

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board

David Hutchison

*Lancaster University, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Friedemann Mattern

*ETH Zurich, Switzerland*

John C. Mitchell

*Stanford University, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

Oscar Nierstrasz

*University of Bern, Switzerland*

C. Pandu Rangan

*Indian Institute of Technology, Madras, India*

Bernhard Steffen

*University of Dortmund, Germany*

Madhu Sudan

*Massachusetts Institute of Technology, MA, USA*

Demetri Terzopoulos

*University of California, Los Angeles, CA, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Moshe Y. Vardi

*Rice University, Houston, TX, USA*

Gerhard Weikum

*Max-Planck Institute of Computer Science, Saarbruecken, Germany*

Radu Prodan Thomas Fahringer

# Grid Computing

Experiment Management, Tool Integration,  
and Scientific Workflows



Springer

## Authors

Radu Prodan

Thomas Fahringer

University of Innsbruck, Institute for Computer Science

Technikerstr. 21a, 6020 Innsbruck, Austria

E-mail: {radu,thomas.fahringer}@dps.uibk.ac.at

Library of Congress Control Number: 2006939015

CR Subject Classification (1998): C.2, D.4, F.3, H.4, H.3, C.4, I.2.8

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743

ISBN-10 3-540-69261-4 Springer Berlin Heidelberg New York

ISBN-13 978-3-540-69261-4 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

[springer.com](http://springer.com)

© Springer-Verlag Berlin Heidelberg 2007

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Markus Richter, Heidelberg

Printed on acid-free paper SPIN: 11965800 06/3142 5 4 3 2 1 0

*To our families*

---

## Preface

In the last decade, the interest in computational Grids has increasingly grown in the scientific community as a means of enabling the application developers to aggregate resources scattered around the globe for solving large-scale scientific problems. As applications get larger, more complex and dynamic, the use of software tools becomes vital for tuning application parameters, identifying performance leaks, or detecting program defects. Extensive efforts within academia and industry over the last decade resulted in a large collection of tools for practical application engineering. Available tools of broad interest include program source and structure browsers, editors, static program analysers, performance predictors, optimising compilers, schedulers, execution control and monitoring environments, sequential and parallel debuggers (providing deadlock detection and deterministic replay mechanisms), check-pointers, data and execution visualisers, performance analysers, or various program tracers.

Despite all these extensive efforts, building applications that can effectively utilise the Grid still remains an art due to the lack of appropriate high-level tools to support the developers. In this monograph, we address four critical software development aspects for the engineering and execution of applications on parallel and Grid architectures.

First of all, existing available performance analysis tools target single application execution which is not sufficient for effective performance tuning of parallel applications. The most popular performance metrics such as speedup or efficiency require repeated execution of the application for various machine sizes for which no automatic tool support exists so far. Additionally, parallelising and tuning of applications for a certain compute platform requires repeated experimentation for various data distributions, loop iteration scheduling strategies, or compiler optimisation options, which is known to be an NP-complete problem.

Second, tool portability is critical since tools are often based on a monolithic design that does not isolate the inherent platform dependencies required to support advanced and in-depth analysis. For example, when using a new

parallel system the users must in most cases learn about and familiarise themselves with new tools with different functionality and interfaces, which is in many cases very time consuming and can be a major barrier in using novel modern computer architectures.

Third, existing tools cannot be used in cooperation on the same application instance to enhance the performance and correctness debugging engineering process since they are not designed for interoperability and often are based on incompatible instrumentation or monitoring systems.

Fourth, the workflow model that recently emerged as a new attractive paradigm for programming loosely coupled Grid infrastructures requires novel tools that offer appropriate high-level support, including abstract specification mechanisms, optimised scheduling, and scalable fault-tolerant execution, which are of paramount importance to effectively running distributed large-scale applications. These topics attract a lot of interest within the Grid community that aims to evolve the Grid to a commodity platform that transparently aggregates high-performance resources scattered around the globe in a single virtual supercomputer for performing scientific simulations.

