The GENESIS Benchmark Suite: Current State and Results

V.S. Getov, A.J.G. Hey, R.W. Hockney, I.C. Wolton
Department of Electronics and Computer Science
University of Southampton
Southampton SO9 5NH, U. K.
{vsg,ajgh,rwh,icw}@ecs.soton.ac.uk

Proceedings of 1st Workshop on
Performance Evaluation of Parallel Systems - PEPS'93.
Coventry, November 1993.

Abstract

The motivation behind the development of the GENESIS benchmark suite has been to assemble benchmarks for a range of important scientific and engineering problems, which require high performance computing. The suite is being used to evaluate the relative strengths and weaknesses of current distributed-memory (DM) machines and to enable predictions of performance on future parallel computer architectures. In this paper the current release of the GENESIS benchmarks is first presented. Both message-passing and subset HPF versions exist for most of the codes in addition to the standard Fortran77 versions, with the PARMACS macros being used to express the necessary extensions to Fortran77 for task creation and message passing in the distributed versions, and for conferring portability across target systems. Some recent results from selected benchmarks running on the Intel iPSC/860 and the Meiko i860 Computing Surface are finally described and related to basic system parameters.

1 Introduction

Most of the GENESIS benchmarks were initially developed within ESPRIT Project P2702 – GENESIS [4] by a number of project partners including Liverpool and Southampton Universities, PALLAS GmbH, the European Centre for Medium Range Weather Forecasting and the Universitat Politechnica de Catalunya. They have been prepared for public release by Southampton Novel Architecture Research Centre.

The GENESIS benchmarks are required to be written in standard Fortran77, with both a sequential (base) version and a parallel version being supplied. The sequential version of the benchmark defines the problem and algorithm to be used. Although based on the same algorithm, the parallel version (message-passing and/or subset HPF) of the benchmark is a different program which conforms to the corresponding programming model. A description of each benchmark and instructions on how to run it are given in individual `ReadMe` files. Current work on the GENESIS benchmark suite is concentrated on developing PVM versions of the codes. The future development of the suite is planned to take place in coordination with other parallel benchmarking initiatives in the US and Europe.

2 Parallel versions of GENESIS benchmarks

2.1 The PARMACS macros

The distributed version of the benchmarks use the PARMACS macros [2] to provide portability with respect to basic message passing and task creation. The PARMACS macros, developed at GMD and Argonne National Laboratories, allow the same Fortran code to be run without modification on a number of parallel systems. A package of macros is used to express functions such as process creation, message passing and synchronisation. These macros are invoked from the user Fortran program and passed through the standard UNIX "m4" macro expander, which replaces the macros with Fortran text appropriate to the system on which the program is to be run. Some macros expand into a set of Fortran statements whereas others expand into calls to library functions, which are supplied with the PARMACS package for each system. In the UNIX version of the suite expansion of the macros is handled by the `make`-files.

The basic programming model is based on message passing but this can easily be adapted to allow implementations on shared-memory machines as well as distributed-memory systems. Implementations currently exist for the following machines: Cray Y-MP & C-90, iPSC/860 & iPSC/2, Meiko CS (i860 or transputer) & CS-2, nCUBE 2, transputer based systems (on top of VCR & Inmos Toolset), network of Suns (using CS Tools), network of IBM RS/6000s and other UNIX workstations (on top of PVM), Parsytec, SUPRENUM, etc.

2.2 Subset HPF

The specification of High Performance Fortran (HPF) language [5] represents an extension to Fortran90 which addresses the problem of writing data parallel programs within the SPMD programming model. HPF is fully Fortran90 [6] conformant. The main extension streams of the language are data distribution features (new directives ALIGN, DISTRIBUTE, TEMPLATE, PROCESSORS, etc.), concurrent execution features (FORALL construct and several new intrinsics) and EXTRINSIC procedures. In order to allow early compiler availability subset HPF has been defined within the HPF language specification document. It is intended that subset HPF will be capable of being implemented more rapidly by vendors, which will ensure that subset HPF codes will be able to run at an early date on a variety of different machines. In general, subset HPF includes data distribution features, Fortran90 array language, dynamic storage allocation, FORALL statement and long names. It does not include dynamic data distribution, FORALL construct, EXTRINSIC procedures, generic functions and the Fortran90 free source form.

