
**Dynamic Resource Allocation
in Embedded, High-Performance
and Cloud Computing**

RIVER PUBLISHERS SERIES IN INFORMATION SCIENCE AND TECHNOLOGY

Series Editors

K. C. CHEN
National Taiwan University
Taipei, Taiwan

SANDEEP SHUKLA
Virginia Tech
USA

CHRISTOPHE BOBDA
University of Arkansas
USA

The “River Publishers Series in Information Science and Technology” covers research which ushers the 21st Century into an Internet and multimedia era. Multimedia means the theory and application of filtering, coding, estimating, analyzing, detecting and recognizing, synthesizing, classifying, recording, and reproducing signals by digital and/or analog devices or techniques, while the scope of “signal” includes audio, video, speech, image, musical, multimedia, data/content, geophysical, sonar/radar, bio/medical, sensation, etc. Networking suggests transportation of such multimedia contents among nodes in communication and/or computer networks, to facilitate the ultimate Internet.

Theory, technologies, protocols and standards, applications/services, practice and implementation of wired/wireless networking are all within the scope of this series. Based on network and communication science, we further extend the scope for 21st Century life through the knowledge in robotics, machine learning, embedded systems, cognitive science, pattern recognition, quantum/biological/molecular computation and information processing, biology, ecology, social science and economics, user behaviors and interface, and applications to health and society advance.

Books published in the series include research monographs, edited volumes, handbooks and textbooks. The books provide professionals, researchers, educators, and advanced students in the field with an invaluable insight into the latest research and developments.

Topics covered in the series include, but are by no means restricted to the following:

- Communication/Computer Networking Technologies and Applications
- Queuing Theory
- Optimization
- Operation Research
- Stochastic Processes
- Information Theory
- Multimedia/Speech/Video Processing
- Computation and Information Processing
- Machine Intelligence
- Cognitive Science and Brain Science
- Embedded Systems
- Computer Architectures
- Reconfigurable Computing
- Cyber Security

For a list of other books in this series, www.riverpublishers.com

Dynamic Resource Allocation in Embedded, High-Performance and Cloud Computing

Leandro Soares Indrusiak

Piotr Dziurzynski

Amit Kumar Singh


River Publishers

Published, sold and distributed by:

River Publishers
Alsbjergvej 10
9260 Gistrup
Denmark

River Publishers
Lange Geer 44
2611 PW Delft
The Netherlands

Tel.: +45369953197
www.riverpublishers.com

ISBN: 978-87-93519-08-4 (Hardback)
978-87-93519-07-7 (Ebook)

©2016 Leandro Soares Indrusiak, Piotr Dziurzanski and Amit Kumar Singh

Cover design by Andrea Monteiro and Leandro Soares Indrusiak

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, mechanical, photocopying, recording or otherwise, without prior written permission of the authors.

Contents

Preface	xi
Acknowledgements	xiii
List of Figures	xv
List of Tables	xix
List of Algorithms	xxi
List of Abbreviations	xxiii
1 Introduction	1
1.1 Application Domains	2
1.2 Related Work	4
1.2.1 Allocation Techniques for Guaranteed Performance	4
1.2.2 Allocation Techniques for Energy-efficiency	6
1.3 Challenges	7
1.3.1 Load Representation	7
1.3.2 Monitoring and Feedback	8
1.3.3 Allocation of Modal Applications	8
1.3.4 Distributed Allocation	9
1.3.5 Value-based Allocation	9
2 Load and Resource Models	11
2.1 Related Work	12
2.2 Requirements	13
2.2.1 Requirements on Modelling Load Structure	13
2.2.1.1 Singleton	14
2.2.1.2 Independent jobs	14

