Distinguished Dissertations

Springer
London
Berlin
Heidelberg
New York
Barcelona
Hong Kong
Milan
Paris
Singapore
Tokyo

Other titles published in this Series:

Extensional Constructs in Intensional Type Theory
Martin Hoffman

Search and Planning Under Incomplete Information: A Study Using Bridge Card Play
Ian Frank

Theorem Proving with the Real Numbers
John Harrison

Games and Full Abstraction for a Functional Metalanguage with Recursive Types
Guy McCusker

Hardware Evolution: Automatic Design of Electronic Circuits in Reconfigurable Hardware by Artificial Evolution
Adrian Thompson

Models of Sharing Graphs: A Categorical Semantics of **let** *and* **letrec**
Masahito Hasegawa

Large Scale Collaborative Virtual Environments
Chris Greenhalgh

Radu C. Calinescu

Architecture-Independent Loop Parallelisation

Springer

Radu C. Calinescu, DPhil, MSc
Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, OX1 3QD, UK

Series Editor
Professor C.J. van Rijsbergen
Department of Computing Science, University of Glasgow, G12 8RZ, UK

ISSN 1439-9768

ISBN-13: 978-1-4471-1197-9 Springer-Verlag London Berlin Heidelberg

British Library Cataloguing in Publication Data
Calinescu, Radu C.
Architecture-independent loop parallelisation.-
(Distinguished dissertations)
1.Parallel processing (Electronic computers)
I.Title
004.3'5
ISBN-13: 978-1-4471-1197-9

Library of Congress Cataloging-in-Publication Data
Calinescu, Radu., 1968-
Architecture-independent loop parallelisation / Radu C. Calinescu.
p. cm. -- (Distinguished dissertations)
Includes bibliographical references and index.

ISBN-13: 978-1-4471-1197-9 e-ISBN-13: 978-1-4471-0763-7
DOI: 10.1007/978-1-4471-0763-7

1. Parallel processing (Computer science) 2. Computer architecture. I. Title. II. Distinguished dissertations (Springer-Verlag)
QA76.58.C34 2000
004'.35--dc21 00-037369

Softcover reprint of the hardcover 1st edition 2000

Typesetting: Camera-ready by author

34/3830-543210 Printed on acid-free paper SPIN 10760296

Preface

This book addresses the automatic parallelisation of regular loop computations involving dense data structures. In order to achieve parallel code which is architecture-independent, scalable and of analytically predictable performance, scheduling in the bulk-synchronous parallel model of computation is considered.

Our parallelisation approach combines two types of scheduling in a novel way. A class of parallelisation techniques termed template-matching scheduling is used to build the parallel version of certain loop computations starting from predefined, highly optimised schedule skeletons. A more complicated technique called generic loop nest scheduling tackles the parallelisation of nested loops whose structure matches none of the recognised computation templates.

A collection of template-matching parallelisation methods is developed in the book. This collection builds on recent advances in automatic parallelisation and architecture-independent parallel programming, and includes two categories of scheduling techniques. The subset of techniques belonging to the first category is dedicated to the parallelisation of uniform-dependence perfect loop nests. The second category of techniques addresses the parallelisation of several loop constructs that appear frequently in imperative programs and comprise non-uniform dependences.

We also introduce a new scheme for the parallelisation of generic, untightly nested loops. This scheme comprises four steps: data dependence analysis, potential parallelism identification, data and computation partitioning, and communication and synchronisation generation. Due to the new algorithms employed in its last three steps, the scheme is able to identify coarse-grained potential parallelism, and to map it efficiently on the processor/memory units of a general purpose parallel computer.

The effectiveness of architecture-independent loop parallelisation is assessed through a series of case studies addressing the parallelisation of several scientific computing problems. For each problem, the best known parallel solution is compared with the one obtained using the automatic scheduling techniques, as well as with the parallel schedule generated by a research tool that implements a subset of these techniques. This study reveals that the new parallelisation approach is feasible, and can be successfully applied to many scientific computations involving dense data structures.

