



## Capturing Parallel Performance Dynamics

Zoltán Péter Szebenyi



Forschungszentrum Jülich GmbH  
Institute for Advanced Simulation (IAS)  
Jülich Supercomputing Centre (JSC)

# Capturing Parallel Performance Dynamics

Zoltán Péter Szebenyi

Schriften des Forschungszentrums Jülich  
IAS Series

Volume 12

---

ISSN 1868-8489

ISBN 978-3-89336-798-6

# Contents

<b>Contents</b>	<b>xi</b>
<b>List of Figures</b>	<b>xv</b>
<b>List of Tables</b>	<b>xxi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Supercomputers . . . . .	1
1.2 Parallel programming . . . . .	2
1.3 Serial performance tools . . . . .	3
1.4 Parallel performance measurement and analysis . . . . .	4
1.4.1 MPI measurements and metrics . . . . .	5
1.4.2 Profiling vs. tracing . . . . .	5
1.4.3 Scalasca toolset . . . . .	6
1.5 Motivation: Sweep3D . . . . .	9
<b>2 Experimental Setup</b>	<b>17</b>
2.1 Introduction . . . . .	17
2.2 Experiment configuration . . . . .	18
2.2.1 Initial measurements . . . . .	19
<b>3 Performance Dynamics</b>	<b>27</b>
3.1 Analysis methods . . . . .	30
3.2 Temporal patterns . . . . .	33
3.3 Case study: PEPC . . . . .	33
3.3.1 Experimental results . . . . .	35
3.3.2 Conclusion . . . . .	42
3.4 Summary . . . . .	44

## CONTENTS

---

<b>4</b>	<b>Compression of Time-series Profiles</b>	<b>47</b>
4.1	Compression algorithm . . . . .	48
4.1.1	Clustering . . . . .	48
4.1.2	Distance function . . . . .	49
4.1.3	Emphasizing the baseline . . . . .	50
4.1.4	Call-tree equivalence . . . . .	51
4.1.5	Reconstructing aggregate profiles . . . . .	52
4.2	Evaluation . . . . .	54
4.2.1	Off-line version . . . . .	55
4.2.2	Quality assessment . . . . .	55
4.2.3	Processing time . . . . .	61
4.2.4	Memory requirements . . . . .	62
4.2.5	Comparison with PEPC . . . . .	63
4.3	Visual evaluation . . . . .	63
4.4	Detailed Example: PEPC . . . . .	66
4.5	Summary . . . . .	68
<b>5</b>	<b>Combining Sampling and PMPI Event Profiling</b>	<b>71</b>
5.1	Hybrid call-path profiling . . . . .	73
5.1.1	Fast call-path unwinding . . . . .	74
5.1.2	Hybrid sampling methodology . . . . .	77
5.1.3	Implementation challenges . . . . .	79
5.2	Experimental evaluation . . . . .	81
5.2.1	Direct instrumentation . . . . .	81
5.2.2	Hybrid sampling . . . . .	82
5.3	C++ application example: DROPS . . . . .	86
5.3.1	Execution time metric . . . . .	86
5.3.2	MPI collective communication time metric . . . . .	88
5.3.3	Deciding the question . . . . .	89
5.3.4	Conclusion . . . . .	92
5.4	Compression of hybrid time-series profiles . . . . .	92
5.4.1	Exact measurement of iteration and marked region times . . . . .	92
5.4.2	Modified call-tree comparison methods . . . . .	93
5.4.3	Evaluation . . . . .	94
5.5	Summary . . . . .	98

<b>6</b>	<b>Evaluation of On-line Compression</b>	<b>101</b>
6.1	Evaluation . . . . .	101
6.1.1	Error rates for entire iterations . . . . .	102
6.1.2	Error rates for individual call paths . . . . .	104
6.1.3	Quantized call-path error rates . . . . .	104
6.1.4	Example: 128.GAPgeofem . . . . .	108
6.1.5	Example: 129.tera_tf . . . . .	108
6.1.6	Example: 132.zeusmp2 . . . . .	111
6.1.7	Example: 143.dleslie . . . . .	111
6.1.8	Example: PEPC . . . . .	116
6.1.9	Example: DROPS . . . . .	119
6.1.10	Overview . . . . .	119
6.2	Summary . . . . .	124
<b>7</b>	<b>Related Work</b>	<b>125</b>
7.1	Performance dynamics . . . . .	125
7.2	Compression of time-series profiles . . . . .	126
7.3	Combining sampling and direct instrumentation . . . . .	126
<b>8</b>	<b>Summary and Outlook</b>	<b>129</b>
8.1	Future work . . . . .	132
	<b>Appendices</b>	<b>135</b>
<b>A</b>	<b>Detailed Analysis of the Application Suite</b>	<b>137</b>
A.1	Initial measurements . . . . .	142
A.2	Basic Scalasca measurements . . . . .	144
<b>B</b>	<b>Overview of the Application Suite's Time-dependent Behavior</b>	<b>151</b>
B.1	121.pop2 . . . . .	152
B.2	125.RAxML . . . . .	154
B.3	125.RAxML zoom . . . . .	156
B.4	126.lammps . . . . .	158
B.5	128.GAPgeofem . . . . .	160
B.6	129.tera_tf . . . . .	162
B.7	132.zeusmp2 . . . . .	164

## CONTENTS

---

B.8	137.lu . . . . .	166
B.9	142.dmilc . . . . .	168
B.10	143.dleslie . . . . .	170
B.11	143.dleslie zoom . . . . .	172
B.12	145.lGemsFDTD . . . . .	174
B.13	147.l2wrf2 . . . . .	176
B.14	DROPS . . . . .	178
B.15	PEPC . . . . .	180
<b>References</b>		<b>183</b>

Runtime call-path profiling is a conventional, well-known method used for collecting summary statistics of a parallel application's execution such as the time spent in different call paths of the code. However, these kinds of measurements give the user only a summary overview of the entire execution, without regard to changes in performance behavior over time. As present day scientific applications tend to be run for extended periods of time, understanding the patterns and trends in the performance data along the time axis becomes crucial.

As shown by our analysis of a representative set of scientific codes, profiling every iteration separately provides a wealth of new data that often leads to invaluable new insights. However, with the introduction of the time dimension, memory usage and file sizes grow considerably. To counter this problem, a low-overhead on-line compression algorithm was developed that exploits similarities between different iterations.

While standard, direct instrumentation, which is assumed by the initial version of the compression algorithm, results in fairly low overhead with many scientific codes, in some cases the high frequency of events makes such measurements impractical. Therefore, a hybrid solution was developed that seamlessly integrates sampling and direct instrumentation techniques in a single unified measurement, using direct instrumentation for message passing constructs, while sampling the rest of the code. Finally, the compression algorithm was adapted to the hybrid profiling approach, avoiding the overhead of pure direct instrumentation.

Evaluation of the above methodologies shows that our similarity-based compression algorithm provides a very good approximation of the original data with very little measurement dilation, while the hybrid combination of sampling and direct instrumentation fulfills its purpose by showing the expected reduction of measurement dilation in cases unsuitable for direct instrumentation.

This publication was written at the Jülich Supercomputing Centre (JSC) which is an integral part of the Institute for Advanced Simulation (IAS). The IAS combines the Jülich simulation sciences and the supercomputer facility in one organizational unit. It includes those parts of the scientific institutes at Forschungszentrum Jülich which use simulation on supercomputers as their main research methodology.