UNICOM

APPLIED INFORMATION TECHNOLOGY 13

Parallel Processing and Data Management

Edited by P. Valduriez

CHAPMAN & HALL

Parallel Processing and Data Management

UNICOM

APPLIED INFORMATION TECHNOLOGY 13

Edited by **P. Valduriez**
Director of Research, INRIA, France

CHAPMAN & HALL
London · New York · Tokyo · Melbourne · Madras

Published by Chapman & Hall, 2–6 Boundary Row, London SE1 8HN

Chapman & Hall, 2–6 Boundary Row, London SE1 8HN, UK

Blackie Academic & Professional, Wester Cleddens Road, Bishopbriggs,
Glasgow G64 2NZ, UK

Van Nostrand Reinhold Inc., 115 5th Avenue, New York NY10003, USA

Chapman & Hall Japan, Thomson Publishing Japan, Hirakawacho Nemoto
Building, 6F, 1–7–11 Hirakawa-cho, Chiyoda-ku, Tokyo 102, Japan

Chapman & Hall Australia, Thomas Nelson Australia, 102 Dodds Street,
South Melbourne, Victoria 3205, Australia

Chapman & Hall India, R. Seshadri, 32 Second Main Road, CIT East,
Madras 600 035, India

First edition 1992

Printed in Great Britain by T J Press (Padstow) Ltd, Cornwall

ISBN 0 412 42800 8   0 442 31580 5 (USA)

The publisher makes no representation, express or implied, with regard to the accuracy of the information contained in this book and cannot accept any legal responsibility or liability for any errors or omissions that may be made.

A catalogue record for this book is available from the British Library

Library of Congress Cataloging-in-Publication data available

Parallel Processing and Data Management

UNICOM Applied Information Technology

Each book in the series is based upon papers given at a seminar organized by UNICOM Seminars Ltd. The reports cover subjects at the forefront of information technology, and the contributors are all authorities in the subject on which they are invited to write, either as researchers or as practitioners.

1 **Fourth-Generation Systems**
Their scope, application and methods of evaluation
Edited by Simon Holloway

2 **Evaluating Supercomputers**
Strategies for exploiting, evaluating and benchmarking
Edited by A. van der Steen

3 **Failsafe Control Systems**
Applications and emergency management
Edited by Kevin Warwick and Ming T. Tham

4 **Computer Vision and Image Processing**
Edited by Anthony Barrett

5 **The Distributed Development Environment**
The art of using CASE
Edited by Simon Holloway

6 **Software Quality and Reliability**
Tools and methods
Edited by Darrel Ince

7 **Open Systems for Europe**
Edited by Tony Elliman and Colston Sanger

8 **Hypermedia/Hypertext**
And Object-oriented Databases
Edited by Heather Brown

9 **Software for Parallel Computers**
Edited by R. H. Perrott

10 **Object-oriented Programming Systems**
Tools and applications
Edited by J. J. Florentin

11 **Object-oriented Design**
Edited by Peter Robinson

12 **Software Reuse and Reverse Engineering in Practice**
Edited by P. A. V. Hall

13 **Parallel Processing and Data Management**
Edited by P. Valduriez

14 **Creating a Business-based IT strategy**
Edited by A. Brown

CONTRIBUTORS

D. R. Audley and A. E. Terrano
Financial Strategies Group
Prudential-Bache Securities
Prudential-Bache Building
New York
NY 10292
USA

P. Bryant
Stratus Computer Ltd
Central House
Lampton Rd
Hounslow
Middlesex
UK

C. Chachaty, P. Borla-Salamet and B. Bergsten
BULL Corporate Research Center
Avenue Jean Jaurès
78340 Les Clayes Sous Bois
France

J. P. Cheiney and Y. N. Huang
Ecole Nationale Supérieure des Telecommunications
46 Rue Barrault
74634 France

A. Davison
The Parlog Group
Dept of Computing
Imperial College
180 Queen's Gate
London
UK

D. DeGroot, E. Meyer and D. Wells
Computer Science Centre
Texas Instruments
Mailstation 238
Dallas
Texas
USA

T. C. Fogarty, J. G. Gammack, S. A. Battle and R. G. Miles
The Transputer Centre
Bristol Polytechnic
Coldharbour Lane
Bristol
UK

