

# Das ParkBench Projekt: PARallel Kernels and BENCHmarks

Erich Strohmaier  
Computer Science Department  
University of Tennessee, Knoxville  
erich@cs.utk.edu

## Abstract

PARKBENCH (PARallel Kernels and BENCHmarks) ist eine hierarchische Benchmark Sammlung für Supercomputer. Sie beinhaltet bewährte Codes zur Messung der Leistung von Parallel- und Vektorrechnern, insbesondere die NAS Parallel Benchmarks. Mit der Version 2 von PARKBENCH ist eine vollständige Implementierung aller verwendeten Codes in PVM und MPI erhältlich. Zudem wurden die verwendeten Problemgrößen den heutigen Systemen angepasst. PARKBENCH 2 bietet damit einen idealen Startpunkt für Leistungsvergleiche sowohl von verschiedenen Systemen als auch von den verschiedenen Message Passing Dialekten.

## 1 Hintergrund

Auf der Supercomputing 92 in Minneapolis wurde von den Mitgliedern der wichtigsten Initiativen für das Benchmarking von Parallelrechnern das sogenannte PARKBENCH (PARallel Kernels and BENCHmarks) Komitee gegründet [1]. Die wesentlichen Ziele dieser Gruppe waren zunächst:

1. Die Etablierung einer kompakten Sammlung von Benchmarks für Parallelrechner, die sowohl von den Hersteller als auch von den Benutzer dieser Systeme akzeptiert wird.
2. Die Gruppe sollte als Fokus für die Aktivitäten im Bereich der parallelen Benchmarks dienen und damit die unnötige Duplikation von Arbeit vermeiden.
3. Es sollten Standards für die Methodologie des Benchmarkings und die Publikation von Resultaten etabliert werden und es sollte eine Datenbank mit diesen Ergebnissen aufgebaut werden.
4. Die Benchmark Programme wie auch die Resultate sollten frei zugänglich gemacht werden.

Für die Version 1 von PARKBENCH wurden Codes von Euroben, Genesis und der NAS Parallel Benchmarks (NPB) verwendet [1]. Diese Version beinhaltete neben sequentiellen Fortran Programmen, Codes die mit PVM als Message Passing für skalierbare Parallelrechner mit verteiltem Speicher geschrieben waren.

Als Datenbank für Ergebnisse wurde der PDS (performance database server) zusammen mit dem graphischen Interface GBIS (graphical benchmark information

system) entwickelt. Diese sind über das WWW am netlib<sup>1</sup> und in Southampton<sup>2</sup> erreichbar. Eine gute Beschreibung dieser Version 1 von PARKBENCH ist in [2] zu finden. Die Erfahrungen mit den ersten Versionen von NPB und PARKBENCH führten schliesslich Ende 95/Anfang 96 zu den derzeit aktuellen Versionen 2 für beide Benchmark Sammlungen [3] [4].

## 2 ParkBench - Version 2

Die derzeitige Version 2 der PARKBENCH Sammlung besitzt die gleiche hierarchische Struktur wie Version 1 [2]. Sie umfasst die drei Sektionen 'Low Level', 'Kernels' und 'Compact Applications'. Nachdem sich MPI als Standard für Message Passing weitgehend etabliert hat, wurden von allen Codes zusätzlich zu den PVM Implementierungen MPI Versionen erstellt.

Die in Version 1 benutzten PVM Implementierungen der NPB wurden durch die neuen MPI Codes der Version 2 der NPB ersetzt, da diese wesentlich effizienter implementiert wurden. Um eine auf PVM ablauffähige Version dieser Codes bereitzustellen, wurde für die verwendeten MPI Aufrufe eine PVM Implementierung geschrieben. Durch Linken der ungeänderten NPB-MPI Programme zu dieser 'MPI2PVM' Bibliothek sind diese auf PVM ablauffähig. Im Folgenden werden die verschiedenen Benchmarks nur sehr kurz dargestellt.

### 2.1 Low Level

Mit den Low Level Benchmarks sollen Basisparameter der untersuchten Systeme vermessen werden. Dies umfasst sowohl die Leistung der Prozessoren und ihre Speicherzugriffsgeschwindigkeit, als auch die Kommunikationsleistung zwischen einzelnen Prozessoren. Es handelt sich durchwegs um synthetische Benchmarks, d.h. um einfache Programme die speziell zu diesem Zweck geschrieben wurden. Alle Leistungen werden für verschiedene Vektorlängen, Numerische Intensitäten und Message Längen gemessen und die zugehörigen Hockney Parameter ermittelt [2]. Zusätzlich wird die Genauigkeiten der verwendeten Timer ermittelt.

Table 1: Die Low-Level Benchmarks in Parkbench 2.0, zusammen mit den gemessenen Parametern. (perf. = performance, arith. = arithmetic, comm. = communication, ops. = operations)

Benchmark	Messung	Parameter	Implementierung	
<b>SINGLE PROCESSOR:</b>				
TICK1	Timer Auflösung	clock tick	seq	
TICK2	Timer Genauigkeit	wall-clock	seq	
RINF1	Basic arith. ops.	$(r_{\infty}, N_{1/2})$	seq	
POLY1	Cache bottleneck	$(f_{\infty}, f_{\frac{1}{2}})$	seq	
POLY2	Memory bottleneck	$(f_{\infty}, f_{\frac{1}{2}})$	seq	
<b>MULTIPROCESSOR:</b>				
COMMS1	Basic message perf.	$(r_{\infty}, N_{1/2})$	MPI	PVM
COMMS2	Message exch. perf.	$(r_{\infty}, N_{1/2})$	MPI	PVM
COMMS3	Max. Bandbreite	$(r_{\infty}, N_{1/2})$	MPI	PVM
POLY3	Comms. Engpass	$(f_{\infty}, f_{\frac{1}{2}})$	MPI	PVM
SYNCH1	Barrier synch.	barr/s	MPI	PVM