The current set of GENESIS benchmarks includes regular problems in one, two, three and four dimensions. From this point of view all benchmarks conform to the SPMD programming model and allow HPF transformations. The current release (2.2) of the suite is based on subset HPF, whilst future releases will be able to take advantage of dynamic data distribution, data parallel programming (FORALL construct) and code tuning by EXTRINSIC procedures. A subset HPF conversion strategy, which assumes that the conversion process is divided into two phases (see Fig. 1), has been adopted. During the first phase the work is concentrated on the development of Fortran90 version of the benchmarks. The main task in this phase

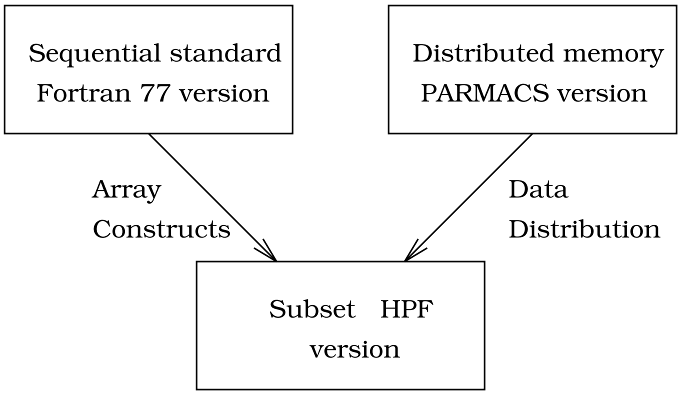

Figure 1: Subset HPF conversion strategy

is the introduction of array constructs into the codes. The second phase addresses the data distribution, resulting in the subset HPF version of benchmarks.

3 Organisation and Contents of Benchmark Suite

3.1 Currently Available Benchmarks

The latest public release (2.2) of the GENESIS benchmark suite comprises sixteen codes that vary from synthetic code fragments 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 non-parallelisable sections of code, and memory, communication or I/O bottlenecks may become important.

1. **Synthetic code fragments.** These measure basic machine parameters connected with communication, synchronisation and vectorisation.

   TICK1 Measures resolution of system clock
   Versions F-77, Subset HPF

   TICK2 Measures accuracy of system wall-clock time
   Versions F-77, Subset HPF

   RINF1 Measurement of vectorization parameters r_∞ and $n_{1/2}$
   Versions F-77, Subset HPF

   COMMS1 Pingpong benchmark
   Versions F-77, PARMACS

   COMMS2 Message exchange benchmark
   Versions F-77, PARMACS

   SYNCH1 Barrier synchronisation cost
   Versions F-77, PARMACS

2. **Application kernels.** These are larger, more complex codes that represent the computationally intensive kernels common to a variety of different applications. Analysis of the

benchmark performance is generally possible in terms of the basic machine parameters. They provide valuable insight into the likely behaviour of full applications.

TRANS1 Matrix transpose using naive algorithm
Versions F-77, PARMACS, Subset HPF

IO1 Measurement of parallel input/output transfer rate
Versions F-77, PARMACS, Subset HPF

FFT1 One-dimensional FFT
Versions F-77, PARMACS, Subset HPF

QCD2 Conjugate gradient kernel from QCD
Versions F-77, PARMACS

PDE1 Three-dimensional Poisson solver using red-black relaxation
Versions F-77, PARMACS, Subset HPF

PDE2 Two-dimensional multi-grid Poisson solver
Versions F-77, PARMACS, Subset HPF

3. **Full application codes.** These are typical of medium-sized research codes where effects due to non-parallelizable sections of code, and memory or I/O bottlenecks may become important.

QCD1 Quantum chromodynamics simulation using a Monte Carlo method
Versions F-77, PARMACS

GR1 General relativity application
Versions F-77, PARMACS

LPM1 Electronic device simulation
Versions F-77, PARMACS

MD1 Molecular dynamics code
Versions F-77, PARMACS, Subset HPF

The GENESIS benchmarks are publicly available by anonymous `ftp` from `ecs.soton.ac.uk`, directory `pub/benchmarks`.

3.2 UNIX Directory Structure

The GENESIS benchmark suite is available as a single UNIX directory structure (see Fig. 2). Apart from the benchmarks themselves, the top level directory includes the general `ReadMe` file and two auxiliary directories - `Incl` and `Lib`. The `Incl` directory contains various `make`-files, whilst the `Lib` contains precompiled and non-standard Fortran libraries. Every benchmark directory contains a `ReadMe` file and reference output files in the `StdRes` directory. The standard Fortran77 code for single-node execution is placed in the `s77` directory, the distributed memory code (Fortran77 + PARMACS extensions) is to be found in the `d77` directory and the `hpf` directory contains the subset HPF code.