M. Freeston
European Computer-Industry Research Centre
Arabellastrasse 17
D-8000 Munich 81
Germany

O. Gruber
INRIA-Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France

G. F. J. Handley
Cray Research (UK) Ltd
Bracknell
Berkshire
UK

G. Haworth
ICL
Kings House
33 Kings Rd
Reading
Berkshire
UK

S. Jakobek
Meiko Ltd
650 Aztec West
Bristol
UK

R. G. Johnson, N. J. Martin, T. Wu and X. Zhao
Dept of Computer Science
Birbeck College
University of London
London
UK

J. Kerridge
Dept of Computer Science
University of Sheffield
Portobello Centre
Sheffield
UK

S. Lavington
Dept of Computer Science
University of Essex
Wivenhoe Park
Colchester
Essex
UK

P. A. Lee
Centre for Multiprocessors
Computing Laboratory
University of Newcastle upon Tyne
Claremont Rd
Newcastle upon Tyne
UK

J. C. Manley
Hewlett-Packard Laboratories
Filton Rd
Stoke Gifford
Bristol
UK

C. de Maindreville
INRIA-Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France

D. Moody
Intel Scientific Computers
Pipers Way
Swindon
Wiltshire
UK

M. Paci
European Computer-Industry Research Centre GmbH
Arabellastrasse 17
D-8000 Munich 81
Germany

J. Page
Teradata UK Ltd
Alwyn House
31 Windsor St
Chertsey
Surrey
UK

V. F. Rich
Encore Computer (UK) Ltd
Marlborough House
Mole Business Park
Leatherhead
Surrey
UK

J. Spiers
Oracle Corporation UK Ltd
Oracle Centre
The Ring
Bracknell
Berkshire
UK

K. Steer
Consultant to Strand Software
Technologies Inc.

P. Valduriez
INRIA-Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France

K-F. Wong
European Computer-Industry
Research Centre GmbH
Arabellastrasse 17
D-8000 Munich 81
Germany

CONTENTS

Contributors vii

Part One Practical experiences with parallel database 1

1 Information flow in an enterprise 3
S. Jakobek

2 Computational finance using the Intel iPSC family of concurrent supercomputers 13
D. Moody

3 Parallel processing in the securities industry interactive supercomputing 23
D. R. Audley and A. E. Terrano

Part Two Parallel database servers 31

4 High performance database for client/server systems 33
J. Page

5 Mapping a 4th generation DBMS to a range of parallel machines 53
J. Spiers

Part Three New architectures for parallel database systems 69

6 Developing concurrent databases using distributed objects 71
K. Steer

7 The development of advanced databases exploiting parallel transputers 93
R. G. Johnson, N. J. Martin, T. Wu, X. Zhao

8 Transputer topologies for data management 107
J. Kerridge

9 The Cray Research architecture for high performance applications and data management 121
G. F. J. Handley

Part Four Concurrency and fault-tolerance in parallel systems 127

10 Concurrency within the Mach operating system and its use in DB server applications 129
V. F. Rich

11 Multiprocessor system for performance and fault tolerance 135
P. A. Lee

12 Parallel processing, fault tolerance and open systems 147
P. Bryant

Part Five Parallel knowledge base systems 165

13 KbProlog: a language for knowledge management 167
J. C. Manley

14 A parallel platform for a KBMS 179
M. Freeston

15 Intelligent decision support using on-line databases 187
T. C. Fogarty, J. G. Gammack, S. A. Battle, R. G. Miles

Part Six Parallel data processing 193

16 Issues in parallelizing object oriented database systems 195
D. DeGroot, E. Meyer, D. Wells

17 Object oriented databases in Polka 207
A. Davison

18 Parallel processing of AI structures 225
S. Lavington

19 Cyclic computation of direct transitive closures in a shared-nothing architecture 239
J. P. Cheiney, Y. N. Huang, C. de Maindreville

Part Seven The EDS parallel database server 259

20 Information servers, present and future 261
G. Haworth

21 Object management in parallel database servers 275
O. Gruber, P. Valduriez

22 Capturing parallel data processing strategies within a compiled language 295
C. Chachaty, P. Borla-Salamet, B. Bergsten

23 Performance evaluation of an OLTP application on the EDS database server using a behavioural simulation model 317
K-F. Wong and M. Paci

Part One

Practical experiences with parallel database

1 Information flow in an enterprise

S. Jakobek
Meiko

True information flow in an enterprise – making both applications and data available wherever they are needed in a timely fashion – can only be achieved when we are able to distribute the information processing load across the enterprise.

