
**Automatic Performance Analysis for
Memory Hierarchies and Threaded
Applications on SMP Systems**

Edmond Kereku

Technische Universität München
Lehrstuhl für Rechnertechnik und Rechnerorganisation

**Automatic Performance Analysis for Memory
Hierarchies and Threaded Applications on SMP
Systems**

Edmond Kereku

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

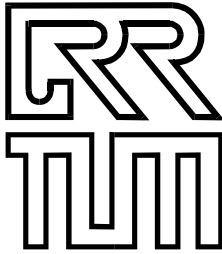
genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Martin Bichler

Prüfer der Dissertation:

1. Univ.-Prof. Dr. Hans Michael Gerndt
2. Univ.-Prof. Dr. Wolfgang E. Nagel,
Technische Universität Dresden

Die Dissertation wurde am 30.05.2006 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 13.07.2006 angenommen.



Research Report Series

Lehrstuhl für Rechnertechnik und
Rechnerorganisation (LRR-TUM)
Technische Universität München

<http://www.bode.in.tum.de/>

Editor: Prof. Dr. A. Bode

Vol. 32

**Automatic Performance Analysis for
Memory Hierarchies and Threaded
Applications on SMP Systems**

Edmond Kereku

SHAKER

V E R L A G

Aachen 2006

Bibliographic information published by Die Deutsche Bibliothek

Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the internet at <http://dnb.ddb.de>.

Zugl.: München, Techn. Univ., Diss., 2006

Copyright Shaker Verlag 2006

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the publishers.

Printed in Germany.

ISBN-10: 3-8322-5408-0

ISBN-13: 978-3-8322-5408-7

ISSN 1432-0169

Shaker Verlag GmbH • P.O. BOX 101818 • D-52018 Aachen

Phone: 0049/2407/9596-0 • Telefax: 0049/2407/9596-9

Internet: www.shaker.de • e-mail: info@shaker.de

Abstract

SMP systems rank among the most commonly used parallel computer architectures. As powerful single machines or as nodes of clusters and supercomputers, SMPs provide the computing power needed by a wide range of HPC applications. Besides concurrently using the multiple resources offered by SMPs, a primary requirement for such applications is the efficient use of resources. Thus, software developers spend a considerable effort on finding the performance bottlenecks and on optimizing their programs.

One of the common sources of inefficiency in parallel programs is the poor utilization of the memory architectures. Due to the high complexity of parallel computers, finding such problems in the application is a difficult task even for experienced developers. This thesis addresses this problem by introducing an approach for the automatic detection of memory hierarchy-related performance bottlenecks of parallel programs running on SMP systems.

The approach includes new concepts for online monitoring of multithreaded applications. A highly configurable monitoring architecture is developed which detects cache-related performance problems for selected code regions and data structures of the program by using the performance counters available in most microprocessors. A well defined interface, named MRI, is used by the performance tools to configure the monitoring architecture and to retrieve the collected performance data in the form of profiles, traces, and memory access histograms.

The main research topic is the automatic analysis of the memory access behavior of multithreaded applications based on the performance data delivered by the monitoring architecture. The collected performance data is captured in an object-oriented data model and the performance bottlenecks are defined in terms of performance properties using the APART Specification Language (ASL).

Several strategies are specified which perform an online search for performance properties on the code regions and the data structures of the program. The strategies are built using so called strategy bricks. A strategy brick specifies an algorithm which may refine the search either on code regions and data structures or on the performance problem itself. The automatic analysis concepts are implemented in an automatic performance tool called AMEBA.

Kurzfassung

Von großer Bedeutung für die effiziente Nutzung heutiger Parallelrechner ist neben der Parallelisierung der Anwendungen die Optimierung des Speicherzugriffsverhaltens. Im Rahmen dieser Doktorarbeit wurden neue Methoden zur automatischen Analyse der Speicherhierarchienutzung entwickelt. Der Focus liegt hierbei auf der Optimierung von OpenMP-Anwendungen auf SMP-Systemen, die als einzelne Parallelrechner, aber auch als Komponenten hochparalleler Systeme eingesetzt werden.

Die Analyseumgebung basiert auf einer Hardware zur adressbezogenen Erfassung von Ereignissen in der Speicherhierarchie, z.B., Cache-Fehler und -Treffer für die einzelnen Cache-Stufen. Diese Hardware-Zähler, die u.a. auf Itanium-Prozessoren verfügbar sind, werden durch ein weitgehend konfigurierbares Monitorsystem kontrolliert. Die Konfiguration erfolgt über das Monitor Request Interface (MRI).

Die eigentliche Analyse des Zugriffsverhaltens erfolgt automatisch anhand einer vordefinierten Menge von Leistungseigenschaften, die in der APART Spezifikationsprache (ASL) vorgegeben werden. Die Analysekomponente verwendet anwendungsspezifische Suchstrategien um Leistungsengpässe automatisch zu erkennen.

Acknowledgments

During my research and the writing of this thesis I received the help and support of many people, and I would like to take this opportunity to thank them. This thesis was written at the Lehrstuhl für Rechnerorganisation und Rechnerorganisation der TU-München and I would like to thank the head of the chair, Prof. Dr. Arndt Bode, and all my colleagues at LRR for providing an excellent work and research environment. I cannot be grateful enough to my adviser, Prof. Dr. Michael Gerndt for all the support, ideas, and guidance he provided throughout my research. I would especially like to thank him for all the time spent reviewing my thesis and polishing my English. Many thanks to Prof. Dr. Wolfgang Nagel for serving as second referee and to Prof. Dr. Martin Bichler, the head of my thesis commission.