2.2.1.3	Single-dependency jobs	14
2.2.1.4	Communicating jobs	14
2.2.1.5	Multi-dependency jobs	14
2.2.2	Requirements on Modelling Load Temporal Behaviour	14
2.2.2.1	Single appearance	15
2.2.2.2	Strictly periodic	15
2.2.2.3	Sporadic	15
2.2.2.4	Aperiodic	15
2.2.2.5	Fully dependent	15
2.2.2.6	N out of M dependent	15
2.2.3	Requirements on Modelling Load Resourcing Constraints	15
2.2.3.1	Untyped job	16
2.2.3.2	Single-typed job	16
2.2.3.3	Multi-typed job	16
2.2.4	Requirements on Modelling Load Characterisation	16
2.2.4.1	Fixed load	16
2.2.4.2	Probabilistic load	16
2.2.4.3	Typed fixed load	16
2.2.4.4	Typed probabilistic load	16
2.3	An Interval Algebra for Load and Resource Modelling	17
2.3.1	Modelling Load Structure	19
2.3.2	Modelling Load Temporal Behaviour	19
2.3.3	Modelling Load Resourcing Constraints	20
2.3.4	Modelling Load Characterisation	22
2.3.5	Stochastic Time	22
2.4	Summary	23
3	Feedback-Based Admission Control Heuristics	25
3.1	System Model and Problem Formulation	26
3.1.1	Platform Model	26
3.1.2	Application Model	27
3.2	Distributed Feedback Control Real-Time Allocation	27
3.3	Experimental Results	29
3.3.1	Controller Tuning	29
3.3.2	Stress Tests	32
3.3.3	Random Workloads	34

3.4	Dynamic Voltage Frequency Scaling	35
3.5	Applying Controllers to Steer DVFS	37
3.6	Experimental Results	40
3.6.1	Controller Tuning	40
3.6.2	Random Workloads	42
3.7	Related Work	46
3.8	Summary	48
4	Feedback-Based Allocation and Optimisation	
	Heuristics	51
4.1	System Model and Problem Formulation	52
4.1.1	Application Model	53
4.1.2	Platform Model	53
4.1.3	Problem Formulation	54
4.2	Performing Runtime Admission Control and Load Balancing to Cope with Dynamic Workloads	54
4.3	Experimental Results	58
4.3.1	Number of Executed Tasks, Rejected Tasks and Schedulability Tests	60
4.3.1.1	Periodic workload	60
4.3.1.2	Random workload	62
4.3.2	Dynamic Slack, Setpoint and Controller Output	64
4.3.2.1	Periodic workload	64
4.3.2.2	Light workload	66
4.3.3	Core Utilization	66
4.3.4	Case Study: Industrial Workload Having Dependent Jobs	68
4.4	Related Work	70
4.5	Summary	72
5	Search-Based Heuristics for Modal Application	73
5.1	System Model and Problem Formulation	74
5.1.1	Application Model	74
5.1.2	Platform Model	75
5.1.3	Problem Formulation	77
5.2	Proposed Approach	78
5.2.1	Mode Detection/Clustering	78
5.2.2	Spanning Tree Construction	79
5.2.3	Static Mapping for Initial Mode	80

5.2.4	Static Mapping for Non-Initial Modes	82
5.2.5	Schedulability Analysis for Taskset During Mode Changes	84
5.2.6	On-Line Steps	90
5.3	Related Works	91
5.4	Summary	93
6	Swarm Intelligence Algorithms for Dynamic Task Reallocation	95
6.1	System Model and Problem Formulation	96
6.1.1	Load Model	96
6.1.2	Platform Model	98
6.1.3	Problem Statement	98
6.2	Swarm Intelligence for Resource Management	99
6.2.1	PS – Pheromone Signalling Algorithm	99
6.2.2	PSRM – Pheromone Signalling Supporting Load Remapping	102
6.3	Evaluation	108
6.3.1	Experiment Design	108
6.3.1.1	Metrics	109
6.3.1.2	Baseline Remapping Techniques	110
6.3.2	Experimental Results	110
6.3.2.1	Comparison between clustered approaches	110
6.3.2.2	Comparison regarding video processing performance	111
6.3.2.3	Comparison regarding communication overhead	112
6.3.2.4	Comparison regarding processor utilisation	113
6.3.3	Outlook	115
6.4	Summary	116
7	Value-Based Allocation	119
7.1	System Model and Problem Formulation	120
7.1.1	Many-Core HPC Platform Model	120
7.1.2	Job Model	121
7.1.3	Value Curve of a Job	121
7.1.4	Energy Consumption of a Job	122
7.1.5	Problem Formulation	122

