

# History-Based Batch Job Scheduling on a Network of Interactively Used Workstations

**Inauguraldissertation**

zur

Erlangung der Würde eines Doktors der Philosophie

vorgelegt der

Philosophisch-Naturwissenschaftlichen Fakultät

der Universität Basel

von

Andreas Wespi

aus Schüpfheim, LU

Basel, 1999

Genehmigt von der Philosophisch-Naturwissenschaftlichen Fakultät auf  
Antrag der Herren

Prof. Dr. Helmar Burkhart

Prof. Dr. Clemens H. Cap

Basel, den 17. November 1998

Prof. Dr. Stefan M. Schmid, Dekan

Research Reports in Computer Science

Band 7

**Andreas Wespi**

**History-Based Batch Job Scheduling on a  
Network of Interactively Used Workstations**

Shaker Verlag  
Aachen 2000

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

*Wespi, Andreas:*

History-Based Batch Job Scheduling on a Network of  
Interactively Used Workstations / Andreas Wespi.

Aachen : Shaker, 2000

(Research Reports in Computer Science ; Bd. 7)

Zugl.: Basel, Univ., Diss., 2000

ISBN 3-8265-8295-0

Copyright Shaker Verlag 2000

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, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the publishers.

Printed in Germany.

ISBN 3-8265-8295-0

ISSN 1436-6967

Shaker Verlag GmbH • P.O. BOX 1290 • D-52013 Aachen

Phone: 0049/2407/9596-0 • Telefax: 0049/2407/9596-9

Internet: [www.shaker.de](http://www.shaker.de) • eMail: [info@shaker.de](mailto:info@shaker.de)

To Sarah, our beloved little baby,  
who stayed with us for only a short time.



# Acknowledgements

I am deeply indebted to my advisor Prof. Dr. Helmar Burkhart for his guidance, understanding, and support during the evolution of this thesis. I am also grateful to Prof. Dr. Clemens H. Cap, who kindly agreed to be my co-advisor, for his interest in my work and the animated discussions we had.

Many thanks to the IBM Zurich Research Laboratory, which gave me the opportunity to work on this thesis. My sincere thanks go particularly to Dr. Ernst Rothauser for his continuous help, guidance, encouragement, and the many hours of lively and fruitful discussions. I also thank my colleagues of the Information Systems group for their technical advice and understanding, especially when my data collection processes filled up too much disk space. I am indebted to Lilli Pavka for proofreading this text.

I am grateful to all those many people who contributed to this thesis with advice and expertise.

I thank my parents, who always supported me in good times and bad, and who taught me to love to learn. Most of all I thank my wife Rahel, who supported me with her understanding even though I often tried her patience working late in the evenings and on weekends.





# Abstract

Networks of workstations are the computing environment at many sites nowadays. Workstations are used for daily interactive work but also, owing to their excellent price/performance ratio, as dedicated computing servers. As it is known that many workstations sit idle in a network of interactively used workstations, approaches have been developed to make use of the idle resources by employing them for computation-intensive tasks.

Distributed Batch Job Scheduling Systems (DBJSSs) have been built that allow batch jobs to be run on interactively used workstations in a way that the impact on interactive users is kept to a minimum. Whereas batch job scheduling in a dedicated cluster of computing servers is well understood, scheduling in a network of interactively used workstations, due to the unpredictability of the available resources, is quite different and has only marginally been investigated in the past. As a consequence, either simple scheduling strategies are implemented in current DBJSSs or the system administrators are given the means to implement their own scheduling strategy without any further assistance.

I have analyzed the workstation utilization data of the distributed computing environment of the IBM Zurich Research Laboratory for more than one year. Based on a new approach, I have evaluated the impact of different definitions of the workstation idle state on the total amount of resources available for batch job processing. Furthermore, I could show that many workstations have a regular usage profile.

This observation has led to the development of novel, history-based batch job scheduling algorithms for networks of interactively used workstations. To quantify the performance benefits of the newly developed algorithms, the SIByL simulation environment has been built to analyze different batch job scheduling approaches under the same environment

conditions.

Simulation results show that history-based batch job scheduling algorithms outperform the traditional scheduling approaches implemented in many DBJSSs. The traditional approaches are based only on the state of the workstations at the time when the scheduling decision takes place, whereas history-based algorithms also consider the workstation usage in the past. For moderately loaded systems and environments where job migration is of limited use, the history-based algorithms result in mean job turnaround times that are about 30% shorter than the times obtained with the traditional scheduling approaches.