In this monograph, we first propose a new directive-based language called ZEN for compact specification of wide value ranges of interest for arbitrary application parameters, including problem or machine sizes, array or loop distributions, software libraries, interconnection networks, or target execution machines. We design the ZEN directives as problem-independent with global or fine-grain scopes that do not change the semantics of the application, nor require any application modification or special preparation. Irrelevant or meaningless experiments can be filtered through well-defined constraints. Additionally, the ZEN directives can be used to request a wide range of performance metrics to be collected from the application for arbitrary code regions.

Based on the ZEN language, we develop a novel experiment management tool called ZENTURIO for automatic experiment management of large-scale performance and parameter studies on parallel and Grid architectures. ZENTURIO offers automatic analysis and visualisation support across multiple experiments based on the performance and output data collected and organised in a common shared data repository. In contrast to existing parameter study tools, ZENTURIO requires no special preparation of the application, nor does it restrict the parametrisation to input files or to global input arguments. We validated the functionality and usefulness of ZENTURIO on several real-world parallel applications from various domains, including theoretical chemistry, photonics, finances, and numerical mathematics.

We designed ZENTURIO as a comprehensive distributed service-oriented architecture for interoperable tool development based on the latest state-of-the-art Web and Grid services technologies. We illustrate how a service-oriented architecture facilitates the integration of a broad set of tools and enables a range of useful tool interoperability scenarios that facilitate the

engineering effort of applications. We illustrate a variety of novel adaptations of state-of-the-art Web technologies for Grid computing which anticipated several existing standardisation efforts.

Based on the ZENTURIO experiment management architecture, we propose a generic optimisation framework that integrates general-purpose meta-heuristics for solving NP-complete performance and parameter optimisation problems in an exponential search space specified using the ZEN experiment specification language. We illustrate a generic problem-independent realisation of the search engine using a genetic algorithm that allows new optimisation problems to be formulated through appropriate objective functions, for example, a performance metric using the ZEN language. We illustrate three case studies that instantiate the framework for Grid workflow scheduling, throughput scheduling of parameter studies, and performance tuning of parallel applications on the Grid using irregular array distributions.

Finally, we propose a timely approach for modelling and executing scientific workflows in dynamic and heterogeneous Grid environments. We introduce an abstract formal model for hierarchical representation of complex directed graph-based workflows using composite activities (such as parallel and sequential loops or conditional activities) interconnected through control and data flow dependencies comprising advanced collective communication patterns such as broadcast, scatter, and gather. We propose and comparatively analyse three heuristic-based algorithms for scheduling two real-world scientific workflows from material science and meteorology domains. The scheduled applications achieve good performance on the Austrian Grid environment using advanced runtime techniques such as partitioning, workflow optimisation, and load balancing. We design a steering algorithm that performs runtime monitoring and workflow schedule adaptations which ensure that certain quality of service performance contracts are preserved during execution of the workflow. We conclude with a classification of the most important performance overheads that may slow down the performance of scientific workflows and validate them through several experiments.

Innsbruck, October 2006

Radu Prodan  
Thomas Fahringer

---

# Contents

<b>1</b>	<b>Introduction</b>	1
1.1	Motivation	2
1.1.1	Performance Tuning	2
1.1.2	Parameter Studies	3
1.1.3	Optimisation	3
1.1.4	Scheduling	3
1.1.5	Parametrisation Language	4
1.1.6	Instrumentation	4
1.1.7	Portability	5
1.1.8	Tool Interoperability	5
1.1.9	Grid Services	5
1.1.10	Scientific Workflows	6
1.2	Goals	6
1.2.1	Experiment Specification Language	6
1.2.2	Experiment Management Tool	7
1.2.3	Optimisation	8
1.2.4	Scientific Workflows	8
1.2.5	Service-Oriented Grid Architecture	9
1.2.6	Grid Services	10
1.3	Outline	10
<b>2</b>	<b>Model</b>	13
2.1	Introduction	13
2.2	Distributed Technology History	14
2.3	Web Services	15
2.3.1	Web Services Stack	16
2.3.2	Web Services Runtime Environment	18
2.4	Grid Security Infrastructure	19
2.5	Globus Toolkit	20
2.6	Grid Architectural Model	22
2.6.1	Machine Layer	22
2.6.2	Grid Services Layer	28