7.2	The Solution	123
7.2.1	Profiling Based Approach (PBA)	123
7.2.2	Non-profiling Based Approach (NBA)	125
7.3	Evaluations	128
7.3.1	Experimental Baselines	129
7.3.2	Value and Energy Consumption at Different Arrival Rates	130
7.3.3	Value and Energy Consumption with Varying Number of Nodes	131
7.3.4	Value and Energy Consumption with Varying Number of Cores in Each Node	131
7.3.5	Percentage of Rejected Jobs	132
7.4	Related Works	133
7.5	Summary	134
	References	135
	About the Authors	151

Preface

The availability of many-core computing platforms enables a wide variety of technical solutions for systems across the embedded, high-performance and cloud computing domains. However, large scale many-core systems are notoriously hard to optimise. Choices regarding resource allocation alone can account for wide variability in timeliness and energy dissipation (up to several orders of magnitude). This book covers dynamic resource allocation heuristics for many-core systems, aiming to provide appropriate guarantees on performance and energy efficiency. It addresses different types of systems, aiming to harmonise the approaches to dynamic allocation across the complete spectrum between systems with little flexibility and strict performance guarantees all the way to highly flexible systems with soft performance guarantees.

Resource allocation is one of the most complex problems in large multiprocessor and distributed systems, and in general it is considered NP-hard. The theoretical evidence shows that the number of possible allocations of application tasks grows exponentially with the increase of the number of processing cores. The empirical evidence points in the same direction, with case studies showing that for a realistic multiprocessor embedded system (40–60 application components, 15–30 processing cores) a well-tuned search algorithm had to statically evaluate hundreds of thousands of distinct allocations before it finds one that meets the systems performance requirements.

In this book, we argue that the only way to cope with such complexity is to design systems that are capable to explore the allocation space during runtime. This is commonly done in cloud and high-performance computing, mainly because the workload of such systems cannot be accurately predicted in advance and static allocations are thus impossible. In embedded systems, the workload is more predictable in terms of its worst-case behaviour, but static allocations that take such characterisation into account tend to produce underutilised platforms. We therefore set the scene for dynamic resource allocation mechanisms by identifying and evaluating allocation heuristics that can be used to provide different levels of performance guarantees, and that cope with different levels of dynamism on the application workload.

The book starts with a description of the common practices and challenges in dynamic resource allocation, highlighting the peculiarities of each domain: embedded, HPC and cloud computing. Then, each of the challenges is addressed in detail within the following chapters, which are largely self-contained and therefore can be read in any order. To facilitate understanding, all of them follow the same structure: a specific challenge is motivated and the respective problem is precisely formulated; a detailed description of a solution to the problem is then given, followed by experimental work showing quantitative evidence of the strengths and weaknesses of that solution; related work is reviewed; and a summary of the chapter is given at the end.

The technical work that resulted in this book was done within the frame of the DreamCloud project, and the project website¹ makes available a number of reference implementations of the models and heuristics described here. Updates to this book will also be made available on that website.

Leandro Soares Indrusiak,
Piotr Dziurzanski,
and Amit Kumar Singh,

York, summer of 2016.

¹<http://www.dreamcloud-project.org>

Acknowledgements

The research work that resulted in this book was done within the frame of the DreamCloud project, funded by the European Commission under Framework Programme 7 (Ref. 611411). The authors would like to acknowledge and thank the commission for the funding, as well as the project officers and reviewers for their suggestions and feedback on the project outcomes.

