PARKBENCH:

Methodology, Relations and Results

Jack J. Dongarra
University of Tennessee, Knoxville

Tony Hey
University of Southampton

Erich Strohmaier
University of Tennessee, Knoxville

e-mail: parkbench-comments@cs.utk.edu

HPCN Europe 1996
Brussels
16.-18. April 1996

ParkBench

(PARallel Kernels and BENCHmarks)

- Founded at a special session of SC'92, convened by Dongarra & Hey

- Membership open to all - voting rights acquired by attendance at meetings

- Six monthly meetings with interim email discussion
  - Mail reflector - parkbench-comm@cs.utk.edu

- Supported by most benchmarking groups

- Supported by computer vendors - Cray Research, Intel, IBM, SGI, Meiko, Sun,...

ParkBench Objectives

- To establish a comprehensive set of parallel benchmarks that is generally accepted by both users and vendors of parallel systems.

- To provide a focus for parallel benchmark activities and avoid unnecessary duplication of effort and proliferation of benchmarks.

- To set standards for benchmarking methodology and result-reporting together with a control database/repository for both benchmarks and the results.

- To make the benchmarks and their results freely available in the public domain.

ParkBench Motivation

- Lack of distributed memory message passing benchmarks that were generally acceptable to both users and system vendors

ParkBench Outputs

- First release of Low-level and Kernel benchmark suite assembled Compromises codes from Genesis, LAPACK & NAS Based on Fortran 77 + PVM for message-passing.

- Email discussion group in use since 1993 - parkbench-comm@cs.utk.edu

- Performance Database Server in use on Xnetlib and the WWW via Mosaic provides access to benchmarks results and literature.

- Proposed applications areas and procedure for submission of further compact applications identified

ParkBench Outputs - cont'd

- ParkBench Report: Describes philosophy, methodology, current benchmark suite, procedure for submission of compact applications.

  - R.W. Hockney and M.W. Berry, "Public International Benchmarks for Parallel Computers", PARKBENCH Committee report number 1, Scientific Programming 3(2), 101-146, 1994.

- Second release of ParkBench including Low-level, Kernel and Compact Applications benchmark suite assembled. All Codes available as Fortran 77 with PVM and MPI for message passing.

Structure of Version 2

- Low-Level section for measuring basic system parameters

- Kernel section for solver routines and building blocks of programs

- Compact Application section to test performance of real world application programs

- All Benchmarks are available in Fortran77 with PVM and MPI

Directory Structure of Version 2

```
Comp_Apps/
Kernels/
Low_Level/
Makefile
NPB2.1/
Results/
Submission/
bin/
conf/
include/
lib/

Info.PVM.dependencies
README
make.local.def
pvmhostfile
runrules.draft6
```

Low Level Directory

```
Makefile
ReadMe
comms1/
comms2/
comms3/
poly1/
poly2/
poly3/
rinf1/
synch1/
tick1/
tick2/
```

comms1 Directory

```
Makefile
ReadMe
comms1.dat
src_mpi/
src_pvm/
```

Kernel Directory

```
FT/
LU_solver/
MATMUL/
MG/
Makefile
QR/
ReadMe
TRANS/
TRD/
```

Compact Applications Directory

```
BT/
LU/
Makefile
PSTSWM/
SP/
```

lib Directory

```
BLACS/
BLACS.MPI/
BLAS/
Low_Level/
MPI2PVM/
Makefile
PBLAS/
SCALAPACK/
TOOLS/
```

Low Level lib Directory

ADDLEN.f	CHECK.f
DCOPY.f	DUMMY.f
ESTOV.f	GETLEN.f
GETOPT.f	HEADER.f
INLIST.f	INSERT.f
LSTSQ.f	SATIME.f
TOVER.f	

launch.f

time.c	time_mpi.f

Makefile

README

conf Directory

```
Makefile
make.def.ALPHA
make.def.CM5
make.def.CSPP.x
make.def.Generic
make.def.HPPA
make.def.PGON.x
make.def.RS6K
make.def.SGI5
make.def.SP2MPI
make.def.SUN4
makeconf*
pvmgetarch*
```

Major Makefile Targets

```
make conf

make all
make all.pvm
make all.mpi

make Low_Level[ ,seq,mpi,pvm]

make Kernels[ ,mpi,pvm]

make Comp_Apps[ ,mpi,pvm]

make NPB[ ,mpi,pvm]

make clean
make clobber
```

Definitions in make.local.def

```
PVM_ROOT =      /src/icl/pvm/pvm3

MPICH_ROOT =    /src/icl/MPI/mpich
   or
MPI_INCDIR =    $(MPI_ROOT)/include
MPI_LIB =       -L$(MPI_ROOT)/lib/$(MPI_ARCH)/$(MPIdev) -lmpi

LIBS_TO_MAKE =  BLAS BLACS.pvm PBLAS SCALAPACK TOOLS

BLAS =          $(ParkBench_libdir)/blas_subset.a
SCALAPACK =     $(ParkBench_libdir)/scalapack_subset.a
PBLAS =         $(ParkBench_libdir)/pblas_subset.a
TOOLS =         $(ParkBench_libdir)/tools_subset.a

PVM_BLACSdir =  $(ParkBench_home)/lib/BLACS/LIB
MPI_BLACSdir =  $(ParkBench_home)/lib/BLACS.MPI/LIB

TOTMEM =        24000000
```

