

Jun Qin · Thomas Fahringer

Scientific Workflows

Programming, Optimization, and
Synthesis with ASKALON and AWDL

 Springer

Scientific Workflows

Jun Qin • Thomas Fahringer

Scientific Workflows

Programming, Optimization, and
Synthesis with ASKALON and AWDL

 Springer

Jun Qin
Amadeus Data Processing GmbH
Erding
Germany

Thomas Fahringer
Universität Innsbruck
Innsbruck
Austria

ISBN 978-3-642-30714-0 ISBN 978-3-642-30715-7 (eBook)
DOI 10.1007/978-3-642-30715-7
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012944567

ACM Computing Classification (1998): H.4, C.2, C.3, J.3, I.2

© Springer-Verlag Berlin Heidelberg 2012

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

dedicated to our families

Preface

Distributed systems enable the sharing, selection, and aggregation of a wide variety of distributed resources for solving large-scale, resource (computation and/or data)-intensive problems in science. Workflows represent a main programming model for the development of scientific applications. Creating scientific workflow applications, however, is still a very challenging task for scientists due to the complexity of distributed computing environments, the complex control and data flow requirements of scientific applications, and the lack of high-level languages and tool support. Particularly, sophisticated expertise in distributed computing is often required to determine the software entities to perform computations of workflow tasks, the computers on which workflow tasks are executed, the execution order of workflow tasks, and the data communications among them. The composition of an optimized scientific workflow can be even more difficult. Existing work suffers from being limited to specific implementation technologies, modeling low-level tasks, limited mechanisms to express iterative, conditional, and parallel execution, naive dataset distribution, no semantic support, no automatic workflow composition, or automatic workflow composition limited for special cases, etc.

This book presents a novel workflow language called Abstract Workflow Description Language (AWDL) and the corresponding standards-based, knowledge-enabled tool support with the aim to simplify the development of scientific workflow applications, as well as to optimize and to synthesize them. AWDL is an XML-based language for describing scientific workflow applications at a high level of abstraction. AWDL is designed such that users can concentrate on specifying scientific workflow applications without dealing with either the complexity of distributed computing environments or any specific implementation technology. A rich set of control flow constructs is provided in AWDL to simplify the specification of scientific workflow applications which includes directed acyclic graphs, conditional branches, parallel and sequential iterative constructs, alternative executions, etc. Properties and constraints can be specified in AWDL to provide additional information for workflow runtime environments to optimize and steer the execution of workflow applications. The AWDL modularization mechanism, including sub-workflows and workflow libraries, enables easy reuse and sharing of

scientific workflow applications among research groups. To streamline scientific workflow composition, a standards-based approach for modeling scientific workflow applications using the Unified Modeling Language (UML) Activity Diagram is presented, along with the corresponding graphical scientific workflow composition tool, which can automatically generate AWDL code based on graph representations of scientific workflows.

To meet the complex dataset-oriented data flow requirements of scientific workflow applications, AWDL introduces a data collection concept and the corresponding collection distribution constructs, which are inspired by High Performance Fortran (HPF). With these constructs, more fine-grained data flow can be specified at an abstract workflow language level, such as mapping a portion of a dataset to an activity, and independently distributing multiple collections onto parallel loop iterations. The use of these constructs improves the performance of scientific workflows by reducing data duplications and simplifies the effort to port scientific workflow applications onto distributed systems.

With the help of Semantic Web technologies, AWDL introduces a novel semantic based approach for scientific workflow composition. This approach features separation of concerns between data semantics and data representations and between Activity Functions (AFs) and Activity Types (ATs). It simplifies scientific workflow composition by enabling knowledge support and automatic data conversion. On the basis of this semantic approach, an Artificial Intelligence planning based algorithm for automatic control flow composition of scientific workflows using an Activity Function Data Dependence (ADD) is presented. The algorithm employs progression to create an ADD graph and regression to extract workflows, including alternative ones if available. The extracted workflows are then optimized based on data dependence analysis. Following automatic control flow composition, data flow composition is automated by semantically matching each data sink in scientific workflows against the corresponding data sources obtained through backward control flow traversing.

The newly introduced techniques and algorithms are implemented in the framework of the ASKALON development and runtime environment for workflows on distributed systems. The effectiveness of our techniques is demonstrated through a variety of experiments on a distributed computing infrastructure.

The topic covered in this book is of interest to a broad range of computer science researchers, domain scientists who are interested in applying workflow technologies in their work, and engineers who want to develop workflow systems, languages, and tools.