The authors would like to thank Hashan Roshantha Mendis, whose work provided most of the foundations and the experimental results reported in Chapter 6.

The authors would also like to thank all the members of the DreamCloud consortium, particularly Scott Hansen, Björn Saballus, Devendra Rai, Manuel Selva, Abdoulaye Gamatié, Gilles Sassatelli, Luciano Copello Ost, Leonardo Zordan, David Novo, Alexey Cheptsov, Dennis Hoppe, Thomas Baumann, Fridtjof Siebert, Malek Ben Salen, Raj Patel, José Miguel Montañana and Neil Audsley.

List of Figures

Figure 1.1	Application domains and their characteristics with regard to dynamicity, resource utilisation and performance predictability.	3
Figure 2.1	Example of a probability mass function of a discrete random variable describing a job's execution time.	23
Figure 3.1	Block diagrams of control system architectures: feed-forward (<i>above</i>) and feedback (<i>below</i>).	26
Figure 3.2	Distributed feedback control real-time allocation architecture.	28
Figure 3.3	Maximum normalised task lateness (with execution time equal to 50,000 ns) in step responses for a number of tasks (3 clusters, 3 cores in each).	30
Figure 3.4	Maximum normalised task lateness step response for 500 tasks (with execution time equal to 50,000 ns) released at 5,000 ns (3 clusters, 3 cores in each).	31
Figure 3.5	Cores utilisation step response for 500 tasks (with execution time equal to 50,000 ns) released at 0 ns (1 cluster with 3 cores).	31
Figure 3.6	Core utilisation measured during the first 500 ns of the simulation.	33
Figure 3.7	Control signal observed during the first 500 ns of the simulation.	34
Figure 3.8	Distributed feedback control real-time allocation with DVFS architecture.	37
Figure 3.9	Pseudo-code of the proposed admission controller functionality.	39
Figure 3.10	Tasks executed before their deadline in random workload scenarios for DVFS with $\Gamma = 50$ ms (red), $\Gamma = 300$ ms (green) and $\Gamma = \infty$ (blue) – three processors with three cores (<i>top</i>) and two processors with four cores (<i>bottom</i>) systems.	43

Figure 3.11	Tasks rejected in random workload scenarios for DVFS with $\Gamma = 50$ ms (red), $\Gamma = 300$ ms (green) and $\Gamma = \infty$ (blue) – three processors with three cores (<i>top</i>) and two processors with four cores (<i>bottom</i>) systems.	44
Figure 3.12	Normalized dynamic energy dissipation in random workload scenarios for DVFS with $\Gamma = 50$ ms (red), $\Gamma = 300$ ms (green) and $\Gamma = \infty$ (blue) – three processors with three cores (<i>top</i>) and two processors with four cores (<i>bottom</i>) systems.	45
Figure 3.13	Normalized dynamic energy dissipation per task meeting its deadline in random workload scenarios for DVFS with $\Gamma = 50$ ms (red), $\Gamma = 300$ ms (green) and $\Gamma = \infty$ (blue).	46
Figure 4.1	Building blocks of the proposed approach.	53
Figure 4.2	A proposed many-core system architecture.	53
Figure 4.3	Illustration of task $\tau_{i,j}$ slack in three cases from Equation (4.1).	55
Figure 4.4	Number of executed tasks (<i>top</i>) and number of tasks rejected by the exact schedulability test (<i>bottom</i>) in closed-loop WCET, closed-loop ET and open-loop systems for the periodic task workload simulation scenario.	61
Figure 4.5	Number of tasks executed before their deadlines (<i>top</i>), the number of rejected tasks (<i>centre</i>) and number of the exact schedulability test executions (<i>bottom</i>) in baseline open-loop and proposed closed-loop ET systems for the random workloads simulation scenario with different weight of workloads.	63
Figure 4.6	Number of tasks executed before their deadlines (<i>top</i>), the number of rejected tasks (<i>centre</i>) and number of the exact schedulability test executions (<i>bottom</i>) in baseline open-loop and proposed closed-loop ET systems with different number of processing cores for the random workloads simulation scenario.	64

