PROCEEDINGS / KONGRESOPSOMMINGS

6th
SOUTHERN AFRICAN COMPUTER SYMPOSIUM

6de
SUIDELIKE-AFRIKAANSE REKENAARSIMPOSIUM

De Overberger Hotel, Caledon
2 - 3 JULY 1991

SPONSORED by

ISM
FRD
GENMIN

EDITED by
M H LINCK
Department of Computer Science
University of Cape Town
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreword</td>
<td>1</td>
</tr>
<tr>
<td>Organising Committee</td>
<td>2</td>
</tr>
<tr>
<td>Referees</td>
<td>3</td>
</tr>
<tr>
<td>Program</td>
<td>5</td>
</tr>
<tr>
<td>Papers (In order of presentation)</td>
<td>9</td>
</tr>
<tr>
<td>&quot;A value can belong to many types&quot;</td>
<td></td>
</tr>
<tr>
<td>B H Venter, University of Fort Hare</td>
<td>10</td>
</tr>
<tr>
<td>&quot;A Transputer Based Embedded Controller Development System&quot;</td>
<td></td>
</tr>
<tr>
<td>M R Webster, R G Harley, D C Levy &amp; D R Woodward, University of Natal</td>
<td>16</td>
</tr>
<tr>
<td>&quot;Improving a Control and Sequencing Language&quot;</td>
<td></td>
</tr>
<tr>
<td>G Smit &amp; C Fair, University of Cape Town</td>
<td>25</td>
</tr>
<tr>
<td>&quot;Design of an Object Orientated Framework for Optimistic Parallel Simulation on Shared-Memory Computers&quot;</td>
<td></td>
</tr>
<tr>
<td>P Machanick, University of Witwatersrand</td>
<td>40</td>
</tr>
<tr>
<td>&quot;Using Statecharts to Design and Specify the GMA Direct-Manipulation User Interface&quot;</td>
<td></td>
</tr>
<tr>
<td>L van Zijl &amp; D Mitton, University of Stellenbosch</td>
<td>51</td>
</tr>
<tr>
<td>&quot;Product Form Solutions for Multiserver Centres with Heirarchical Classes of Customers&quot;</td>
<td></td>
</tr>
<tr>
<td>A Krzesinski, University of Stellenbosch and R Schassberger, Technische Universität Braunschweig</td>
<td>69</td>
</tr>
<tr>
<td>&quot;A Reusable Kernel for the Development of Control Software&quot;</td>
<td></td>
</tr>
<tr>
<td>W Fouche and P de Villiers, University of Stellenbosch</td>
<td>83</td>
</tr>
<tr>
<td>&quot;An Implementation of Linda Tuple Space under the Helios Operating System&quot;</td>
<td></td>
</tr>
<tr>
<td>P G Clayton, E P Wentworth, G C Wells and F de Heer-Menlah, Rhodes University</td>
<td>95</td>
</tr>
<tr>
<td>&quot;The Design and Analysis of Distributed Virtual Memory Consistency Protocols in an Object Orientated Operating System&quot;</td>
<td>107</td>
</tr>
<tr>
<td>K Macgregor, University of Cape Town &amp; R Campbell University of Illinois at Urbana-Champaign</td>
<td></td>
</tr>
</tbody>
</table>
· "Concurrency Control Mechanisms for Multidatabase Systems"
  A Deacon, University of Stellenbosch  118

· "Extending Local Recovery Techniques for Distributed Databases"
  H L Victor & M H Rennhackkamp, University of Stellenbosch  135

· "Analysing Routing Strategies in Sporadic Networks"
  S Melville, University of Natal  148

· "The Design of a Speech Synthesis System for Afrikaans"
  M J Wagener, University of Port Elizabeth  167

· "Expert Systems for Management Control: A Multiexpert Architecture"
  V Ram, University of Natal  177

· "Integrating Similarity-Based and Explanation-Based Learning"
  G D Oosthuizen and C Avenant, University of Pretoria  187

· "Efficient Evaluation of Regular Path Programs"
  P Wood, University of Cape Town  201

· "Object Orientation in Relational Databases"
  M Rennhackkamp, University of Stellenbosch  211

· "Building a secure database using self-protecting objects"
  M Olivier and S H von Solms, Rand Afrikaans University  228

· "Modelling the Algebra of Weakest Preconditions"
  C Brink and I Rewitsky, University of Cape Town  242

· "A Model Checker for Transition Systems"
  P de Villiers, University of Stellenbosch.  262

· "A New Algorithm for Finding an Upper Bound of the Genus of a Graph"
  D I Carson and O R Oellermann, University of Natal  276
FOREWORD

The 6th Computer Symposium, organised under the auspices of SAICS, carries on the tradition of providing an opportunity for the South African scientific computing community to present research material to their peers.