2.6.3	Application Layer .....	30
2.7	Summary .....	35
<b>3</b>	<b>The ZEN Experiment Specification Language .....</b>	<b>37</b>
3.1	Functionality and Use Cases .....	37
3.1.1	Shared Memory Application Scalability .....	38
3.1.2	ZEN Transformation System .....	39
3.1.3	Shared Memory Loop Scheduling .....	40
3.1.4	Distributed Processor Arrays .....	41
3.1.5	Distributed Memory Arrays .....	41
3.1.6	Work Distribution .....	43
3.1.7	Parameter Studies .....	43
3.2	Formal Language Specification .....	44
3.2.1	ZEN Set .....	44
3.2.2	ZEN Directives .....	49
3.2.3	ZEN Substitute Directive .....	50
3.2.4	Local Substitute Directive .....	51
3.2.5	Homonym ZEN Variables .....	51
3.2.6	ZEN Assignment Directive .....	53
3.2.7	Multi-dimensional Value Set .....	54
3.2.8	ZEN Constraint Directive .....	55
3.2.9	ZEN Performance Directive .....	59
3.2.10	Parameter Study Experiment .....	62
3.2.11	Experiment Generation Algorithm .....	62
3.2.12	Online Monitoring and Analysis .....	65
3.3	Summary .....	68
<b>4</b>	<b>ZENTURIO Experiment Management Tool .....</b>	<b>69</b>
4.1	User Portal Functionality .....	69
4.1.1	ZEN Editor .....	70
4.1.2	Experiment Preparation .....	71
4.1.3	Experiment Monitor .....	73
4.1.4	Application Data Visualiser .....	73
4.2	Performance Studies .....	76
4.2.1	Ocean Simulation .....	76
4.2.2	Linearised Augmented Plane Wave .....	79
4.2.3	Three-Dimensional Particle-in-Cell .....	84
4.2.4	Benders Decomposition .....	86
4.2.5	Three-Dimensional FFT Benchmarks .....	89
4.3	Parameter Studies .....	94
4.3.1	Backward Pricing .....	94
4.4	Architecture .....	105
4.4.1	Experiment Generator .....	107
4.4.2	Experiment Executor .....	108
4.4.3	Experiment State Transition Diagram .....	110

4.4.4	Experiment Data Repository .....	110
4.5	Summary .....	111
<b>5</b>	<b>Tool Integration .....</b>	<b>113</b>
5.1	Architecture .....	114
5.2	Interoperable Tool Set .....	116
5.2.1	Object Code Browser .....	117
5.2.2	Function Profiler (Z_prof) .....	117
5.2.3	Function Tracer (Z_trace) .....	118
5.2.4	Function Coverager (Z_cov) .....	119
5.2.5	Sequential Debugger (Z_debug) .....	121
5.2.6	Memory Allocation Tool (Z_MAT) .....	121
5.2.7	Resource Tracker (Z_RT <sup>2</sup> ) .....	122
5.2.8	Deadlock Detector (Z_deadlock) .....	122
5.3	Tool Interoperability .....	122
5.3.1	Classification .....	122
5.3.2	Interaction with a Browser .....	123
5.3.3	Performance Steering .....	124
5.3.4	Just-in-Time Debugging .....	126
5.3.5	Interaction with a Debugger .....	127
5.4	The Monitoring Layer .....	128
5.4.1	Dynamic Instrumentation .....	128
5.4.2	The Process Manager .....	130
5.4.3	Dynamic Instrumentation of MPI Applications .....	134
5.5	The Grid Services Layer .....	136
5.5.1	Web Application and Services Platform (WASP) ....	138
5.5.2	Service Repository .....	139
5.5.3	Abstract Grid Service .....	140
5.5.4	Factory .....	142
5.5.5	Registry .....	143
5.5.6	WSDL Compatibility .....	144
5.5.7	Dynamic Instrumentor .....	144
5.5.8	Aggregator .....	145
5.6	Event Framework .....	146
5.6.1	Representation .....	146
5.6.2	Implementation .....	149
5.6.3	Filters .....	151
5.7	Firewall Management .....	151
5.8	WASP Versus GT3 Technology Evaluation .....	152
5.8.1	Stub Management .....	153
5.8.2	Service Lifecycle .....	154
5.8.3	UDDI-Based Service Repository .....	155
5.8.4	Service Data .....	155
5.8.5	Events .....	155
5.8.6	Registry .....	156

