Dejan S. Milojicic

Load Distribution
Dejan S. Milojicic

**Load Distribution**
Implementation for the Mach Microkernel

Christoph W. Keßler (Ed.)

**Automatic Parallelization**
New Approaches to Code Generation, Data Distribution, and Performance Prediction

R. Jungclaus

**Modeling of Dynamic Object Systems**
A Logic-based Approach

J.M. Schneider

**Protocol Engineering**
A Rule-based Approach

R. Gotzhein

**Open Distributed Systems**
On Concepts, Methods, and Design from a Logical Point of View

Spiro N. Pollalis

**Computer-Aided Project Management**
A Visual Scheduling and Management System

E. Eder

**Relative Complexities of First Order Calculi**
Dejan S. Milojicic

Load Distribution

Implementation for the Mach Microkernel

With a Foreword by Jürgen Nehmer
This book represents the PhD thesis defended at the University of Kaiserslautern. Some of the material in this book has been presented at the USENIX Symposiums and IEEE Workshop, as referenced in the following material:


Verlag Vieweg, P.O. Box 58 29, D-65048 Wiesbaden

All rights reserved
© Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 1994
Softcover reprint of the hardcover 1st edition 1994

No part of the publication may be reproduced, stored in a retrieval system or transmitted, mechanical, photocopying or otherwise, without prior permission of the copyright holder.

Printed on acid-free paper

DOI: 10.1007/978-3-322-85872-6
To my Grandparents Milka and Dušan Generalović
Load distribution is a very important concept for distributed systems in order to achieve better performance, resource utilization and response times. Providing efficient mechanisms for the transparent support of load distribution has proven to be an extremely difficult undertaking. As a matter of fact, there is no commercially available system which provides transparent load distribution right now. The monograph by D. Milojicic presents a novel load distribution scheme based on modern microkernel architectures. The remarkable results of D. Milojicic's approach show evidence for his hypothesis that load distribution is feasible even under strong efficiency constraints if built upon microkernel architectures. Based on a complete implementation using the NORMA-version of Mach, D. Milojicic shows that substantial performance improvements of his load distribution scheme on top of Mach result from the dramatic reduction of state information to be managed in course of a task migration.

For readers not familiar with the topic, the monograph gives a good survey of the load distribution problem and puts existing approaches into perspective.
## Contents

Preface ...................................................... xvii

1 Introduction .............................................. 1
   1.1 Motivation ............................................ 1
   1.2 Load Distribution ..................................... 3
   1.3 Research Contributions .............................. 5
   1.4 Thesis Outline ....................................... 6

2 Background and Related Work .............................. 9
   2.1 Introduction .......................................... 9
   2.2 Migration ............................................. 9
      2.2.1 Design ............................................ 11
      2.2.2 Issues ........................................... 12
      2.2.3 Previous Work .................................... 14
   2.3 Load Information Management ....................... 19
      2.3.1 Design ............................................ 20
      2.3.2 Issues ........................................... 20
      2.3.3 Previous Work .................................... 21
   2.4 Distributed Scheduling ............................... 23
      2.4.1 Design ............................................ 23
      2.4.2 Issues ........................................... 24
      2.4.3 Previous Work .................................... 25
   2.5 Summary ............................................... 29

3 Mach and Load Distribution ............................... 31
   3.1 Introduction .......................................... 31
   3.2 Mach .................................................. 31
   3.3 Mach NORMA Version .................................. 37
   3.4 Mach Support for Load Distribution ................. 39
   3.5 Load Distribution Architecture, Overview ......... 42
   3.6 Summary ............................................... 43
## Contents

### 4 Task Migration

4.1 Introduction ........................................... 45  
4.2 General Principles and Architecture .................. 45  
4.3 Requirements for Microkernels .......................... 48  
4.4 Implementation ....................................... 50  
  4.4.1 Task Migration Algorithm .......................... 50  
  4.4.2 Necessary Modifications to the Mach Microkernel ... 55  
  4.4.3 Simple Migration Server ............................ 56  
  4.4.4 Optimized Migration Server ........................ 57  
  4.4.5 In-Kernel Task Migration ........................... 60  
4.5 Performance Measurements .............................. 62  
  4.5.1 Migration Server Measurements ...................... 64  
  4.5.2 WPI Benchmarks ..................................... 67  
  4.5.3 Parallel Make and Other Applications ............... 70  
4.6 Related Work ......................................... 70  
4.7 Summary .............................................. 72  