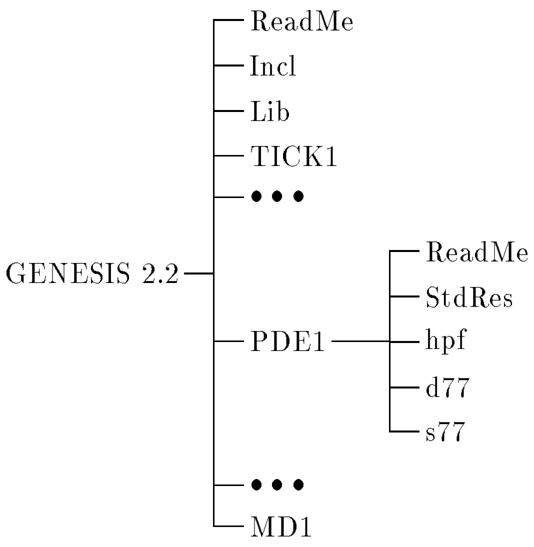

Figure 2: Structure of the GENESIS benchmark suite (Release 2.2)

4 Use of the GENESIS Benchmark Suite in Evaluating Parallel Systems

The first release of the GENESIS benchmarks was used to evaluate the following wide range of parallel computers for the GENESIS pre-study in 1989: ISIS, Cray-XMP, ETA-10, Alliant FX-8, nCUBE-1, Supernode, Meiko, iPSC/2, Amtek/Symult 2010. The pace of change in parallel computation is such that all these machines have now been superseded as have some of the conclusions drawn at the time. However, the GENESIS benchmark suite has allowed us to monitor the changes in performance of different systems as technology has advanced.

The development and application of the benchmark suite during 1990 for the evaluation of the SUPRENUM machine [1], demonstrated that supercomputer performance could be achieved with DM architectures. The best result was obtained for matrix multiplication with 870 Mflop/s from 64 SUPRENUM nodes on a large optimally shaped matrix. A highly optimized code for quantum chromodynamics achieved over 200 Mflop/s on 64 nodes. A two dimensional version of the red-black relaxation Poisson solver gave 264 Mflop/s on 64 nodes, whilst the multigrid Poisson solver ran at 160 Mflop/s.

Usually a high degree of code optimization is required to achieve such performances. The evaluation exercise confirmed a number of anticipated strengths and weaknesses of the SUPRENUM architecture. In particular, the fast access from the vector floating-point unit to vector cache memory but the long vector start-up time, the high communications bandwidth within a cluster but the long message start-up time. The poor start-up times meant that high performance could only be achieved on problems with a coarse granularity and by code using relatively few, long messages. In rewriting the codes on the basis of PARMACS during 1991 most of the specific SUPRENUM optimizations have been lost, but it has still been possible to obtain good performance from the i860-based DM computers on which measurements have so far been made.

The benchmarks have been run on two i860-based systems: the Intel iPSC/860 and the Meiko i860 Computing Surface. The Portland Group compiler was used on the iPSC/860, whilst the Greenhills compiler in association with the Pacific-Sierra vectorizer VAST-2/i860 was used on

Table 1: Best single-node performance in Mflop/s on the Meiko MK096 and the iPSC/860 for various benchmarks

Computer System	Benchmark					
	FFT1	QCD2	PDE1	PDE2	QCD1	GR1
Meiko MK096	6.2	9.7	11.9	5.0	5.7	5.7
iPSC/860	4.7	6.7	4.6	4.0	3.4	2.4

a small 8-node demonstrator Meiko machine with the MK096 board design.

Good performance on a DM benchmark can only be obtained if the basic code of the computation performs well on a single node. It was found that the combination of the VAST-2 vectorizer with the Greenhills compiler on Meiko MK096 boards gave significantly better sequential performance than the Portland compiler on the iPSC/860. Best performance was of course dependent on the problem size being large enough to allow efficient vectorization. Table 1 shows the best vectorized single node performances obtained on both machines for benchmarks where results are available to make a comparison. Generally the maximum performance per node will be less than the best single-node value. An important factor which reduces performance per node for distributed programs is the communications overhead. The success of a DM computer depends on keeping the message start-up time low and providing a high asymptotic bandwidth between processors. If this is not done, one can easily find that time of execution is completely dominated by message communication time.

4.1 1-dimensional FFT (FFT1)

The motivation for including this kernel in the GENESIS suite has been to provide a popular test of both parallelisation and vectorisation properties of parallel computer systems. The benchmark performs a classical 'in-place' derivation of the Cooley-Tukey's algorithm in one dimension. There is no doubt that multidimensional FFTs have incontestable advantages for performance improvements in parallel processing environments. The FFT1 benchmark, however, provides a hard and useful test by changing the vector length from 1 to hundreds of thousands on a single processor and also by performing a hypercube-type communications.