5.8.7	Security .....	159
5.8.8	Grid Service Throughput .....	160
5.8.9	Comparison .....	163
5.9	Summary .....	164
<b>6</b>	<b>Optimisation Framework .....</b>	<b>165</b>
6.1	Workflow Scheduling .....	167
6.1.1	Schedule Dependencies .....	169
6.1.2	Objective Function .....	170
6.2	Genetic Search Engine .....	174
6.2.1	Initial Population .....	175
6.2.2	Selection .....	177
6.2.3	Crossover .....	177
6.2.4	Mutation .....	178
6.2.5	Elitist Model .....	178
6.2.6	Fitness Scaling .....	179
6.2.7	Convergence Criterion .....	180
6.3	Genetic Workflow Scheduling .....	180
6.3.1	WIEN2k .....	180
6.4	Throughput Scheduling .....	192
6.5	Performance Tuning of Parallel Applications .....	194
6.5.1	Parallel Applications on the Grid .....	195
6.6	Summary .....	201
<b>7</b>	<b>Scientific Grid Workflows .....</b>	<b>203</b>
7.1	Workflow Model .....	204
7.1.1	Computational Activity .....	205
7.1.2	Control Flow Dependencies .....	206
7.1.3	Data Flow Dependencies .....	207
7.1.4	Conditional Activity .....	207
7.1.5	Parallel Loop Activity .....	208
7.1.6	Sequential Loop Activity .....	211
7.1.7	Workflow Activity .....	213
7.2	Scheduler .....	214
7.2.1	Workflow Converter .....	214
7.2.2	Scheduling Engine .....	220
7.2.3	Layered Partitioning .....	226
7.2.4	WIEN2k .....	227
7.2.5	Invmod .....	231
7.3	Enactment Engine .....	235
7.3.1	Workflow Partitioning .....	236
7.3.2	Control Flow Management .....	241
7.3.3	Data Flow Management .....	242
7.3.4	Virtual Single Execution Environment .....	243
7.3.5	Workflow Steering .....	244

7.3.6	Fault Tolerance .....	248
7.3.7	WIEN2k Execution Experiments .....	253
7.3.8	Steering Experiments .....	255
7.4	Overhead Analysis .....	260
7.4.1	Experiments .....	263
7.5	Summary .....	269
<b>8</b>	<b>Related Work .....</b>	<b>271</b>
8.1	Experiment Management .....	271
8.2	Performance Study .....	271
8.3	Parameter Study .....	273
8.4	Optimisation and Scheduling .....	273
8.5	Tool Integration .....	274
8.5.1	Scientific Workflows .....	276
<b>9</b>	<b>Conclusions .....</b>	<b>279</b>
9.1	Contributions .....	279
9.1.1	Experiment Specification .....	279
9.1.2	Experiment Management .....	280
9.1.3	Optimisation .....	281
9.1.4	Tool Integration Design .....	281
9.1.5	Web Services for the Grid .....	283
9.1.6	Scientific Workflows .....	283
<b>10</b>	<b>Appendix .....</b>	<b>285</b>
10.1	Notations .....	285
10.2	Code Regions .....	288
10.3	Abbreviations .....	289
10.4	Performance Metrics .....	292
	<b>References .....</b>	<b>297</b>
	<b>Index .....</b>	<b>311</b>

