On the contrary, NPB version 1 was designed for parallel systems with quite different architectures. Since no standard for programming parallel systems existed at the time (1991) NPB was initially designed, the developers decided to specify a pencil and paper benchmark. Now, as the Message Passing Inteface (MPI) standard gains acceptance, the NPB developers are switching to MPI.

4 Results

The results of version 1 of PARKBENCH and early results of version 2 are available on the Web by using the PDS/GBIS interface. In addition to tables, customized graphs can also be obtained, which show the performance of one benchmark over different numbers of processors for different systems. Here we present results for a few selected benchmarks.

As our first example we select RINF1, which is designed to analyze basic arithmetic operations. It takes a set of 17 common Fortran DO-loops and evaluates their time of execution in terms of the two parameters, RINF and NHALF. RINF is the asymptotic performance rate in Mflops that is approached as the loop (or vector) length n becomes longer. NHALF (the half-performance length) expresses how rapidly, in terms increasing vector length, the actual performance r approaches RINF. It is defined as the vector length required to achieve a performance of one half of RINF. In Figure 1 we show the results for loop number 4, which is the calculation of a triad $A(I) = B(I) * C(I) + D(I)$ with a data access of stride 8. The influence of the cache on the performance can clearly be seen. The out of cache performance for all measured processors is very poor, which is due to the data access with a stride of a power of 2.

We now present the results for the communication benchmarks COMMS2. COMMS2, or pingpong, measures the basic communication properties of a message-passing MIMD computer. A message of variable length n is simultaneously exchanged between a master node and a slave node. In this case advantage can be taken of bidirectional links, and a greater bandwidth can be obtained than is possible with COMMS1. In Figure 2 we see the increase and final value for the transfer rate of this type of two-node communication.

As an example we show the results for IS, one of the NPB kernels. IS is one of the least parallelizable benchmarks in the NPB suite. Even for problems in class B, which represents larger and therefore better scaling problems than those in class A, we see in Figure 3, that vector mainframes still show good performance compared with massively parallel processors.

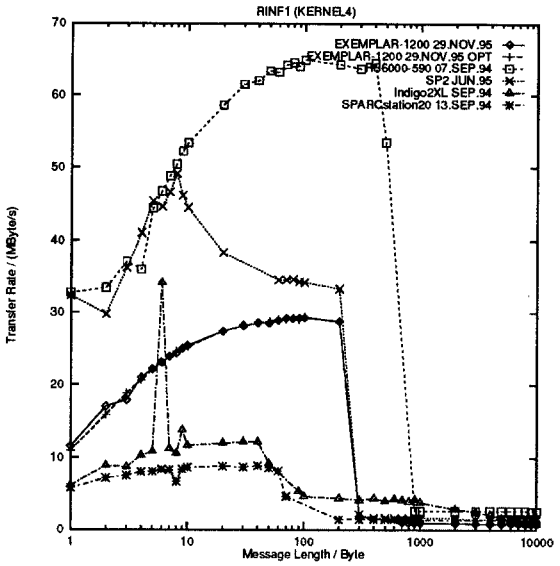

Fig. 1. Computational performance over the vector length for the LowLevel benchmark RINF1 for a triad with stride 8.

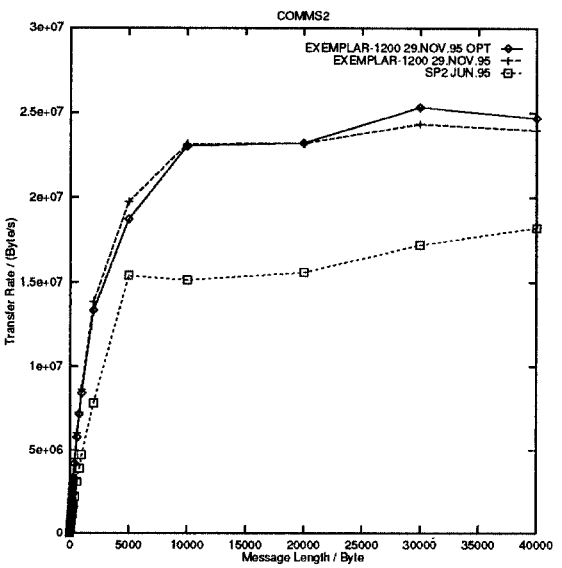

Fig. 2. The communication performance over the message length for the LowLevel benchmark COMMS2 based on PVM

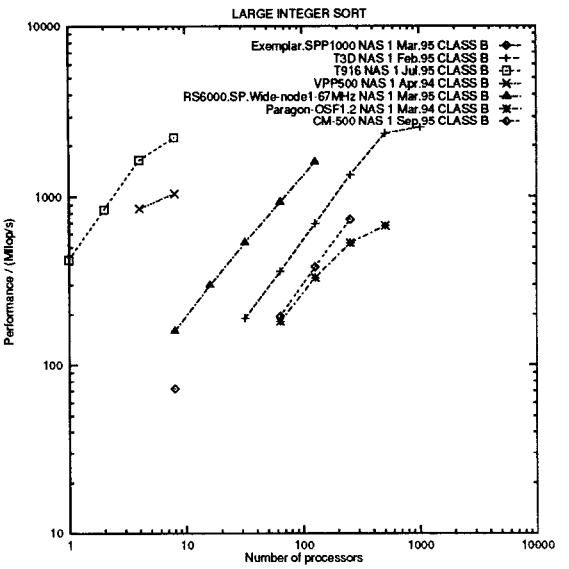

Fig. 3. The performance of the NPB kernel IS over the number of processors for the problem size class B

In Figure 4 we show the results for the larger problem size of PSTSWM[6], which is part of the Compact Applications. This application solves the nonlinear shallow water equations on a rotating sphere by using the spectral transform method. While the scaling of even the small problem size is good, the positive effect of the increased problem size on the absolute performance and on the scaling can easily be seen.