Figure 4.7	Dynamic slack (<i>top</i>), setpoint (<i>centre</i>) and controller output (<i>bottom</i>) during the simulation for the periodic task workload simulation scenario executed by a 3 core system.	65
Figure 4.8	Dynamic slack (<i>top</i>), setpoint (<i>centre</i>) and controller output (<i>bottom</i>) during the simulation for the selected light workload simulation scenario executed by a 3 core system.	67
Figure 4.9	Core utilisation during the simulation for the periodic (<i>top</i>) and light (<i>bottom</i>) workload simulation scenario with 3 core system.	68
Figure 4.10	Number of executed jobs (<i>top</i>) and number of schedulability test executions (<i>bottom</i>) for systems configured in four different ways for the industrial workloads simulation scenario.	69
Figure 5.1	Flow graph of the DemoCar example; the runnables belonging to the same task are highlighted with the same colour, labels are not highlighted. Some flows are drawn in different colours for readability.	76
Figure 5.2	Finite State Machine describing mode changes in DemoCar use case: before (<i>upper part</i>) and after (<i>lower part</i>) the clustering step.	77
Figure 5.3	An example many-core system platform.	77
Figure 5.4	Steps of dynamic resource allocation method benefiting from modal nature of applications.	78
Figure 5.5	Spanning tree construction for DemoCar.	80
Figure 5.6	Example of two different mappings (m_α, m_β) of runnables τ_i, τ_j, τ_k into cores π_a and π_b	85
Figure 5.7	Tasks' stages in DemoCar: green – runnable execution, red – write to labels; release times and deadlines are provided in ms.	86
Figure 6.1	System overview diagram.	97
Figure 6.2	PS pheromone propagation	102
Figure 6.3	Sequence diagram of PSRM algorithm related events. Time triggered (periodic): <i>PSDifferentiation</i> , <i>PSDecay</i> and <i>Remapping</i> cycles; Event triggered: <i>PSPropagation</i>	106

Figure 6.4	Task remapping example. (Q = queen nodes; D = Dispatcher; $[\tau_1, \tau_2]$ are late tasks; Blue lines represent communication.	107
Figure 6.5	Comparison of $CCPRM_{V1}$ (original) and $CCPRM_{V2}$ (improved). (a) Cumulative job lateness improvement. (b) Communication overhead.	111
Figure 6.6	Distribution of cumulative job lateness improvement after applying remapping.	112
Figure 6.7	Comparison of fully schedulable video streams for each remapping technique.	112
Figure 6.8	Communication overhead of the remapping approaches.	113
Figure 6.9	Comparison of PE utilisation for all remapping techniques. (a) Distribution of PE utilisation across a 10×10 NoC. (b) Histogram of PE busy time (normalised; 20 bins).	114
Figure 7.1	System model adopted in this chapter. A cloud data center containing different nodes (servers) with dedicated cores (PEs) to execute jobs submitted by multiple users.	120
Figure 7.2	An example job model and its value curve.	121
Figure 7.3	Profiling and non-profiling based approaches.	123
Figure 7.4	Voltage/frequency identification by FCPS and FTFS.	130
Figure 7.5	Value and energy at different arrival rates.	130
Figure 7.6	Value and energy with varying number of nodes.	131
Figure 7.7	Value and energy with varying number of cores at each node.	132