Low Level Benchmarks

Benchmark	Measures	Parameter	Implementation	
SINGLE PROCESSOR:				
TICK1	Timer resolution	clock tick	seq	
TICK2	Timer value	wall-clock	seq	
RINF1	Basic arith. ops.	$(r_\infty, n_{\frac{1}{2}})$	seq	
POLY1	Cache bottleneck	$(\hat{r}_\infty, f_{\frac{1}{2}})$	seq	
POLY2	Memory bottleneck	$(\hat{r}_\infty, f_{\frac{1}{2}})$	seq	
MULTIPROCESSOR:				
COMMS1	Basic message perf.	$(r_\infty, n_{\frac{1}{2}})$	MPI	PVM
COMMS2	Message exch. perf.	$(r_\infty, n_{\frac{1}{2}})$	MPI	PVM
COMMS3	Saturation bandwidth	$(r_\infty, n_{\frac{1}{2}})$	MPI	PVM
POLY3	Comms. bottleneck	$(\hat{r}_\infty, f_{\frac{1}{2}})$	MPI	PVM
SYNCH1	Barrier synch.	barr/s	MPI	PVM

Kernel Benchmarks

Benchmark	Algorithm	Implementation	
LINALG:			
MATMUL	Matrix Multiply	BLACS	
TRANS	Transpose	BLACS	
LU_solver	Dense LU factorization	BLACS	
QR	QR decomposition	BLACS	
TRD	Matrix tridiagonalization	BLACS	
NPB:			
EP	Embarrassingly parallel (nya)	MPI	MPI2PVM
IS	Integer sort (nya)	MPI	MPI2PVM
CG	Conjugate gradient (nya)	MPI	MPI2PVM
MG	Multigrid solver	MPI	MPI2PVM
FT	3D FFT	MPI	MPI2PVM

Compact Applications Benchmarks

Benchmark	Algorithm	Implementation	
NPB CFD Codes:			
LU	SSOR solver	MPI	MPI2PVM
SP	scalar pentadiag. system	MPI	MPI2PVM
BT	block-tridiagonal	MPI	MPI2PVM
PSTSWM	Shallow water model	PICL	

Low-Level Benchmarks

Low-Level benchmarks measure basic machine parameters for computation as well as communication.

Single Processor Benchmarks	
TICK1	Measurement of clock resolution
TICK2	Measurement of clock correctness
RINF1	Measurement of vectorization parameters r_∞ and $n_{1/2}$
POLY1	In-cache memory bottleneck
POLY2	Out-of-cache memory bottleneck
Two Processor Benchmarks	
POLY3	Communications-bottleneck
COMMS1	Unidirectional single message transfer ('pingpong' benchmark)
COMMS2	Bidirectional exchange of two messages ('pingping' benchmark)
Multi Processor Benchmarks	
COMMS3	Saturation Bandwidth
SYNCH1	Barrier synchronization cost

Kernel Benchmarks

Linear Algebra Kernels (ScaLAPACK/BLACS)

MATMUL	Dense matrix multiply
TRANS	Transpose
LU_solver	Dense LU factorization with partial pivoting
QR	QR Decomposition
TRD	Matrix tridiagonalization

NPB Kernels - Version 2.1

FT	3-D FFT
MG	Multigrid
EP	Embarrassingly Parallel (nya)
IS	Large Integer Sort (nya)
CG	Conjugate Gradient (nya)

Compact Applications

- NPB Version 2.1 - CFD codes
  - BT - block-tridiagonal
  - SP - scalar pentadiag. system
  - LU - CFD code - SSOR solver
  - Native MPI codes
  - PVM version by usage of MPI2PVM wrapper Library
- PSTSWM
  - Parallel Spectral Transform Shallow Water Model from ORNL
  - PVM and MPI versions available

New in Version 2

- Full Integration of Kernel and Compact Application Benchmarks
- All Benchmarks are Fortran 77 based and available with MPI and PVM
- Integration of all necessary Libraries
- New 'orthogonal' Makefile Structure
- New Set of Run Rules is available
- More efficient NPB Benchmarks included
- Maximal message sizes extended to 10MB

Run Rules - Baseline Runs

- Code modifications are not allowed

- Run the codes “as is”,
  must get the correct solution.

- Compiler or loader flags which are supported and documented by the supplier are allowed.

- Linking to optimized versions of the following libraries is allowed:
  BLACS, BLAS, LAPACK, MPI, PVM, ScaLaPack.

Run Rules - Optimized Runs

- Compiler directives and Code modifications are allowed

- Limitations of Optimizations
  - Optimized assembly modules should be disclosed
  - The calculation should be carried out in full precision.
  - Exchanged mathematical algorithm must be as robust as the baseline algorithm.
  - Using the knowledge of the solution is not permitted.
  - Code to circumvent the actual computation is not permitted.