Munich, Germany
Innsbruck, Austria
April 2012

Jun Qin
Thomas Fahringer

Contents

Part I Overview

1	Introduction	3
1.1	Motivation	5
1.1.1	Adaptability	5
1.1.2	Abstraction	6
1.1.3	Single System Image	6
1.1.4	Complexity of Scientific Applications	6
1.1.5	Workflow Modularization	6
1.1.6	Standard Based Graphical Tools	7
1.1.7	Data Flow Optimization	7
1.1.8	Semantics Based Scientific Workflow Composition	7
1.1.9	Automatic Scientific Workflow Composition	8
1.2	Research Goals	8
1.2.1	Abstract Workflow Description Language	8
1.2.2	Workflow Modularization	9
1.2.3	UML-Based Scientific Workflow Modeling	9
1.2.4	Data Flow Optimization	10
1.2.5	Semantics-Based Scientific Workflow Composition	10
1.2.6	Automatic Scientific Workflow Composition	10
1.3	Organization	11
1.3.1	Part I: Overview	11
1.3.2	Part II: Programming	11
1.3.3	Part III: Optimization	12
1.3.4	Part IV: Synthesis	12
1.3.5	Part V: Related Work	13
1.3.6	Part VI: Conclusions	13
1.3.7	Appendices	13

2	Prerequisites	15
2.1	Distributed Systems	15
2.1.1	Programming Architecture	15
2.1.2	Characteristics	17
2.2	Programming for Distributed Systems	17
2.2.1	Programming Issues	17
2.2.2	Programming Models	18
2.3	Scientific Workflow	21
2.3.1	Comparison of Scientific Workflows with Business Workflows	21
2.3.2	Scientific Workflow Basics	23
2.4	Semantics	24
2.4.1	Ontology	24
2.4.2	Web Ontology Language	25
2.5	ASKALON	25
2.5.1	Workflow Composition	26
2.5.2	Resource Management	26
2.5.3	Workflow Scheduling	27
2.5.4	Workflow Execution	27
2.5.5	Performance Analysis and Prediction	27
2.6	Summary	28
 Part II Programming		
3	Abstract Workflow Description Language	31
3.1	System Overview	31
3.2	Specification	32
3.2.1	Atomic Activity	33
3.2.2	AWDL Control Flow	35
3.2.3	AWDL Data Flow	47
3.2.4	Properties and Constraints	52
3.2.5	The Structure of an AWDL Workflow	53
3.3	AWDL Modeling: A Real-World Scientific Workflow Example	54
3.4	Reification and Execution	57
3.5	Summary	60
4	Workflow Modularization	63
4.1	Introduction	63
4.2	Sub-workflows	63
4.3	Workflow Library	64
4.4	Invocation of (Sub-)workflows	65
4.4.1	Detection of Infinite Sub-workflow Invocation	67
4.4.2	Workflow Invocation	69
4.5	Case Study: WIEN2k	70
4.6	Summary	73

- 5 UML-Based Scientific Workflow Modeling** 75
 - 5.1 Introduction 75
 - 5.2 UML Activity Diagram 75
 - 5.3 Modeling Scientific Workflows with the UML Activity Diagram 76
 - 5.4 Graphical Scientific Workflow Composition Tool 79
 - 5.4.1 Graphical User Interface 80
 - 5.4.2 Model Traverser 83
 - 5.4.3 Model Checker 83
 - 5.5 ASKALON Workflow Hosting Environment 86
 - 5.6 Real-World Scientific Workflow Modeling and Code Generation 87
 - 5.7 Summary 89

Part III Optimization

- 6 Collection-Oriented Data Flow Support for Scientific Workflows** 93
 - 6.1 Introduction 93
 - 6.2 Data Collections and Collection Distribution Constructs 94
 - 6.2.1 Data Collections 94
 - 6.2.2 Collection Distribution Constructs 95
 - 6.3 Case Study 100
 - 6.3.1 WIEN2k 100
 - 6.3.2 MeteoAG 102
 - 6.3.3 AstroGrid 102
 - 6.3.4 GRASIL 104
 - 6.4 Experimental Results 105
 - 6.4.1 Results of the WIEN2k Workflow 106
 - 6.4.2 Results of the MeteoAG Workflow 109
 - 6.5 Summary 111

Part IV Synthesis

- 7 Semantic-Based Scientific Workflow Composition** 115
 - 7.1 Introduction 115
 - 7.2 Ontologies 117
 - 7.3 Ontology-Based Scientific Workflow Representation 120
 - 7.3.1 Activity Function 120
 - 7.3.2 Activity Type 121
 - 7.3.3 From Semantics to Syntax 123
 - 7.4 Implementation 127
 - 7.5 Experimental Results 131
 - 7.6 Summary 134