### 5 Load Information Management

5.1 Introduction ........................................... 75  
5.2 Load Information Collection ............................ 78  
  5.2.1 Information on Processing .......................... 79  
  5.2.2 Information on Network IPC ........................ 79  
  5.2.3 Information on XMM .................................. 80  
5.3 Information Dissemination and Negotiation ............. 81  
5.4 Performance Measurements .............................. 83  
5.5 Characterization of Mach Applications .................. 85  
5.6 Summary .............................................. 87  

### 6 Distributed Scheduling

6.1 Introduction ........................................... 89  
6.2 Distributed Scheduling Algorithms ..................... 90  
6.3 Artificial Load ....................................... 94  
6.4 Performance Measurements .............................. 95  
  6.4.1 Comparison of Various Strategies ................. 96  
  6.4.2 Considering Information on Communication ........... 99  
  6.4.3 Task Migration v. Initial Placement ................ 102  
6.5 Summary .............................................. 104  

### 7 Lessons Learned

7.1 Introduction ........................................... 107  
7.2 Task Migration Implementation .......................... 107  
7.3 Task Migration is not Always Enough ................... 109  
7.4 Task Migration v. Initial Placement .................... 111  
7.5 Microkernels are the Right Level for LD ............... 111
CONTENTS

7.6 Experiences with Network IPC ...................................... 113
7.7 Fault Tolerance is Hard to Support ................................. 114
7.8 Summary ............................................................. 116

8 Conclusions and Future Work ................................. 117
  8.1 Introduction .......................................................... 117
  8.2 Summary of Results ................................................ 117
  8.3 Future Work .......................................................... 118
  8.4 Conclusion ........................................................... 120

Bibliography .............................................................. 121

Index ................................................................. 146
List of Figures

1.1 Monolithic v. Microkernel Architecture .................................. 3
2.1 Load Distribution Elements ................................................ 10
2.2 Load Information Management ............................................ 19
2.3 Distributed Scheduling ...................................................... 24
3.1 The History of Mach Development ........................................ 35
3.2 Supporting the OS Personality Calls on Mach .......................... 36
3.3 Extending IPC to Network Environment Using Netmsg Server ...... 38
3.4 NORMA Support for Distributed Shared Memory ..................... 39
3.5 Monolithic v. Microkernel Architectures .............................. 40
3.6 LD Architecture .............................................................. 42
4.1 Task Migration Design ...................................................... 46
4.2 Interposing Task Kernel Port, External Perspective .................. 52
4.3 The Transferred Task State ............................................... 53
4.4 Interposing the Task Kernel Port - Internal Perspective ........... 55
4.5 Lazy Copying Address Space with OMS .................................. 58
4.6 Task State Transfer for Various Task Migration Versions ........... 62
4.7 Transfer Time v. Memory Size ............................................ 63
4.8 Transfer Time v. Number of Threads ..................................... 63
4.9 Transfer Time v. Number of Receive Capabilities .................... 64
4.10 Transfer Time v. Number of Capabilities ............................ 64
4.11 Transfer Time v. Memory Contiguity ................................... 65
4.12 Execution Time v. Memory Size ......................................... 65
4.13 Transfer Time v. Number of Receive Capabilities for SMS and OMS 66
4.14 Execution Time v. Remote/Local IPC .................................... 66
4.15 Solving Time v. Jigsaw Puzzle Size ................................... 68
4.16 Paging v. Jigsaw Puzzle Size ........................................... 68
4.17 Make Ratio ................................................................. 69
5.1 Common Types of NORMA Applications ............................... 76
5.2 IPC-Type Task Execution Time v. Local/Remote Execution .......... 78
5.3 XMM-Type Task Execution Time v. Local/Remote Execution ......... 78
5.4 Network and Local IPC Paths and Accounting ........................ 81
<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5</td>
<td>Network and Local Paging Paths and Accounting</td>
<td>82</td>
</tr>
<tr>
<td>5.6</td>
<td>Average Execution Time as a Function of the IPC and Processing</td>
<td>83</td>
</tr>
<tr>
<td>5.7</td>
<td>Average Execution Time as a Function of the XMM and Processing</td>
<td>83</td>
</tr>
<tr>
<td>5.8</td>
<td>Profiling Application’s IPC Activity</td>
<td>85</td>
</tr>
<tr>
<td>6.1</td>
<td>Distributed Scheduling Algorithm</td>
<td>92</td>
</tr>
<tr>
<td>6.2</td>
<td>Artificial Load for Distributed Scheduling Experiments</td>
<td>95</td>
</tr>
<tr>
<td>6.3</td>
<td>Average Execution Time v. Homogeneous Load</td>
<td>96</td>
</tr>
<tr>
<td>6.4</td>
<td>Average Execution Time v. Unequal System Load</td>
<td>97</td>
</tr>
<tr>
<td>6.5</td>
<td>Average Execution Time for Symmetrical Strategy v. System Load</td>
<td>98</td>
</tr>
<tr>
<td>6.6</td>
<td>Average Execution Time for Sender-Initiated Strategy v. System Load</td>
<td>98</td>
</tr>
<tr>
<td>6.9</td>
<td>Average Execution Time for Sender-Initiated Strategy v. XMM Load</td>
<td>100</td>
</tr>
<tr>
<td>6.10</td>
<td>Average Execution Time for Receiver-Initiated Strategy v. XMM Load</td>
<td>100</td>
</tr>
<tr>
<td>6.11</td>
<td>MPP Architecture</td>
<td>102</td>
</tr>
<tr>
<td>6.12</td>
<td>Average Execution Time as a Function of the Amount of IPC for Initial Placement and Task Migration</td>
<td>103</td>
</tr>
<tr>
<td>7.1</td>
<td>Average Execution Time for ALT with Dhrystone.</td>
<td>109</td>
</tr>
<tr>
<td>7.2</td>
<td>Average Execution Time for ALT with Simple Processing Code</td>
<td>109</td>
</tr>
<tr>
<td>7.3</td>
<td>Residual Dependency of Migrated Task</td>
<td>115</td>
</tr>
<tr>
<td>8.1</td>
<td>Load Distribution and PVM</td>
<td>119</td>
</tr>
</tbody>
</table>
List of Tables