Run Rules

- Baseline Runs
- Optimized Runs
- We reserve the right to verify the results of the benchmarks.
- Results should be sent to parkbench-comments@cs.utk.edu

Low Level: rinf1 Benchmark

- Measures Basic Arithmetic Operations
- It uses 17 common Fortran DO-loops
- Analyses their time of execution in terms of the two parameters (r_∞,$n_{\frac{1}{2}}$).
- Timing equations used:

$$t = q * \frac{(n + n_{\frac{1}{2}})}{r_\infty} \qquad (1)$$

Parameters used:

t	time of execution
q	floating-point operations per iteration
n	loop (or vector) length n
r_∞	asymptotic performance rate in Mflop/s
$n_{\frac{1}{2}}$	the half-performance length

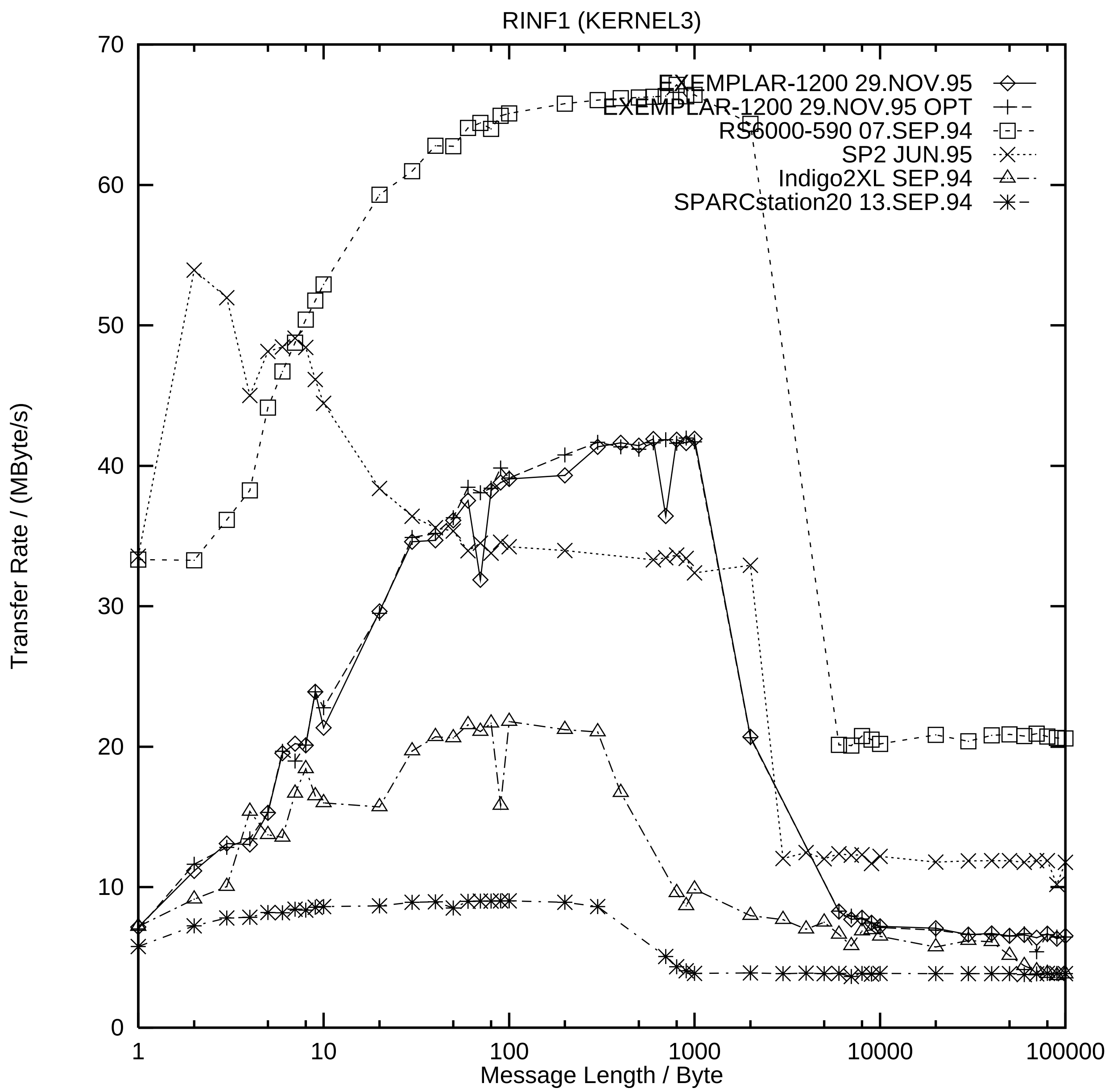
Low Level: rinf1 Benchmark

Used DO-loops:

1	Contiguous Dyads:	A(I)=B(I)*C(I)
2	DYADS, Stride=8:	A(I)=B(I)*C(I)
3	Contiguous Triads:	A(I)=B(I)*C(I)+D(I)
4	Triads, Stride=8:	A(I)=B(I)*C(I)+D(I)
5	Random Scatter/Gather:	
6	Contiguous 4-OP:	A(I)=B(I)*C(I)+D(I)*E(I)+F(I)
7	Inner Product:	S=S+B(I)*C(I)
8	1. Order Recurrence:	A(I)=B(I)*A(I-1)+D(I)
9	Charge Assignment:	A(J(I))=A(J(I))+S
10	Transposition:	B(I,J)=A(J,I)
11	Matrix Mult by Inner Product	
12	Matrix Mult by Middle Product	
13	Matrix Mult by Outer Product	
14	Dyads, Stride=128:	A(I)=B(I)*C(I)
15	Dyads, Stride=1024:	A(I)=B(I)*C(I)
16	Contiguous DAXPY:	A(I)=S*B(I)+C(I)
17	Indirect DAXPY:	A(J(I))=S*B(K(I))+C(L(I))

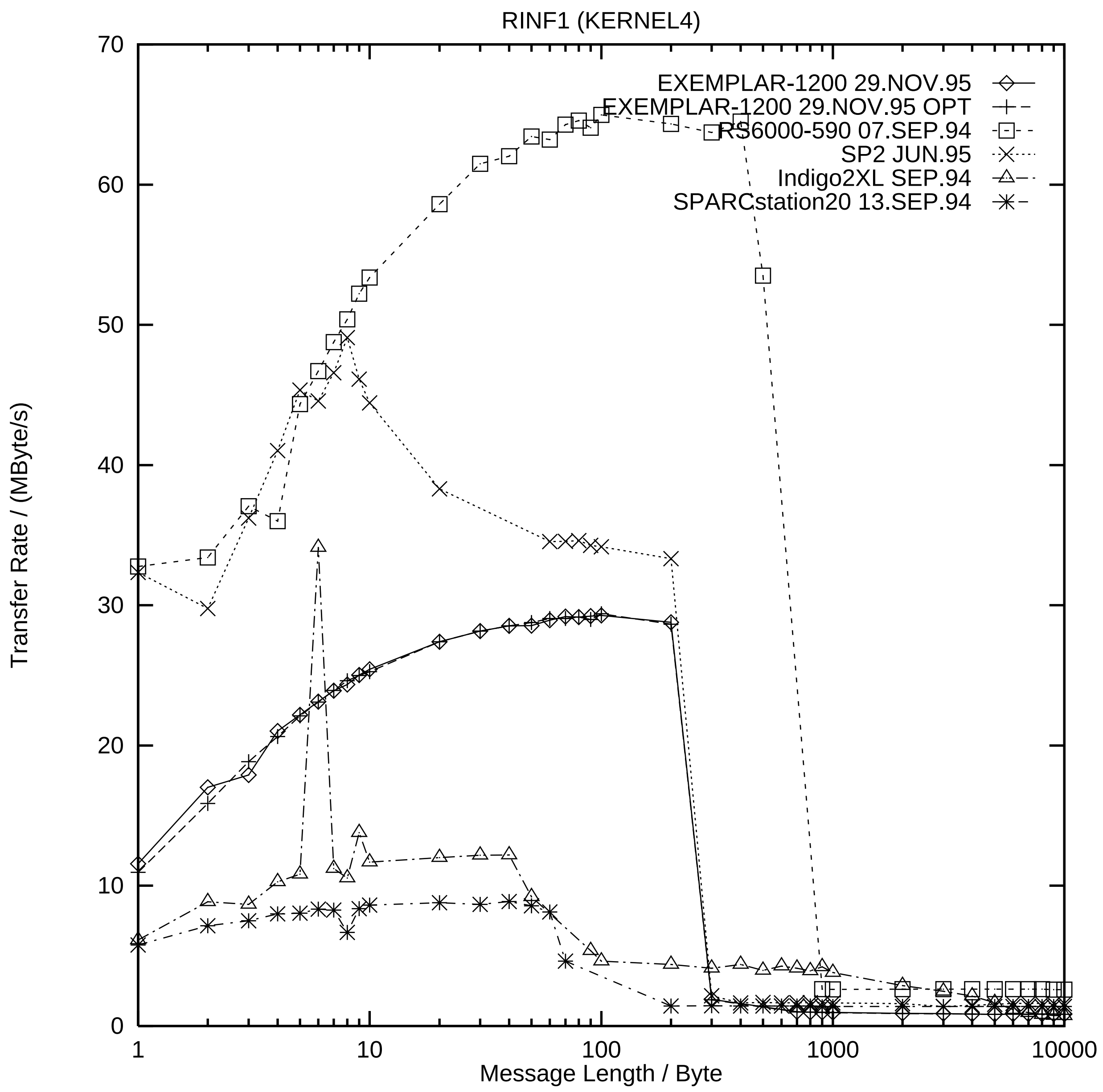
Low Level: rinf1 Benchmark

Loop 3: Contiguous Triads



Low Level: rinf1 Benchmark

Loop 4: TRIADS, STRIDE=8



Low Level: comms2 Benchmark

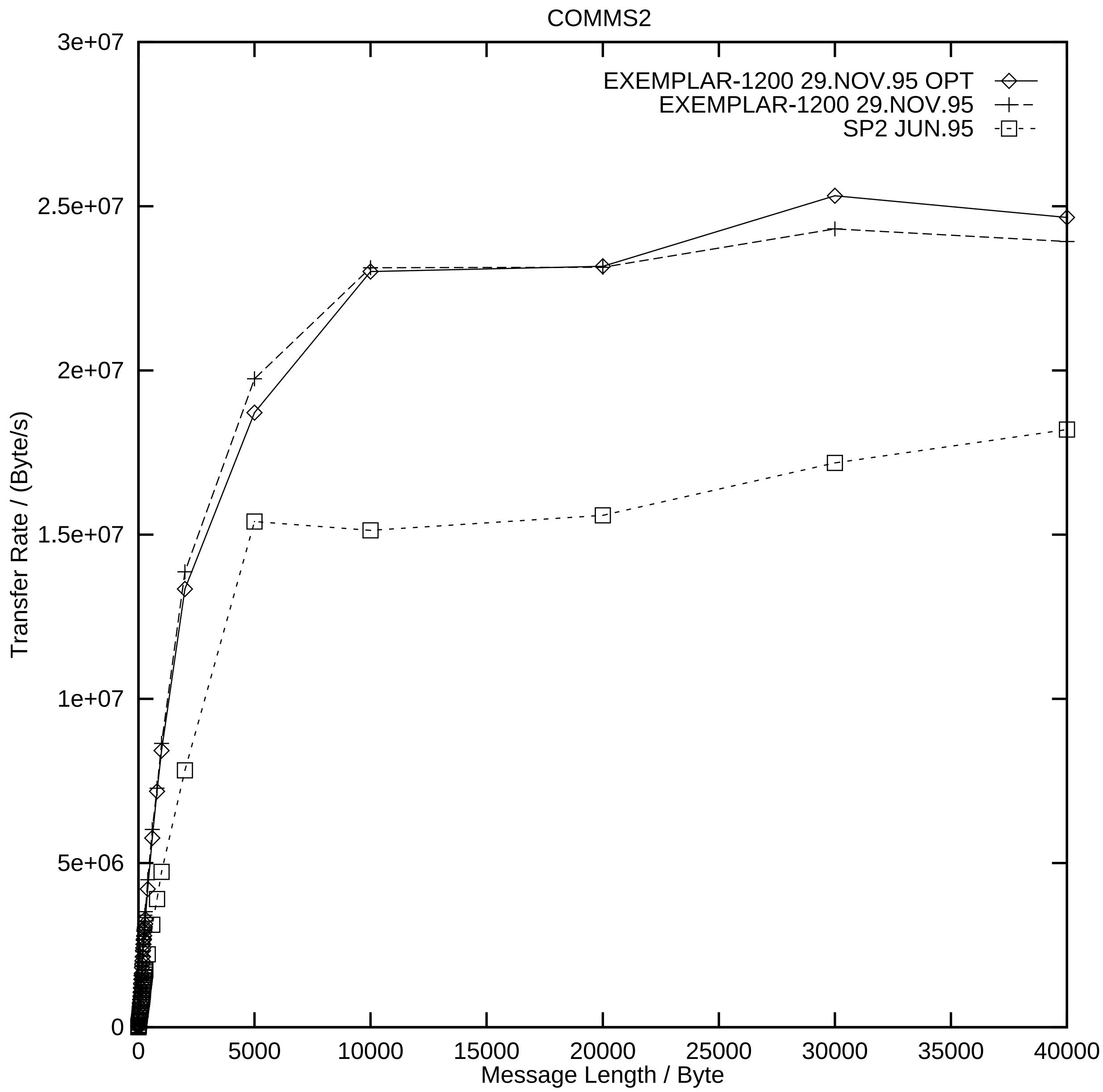
- Two Processor Communication
- Uses simultaneously messages exchange
- Analyses their time of execution in terms of the two parameters (r_∞,$n_{\frac{1}{2}}$).
- Timing equations used:

$$t = \frac{(n + n_{\frac{1}{2}})}{r_\infty} \qquad (2)$$

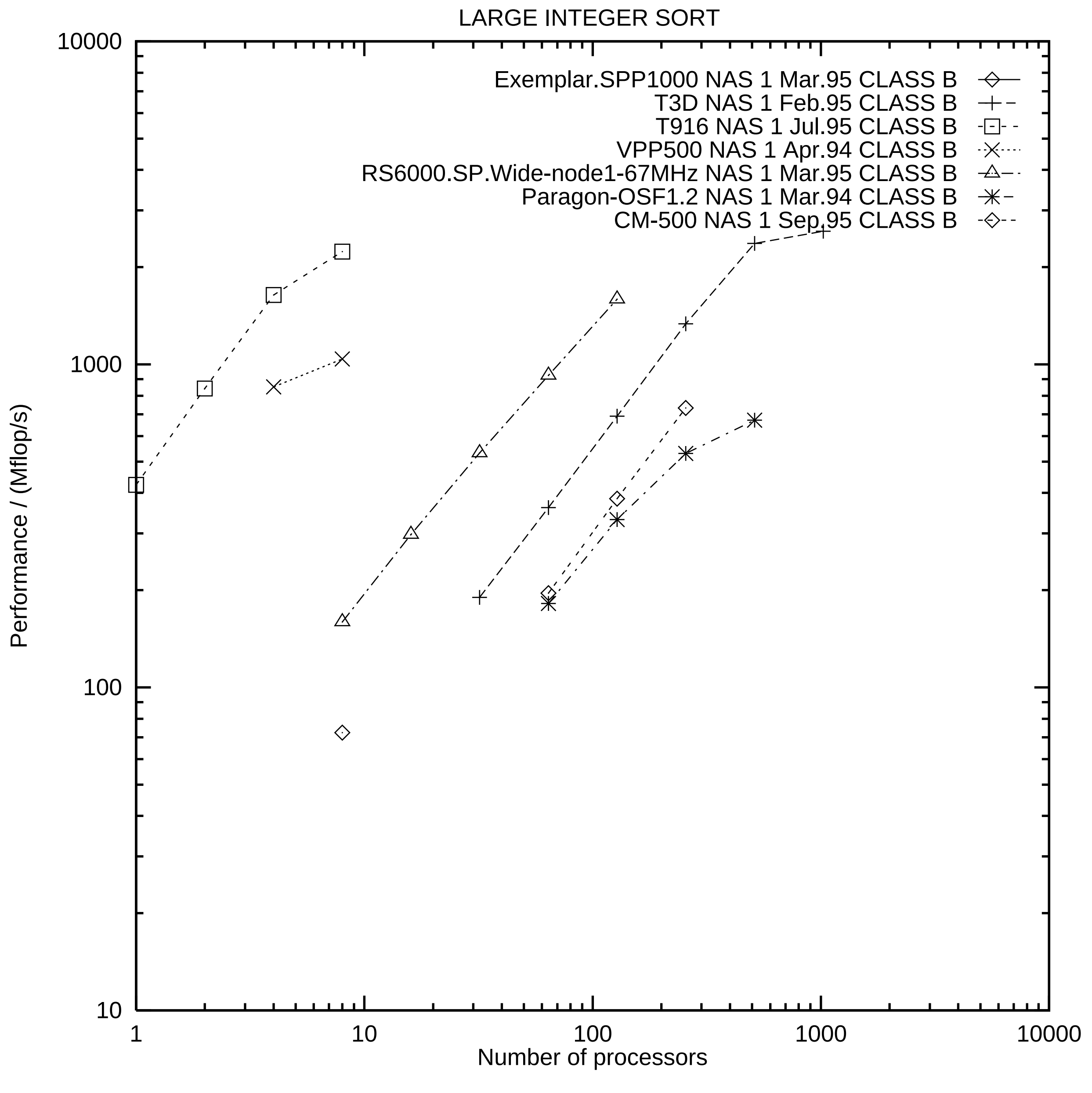
Parameters used:

t	time of execution
n	message length n
r_∞	asymptotic bandwidth in MB/s
$n_{\frac{1}{2}}$	the half-performance message length