8	Automatic Scientific Workflow Composition	135
8.1	Introduction	135
8.2	Activity Function	136
8.3	Definition of the Scientific Workflow Composition Problem	137
8.4	ADD Graph and Notations	138
8.5	Control Flow Composition Algorithm	143
8.5.1	ADD Graph Creation	143
8.5.2	Workflow Extraction	143
8.5.3	Workflow Optimization	146
8.5.4	Algorithm Analysis	149
8.5.5	Branches and Loops	152
8.6	Data Flow Composition Algorithm	154
8.7	Experimental Results	156
8.8	Summary	164

Part V Related Work

9	Related Work	169
9.1	Overview of Main Workflow Systems and Languages	169
9.2	Programming	172
9.2.1	Abstraction	173
9.2.2	Type System	175
9.2.3	Control Flow	177
9.2.4	Data Flow	180
9.2.5	Modularization and Reuse	181
9.2.6	Modeling Support	182
9.2.7	Semantic Support	183
9.3	Optimization	185
9.4	Synthesis	187

Part VI Conclusions

10	Conclusions	191
10.1	Contributions	191
10.1.1	Scientific Workflow Language	191
10.1.2	Scientific Workflow Modeling	192
10.1.3	Data Flow Optimization	192
10.1.4	Semantic Scientific Workflows	193
10.1.5	Automatic Scientific Workflow Composition	194
10.2	Future Research	194

Part VII Appendices

A	Acronyms	199
----------	-----------------------	-----

Contents	xiii
B Symbols	201
References	205
Index	217

List of Figures

Fig. 2.1	Distributed system programming architecture compared with conventional programming architecture	16
Fig. 2.2	The architecture of ASKALON	26
Fig. 3.1	The development process of scientific workflows in ASKALON	32
Fig. 3.2	Atomic activity	33
Fig. 3.3	Activity type, activity deployment, and activity instance	34
Fig. 3.4	The atomic activity <i>CalcKPoint</i>	35
Fig. 3.5	dag activity	37
Fig. 3.6	The dag activity <i>dagRAMS</i>	38
Fig. 3.7	The graphical representation of the dag activity <i>dagRAMS</i>	38
Fig. 3.8	sequence activity	39
Fig. 3.9	parallel activity	39
Fig. 3.10	if activity	40
Fig. 3.11	The if activity <i>runhist</i>	41
Fig. 3.12	for activity	42
Fig. 3.13	The for activity <i>pForAkmin</i>	43
Fig. 3.14	while activity	44
Fig. 3.15	The while activity <i>whileConv</i>	45
Fig. 3.16	alt activity	46
Fig. 3.17	The alt activity <i>initSimCaseAlt</i>	47
Fig. 3.18	Data flow in dag activity	48
Fig. 3.19	Data flow in if activity	49
Fig. 3.20	Data flow in while activity	50
Fig. 3.21	Data flow in for activity with <code>parallel="yes"</code>	51
Fig. 3.22	Properties and constraints	52
Fig. 3.23	Examples of properties and constraints	53
Fig. 3.24	The structure of an AWDL workflow	54
Fig. 3.25	The WIEN2k workflow	55
Fig. 3.26	An excerpt of the AWDL representation of the WIEN2k workflow	56