Except for a few minor corrections, this book represents the author's University of Oxford D.Phil. thesis.

Acknowledgements

I would like to express my gratitude to Bill McColl, my supervisor; he has provided invaluable advice and encouragement throughout the development of this thesis. I am also indebted to Alex Gerbessiotis, Constantinos Siniolakis, Stephen Donaldson, Alexandre Tiskin, Fabrizio Petrini, Jonathan Hill, Ronald Sujithan and all other current and former members of the Oxford BSP group for their comments on my work.

Special thanks are due to David J. Evans, Mike Giles, and Dan Stefanescu for their suggestions at various moments during my D.Phil. course, and to Stephen Donaldson and Tom Costello for proof-reading parts of the thesis.

Gaétan Hains and Fabrizio Petrini kindly accepted to examine this work, their valuable comments having led to several improvements in the final version of the thesis. I am also grateful to my examiners for suggesting that my thesis should be submitted to the Distinguished Dissertation competition.

To a great extent, this thesis represents the result of over twenty years of formal education in my life. I owe my success over all these years to many remarkable teachers. I am especially indebted to Octav Pastravanu for introducing me to the challenging world of research.

I am deeply grateful to my parents, Rodica and Dumitru, and to my wife, Ani, for their love, support and understanding; and to little Maria for reminding me that there is much more to life than computer science.

This work was funded by an Overseas Research Studentship, a Dulverton Scholarship, and an Oxford Overseas Bursary.

Contents

Glossary of Notations

Arithmetic

$\lvert x \rvert$	the absolute value of x
$\lceil x \rceil$	the ceiling of x ($y \in \mathbf{Z}$ such that $y - 1 < x \leq y$)
$\lfloor x \rfloor$	the floor of x ($y \in \mathbf{Z}$ such that $y \leq x < y + 1$)
$\gcd(x_1, x_2, \ldots, x_n)$	the greatest common divisor of $x_1, x_2, \ldots, x_n$
$\min\{x_1, x_2, \ldots, x_n\}$	the minimum of $x_1, x_2, \ldots, x_n$
$\max\{x_1, x_2, \ldots, x_n\}$	the maximum of $x_1, x_2, \ldots, x_n$
$x \bmod y$	the remainder of the integer division of x by y

Sets

$\{\}$	the empty set
$\{x_1, x_2, \ldots, x_n\}$	the set containing elements $x_1, x_2, \ldots, x_n$
$\#X$	the number of elements in set X
$A, B, C, \ldots$	sets
$\mathbf{N}$	the set of natural numbers ($\{0, 1, 2, \ldots\}$)
$\mathbf{Z}$	the set of integer numbers ($\{\ldots, -2, -1, 0, 1, 2, \ldots\}$)
$x \in X$	set membership
$\{x \in T \mid P(x)\}$	set comprehension (the set of all x in T such that $P(x)$ holds)
$\mathbf{P}A$	the powerset of A ($\{X \mid X \subseteq A\}$)
$x..y$	the set $\{k \in \mathbf{Z} \mid x \leq k \leq y\}$, where $x, y \in \mathbf{Z}$
$X \subseteq Y$	set inclusion ($\forall x \in X \bullet x \in Y$)
$X \setminus Y$	set difference ($\{x \in X \mid x \notin Y\}$)
$X \cup Y$	set union ($\{x \mid x \in X \vee x \in Y\}$)
$X \cap Y$	set intersection ($\{x \in X \mid x \in Y\}$)
$X \times Y$	the Cartesian product of sets X and Y ($\{(x, y) \mid x \in X \wedge y \in Y\}$)

Logic

$\neg x$	negation (not x)
$x \vee y$	disjunction (x or y)
$x \wedge y$	conjunction (x and y)

$x \Rightarrow y$	implication (if x, then y)
$\forall x \in X \bullet pred$	universal quantification ($pred$ holds for all $x \in X$)
$\exists x \in X \bullet pred$	existential quantification ($pred$ holds for at least one $x \in X$)