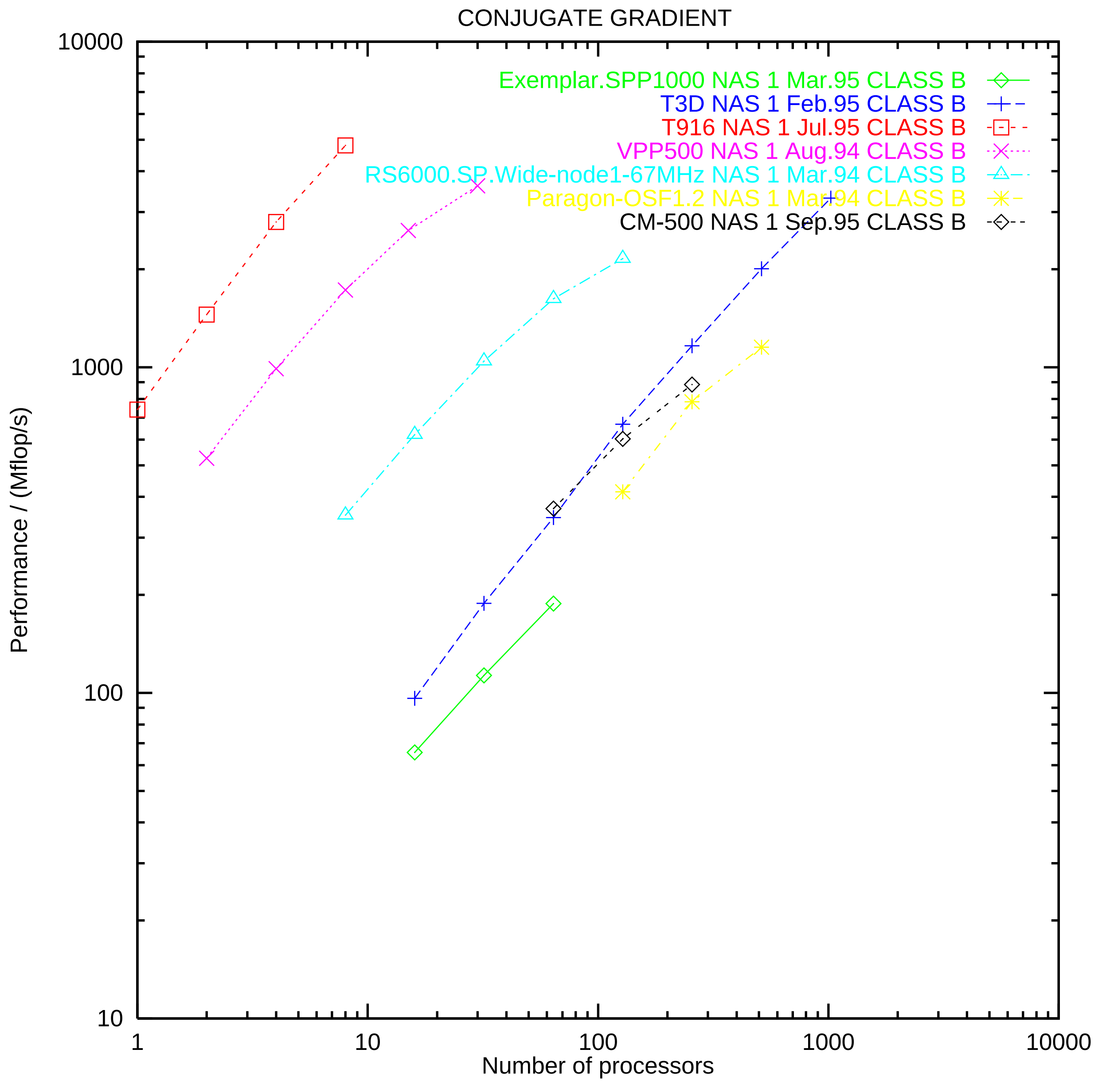
Low Level: comms2 Benchmark



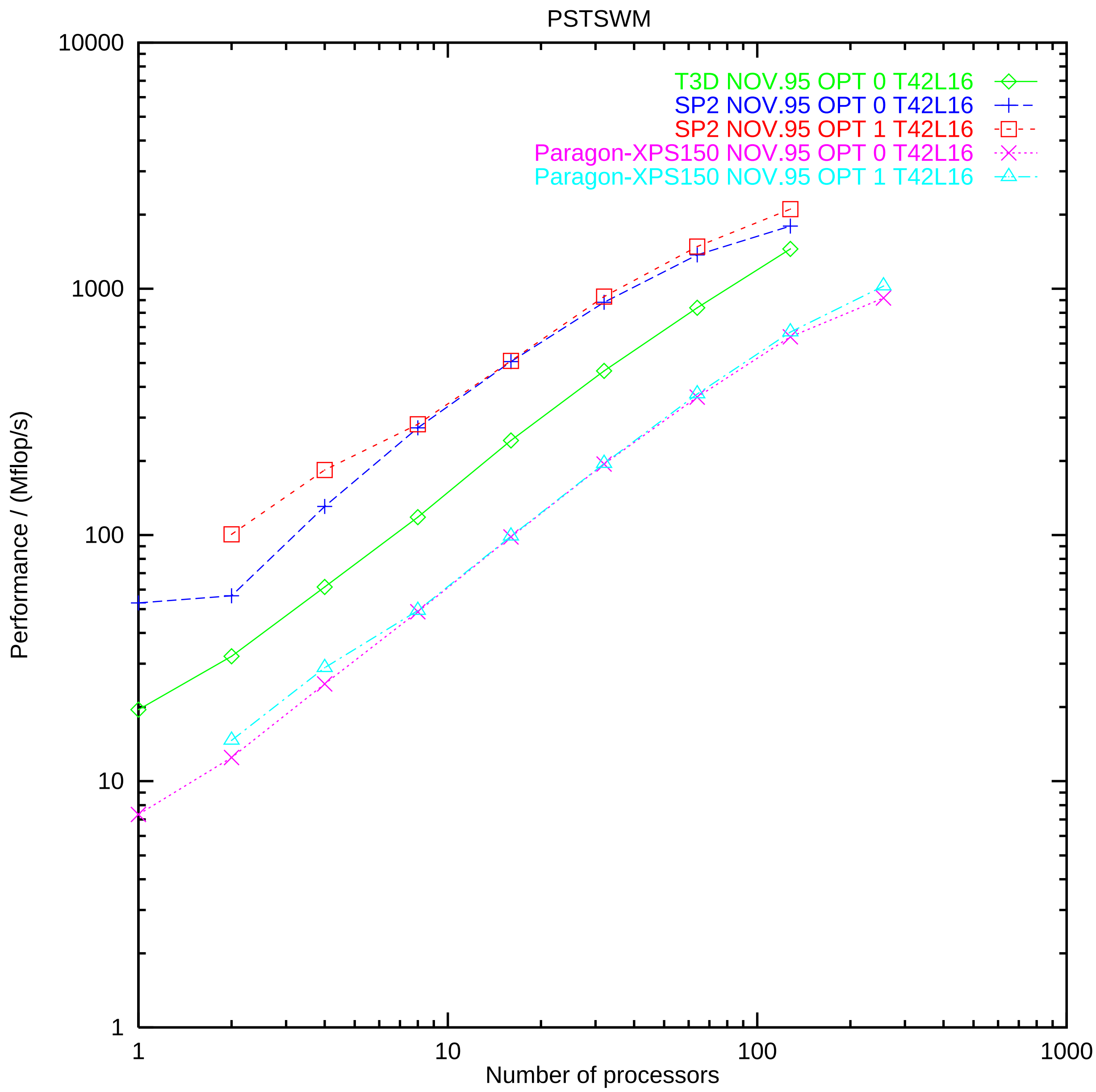
Kernel: IS - Class B



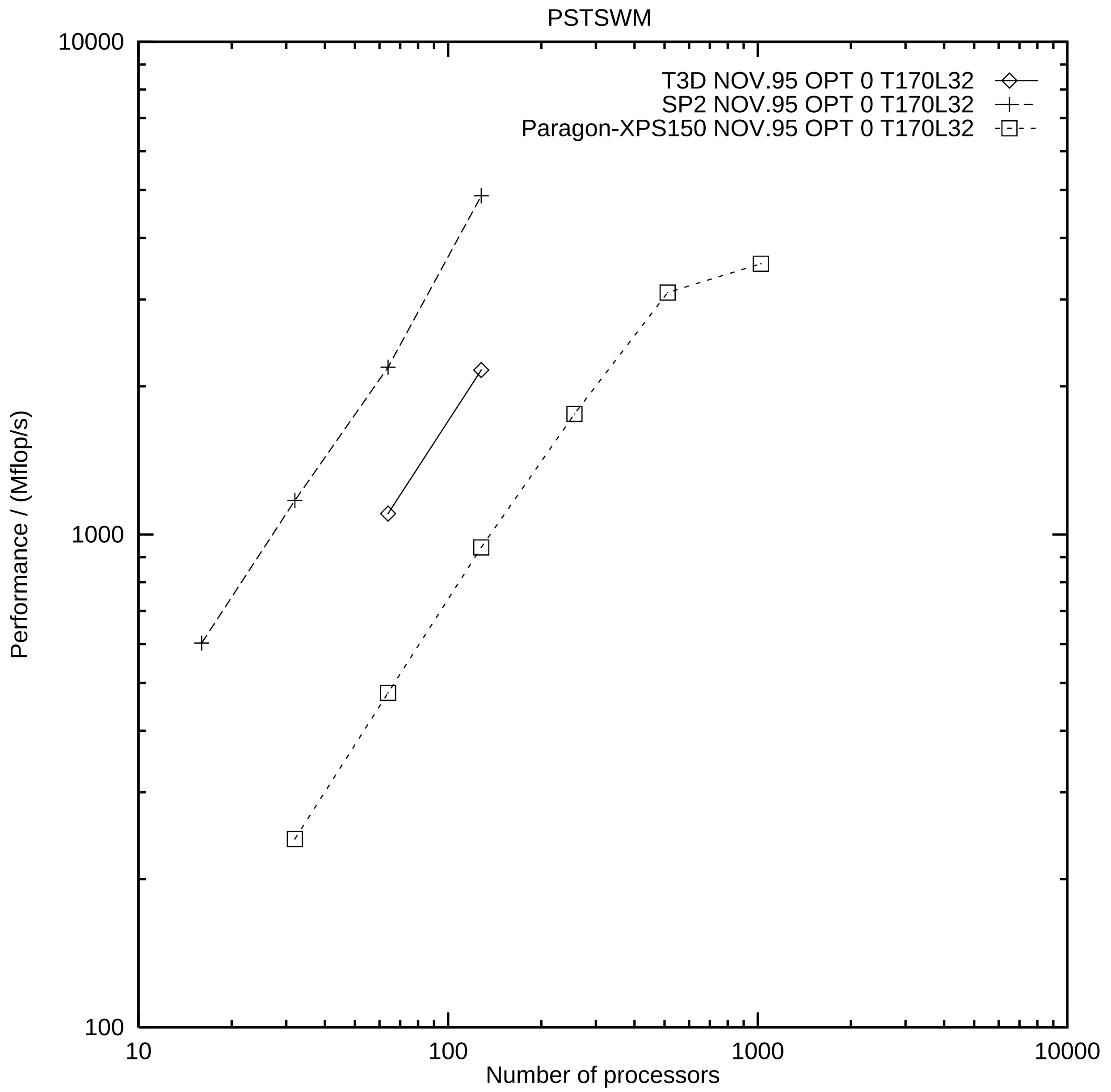
Kernel: CG - Class B



Compact Applications: PSTSWM - small



Compact Applications: PSTSWM - large



ParkBench Availability

- ParkBench Home page
  - http://www.netlib.org/parkbench/html/
- PDS
  - http://performance.netlib.org/performance/html/
    PDStop.html
- GBIS
  - http://hpcc.soton.ac.uk/RandD/gbis/ papiani-
    new-gbis-top.html
    http://www.netlib.org/parkbench/gbis/html/gbis.html
- National HPCC Software Exchange (NHSE)
  - http://www.netlib.org/nhse/home.html

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.

- 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.

- The next PARKBENCH workshop will take place in Knoxville at the 25. April.
  For Informations see:
  http://www.netlib.org/parkbench