Fig. 3.27	Reification and execution	58
Fig. 3.28	The activity <i>LAPW0</i> in the abstract workflow	58
Fig. 3.29	The activity <i>LAPW0</i> in the concrete workflow	59
Fig. 4.1	<i>subWorkflow</i>	64
Fig. 4.2	Activity type invocation	66
Fig. 4.3	Sub-workflow invocation	66
Fig. 4.4	Workflow invocation	66
Fig. 4.5	Examples of incorrect recursive sub-workflow invocations	67
Fig. 4.6	The WIEN2k workflow	70
Fig. 4.7	The WIEN2k workflow using sub-workflows	71
Fig. 4.8	The AWDL representation of the WIEN2k workflow using sub-workflows	72
Fig. 5.1	A subset of modeling elements of UML activity diagram	76
Fig. 5.2	The definition and usage of the stereotype <i>Activity</i>	77
Fig. 5.3	The UML activity diagram representations of AWDL control flow constructs	78
Fig. 5.4	Modeling data flow	79
Fig. 5.5	Main GUI	80
Fig. 5.6	Data flow composition: specific data	81
Fig. 5.7	Data flow composition: source data port	82
Fig. 5.8	Activity constraint: <i>awdl:node</i>	82
Fig. 5.9	Data port property: <i>awdl:datasize</i>	83
Fig. 5.10	Model traverser	84
Fig. 5.11	Connection rules	85
Fig. 5.12	Error examples detectable by the static model checker	85
Fig. 5.13	Error examples detectable by the dynamic model checker	86
Fig. 5.14	ASKALON workflow hosting environment (AWHE)	87
Fig. 5.15	Searching scientific workflows in AWHE	87
Fig. 5.16	The UML activity diagram representation of the MeteoAG workflow	88
Fig. 6.1	Data flow problems	94
Fig. 6.2	A data collection <i>a</i> with <i>n</i> data elements	95
Fig. 6.3	The <i>BLOCK</i> distribution	97
Fig. 6.4	The <i>BLOCK(5)</i> distribution	98
Fig. 6.5	The <i>BLOCK(6,3)</i> distribution	99
Fig. 6.6	The <i>REPLICA(4)</i> distribution	100
Fig. 6.7	The WIEN2k workflow	101
Fig. 6.8	The MeteoAG workflow	103
Fig. 6.9	The AstroGrid workflow	104
Fig. 6.10	The GRASIL workflow	105
Fig. 6.11	Collection distribution constructs used in the WIEN2k workflow	107
Fig. 6.12	Experimental results of the WIEN2k workflow	108

Fig. 6.13	Performance analysis of the execution of <i>pForLAPW2</i> with <i>kpoint=252</i> on 5 nodes	109
Fig. 6.14	The MeteoAG experimental results	110
Fig. 7.1	Two scientific workflow composition problems	116
Fig. 7.2	The upper ontology	118
Fig. 7.3	Ontology for DataRepresentation	119
Fig. 7.4	Activity function <i>RAMSInit</i>	121
Fig. 7.5	Activity type <i>rams_init_c</i>	122
Fig. 7.6	Activity type <i>month-str-int</i>	123
Fig. 7.7	Mapping an AF to ATs	124
Fig. 7.8	Workflow mapping process	126
Fig. 7.9	The main data classes of the meteorology ontology	128
Fig. 7.10	The main function classes of the meteorology ontology	129
Fig. 7.11	The Meteo2 workflow	130
Fig. 7.12	The architecture of the prototype	131
Fig. 7.13	Execution time of searching for AFs based on free text and based on input and output data	132
Fig. 7.14	Execution time of mapping AF <i>RAMSMakevfile</i> (direct match), mapping AF <i>RAMSAll</i> (indirect match), and mapping the entire workflow Meteo2	133
Fig. 8.1	The activity function <i>RAMSInit</i>	136
Fig. 8.2	An Activity Function Data Dependence (ADD) graph and its <i>ncDC(S)</i> , <i>ncAF(S)</i> , <i>altDC(S)</i> , and <i>altAF(S)</i>	139
Fig. 8.3	Alternative AF combinations	142
Fig. 8.4	Automatically composed scientific workflows	146
Fig. 8.5	Automatically optimized scientific workflows	148
Fig. 8.6	Improvement of workflow optimization	149
Fig. 8.7	An ADD graph for generation of sequential loops	153
Fig. 8.8	Data flow among source and sink data ports	155
Fig. 8.9	Execution time of the control flow composition algorithm in the worst case	158
Fig. 8.10	Memory usage of the control flow composition algorithm in the worst case	159
Fig. 8.11	Automatic composition of the Meteo2 workflow	160
Fig. 8.12	Execution time of the optimized and nonoptimized Meteo2 workflows	162
Fig. 8.13	Speedup of the optimized and nonoptimized Meteo2 workflows	162
Fig. 8.14	Comparison of the numbers of available data sources for data ports in Meteo2	163
Fig. 8.15	Automatic data flow composition plugin	164

List of Tables

Table 5.1	Extending UML activity diagram to model AWDL constructs	77
Table 6.1	Notations	97
Table 6.2	The nodes used for the experiments of collection distribution.....	105
Table 7.1	DataRepresentation properties	119
Table 8.1	ncAF(S) and their input/output DCs	153
Table 8.2	The nodes used for the execution of both Meteo2 workflows	161

List of Algorithms

- 4.1 Find incorrect recursive sub-workflow invocation 68
- 7.1 Workflow mapping: mapping() 125
- 8.1 ADD graph creation 144
- 8.2 Workflow extraction: extractWorkflows() 145
- 8.3 Calculate alternative AF combinations: calcAltAF() 145
- 8.4 Workflow optimization 147
- 8.5 Sequential loop generation 154
- 8.6 Automatic data flow composition 156

Part I

Overview