<sup>1</sup> <http://netlib2.cs.uk.edu/performance/html/PD8top.html>

<sup>2</sup> <http://www.soton.ac.uk>

Fünf der Codes in Tabelle 1 sind rein sequentiell, beinhalten aber Messungen für bis zu 17 verschiedenen DO-Loops. Drei Codes beinhalten Messungen zwischen je zwei Prozessoren, während die beiden restlichen Codes beliebig viele Prozessoren benutzen können. Alle fünf parallele Programme sind sowohl in PVM als auch in MPI implementiert.

## 2.2 Kernels

In der Sektion Kernels soll durch das Vermessen von wichtigen Teilproblemen echter Anwendungsprogramme ein erster Rückschluss auf tatsächlich erreichbare Leistungen ermöglicht werden. Im Gegensatz zu den vollständigen Anwendungsprogrammen der Sektion 'Compact Applications' sind Kernel Programme typischerweise kompakter und leichter optimierbar.

Oftmals lassen sich Implementierungen von Kernelproblemen in optimierten Bibliotheken finden. Fünf der verwendeten Kernel Benchmarks in Tabelle 2 bestehen daher aus Treibroutinen, die *SciLAPACK* Routinen [5] aus dem Bereich der Linearen Algebra aufrufen. Da die *SciLAPACK* Library aufbauend auf der Kommunikations Bibliothek *BLACS* in MPI und PVM implementiert ist (wie im Übrigen auch in einigen proprietären Message Passing Dialekten), können diese Linear Algebra Kernels einfach durch Einbinden der jeweiligen Bibliothek auf PVM oder MPI zum Laufen gebracht werden. Die Benutzung dieser Bibliotheken stellt eine effiziente Kommunikation und Berechnung sicher.

Zusätzlich zu diesen Kernels aus dem Bereich der Linearen Algebra sind alle erhältlichen NPB Kernels der Version 2.1 in dieser Sektion integriert (MG, FT). Die restlichen bisher fehlenden NPB Kernels (EP, CG, IS) werden nach ihrer Bereitstellung von NAS integriert werden.

Table 2: Die Kernel Benchmarks in Parkbench 2.0.

Benchmark	Algorithmus	Implementierung	
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 (nys)	MPI	MPI2PVM
IS	Integer sort (nys)	MPI	MPI2PVM
CG	Conjugate gradient (nys)	MPI	MPI2PVM
MG	Multigrid solver	MPI	MPI2PVM
FT	3D FFT	MPI	MPI2PVM

## 2.3 Compact Applications

In der Sektion 'Compact Applications' befinden sich vollständige Anwendungsprogramme. Mit ihnen soll das Verhalten eines Rechners unter einer realistischen Last eines numerisch intensiven Programms getestet werden. Optimierung dieser Benchmarks ist natürlich wesentlich schwieriger als im Falle der Kernel Benchmarks. Typischerweise sind die erzielten Leistungen auch deutlich geringer als für Kernel Benchmarks. In dieser Sektion (Tabelle 3) finden sich die drei CFD Programme der NPB Suite (LU, SP, BT), zusammen mit dem 'Shallow Water Model' PSTSWM [6].

Table 3: Die Compact Applications Benchmarks in Parkbench 2.0.

Benchmark	Algorithms	Implementierung	
NPB CFD Codes:			
LU	SSOR solver	MPI	MPE2PVM
SP	scalar pentadiag. system	MPI	MPE2PVM
BT	block-tridiagonal	MPI	MPE2PVM
PSTSWM	Shallow water model	MPI	PVM

### 3 Zusammenfassung und Ausblick

Die PARKBENCH Benchmark Suite beinhaltet eine Sammlung bewährter Benchmark Codes, die die Messung von grundlegenden Hardware Parametern, der Performance wichtiger Lösungsroutinen und vollständiger Anwendungs-codes erlaubt. Diese hierarchische Struktur bietet den geeigneten Ansatzpunkt, um über die eigentliche Messung hinaus eine Analyse der Leistungsdaten zu ermöglichen.

Mit der neuen Version 2 von PARKBENCH ist zum ersten Mal eine vollständige Implementierung dieser Benchmarks sowohl in MPI als auch in PVM erhältlich. Dies bietet zum ersten Mal die Möglichkeit eines echten Vergleichs der Effizienzen dieser Message Passing Implementierungen.

Aufgrund des stärker werdenden Interesses der High Performance Community an HPF stellt das Design und die Implementierung einer entsprechenden HPF Benchmark Suite einen der nächsten Schritte für das PARKBENCH Komitee dar.

Alle PARKBENCH betreffenden Informationen können im WWW über <http://www.netlib.org/parkbench> abgefragt werden.

### 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] R.W. Hockney. The Science of Computer Benchmarking. SIAM, Philadelphia, 1996.
- [3] D. Bailey, T. Harris, W. Saphir, R. van der Wijngaart, 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.
- [4] J. Dongarra, T. Hey and E. Strohmaier. The PARKBENCH Benchmark. Electronic Benchmarking Journal, to appear 1997.
- [5] 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.
- [6] I.T. Foster, B. Thoenen 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.