5 Conclusions

The PARKBENCH benchmark suite comprises software that ranges from low-level benchmarks measuring basic machine parameters, through important application kernels, to compact research applications. This hierarchical structure allows information derived from the simpler codes to be used in explaining the performance characteristics of the more complicated codes. Thus, the benchmark suite can be used to evaluate performance on a range of levels from simple machine parameters to full applications where effects due to memory, communication or I/O bottlenecks, and nonparallelizable sections of code may become important.

With version 2 of the PARKBENCH suite a complete set of codes implemented in PVM and MPI is available, allowing for the first time a detailed comparison of these different message-passing standards with real applications.

[6] http://www.epm.ornl.gov/chammp/pstswm/

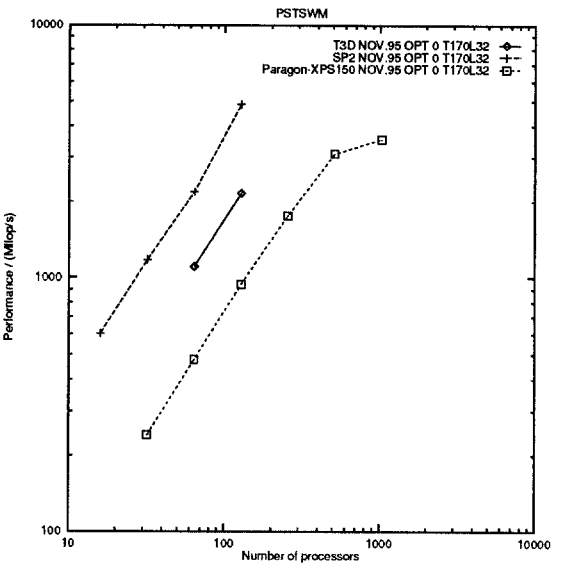

Fig. 4. The performance of the application PSTSWM over the number of processors for the larger problem size T170L32

Because of the rapid change in high-performance computing, benchmarks for these systems must be changed and adapted continuously. As an open forum, the Parkbench committee organizes biannual workshops to discuss further development of the benchmark suite.
Information is available at http://www.netlib.org/parkbench.

References

1. PARKBENCH Committee. Public International Benchmarks for Parallel Computers. Technical Report CS-93-213, Computer Science Department, University of Tennessee, Knoxville, Tennessee, November 1993. (Scientific Programming, 3(2):101-146,1994).
2. A. J. G. Hey. The Genesis Distributed-Memory Benchmarks. *Parallel Computing*, 17:1275–1283, 1991.
3. V.S. Getov, A.J.G. Hey, R.W. Hockney, and I.C. Wolton. The Genesis Benchmark Suite: Current State and Results. In *Proceedings of Workshop on Performance Evaluation of Parallel Systems - PEPS'93.* University of Warwick, Coventry, November 29-30, 1993.
4. D. Bailey, J. Barton, T. Lasinski, and H. Simon (editors). The NAS parallel benchmarks. Technical Report RNR-91-02, NASA Ames Research Center, Moffett Field, CA 94035, January 1991.
5. R.W. Hockney. The Science of Computer Benchmarking. SIAM, Philadelphia, 1996.

6. D. Bailey, T. Harris, W. Saphir, R. van der Wjingaart, A. Wo and M. Yarrow. The NAS parallel benchmarks 2.0. Technical Report NAS-95-020, NASA Ames Research Center, Moffett Field, CA 94035, December 1995.
7. I.T. Foster, B. Toonen and P.H. Worley. Performance of parallel computers for spectral atmospheric models Technical Report ORNL/TM-12986, Oak Ridge National Laboratory, Oak Ridge, TN 37831, April 1995.
8. J. Choi, J. Demmel, I. Dhillon, J. Dongarra, S. Ostrouchov, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. "ScaLAPACK: A Portable Linear Algebra Library for Distributed Memory Computers - Design Issues and Performance". Technical Report UT CS-95-283, LAPACK Working Note #95, University of Tennessee, 1995.
9. J. J. Dongarra, J. Du Croz, S. Hammarling, and I. Duff. A set of level 3 basic linear algebra subprograms. *ACM Transactions on Mathematical Software*, 16(1):1–17, 1990.
10. J. Dongarra and R. C. Whaley. "A User's Guide to the BLACS v1.0". Technical Report UT CS-95-281, LAPACK Working Note #94, University of Tennessee, 1995.