Peter M. Borst

An Architecture for Distributed Interpretation of Mobile Programs

Herbert Utz Verlag · Wissenschaft
München

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Ein Titeldatensatz für diese Publikation ist
bei Der Deutschen Bibliothek erhältlich

Zugleich: Dissertation, Karlsruhe, Univ., 2001

ISBN 3-8316-0103-8

Printed in Germany

Herbert Utz Verlag GmbH, München
Tel.: 089/277791-00 – Fax: 089/277791-01

Contents

Contents xi
List of Figures xvi
List of Tables xix
Abstract xxi
Zusammenfassung xxiii

1 Introduction **1**

1.1 Motivation 1
1.2 Problem Formulation 3
1.3 Organisation of the Dissertation 5

2 Program Mobility in Distributed Systems **7**

2.1 Distributed Systems 7
2.1.1 Basic Concepts 8
2.1.2 Classes of Distribution 9
2.1.3 Communication Models 10
2.1.4 Distributed System Architectures 20
2.2 The Situation in Open Networks 30
2.2.1 Levels of Openness 30
2.2.2 The Demand for Distributed Infrastructures 32
2.2.3 Current Approaches 33
2.3 Program Mobility Models 35

2.3.1 Motivation 35
2.3.2 A Classification of Mobile Programming Approaches 40
2.3.3 Mobile Code 44
2.3.4 Mobile Agents 53
2.3.5 Models for Mobile, Distributed Algorithms 67
2.4 Summary 91

3 Mobile WAVE Technology 95

3.1 The WAVE Model 95
3.1.1 Logic Flow in Active Data 95
3.1.2 Spreading of Wavefronts 98
3.1.3 Data and Control Infrastructures 101
3.1.4 Layered Organisation of the WAVE Automaton 105
3.1.5 Embedding of WAVE in Computer Networks 107
3.2 Mechanisms of the WAVE Technology 111
3.2.1 WAVE as a Family of Languages 111
3.2.2 Program Mobility in WAVE 115
3.2.3 Track Supervision and Mobile Control 121
3.2.4 Distributed Garbage Collection 125
3.2.5 Summary 126
3.3 The WAVE Language 128
3.3.1 Language Syntax 129
3.3.2 Data Types 131
3.3.3 Variables 133
3.3.4 Elementary Operations 135
3.3.5 Control Rules 145
3.3.6 Programmed Echoes 148
3.3.7 Cycle Detection 151
3.4 Programming Examples 154

3.4.1 Elementary Data Processing in WAVE 155
3.4.2 Control Constructs 157
3.4.3 Local and Distributed Loops 163
3.4.4 Code Injection and Recursion 168
3.4.5 Graph Algorithms in WAVE 174
3.4.6 Summary and Discussion 187
3.5 Comparison 190
3.5.1 WAVE and Semantic Network Processing 190
3.5.2 WAVE and Distributed Databases 192
3.5.3 WAVE and Mobile Agents 194
3.5.4 WAVE and Java 195
3.6 Summary 198

4 Distributed Interpretation of Mobile Programs 201

4.1 Introduction and Notion Definitions 202
4.2 Distributed Program Execution Architectures 205
4.2.1 Mobile Code Interpretation 206
4.2.2 Mobile Agent Execution Engines 208
4.2.3 Mobile Wave Interpretation 209
4.2.4 Comparison 211
4.3 Distributed Interpretation in WAVE 212
4.3.1 Main Interpretation Functions 213
4.3.2 The Mobile Execution Environment 216
4.3.3 Stages of Wave Interpretation 220
4.3.4 Mobile Messages in WAVE 222
4.4 Architecture of the Virtual WAVE Machine 226
4.4.1 Main Modules of the WAVE Interpreter 226
4.4.2 Knowledge Network Representation 231
4.4.3 Operations on KN 235

4.4.4 Track Forest Representation 238
4.4.5 Operations on Tracks 241
4.4.6 Control Flow within the Virtual WAVE Machine 248
4.5 Summary 255

5 An Abstract Machine for WAVE Interpretation 257

5.1 Introduction 258
5.1.1 Applications of Abstract Machines 258
5.1.2 Overview of Abstract Machine Techniques 260
5.1.3 Preliminary Considerations for the Abstract WAVE Machine 262
5.1.4 Organisation of the Chapter 265
5.2 Architecture of the AWM 266
5.2.1 Basic Definitions 266
5.2.2 Elementary Machine Transitions 272
5.2.3 The Dispatching Stage 275
5.2.4 Elementary Data Processing 286
5.2.5 Program Decomposition and Splitting 296
5.2.6 Code Injection 299
5.2.7 Processing of Hops 301
5.2.8 External Calls and Terminal I/O 319
5.3 Analysis of the Language Core 322
5.4 Extensions for Track Supervision 325
5.4.1 Definitions 325
5.4.2 Control Rules 329
5.4.3 The Halt Operation 335
5.4.4 Distributed Storage Management 340
5.4.5 Track Supervision Mechanisms 346
5.4.6 Echo Routing 362
5.4.7 Tail Routing 388