---

## List of Figures

2.1	The interoperable Web services stack. ....	16
2.2	The best practices of publishing a Web service into a UDDI Service Repository. ....	18
2.3	The Web services runtime environment. ....	19
2.4	The GSI single sign-on and proxy delegation chain of trust. . .	20
2.5	The Grid architectural model. ....	23
2.6	The von Neumann architecture. ....	24
2.7	The Symmetric Multiprocessor (SMP) architecture. ....	24
2.8	The Cluster of Workstation (COW) architecture. ....	26
2.9	The SMP cluster architecture. ....	26
2.10	The Cache Coherent Non-Uniform Memory Access (ccNUMA) architecture. ....	27
2.11	The Grid hardware architecture. ....	28
2.12	The stateful Grid service design alternatives. ....	30
2.13	The parallel application execution model. ....	31
2.14	The execution model of parallel applications on the Grid. ....	32
3.1	The ZEN Transformation System. ....	40
3.2	The (CYCLIC(2), BLOCK) distribution of array $A(8, 4)$ onto processor array $P(2, 2)$ . ....	42
3.3	The ZEN constraint defined by Example 3.5. ....	44
3.4	The ZEN set element evaluation function. ....	46
3.5	The ZEN file instances generated by Example 3.2. ....	50
3.6	The value set constraint defined in Example 3.21. ....	57
3.7	The experiment generation algorithm data flow. ....	64
4.1	The ZENTURIO User Portal main panel. ....	70
4.2	The ZEN editor. ....	71
4.3	The Experiment Preparation dialog-box. ....	72
4.4	The Application Data Visualiser for performance studies. ....	74
4.5	The Application Data Visualiser for parameter studies. ....	75

4.6	The Stommel model performance results for various intra-node and inter-node machine sizes (I), $200 \times 200$ problem size, 20000 iterations. ....	80
4.7	The Stommel model performance results for various intra-node and inter-node machine sizes (II), $400 \times 400$ problem size, 40000 iterations. ....	81
4.8	The Stommel model performance results (III). ....	82
4.9	The LAPW0 performance results for various machine sizes. ...	85
4.10	The 3DPIC performance results for various machine sizes. ...	87
4.11	The benders decomposition performance results for various machine sizes. ....	90
4.12	The parallel three-dimensional FFT computation. ....	91
4.13	The three-dimensional FFT benchmark results (I). ....	95
4.14	The three-dimensional FFT benchmark results (II). ....	96
4.15	The three-dimensional FFT benchmark results (III). ....	97
4.16	The three-dimensional FFT benchmark results (IV). ....	98
4.17	The three-dimensional FFT benchmark results (V). ....	99
4.18	The three-dimensional FFT benchmark results (VI). ....	100
4.19	The three-dimensional FFT benchmark results (VII). ....	101
4.20	The constraint defined in Example 4.19. ....	103
4.21	The backward pricing parameter study results. ....	104
4.22	The ZENTURIO experiment management tool architecture. .	105
4.23	The Experiment Generator architecture. ....	107
4.24	The experiment state transition diagram. ....	110
4.25	The Experiment Data Repository schema. ....	111
5.1	The tool integration service-oriented architecture. ....	115
5.2	A snapshot of interoperable online software tools. ....	124
5.3	The steering configuration. ....	125
5.4	The cyclic debugging states. ....	126
5.5	A just-in-time debugging scenario. ....	127
5.6	The dynamic instrumentation control flow. ....	129
5.7	The Process Manager architecture. ....	130
5.8	The instrumentation probe class hierarchy. ....	132
5.9	The control flow for starting an MPI(CH) application for dynamic instrumentation. ....	135
5.10	The dynamic MPI library profiling. ....	137
5.11	The state transition diagram of WASP-based Web services. ...	139
5.12	The Grid services hierarchy. ....	141
5.13	The ZENTURIO event architecture. ....	147
5.14	The event hierarchy. ....	148
5.15	The Registry throughput results. ....	158
5.16	The secure versus insecure response time comparison. ....	160
5.17	The throughput results of WASP, GT3, and vanilla Axis services. ....	162