The major issues involved in accomplishing this are: the problems of balancing the transaction processing and decision support load across a machine or a network of machines such that neither impacts the performance of the other; and the problems of streamlining networking and communications so that this load-balancing exercise is transparent to the user.

This presentation will consider these issues in more detail and propose solutions based upon our experience of similar problems encountered when designing and developing the technology for the new distributed memory parallel computers.

The distributed memory 'message passing' model, where hundreds and even thousands of processing nodes are loosely coupled to form one or more machines, owes its very existence to the fact that the inter-processor communications issues have been addressed, making it possible to distribute and balance processing load across many 'machines', and to ensure that data is delivered to these applications in a timely and effective manner.

We will describe how these parallel processing techniques can be applied in the context of Information Technology solutions and how the ORACLE user can now take advantage of them.

Research into changing business practices and new organizational trends has shown that the enterprises most likely to succeed in the 90s are the ones which have a fluid management structure and flexible procedures. If these companies are to fulfill their potential then this more adaptive style of working must also be reflected in their IT systems. We believe that the IT solutions of the 90s will be based upon an entirely new type of machine architecture.

Computer architects generally agree that the use of the classical Von Neuman approach to designing a uni-processor is now well understood, and they are capable of exploiting today's component technology to its limits. The way to improve computer performance with given technology and to improve the performance of computers beyond the limits of available technology involves the introduction of new architectures.

Specialized approaches have been used before, such as vector processing and pipelining, to create machines which are particularly good at scientific calculations. However, these are not general-purpose machines, being mainly deployed for numerically intensive problem solving.

The most hopeful means of achieving a scalable, yet general-purpose computing resource, is to use multiple processing elements within single machines. One approach is to link a number of processing elements all simultaneously executing an identical program but on different parts of the data. This is the SIMD (Single Instruction and Multiple Data) approach. This yields a very special purpose machine although in the IT context SIMD machines have only been successfully deployed in specialist applications such as text searching.

The alternative is to use multiple processing elements executing fragments of a program independently of each other. This is the MIMD approach (Multiple Instruction and Multiple Data). With the MIMD approach there is a further immediate choice available and this is between multiple processing elements sharing access to a single memory system, versus multiple processing elements each of which has its own dedicated memory system.

Superficially the shared memory system appears to be the more attractive. It is obviously simpler to imagine applications where individual processing elements get on with part of a task, but with reference to common data. The disadvantage of this approach is that each individual processor has to access the common data. The effect is that scalability of such a machine is limited by the speed of the memory access mechanism, and the speed of the memory itself. The memory system is therefore a shared resource and the architecture does not address the issue of scaling the size of the resource proportionally as the number of processing elements increases.

Consider the analogy of a busy office, where the tasks involve each worker in frequently accessing a single filing cabinet for filing and for reference to documents. As the office workload increases and as more workers are employed, it is clear that at some point the demands upon the filing system will cause workers to queue at the filing cabinet. There is clearly a finite level of work that this office can support, and this illustrates the limitations of the shared memory approach to computer architecture.

The alternative approach is the distributed memory machine, in which each processor has its own memory system and enjoys unimpeded access to that memory. This has the obvious benefit of allowing the performance of both the aggregate machine and the memory system to be proportional to the number of processing elements employed.

The disadvantage inherent in this system is that, in order to address a single problem, processing elements need to co-operate in passing information between each other. The problem presented to the computer architect in designing a distributed memory MIMD machine is to discover a communications medium that can increase its performance in proportion to the number of processing elements employed, while still allowing the processing elements to co-operate.

There exists a good analogy to this approach in the way in which organizations work, where an organization can grow in its capacity to deal with problems by employing more people, each of whom is capable of thinking and remembering and dealing with problems individually but who co-operate by communicating with one another. This is clearly acceptable within the context of a meeting room or a one-on-one meeting, but is also supported in larger communities by the telephone. The telephone is a good example of a communications resource which can be scaled in size and capacity in response to increasing demands from a growing working community. The essence of a distributed memory MIMD machine is therefore one in which there are many processing elements, each with their own memory, an inter-processor communications resource which allows those processing elements to co-operate on single problems by sending messages to one another, and where this communications resource is scalable, just like the telephone system.