Many ideas relating to this research were influenced from my work on the German EP-Cache project and from the APART group. I would like to express my gratitude to all the people who worked in the EP-Cache project and to all the APART members. I had the pleasure of working with Dr. Josef Weidendorfer whom I would like to thank for the helpful discussions and the exchange of ideas. I would particularly like to thank Thorsten Röder for implementing most of the PAPI-wrapper code. And thanks to Stefan Lukowitz for writing some of the DS-mapping code.

My experiments were performed on the Infiniband Cluster at LRR and on the SGI Altix at LRZ-München. Everything went smoothly thanks to Tobias Klug, the Infiniband Cluster administrator, and thanks to Iris Christadler and Bernhard Aichinger at LRZ.

I want to thank Jürgen Jeitner and Alex Stamatakis who shared with me the ups and downs of being PhD students.

Many thanks to Audrey Nemeth and Alex Stamatakis for proof-reading my thesis and correcting many of its English errors.

I thank my parents for their love and encouragement during all these years of studying. And I am deeply grateful to Sonia Mazuel for all the love, patience, and support especially during the writing of this thesis.

Edmond Kereku
Munich, Germany
May 2006

Contents

1	Introduction	1
1.1	The Memory Wall	1
1.2	Multiprocessor systems	3
1.2.1	SMP systems	4
1.3	Performance analysis	5
1.4	Contribution of this work	6
1.5	Organization of this thesis	8
2	Overview of Performance Analysis	9
2.1	Performance analysis	10
2.1.1	Collecting performance data	10
2.1.2	Data transformation and visualization	14
2.2	Automatic performance analysis	16
2.2.1	Why automate the performance analysis process?	17
2.2.2	What can be automated and how?	17
3	Related Work on Automatic Analysis and our Approach	19
3.1	Paradyn	19
3.2	KOJAK	20
3.3	Other automatic performance analysis approaches	21
3.4	Which are the limitations of existing approaches?	22
3.5	Our approach	23
4	Novel Monitoring Environment	25
4.1	Performance data sources of CMM	27
4.1.1	HWMon	28
4.1.2	Simulator	29
4.1.3	Hardware counters	31
4.2	The ePAPI interface	32
4.3	Instrumentation of code regions	34
4.3.1	The <i>f90inst</i> instrumenter	35
4.3.2	The Standard Intermediate Program Representation (SIR)	37
4.4	Monitoring of data structures	38
4.4.1	Mapping the data structures	39
4.5	The monitoring library	41
4.5.1	Lightweight application interface	42
4.6	The Monitoring Request Interface (MRI)	43

4.6.1	Terms and definitions	44
4.6.2	Specification and management of monitoring requests	46
4.6.3	Results delivery interface	50
4.6.4	Application control	52
4.6.5	Publication interface	52
4.7	Implementation of the CMM	53
4.7.1	Monitoring multithreaded applications	54
4.8	Summary	57
5	Formalizing the Performance Behavior	59
5.1	The performance data model	60
5.2	The APART Specification Language (ASL)	62
5.3	The performance-related data model of the CMM	66
5.3.1	Support for Itanium-related events	68
5.4	Cache-related properties in serial regions	69
5.5	Cache-related properties of threaded regions	70
5.5.1	PropPerThread	71
5.5.2	PropOnAllThreads	72
5.5.3	PropInSomeThreads	74
5.5.4	MeanMissRatePar	75
5.5.5	UnbalancedDMissRateInThreads	76
5.6	ASL properties for Itanium-based ccNUMA architectures	77
5.7	Speedup and efficiency on parallel regions	80
5.8	Organization of properties	82
5.9	Summary	84
6	Novel Online Analysis Strategies	85
6.1	Specification of application phases	86
6.1.1	Classification of phases	87
6.2	Instrumentation of phases in the application	88
6.3	Online evaluation of ASL properties	89
6.3.1	Using the performance data model to evaluate properties	92
6.4	Multi-step analysis	93
6.5	Search strategies	95
6.5.1	SingleStepBrick	95
6.5.2	The refinement strategies	98
6.5.3	CodeRegionRefineBrick	101
6.5.4	PerformanceProblemRefineBrick	103
6.5.5	CallRefineBrick	105
6.5.6	CALLRefineOnlyPARBrick	107
6.6	Creation of new strategies	107
6.7	Summary	107
7	Evaluation	109
7.1	The evaluation environment and the evaluation applications	109
7.1.1	SPEC Benchmark: SWIM	110
7.1.2	NAS Parallel Benchmark: LU	110

7.1.3	NAS Parallel Benchmark: FT	110
7.1.4	Parallel Gauss: PGAUSS	110
7.2	Evaluation of the CMM	111
7.2.1	Instrumentation and monitoring overhead	111
7.2.2	Perturbation of performance results	113
7.3	Evaluation of the automatic approach with AMEBA	115
7.3.1	AMEBA	116
7.3.2	Application analysis with AMEBA	117
7.4	Evaluation of search strategies and ASL properties	120
7.4.1	Evaluation of thread-related strategies and properties on the SGI Altix124	127
7.4.2	Remote memory accesses on ccNUMAs	127
7.5	Summary	128
8	Summary and Future Work	131
8.1	Future work	133
8.2	Concluding remarks	135
	Bibliography	137