5.5 Summary 399

6 Parallel WAVE Interpreter Architecture 403

6.1 Introduction 404
6.1.1 Motivation 404
6.1.2 The WAVE Interpreter as a Separate Co-Processor 406
6.1.3 Pro's and Contras for Parallel Hardware 409
6.1.4 Preliminary Considerations 411
6.2 Components of the Parallel WAVE Interpreter 416
6.2.1 Top-Level Architecture 416
6.2.2 The Language Processor 418
6.2.3 The Data Processor 421
6.2.4 The Control Processor 424
6.2.5 Top-Level Organisation 427
6.3 Interactions between the Modules 430
6.3.1 Data Flow 431
6.3.2 Queueing and Dispatching 439
6.3.3 Control Flow for Wave Processing 444
6.3.4 Control Flow for Track-Related Operations 448
6.3.5 Top-Level Schedule 453
6.3.6 Fully Parallel Organisation of the Interpreter 459
6.4 Analysis 462
6.4.1 The WAVE Application Benchmark 463
6.4.2 Analysis of the Three-Processor Architecture 481
6.4.3 Analysis of the Fully Parallel Architecture 496
6.4.4 Summary of the Analysis 507
6.5 Summary 510

7 Conclusions 513

7.1 Usefulness of Mobile Programs 513
7.2 Contributions 516
7.3 Summary 517
7.4 Future Work 520

References **525**

Chapter 1

Introduction

1.1 Motivation

Since the advent of digital telecommunication, computer networks have been continuously growing. Technical innovation has steadily made even faster and cheaper networks possible and permitted to bridge wide geographic distances. At the time of today computer networks are practically ubiquitous and, in different forms, they are connecting the whole world. Part of the tremendous success of computer networks has been the development of standard communication protocols, for instance TCP/IP [Com91], many of which appeared during the 1980s. Using such standards it has been possible to connect practically every computer of any architecture and from any manufacturer with every other computer. Large *network infrastructures* began growing such as the ARPAnet and later the Internet which is connecting today several millions of computers wordwide. We can characterise such networks as largely heterogeneous, sophisticated in their structure and containing *huge amounts of distributed data.* Another type of networks can be found for the purpose of control of large-scale distributed systems comprising areas such as mobile telecommunication, traffic control and military. These networks are presently more structured, say, hierarchically, and can be characterised to embody a *huge number of distributed events* which are to be monitored, collected and re-distributed in order to implement the necessary control functions.

As the size of the network increases, the problem of handling the distributed data and the distributed events becomes more sophisticated. New distributed

processing techniques are demanded which are *scalable* [BDMS94]. Although the network technologies have been evolving rapidly, it can be observed that the distributed processing dicipline did not respond properly to these ongoing changes. Major achievements have been made in the areas of distributed operating systems and high-performance parallel computing in networks. These are restricted, however, to small-scale, closed networks and can not be applied in the large. In large-scale networks and global networks like the Internet, the major doctrine is still *centralisation*, in most cases of both data and control, which is clearly not scalable.

The only true solution to the problem of scalability are programs which can operate in a fully *decentralised* way. In networks such programs can already be found in the form of distributed service procedures in cases where truely decentralised processing is required. In mobile telecommunication networks, for instance, distributed message handlers are employed which pass the events and status changes along the hierarchy to the central database. In the Internet several distributed services can be found, operating also hierarchically, which accomplish, say, name resolution, email delivery and transport-level message routing. It is clear, however, that the mentioned decentralised services can only work since the necessary procedures exist ad hoc in *all* nodes of the network.

Unfortunately, user-level decentralised programs cannot adopt the same architecture for two reasons. Full-scale replication of the programs contradicts scalability. In large-scale networks, user programs can simply not be installed ad hoc on all nodes. On the other hand, the selection of a suitable, smaller subset of nodes where to install the user programs will be generally not possible since global information would be required onto which to base such a decision. In large-scale networks centralised, global information will not exist.

A very promising way out of the dilemma is pointed out by the mobile program approaches which came up at the end of the 1980s [Sap88], [SG90b]. It has been recognised that in order to realise full-scale decentralised computing, a deviation from the traditional process-centric computing paradigm is necessary. Essentially, not only data has to be communicated between the distributed processes, also *programs have to be communicated*. In this way, a change from the "hard-coded" distributed programs to a general function can be achieved, the latter being instantiated by the communicated programs and carrying out the desired remote processing on behalf of the users. Using mobile programs, the static programming of a distributed system can be replaced by dynamic programming.

Fully decentralised processing can be achieved by *sending programs to the remote data* rather than the usual opposite approach of loading the remote data for local processing. A great impetus to this new ideology of distributed processing has been given by the introduction of mobile code and mobile agent technologies in the middle of the 1990s [Whi94b], [GM95] which have put the idea of program mobility into a modern context.

1.2 Problem Formulation

Obviously, mobile programs are a fascinating, new and unconventional approach for distributed processing in networks. On the other hand they also open several new dimensions of decisions which have to be made for the programs and the distributed infrastructure in which they operate. A few examples of such decisions are the following. Concerning the programs and their properties one could ask whether they should carry data along with them or not, whether they are issued in a client-server style or they can self-migrate autonomously, whether they represent a single thread or replicate themselves in order to carry out different tasks in parallel, and how the programs would interact with each other, with the environment on the visited computing nodes and with the users. Concerning the infrastructure there are degrees of freedom whether the programs should be compiled or interpreted and how to organise the distributed data and functions in the computing nodes so that they can be found and used by the mobile programs.