The purpose of this benchmark is not to measure how fast a particular machine can perform FFT, although such results using vendor's standard subroutine calls would be especially interesting. On the contrary, the purpose of FFT1 is to ensure a comparatively good machine-independence for many existing parallel systems. A relatively detailed investigation of vectorisation properties for five versions of this benchmark [3] showed that the current version of FFT1 is most suitable for this target. We consider this suitability particularly important when seeking a comparative performance evaluation across several parallel computer systems.

On the iPSC/860 the automatic vectorizer has been used in addition to the Fortran77 Portland compiler in order to achieve relatively competitive performance figures. The quoted benchmark measurements are for code based on the PARMACS macros, but in most cases the communications overheads associated with the extra PARMACS layer is relatively small. Figure 3 plots performance against number of processors on both the iPSC/860 and the Meiko for different transform lengths.

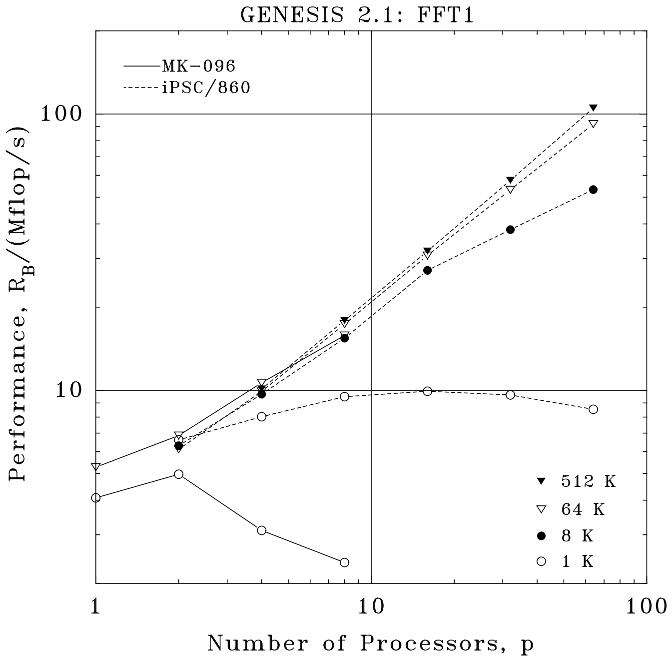

Figure 3: Benchmark performance of the distributed 1-dimensional FFT, $R_B(N;p)$, as a function of the number of processors, p, for different values of transform length, N

4.2 Conjugate Gradient Solver for QCD (QCD2)

This benchmark constitutes the kernel of SU(3) lattice gauge simulation with dynamical Kogut-Susskind fermions. The kernel consists of solving a large sparse system of linear equations by conjugate gradient iteration. This forms the core of several important algorithms for lattice gauge theory with fermions, and typically accounts for about 80% of the total execution time of a full application. The program provides for up to three of the four dimensions of the space-time lattice to be distributed. Communications are required between nearest neighbours in each processor grid dimension, a global summation and a broadcast are also required at each iteration. Virtually all of the code is vectorizable.

Figure 4 plots performance against number of processors, for different lattice sizes on both the Meiko and the iPSC/860. The benchmark performance scales well with the number of processors as long as the problem size is large enough to permit good vector performance on each processor. This is evident in Figure 4 where performance on the iPSC/860 for the largest lattice, 16×16^3, increases linearly with the number of processors within the range of measurements. For smaller problem sizes a distinct flattening of the performance curve is evident as the number of processors is increased. This flattening is due to a reduction of the vector performance on each processor and to the increased relative importance of communications overheads as the local problem size on each processor decreases. As yet it has not been possible to run the benchmark on a large Meiko machine to investigate the scaling of performance with the number of processors.

The performance of this benchmark could probably be improved by increasing the degree of overlapping of communications with computation. Further optimization in the basic single node performance could be obtained by optimizing vector shift routines used in the code. A combination of these optimizations should considerably increase the maximum performance (16×32^3 lattice size) of 325.0 Mflop/s obtained on 64 nodes of the iPSC/860.