6.1	The ZENTURIO optimisation framework design. ....	166
6.2	A sample workflow application. ....	169
6.3	A sample Gantt chart for the workflow depicted in Figure 6.2, assuming that $e_2 = e_3$ (i.e. $\mathcal{S}_{CA_2} = \mathcal{S}_{CA_3}$ ). ....	172
6.4	The genetic operators. ....	179
6.5	The workflow genetic operators. ....	181
6.6	A simplified WIEN2k workflow. ....	183
6.7	The regression functions for LAPW0. ....	184
6.8	The best individual evolution for various application instances. ....	186
6.9	The experimental setup for genetic static scheduler tuning. ..	187
6.10	The genetic scheduler tuning results (I). ....	189
6.11	The genetic scheduler tuning results (II). ....	190
6.12	The genetic scheduler tuning results (III). ....	191
6.13	A sample Gantt chart for the activity set defined in Example 6.13. ....	194
6.14	The default general block array distribution defined in Example 6.18. ....	198
6.15	The default indirect array distribution defined in Example 6.21. ....	201
7.1	A valid and an invalid conditional activity example. ....	208
7.2	The collection transfer patterns. ....	212
7.3	A valid and an invalid sequential loop activity. ....	213
7.4	A sample workflow with two nested conditional activities. ...	216
7.5	The two iteration sequential loop unrolling. ....	218
7.6	The HEFT weights and ranks for a sample workflow. ....	224
7.7	The WIEN2k workflow representation. ....	228
7.8	The scheduling Gantt charts. ....	229
7.9	The WIEN2k scheduling results. ....	232
7.10	The Invmod scientific workflow. ....	233
7.11	The Invmod scheduling results. ....	234
7.12	A workflow partitioning example. ....	239
7.13	A control flow optimisation example. ....	241
7.14	A data flow optimisation example. ....	243
7.15	A VSEE example. ....	245
7.16	A workflow checkpointing example. ....	253
7.17	The WIEN2k execution results (I). ....	256
7.18	The WIEN2k execution results (II). ....	257
7.19	The WIEN2k execution results (III). ....	258
7.20	The workflow steering executions traces. ....	259
7.21	The execution overhead classification. ....	261
7.22	The WIEN2k overhead analysis (I). ....	265
7.23	The WIEN2k overhead analysis (II). ....	266
7.24	The WIEN2k overhead analysis (III). ....	267
7.25	The WIEN2k checkpointing results (I). ....	268
7.26	The WIEN2k checkpointing results (II). ....	269

---

## List of Tables

5.1	The event implementation support. ....	149
5.2	The events supported by ZENTURIO. ....	150
5.3	The open firewall ports. ....	152
5.4	The service data elements. ....	156
5.5	The comparative analysis of WASP versus GT3-based Grid services. ....	163
7.1	The HEFT weight and rank calculations for the sample workflow depicted in Figure 7.6. ....	224
7.2	The Austrian Grid testbed for scheduling experiments. ....	230
7.3	The VSEE results for the WIEN2k workflow. ....	246
7.4	The input and output data checkpointing for the workflow example depicted in Figure 7.16. ....	253
7.5	The Austrian Grid testbed for WIEN2k execution experiments.	254
7.6	The Austrian Grid testbed for overhead analysis experiments.	263



---

# List of Algorithms

1	The experiment generation algorithm.....	64
2	The <b>Z_trace</b> call-graph function tracing algorithm. ....	119
3	The <b>Z_cov</b> function coverage algorithm. ....	120
4	The generational genetic search algorithm.....	176
5	The workflow conversion algorithm (I).....	220
6	The workflow conversion algorithm (II). ....	221
7	The workflow conversion algorithm (III). ....	222
8	The HEFT algorithm.....	225
9	The myopic scheduling algorithm. ....	226
10	The workflow steering algorithm. ....	249