History-based batch job scheduling results in an increased amount of data to be processed to find a good job placement. I show how current DBJSSs can be easily enhanced so that they support history-based batch job scheduling without generating a lot of additional network load or additional computing load on the client workstations. Increased computing load is placed only on the server side. Furthermore, the SIByL simulation tool can be integrated in the DBJSS server such that, based on prior workstation usage data and job execution statistics, the best scheduling algorithm is selected adaptively out of a set of predefined algorithms.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	The Problem . . . . .	2
1.2	The Solution . . . . .	3
1.3	The New Idea . . . . .	4
1.4	Overview . . . . .	5
<b>2</b>	<b>Approaches to Better Workstation Utilization</b>	<b>7</b>
2.1	Terminology . . . . .	7
2.1.1	Workstation Cluster vs. Network of Workstations . . . . .	8
2.1.2	Static vs. Dynamic Load Distribution . . . . .	8
2.1.3	Load Sharing vs. Load Balancing . . . . .	8
2.1.4	Throughput, Response Time, and Fairness . . . . .	9
2.1.5	Idle vs. Available Workstations . . . . .	10
2.2	Approaches . . . . .	10
2.2.1	Distributed Operating System . . . . .	10
2.2.2	Processor Pool . . . . .	11
2.2.3	Parallel Program . . . . .	12
2.2.4	Piranha . . . . .	13
2.2.5	Stealth . . . . .	13
2.2.6	Process Migration . . . . .	14
2.2.7	Distributed Batch Job Scheduling System . . . . .	15
2.3	Comparison . . . . .	15
2.4	Summary . . . . .	17
<b>3</b>	<b>Distributed Batch Job Scheduling Systems</b>	<b>19</b>
3.1	Architectural Components of DBJSSs . . . . .	20
3.2	Properties of DBJSSs . . . . .	22
3.2.1	Scheduler-Related Properties . . . . .	22

3.2.2	General Properties . . . . .	23
3.3	Survey of DBJSSs . . . . .	25
3.3.1	Implementations of DBJSSs . . . . .	26
3.3.2	Job Types . . . . .	31
3.3.3	Attributes . . . . .	32
3.3.4	Queueing . . . . .	36
3.3.5	Fairness . . . . .	40
3.3.6	Placement . . . . .	42
3.3.7	Checkpointing and Migration . . . . .	45
3.3.8	Parallel Job Support . . . . .	47
3.4	Batch Job Scheduling on Interactively Used Workstations	50
3.4.1	Protecting the Workstation Owner . . . . .	50
3.4.2	Supporting the Batch Job Submitter . . . . .	52
3.5	Summary . . . . .	53
<b>4</b>	<b>Workstation Usage Patterns</b>	<b>55</b>
4.1	Motivation . . . . .	56
4.2	Environment . . . . .	56
4.3	Data Acquisition . . . . .	60
4.4	Usage Results . . . . .	62
4.4.1	How Long Does a Workstation Stay Idle After Having Become Idle? . . . . .	62
4.4.2	Load Average and Keyboard Idle Time . . . . .	64
4.4.3	Workstation Usage Profiles . . . . .	67
4.5	Summary . . . . .	74
<b>5</b>	<b>SIByL - Simulating Batch Loads</b>	<b>77</b>
5.1	Motivation . . . . .	78
5.2	SIByL . . . . .	80
5.3	Batch Load Modeling . . . . .	81
5.4	Performance Evaluation Criteria . . . . .	84
5.4.1	Job Measures . . . . .	84
5.4.2	Environment Measures . . . . .	86
5.5	Scheduling Algorithms . . . . .	87
5.5.1	Random Workstation Selection . . . . .	87
5.5.2	Standard Approach . . . . .	88
5.5.3	Availability Index Approach . . . . .	88
5.5.4	Usage Pattern Approach . . . . .	88
5.5.5	“Best Case” Approach . . . . .	90
5.6	Generalized Simulation Environment . . . . .	92