Linear Algebra

$\mathsf{A}, \mathsf{B}, \mathsf{C}, \ldots$	matrices
$\mathsf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$	matrix A has columns $\mathbf{a}_1$, $\mathbf{a}_2$, ..., $\mathbf{a}_n$
$\mathsf{A} = [a_{i,j}]$	matrix A has elements $a_{i,j}$
A^{T}	the transpose of matrix A
$\det \mathsf{A}$	the determinant of matrix A
$\operatorname{rank} \mathsf{A}$	the rank of matrix A
I_n	the $n \times n$ identity matrix
$\mathbf{u}, \mathbf{v}, \mathbf{x}, \ldots$	vectors
$\mathbf{x} = [x_1, x_2, \ldots, x_n]^{\mathrm{T}}$	the n-dimensional vector $\mathbf{x}$ has elements $x_1, x_2, \ldots, x_n$
$(x_1, x_2, \ldots, x_n)$	the point of coordinates $x_1, x_2, \ldots, x_n$ in an n-dimensional space
$\operatorname{diag}(x_1, x_2, \ldots, x_n)$	the matrix $\mathsf{I}_n [x_1, x_2, \ldots, x_K]^{\mathrm{T}}$
$\operatorname{span}\{\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^n\}$	the vector space spanned by the vectors $\mathbf{v}^1$, $\mathbf{v}^2$, ..., $\mathbf{v}^n$
$\mathbf{N}^x$	the vector space of x-dimensional vectors with natural elements

Asymptotic Notation

$\mathrm{O}(f(n))$	$\{g(n) \mid \exists (c > 0, n_0 > 0) \bullet \forall n \geq n_0 \bullet g(n) \leq c f(n)\}$
$\mathrm{o}(f(n))$	$\{g(n) \mid \forall c > 0 \bullet \exists n_0 > 0 \bullet \forall n \geq n_0 \bullet g(n) \leq c f(n)\}$
$\Omega(f(n))$	$\{g(n) \mid \exists (c > 0, n_0 > 0) \bullet \forall n \geq n_0 \bullet c f(n) \leq g(n)\}$
$\omega(f(n))$	$\{g(n) \mid \forall c > 0 \bullet \exists n_0 > 0 \bullet \forall n \geq n_0 \bullet c f(n) \leq g(n)\}$
$\theta(f(n))$	$\{g(n) \mid \exists (c_1 > 0, c_2 > 0, n_0 > 0) \bullet \forall n \geq n_0 \bullet c_1 f(n) \leq g(n) \leq c_2 f(n)\}$

Automatic Parallelisation and the BSP Model

$a, b, c, \ldots$	arrays
$\mathbf{d}, \mathbf{d}^1, \mathbf{d}^2, \mathbf{d}^3, \ldots$	distance vectors
$\mathrm{footprint}(a)$	the footprint of array a
g	the BSP communication parameter
$\mathcal{G}$	dependence graph
$i, i_1, i_2, i_3, \ldots$	loop indices
$\mathbf{i}$	the index vector of a perfect loop nest ($\mathbf{i} = [i_1, i_2, \ldots, i_K]^{\mathrm{T}}$) or an iteration point of a perfect loop nest ($\mathbf{i} = (i_1, i_2, \ldots, i_K)$)
I	the iteration space of a perfect loop nest
K	the number of loops in a perfect loop nest
$l, l_1, l_2, l_3, \ldots$	loops in a computer program
L	the BSP synchronisation parameter
$\mathcal{L}$	loop nest

p	the number of processor/memory units of a BSP computer
$S, S_1, S_2, S_3, \ldots$	statements in a computer program
$S_1 \delta S_2$	flow data dependence between statements S_1 and S_2
$S_1 \bar{\delta} S_2$	data antidependence between statements S_1 and S_2
$S_1 \delta^{\circ} S_2$	output data dependence between statements S_1 and S_2
$S_1 \delta^{*} S_2$	generic data dependence between statements S_1 and S_2
$\mathbf{v}, \mathbf{v}^1, \mathbf{v}^2, \mathbf{v}^3, \ldots$	direction vectors

List of Figures