With this model of computer architecture, there is no limit to the size of machine that can be produced. More and more processing elements can be added to deliver more and more computing power. As the size of the 'telephone network' increases, so more and more internal co-operation resource is also added. The only drawback with this kind of architecture is that of finding problems which are large enough to exploit such machines. I think that we will be able to demonstrate that the ORACLE user community presents the kind of problems which will justify the use of this new breed of machine.

For an Information Technology system to support the operation of an enterprise, there are three key issues which must be addressed.

- The IT solution must be capable of addressing the workload of the enterprise as characterized by the need to enter information, enquire on that information and provide management information in the form of consolidations and reports.

- The IT solution must be able to accommodate growth.
- The IT solution must be flexible so that new functions can be integrated, and changes accommodated in both user and structural requirements.

With a conventional computer system, i.e. a uni-processor environment, the machine is generally sized to support the workload. In reality, however, the machine is often undersized and does not support the workload! This situation arises because it is difficult to predict system loading given the dynamic nature of machine usage, making it hard to deliver an adequate service to users.

When employing a conventional computer system, a number of trade-offs need to be made. A compromise is made between the number of users that the system can support and the response which each user can expect. This is necessary to achieve a balance between the operating system burden imposed by users and the application burden which each of those users presents. *Ad hoc* query and reporting activities also create an unpredictable loading on the system. Usage of these functions is often restricted through system management procedures so that the load does not adversely impinge on any on-line users. Such trade-offs must be made in order that the workload across the enterprise can be supported.

The attraction of a multi-processor computer is that various parts of the total workload can be partitioned and allocated to one or more dedicated processing elements. The number of processing elements dedicated to a particular class of work or function can be determined dynamically, and this maintains a balanced and predictable service to the user community.

If we explore the nature of an Information Technology system workload, it can be characterized by a number of discrete classes of work: support of the operating system itself, which we regard as a significant computational load; secondly, support of the database which can be regarded as a server for the applications load; the applications themselves; and lastly, the processing associated with communications.

The operating system exists to clothe the underlying machine and present a conventional view of the machine to users. The database can be regarded as a service which is consuming processing resource and I/O capability and is used by applications which interact in a variety of different ways with the users. Some applications are transaction-based and provide an on-line operation for users, others are query-based and are used typically for management information enquiries, analyses, etc. The computational load presented by the communications protocol software is significant, but of vital importance if the machine in question is to co-operate with other machines in the enterprise.

ORACLE presents two major classes of workload: one is the applications and the other is the ORACLE kernel. Applications consume quantities of CPU resource per user and correspondingly occupy finite quantities of memory per user. Reports can consume even more memory and variable amounts of computing power depending on the nature of the reporting demanded. This application load controls the amount of work that the database kernel must do. So, the amount of computing resource required by the kernel is proportional to the application load at any point in time.

In an established Information Technology-using community, the number of people using an application can be predicted with each user having relatively predictable patterns. It is not the case that total workload is constant because it probably varies throughout the working day. However, it would be reasonable to say that total workload is repeatable in the short-term and growing in the long-term.

Many companies produce reports as an 'off-line' activity at the end of the working day because of the predictably heavy demands they place on the entire system. Reporting should be regarded as a first-class citizen in terms of workload and on a par with all the other applications which are being run. It need not be relegated to the times of day when its impact will be least disastrous, as is often the case. If this important management information could actually be obtained on-line, then it would provide a real-time picture, enabling better decision-making and potentially providing an important competitive advantage.

The kernel load is directly related to the total application load, so it follows that the number of users the system can support is a function of the capacity of the kernel. If we wish to support more users on the system, we must increase the capacity of the kernel. ORACLE is structured in such a way that one can have multiple copies of the kernel operating on behalf of groups of users. This allows scalable numbers of users to be supported. This is another area where Meiko's specific parallel processing insights have been brought to bear on the implementation of ORACLE.

We took as our goal the design of an architecture which will cater for the variety of demands made by an Information Technology workload, allowing us to deploy varying numbers of processing elements to different classes of work, and achieving a computer which is a balanced resource in all circumstances.

This is not as straightforward as it first seems. In reality, the workload and the mix of classes of work varies during the working day. Techniques such as multi-processor load balancing are used to share out the demands over 'pools' of processing elements. The constraint that we can employ to advantage is that we can restrict each class of work to its processing element