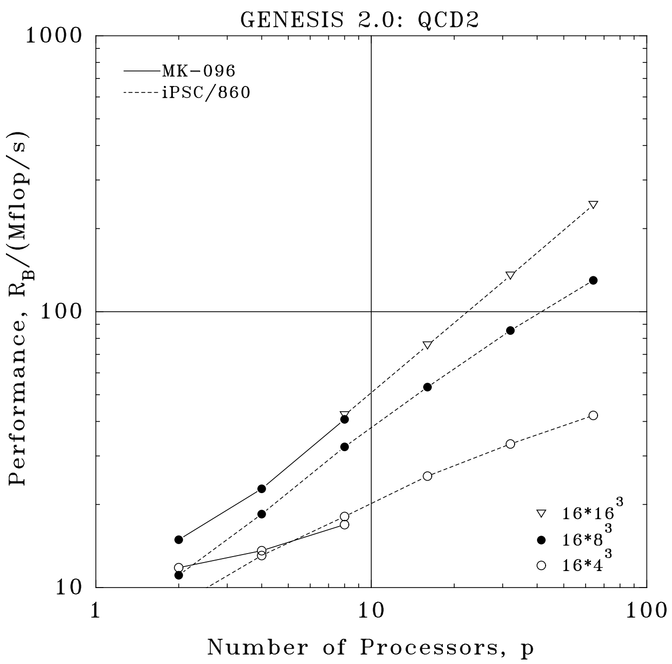

Figure 4: Performance of the Conjugate Gradient QCD Benchmark as a function of the number of processors, p, for different lattice sizes, N

5 Conclusions

The results presented are from a short initial evaluation. There is still much scope for optimization and these results should be considered as representative of the performance that can be obtained from the first naive port of a code, rather than the optimum benchmark performance on the systems. The quoted measurements are for code based on the PARMACS, which introduces an extra overhead to communications above that incurred with the native communication calls. This overhead actually shows the *price of portability*, which one should pay in order to have a portable program across different DM computers. It can be measured with the COMMS1 benchmark, and the timing formulae supplied with the benchmarks allow estimation of the deterioration in performance due to this aspect of the use of PARMACS.

These measurements are presented as an example of how the benchmarks can be used to evaluate the performance of different systems. This performance is influenced by the compiler and message-passing software as well as the machine architecture. As software is being continually upgraded the results presented may not be representative of current system performance.

Both machines were not well balanced on most of the GENESIS benchmarks as regards speed of communication versus speed of computation. This shows up in relatively poor performance speed-up for multi-node measurements when compared with the single-node results. Improved communications on the next generation of machines should do much to improve this imbalance.

Acknowledgments

Many comments, questions, results, etc. have been received since the first public release of the GENESIS distributed-memory benchmarks last year. Amongst many people who have helped, we would like to mention Mark Hill, Mark Debbage, Matthew Peters, Klaus-Dieter Oertel and

Mark Baker for their useful and detailed comments on the benchmark codes and for porting the suite to the transputer based systems running the Virtual Channel Router, the cluster of IBM RS/6000s running PVM and the Meiko Computing Surface based on transputers. Werner Krotz-Vogel has helped significantly to improve and simplify the `make`-interface to the PARMACS installation. We would also wish to thank Craig Chase who put the benchmarks on the ftp-server of Cornell University and Robert Allan for making the suite available from the info-server of Daresbury Laboratory.

References

[1] C. Addison, J. Allwright, N. Binsted, N. Bishop, B. Carpenter, P. Dalloz, D. Gee, V. Getov, A. Hey, R. Hockney, M. Lemke, J. Merlin, M. Pinches, C. Scott, I. Wolton. The GENESIS Distributed Memory Benchmarks. Part I: Methodology and General Relativity Benchmark with Results for the SUPRENUM Computer, *Concurrency: Practice and Experience*, **5**(1), 1993, pp 1-22.

[2] L. Bomans, D. Roose and R. Hempel. The Argone/GMD Macros in FORTRAN for Portable Parallel Programming and their Implementation on the Intel iPSC/2, *Parallel Computing*, **15**(1-3), 1990, pp 119-132.

[3] V. Getov. 1-Dimensional Parallel FFT Benchmark on SUPRENUM. In: D. Etiemble and J.-C. Syre (Eds.) PARLE'92, Parallel Architectures and Languages Europe. *Lecture Notes in Computer Science*, **605**, 1992, pp 163-174.

[4] A. J. G. Hey. The GENESIS Distributed-Memory Benchmarks, *Parallel Computing*, **17**(10-11), 1991, pp 1275-1283.

[5] High Performance Fortran Forum. *High Performance Fortran Language Specification, Version 1.0*, Technical report CRPC-TR-92225, Rice University, Houston, May, 1993.

[6] M. Metcalf and J. Reid, *Fortran-90 Explained*, Oxford Science Publications/OUP, Oxford and New York, 1990.