List of Tables

Table 3.1	Number of rejected tasks, tasks executed before and after their deadlines in various controlling environment configurations for a periodic task workload simulation scenario (3 clusters, 3 cores in each): configuration parameters (<i>above</i>) and obtained results (<i>below</i>)	32
Table 3.2	Total number of rejected tasks, tasks executed before and after their deadlines in various controlling environment configurations for 30 random bursty task workload simulation scenarios (3 clusters, 3 cores in each): configuration parameters (<i>above</i>) and obtained results (<i>below</i>)	34
Table 3.3	Total number of tasks executed before and after their deadlines and rejected tasks with various Υ threshold in the introductory experiment (1 processor with 3 cores)	42
Table 4.1	Average <i>Param</i> values for random workloads generated with different <i>range_min</i> and <i>range_max</i> parameters	60
Table 4.2	Four configuration possibilities with respect to controllers' output usage (OL and CL stands for open-loop and closed-loop, respectively)	69
Table 5.1	Number of hyperperiods (100 ms) required for switching between states <i>PowerUp</i> to <i>Cluster.1</i> in DemoCar depending on router (d_R) and one link latencies (d_L)	90
Table 6.1	PSRM algorithm parameters	108
Table 6.2	PE utilisation distribution statistics. Lower variance (var.) = better workload distribution	115
Table 7.1	Percentage of rejected jobs at different arrival rates	132

List of Algorithms

Algorithm 4.1	Pseudo-code of Admission controller involving DSE algorithm	57
Algorithm 5.1	Pseudo-code of no deadline violation with makespan minimisation algorithm for the initial mode mapping	81
Algorithm 5.2	Pseudo-code of a migration data transfer minimisation algorithm	83
Algorithm 6.1	PS Differentiation Cycle	100
Algorithm 6.2	PS Propagation Cycle	101
Algorithm 6.3	PS Decay Cycle	102
Algorithm 6.4	PSRM Differentiation Cycle	103
Algorithm 6.5	PSRM Remapping	105
Algorithm 7.1	Profiling Based Resource Allocation	125
Algorithm 7.2	Non-profiling Based Resource Allocation	126
Algorithm 7.3	Voltage/frequency Identification	127

List of Abbreviations

ACPI	Advanced Configuration and Power Interface
ALU	Arithmetic Logic Unit
AMIGO	Approximate M-constraint Integral Gain Optimization
AT	Arrival Time
AUTOSAR	Automotive Open System Architecture
BCET	Best Case Execution Time
CL	Closed Loop
CPU	Central Processing Unit
DAG	Directed Acyclic Graph
DBC	Deadline and Budget Constraints
DSE	Design Space Exploration
DSP	Digital Signal Processing
DVFS	Dynamic Voltage and Frequency Scaling
ECG	Electrocardiogram
ECU	Electronic Control Unit
EDF	Early Deadline First
ET	Execution Time
FCPS	Fixing Cores Power States
FIFO	First In First Out
flit	Flow control digIT
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
FTPS	Fixing Tasks Power States
GA	Genetic Algorithm
GoP	Group of Pictures
HLRS	High Performance Computing Center Stuttgart
HP	HPC Platform
HPC	High-Performance Computing
HRT	Hard Real-Time
IA	Interval Algebra

xxiv *List of Abbreviations*

IQR	Inter-Quartile Range
MMKP	Multi-Choice Knapsack Problem
MPSoC	Multiprocessor System-on-Chip
NBA	Non-profiling Based Approach
NoC	Network-on-Chip
OL	Open Loop
OS	Operating System
P	Proportional
PBA	Profiling Based Approach
PE	Processing Element
PG	Platform Graph
PI	Proportional-Integral
PID	Proportional-Integral-Derivative
PID-AC	PID-based Admission Control
PMF	Probability Mass Function
PS	Pheromone Signalling
QoS	Quality of Service
RM	Resource Manager
RMA	Rotating Mapping Algorithm
RTA	Response Time Analysis
RTM	Real-Time Manager
RTS	Real-Time Scheduling
SDF	Synchronous Dataflow
SoC	System-on-Chip
ST	Spanning Tree
TG	Task Graph
TLM	Transaction-Level Modelling
ValOpt	Value Optimization
VC	Value Curve
VM	Virtual Machine
VS	Voltage Scaling
WCET	Worst Case Execution Time
WCRT	Worst Case Response Time