As a conclusion one can say that a large variety of different approaches how to realise mobile programs are possible. While some of them may be highly biased by the application for which they are intended, others may be more general. Similar to the variety of existing programming languages for different purposes, there will be no single mobile programming model. On the other hand, all mobile programming models are akin and they share one common basis: a suitable *general function* or *program interpreter* which resides in the distributed computing nodes and which is the vital element for the mobile programs to operate. This general function will also be the major point of our interest.

This work is based on existing mobile program models, in particular on one of them, the WAVE model [Sap87], [Sap88], [Sap89], [Sap99a]. WAVE belongs to the early models of mobile programs with roots reaching back into the

1970s. At the same time, it is one of the most unconventional and radical approaches. For instance, WAVE introduces a entirely new programming language and concept which is fully tailored to program mobility and decentralised processing with mobile programs, abolishing most of the classical von Neumann computing paradigm. Yet WAVE is a very general model, perhaps one of the most comprehensive ones, including mechanisms such as autonomous, self-migrating programs, replication and a new concept called mobile control which permits to implement complex and highly parallel distributed computations which operate with fully decentralised control.

Starting-point of this work is the abovementioned general function in the computing nodes for the reception and execution of the mobile programs. In WAVE this is an interpreter of the mobile WAVE language. Expectably, such mobile program execution engines have some more responsibilities than normal language processors. For instance, they may have to include means for concurrent execution the mobile programs, they have to organise the interaction between the programs and they have to include protocols for communicating the mobile programs and other messages. Due to the higher complexity, a mobile program interpreter may have opportunities for utilising concurrency of its internal functions as well. The final observation is that mobile program interpreters are largely autonomous functional units and so they would be ideally suited to operate as separate processors on the computing node to which they are attached.

This thesis derives and examines a parallel architecture for the mobile program execution engine which would result for the WAVE model. Since WAVE shares the basic elements with other mobile programming approaches, the main sceleton of the architecture might serve as a framework for the parallel implementation of other mobile program execution engines as well. On the other hand, WAVE introduces a number of novel concepts both with respect to the distributed structures maintained and with the integration of these structures into the mobile programming language. Together these mechanisms lead to a new quality of distributed programming which is unreached by any other mobile program model so far.

The second goal of this thesis is to shed light on the novel mechanisms of WAVE and the basic programming techniques relating to them. Standing alone, these mechanisms are rather simple so that the benefit of including some of them into another, say, mobile code or mobile agent model may be worth a con-

sideration. The study of the WAVE interpreter architecture is carried out to depth in order to clarify all aspects of WAVE, the language and its interpretation. Finally, performance estimates of the parallel architecture are given in order to evaluate the effects of the parallelism and to derive hints for the requirements to a realisation of the WAVE interpreter in hardware or silicon.

The WAVE interpreter has been implemented in software by the author of this work. An object-code distribution for Sun and Silicon Graphics computers has been made available in the Internet in autumn 1995 (www-zorn.ira.uka.de/wave). The system has been ported to Linux in 2000 and is currently being utilised and extended at the University of British Columbia, Canada. An updated status of the available versions as well as a source code licence of the WAVE interpreter for non-commercial use can be requested from Prof. Dr. W. Zorn (www-zorn.ira.uka.de).

1.3 Organisation of the Dissertation

The dissertation is roughly divided into two parts. The first part (Chapter 2 and 3) comprises the study of the WAVE model and technology whereas the second one (Chapter 4 to 6) will deal with the aspects of the machine, developing and analysing the WAVE interpreter architecture.

Chapter 2 gives an introduction to the traditional approaches and the newer, mobile program models for distributed processing in computer networks. The demands to the models for large-scale networks are derived and a classification of the mobile approaches is given. Chapter 3 gives an introduction to the WAVE technology including the model, its mechanisms, the programming language which we will use as the basis for the interpreter architecture, and some programming examples in this language. A comparison of WAVE with other distributed approaches is given at the end of this chapter.

Chapter 4 to 6 look at the techniques and architectural constraints for mobile program execution and WAVE interpretation from different levels. Chapter 4 studies the main elements of distributed program interpretation on top level. In this chapter, the Virtual WAVE Machine is defined, comprising the main modules of the WAVE interpreter, the necessary distributed datastructures and operations, and a concurrent control flow scheme of the internal interpretation process. Chapter 5 gives the definition of the Abstract WAVE Machine as the

complete formalisation of the WAVE interpreter, its structures, functions and protocols. Chapter 6 develops the parallel WAVE interpreter architecture. Two different parallel organisations of the interpreter are proposed and analysed, one applying to a parallel implementation using conventional microprocessors and the other for a fully parallel implementation in hardware or silicon. Chapter 7 reviews the benefits of mobile programs and the WAVE model, discusses the contribution of the dissertation and indicates possible future research directions.