5.1 Applications as a Function of Processing, IPC, and VM .................. 87
6.1 Comparison of Various Strategies ........................................... 101
7.1 Comparison of the LD Mechanisms on Various Architectures .......... 113
Preface

Aims

This book represents my PhD thesis, written and defended at the University of Kaiserslautern, Computer Science Department. Described work has a practical character. All ideas presented have been verified by implementation. The leitmotiv of the work is to provide a widely used, or at least available, load distribution system. Beside the technical merits, this was one of the main reasons for the selection of the Mach microkernel as the underlying environment. Mach has become widely used, which gives a hope that load distribution in such an environment may itself become widely used. Hopefully we have demonstrated that it is possible to provide a useful load distribution on top of the Mach microkernel.

This book is aimed for the researchers and students in the field of computer science, and operating systems in particular. It is intended for "practitioners", since the work has been colored with the implementation from the very beginning.

Described software has been provided to and used by a number of institutions in the world, such as: OSF Research Institute, Grenoble France, "Cluster project"; University of Utah, USA, "Schizo Project"; University of Louvain-la-Neuve, Belgium, "Oscar project"; Technion, Israel, teaching;

Summary

The book describes and evaluates a scheme for load distribution on top of microkernels (μkernels), and demonstrates its implementation for the Mach μkernel. As the first step, a transparent task migration is suggested as a base mechanism for load distribution. Its concepts and architecture are described and it is compared to process migration. The requirements to the underlying μkernel are discussed. Task migration is implemented for the Mach μkernel, in three different versions which have evolved during the course of the research presented in the book. The task migration facility is transparent, flexible and well performing. There are no limitations to its functionality, it supports all system calls a migrated task may issue, and there is no need to relink or recompile the application. Its simplicity and robustness stem from the μkernel interface, on which the load distribution scheme extensively relies, particularly on network Interprocess Communication (IPC) and distributed shared...
memory. However, some modifications to Mach were necessary in order to provide user space task migration.

Load information management is extended to consider information on Virtual Memory (VM) and IPC, beside traditionally supported processor load. Remote VM and IPC are particularly addressed, namely distributed shared memory and network IPC. The \( \mu \)kernel support for load information is outlined, and it is demonstrated in the case of Mach by instrumenting it to account for the information on the network IPC and distributed shared memory, and by providing it to the user with a new interface. Compared to earlier systems that had to consider various kinds of operating system activities, such as disk and other devices access, paging, different types of interprocess synchronization and communication, decisions in modern \( \mu \)kernels can be made only in terms of three main kernel activities: processing, VM, and IPC. This significantly simplifies characterization of applications and using this information for distributed scheduling.