5.7	Summary . . . . .	93
<b>6</b>	<b>Simulation Results</b>	<b>95</b>
6.1	Setup . . . . .	95
6.2	Results . . . . .	96
6.2.1	System Load . . . . .	96
6.2.2	Mean Elapse Time . . . . .	99
6.2.3	CPU Performance . . . . .	100
6.2.4	Interrupt Performance . . . . .	103
6.3	Summary . . . . .	105
<b>7</b>	<b>A DBJSS Supporting History-Based Job Scheduling</b>	<b>107</b>
7.1	Design Principles . . . . .	107
7.2	Measuring the Availability Time . . . . .	108
7.3	History Data Manager . . . . .	110
7.4	Adaptively Selecting the Best Scheduling Algorithm . . . . .	111
7.5	Modifying the Condor DBJSS - A Case Study . . . . .	112
7.5.1	Client Modifications . . . . .	112
7.5.2	Server Modifications . . . . .	115
7.6	Summary . . . . .	118
<b>8</b>	<b>Conclusions and Outlook</b>	<b>119</b>
8.1	Conclusions . . . . .	119
8.2	Outlook . . . . .	121
<b>A</b>	<b>SIByL Manual Page</b>	<b>125</b>
	<b>Bibliography</b>	<b>131</b>
	<b>Index</b>	<b>143</b>



# List of Figures

3.1	Components of a generic DBJSS architecture . . . . .	20
3.2	Distributed batch job scheduling system ancestry . . . . .	25
3.3	LSF system architecture . . . . .	29
4.1	Increase in number of workstations and overall performance of the ZRL network of workstations (1994) . . . . .	59
4.2	Mean performance of a workstation in the ZRL cluster . . . . .	60
4.3	Probability of remaining idle: $P(t > x + d \mid t = x)$ . . . . .	63
4.4	Best versus worst case of pattern classification . . . . .	69
4.5	$rel\_nb\_ws_1$ on non-working days . . . . .	70
4.6	$rel\_nb\_ws_3$ on non-working days . . . . .	71
4.7	$rel\_nb\_ws_1$ on working days . . . . .	72
4.8	$rel\_nb\_ws_3$ on working days . . . . .	73
5.1	The SIByL simulation environment . . . . .	81
5.2	Distribution of job submit times . . . . .	82
5.3	Relative distribution of job submit times . . . . .	84
5.4	Job execution parameters . . . . .	85
5.5	Availability index algorithm . . . . .	89
5.6	Usage pattern algorithm . . . . .	91
6.1	Total vs. available number of workstations for (a) $\lambda = 1$ and $\mu = 1$ and (b) $\lambda = 1$ and $\mu = 20$ . . . . .	97
6.2	Total vs. available number of workstations for (a) $\lambda = 5$ and $\mu = 1$ and (b) $\lambda = 5$ and $\mu = 20$ . . . . .	98
6.3	Total vs. available number of workstations for (a) $\lambda = 10$ and $\mu = 1$ and (b) $\lambda = 10$ and $\mu = 20$ . . . . .	98
6.4	Total vs. available number of workstations for (a) $\lambda = 20$ and $\mu = 1$ and (b) $\lambda = 20$ and $\mu = 20$ . . . . .	98

6.5	Weekly mean elapse time of the five examined algorithms for (a) $\lambda = 20$ and $\mu = 1$ and (b) $\lambda = 20$ and $\mu = 20$ . . .	101
6.6	Weekly CPU performance of the five examined algorithms for (a) $\lambda = 20$ and $\mu = 1$ and (b) $\lambda = 20$ and $\mu = 20$ . . .	102
6.7	Weekly interrupt performance of the five examined algo- rithms for (a) $\lambda = 20$ and $\mu = 1$ and (b) $\lambda = 20$ and $\mu = 20$ . . . . .	104
7.1	A DBJSS supporting history-based job scheduling . . . .	109



# List of Tables

2.1	Approaches to better workstation utilization . . . . .	16
3.1	Job types . . . . .	32
3.2	Static workstation, queue, and job attributes . . . . .	35
3.3	Dynamic workstation, queue, and job attributes . . . . .	37
3.4	Queueing system properties . . . . .	39
3.5	Methods to implement fairness . . . . .	41
3.6	Attributes used by the fairness methods . . . . .	43
3.7	Placement approaches . . . . .	45
3.8	Criteria to sort candidate workstations . . . . .	46
3.9	Checkpointing approaches . . . . .	48
3.10	Parallel batch job support . . . . .	49
4.1	Relative performance and number of workstations . . . . .	58
4.2	Workstation availability on holidays . . . . .	66
4.3	Workstation availability during business hours . . . . .	66
4.4	Workstation availability during off hours . . . . .	66
4.5	Mean workstation availability . . . . .	67
5.1	Hourly job arrival rates . . . . .	83
6.1	Number of jobs per job set . . . . .	96
6.2	<i>idle_ws</i> per algorithm and job set . . . . .	99
6.3	Mean elapse time per algorithm and job set . . . . .	99
6.4	CPU performance per algorithm and job set . . . . .	103
6.5	Interrupt performance per algorithm and job set . . . . .	103