A few well known distributed scheduling algorithms are implemented that beside processing also consider information on network communication. The appropriateness of task migration and the extended information management for load distribution is demonstrated empirically, and performance improvement is observed in the case of all algorithms. It is shown that the integration of three main load distribution elements (task migration, load information management, and distributed scheduling) leads to a harmoniously cooperating load distribution system concept.

During the course of this research some of the Mach deficiencies and performance drawbacks have been observed and some of them corrected. The book reports on the lessons learned while dealing with the Mach interface and on task migration relationship to process migration and the file system.

All features described in the book have been implemented as a part of the load distribution scheme. The task migration has been in use for a year and a half and was a base mechanism for various load distribution experiments. Information management and distributed scheduling have been in use for almost a year. Task migration has been ported to the OSF/1 operating system.

**Acknowledgements**

The work described in this thesis has been supported by the Deutsche Akademische Austausch Dienst (DAAD), the University of Kaiserslautern and by the Institute "Mihajlo Pupin", Belgrade.

I would like to thank a number of people whose contribution has been very important for bringing this thesis to the end. My advisor, professor Jiürgen Nehmer, enabled this project to live by providing me with his time, knowledge and money. He found enough understanding and support to prolong my one-year visit to University of Kaiserslautern into the PhD thesis, overcoming all possible political, administrative and economic issues.
My second program committee member, professor Friedemann Mattern, has improved my manuscript and was the source of very useful feedback on my work.

Professor Dušan Velašević from the Electrical Engineering Faculty, University of Belgrade, has given me initial support and encouragement for the described theme. I would like to thank my students Wolfgang Zint, Andreas Dangel, Peter Giese, Michael Umlauf and Stefan Anslinger. They have significantly contributed to the whole project. Wolfgang Zint has implemented the optimized migration server and IPC profiler. Besides, he co-designed the concept of task migration with me. Andreas Dangel worked with the distributed applications. He collected and profiled over sixty different applications. Peter Giese implemented artificial load for distributed scheduling experiments, as well as a number of scheduling strategies. He co-designed the distributed scheduling part of the project with me. Michael Umlauf has done some initial investigations on the process migration for OSF/1 server. Stefan Anslinger modified the existing load balancing simulation in order to reflect the Mach environment.

The Systems Software group at the University of Kaiserslautern was the source of continuous support. They provided a feedback to my work from the very beginning to the very end. They have improved my German and my presentations in many aspects.

I would like to thank CMU and OSF for providing us with Mach, as well as for the continuous support. The OSF Research Institute, Grenoble France provided to us the first Mach versions, and later the OSF/1 source code license as a part of our collaboration. Joe Barrera from CMU has given me important hints during development. His work on XMM and NORMA IPC was of particular significance for my own work. David Black, Alan Langerman, Steve Sears and many others from OSF Research Institute in Cambridge, as well as Philippe Bernadat, Erik Paire, Jose Rogado, Patrick Roudaud, and many others from OSF Research Institute in Grenoble always found enough time to discuss various issues with me. Jacques Febvre and James Loveluck from OSF Research Institute, Grenoble France, have initiated a collaboration with my University, which contributed to my work in many ways.

Beside the Mach community, I am indebted to the Sprite people. Fred Douglis, Brent Welch and Mike Kupfer have creatively criticized my work. I was not satisfied with my contributions until we found the common stand on monolithic v. microkernels issues and process v. task migration.

Professors Amnon Barak and Andrzej Goscinski continuously supported me in my ideas and my work. Their MOS(IX) and RHODOS systems were excellent examples and guide for my own work.

The following is an alphabetically ordered list of the reviewers of the three papers integrated into thesis. David Black, Henry Chang, Fred Douglis, Orly Kremien, Mike Kupfer, Alan Langerman, Reinhard Lüling, Simon Patience, Laurent Philippe, Bryan Rosenberg, Nikola Šerbedžija, Jelena Vučetić, Brent Welch and Roman Zajciew.

The cumbersome process of thesis reviewing has been performed by Radmilo Božinović, Peter Buhler, Peter Giese, Peter Sturm and the final touch has been
professionally put by Bill Bryant.

Beside me, the hardest part of this thesis has been carried out by my wife Maja and daughters Višnja and Milena. They had reasonable understanding for my overhours and travels.

Finally I would like to thank my Grandparents for raising me up, and teaching me many important lessons in life. I can never thank them enough for what they did for me. I dedicate this thesis to them.

Dejan Milojić, December 1993, Kaiserslautern.