It was heartening that 31 papers were offered for consideration. As before all these papers were refereed. Thereafter a selection committee chose 21 for presentation at the Symposium.

Several new dimensions are present in the 1991 symposium:

* The Symposium has been arranged for the day immediately after the SACLA conference.

* It is being run over only 1 day in contrast to the 2-3 days of previous symposia.

* I believe that it is first time that a Symposium has been held outside of the Transvaal.

* Over 85 people will be attending. Nearly all will have attended both events.

* A Sponsorship package for both SACLA and the Research Symposium was obtained. (This led to reduced hotel costs compared to previous symposia)

A major expense is the production of the Proceedings of the Symposium. To ensure financial soundness authors have had to pay the page charge of R20 per page.

A thought for the future would be consideration of a poster session at the Symposium. This could provide an alternative approach to presenting ideas or work.

I would sincerely hope that the twinning of SACLA and the Research Symposium is considered successful enough for this combination survive. As to whether a Research Symposium should be run each year after SACLA, or only every second year, is a matter of need and taste.

A challenge for the future is to encourage an even greater number of MSc & PhD students to attend the Symposium. Unlike this year, I would recommend that they be accommodated at the same cost as everyone else. Only if it is financially necessary should the sponsored number of students be limited.

I would like to thank the other members of the organising committee and my colleagues at UCT for all the help that they have given me. A special word of thanks goes to Prof. Pieter Kritzinger who has provided me with invaluable help and ideas throughout the organisation of this 6th Research Symposium.

M H Linck
Symposium Chairman
SYMPOSIUM CHAIRMAN

M H Linck, University of Cape Town

ORGANISING COMMITTEE

D Kourie, Pretoria University.
P S Kritzinger, University of Cape Town.
M H Linck, University of Cape Town.

SPONSORS

ISM
GENMIN
FRD
LIST OF REFEREES FOR 6th RESEARCH SYMPOSIUM

<table>
<thead>
<tr>
<th>NAME</th>
<th>INSTITUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barnard, E</td>
<td>Pretoria</td>
</tr>
<tr>
<td>Becker, Ronnie</td>
<td>UCT</td>
</tr>
<tr>
<td>Berman S</td>
<td>UCT</td>
</tr>
<tr>
<td>Bishop, Judy</td>
<td>Wits</td>
</tr>
<tr>
<td>Berman, Sonia</td>
<td>UCT</td>
</tr>
<tr>
<td>Brink, Chris</td>
<td>UCT</td>
</tr>
<tr>
<td>Bodde, Ryn</td>
<td>Networks Systems</td>
</tr>
<tr>
<td>Bornman, Chris</td>
<td>UNISA</td>
</tr>
<tr>
<td>Bruwer, Piet</td>
<td>UOFS</td>
</tr>
<tr>
<td>Cherenack, Paul</td>
<td>UCT</td>
</tr>
<tr>
<td>Cook Donald</td>
<td>UCT</td>
</tr>
<tr>
<td>de Jaeger, Gerhard</td>
<td>UCT</td>
</tr>
<tr>
<td>de Villiers, Pieter</td>
<td>Stellenbosch</td>
</tr>
<tr>
<td>Ehlers, Elize</td>
<td>RAU</td>
</tr>
<tr>
<td>Eloff, Jan</td>
<td>RAU</td>
</tr>
<tr>
<td>Finnie, Gavin</td>
<td>Natal</td>
</tr>
<tr>
<td>Gaynor, N</td>
<td>AECI</td>
</tr>
<tr>
<td>Hutchinson, Andrew</td>
<td>UCT</td>
</tr>
<tr>
<td>Jourdan, D</td>
<td>Pretoria</td>
</tr>
<tr>
<td>Kourie Derrick</td>
<td>Pretoria</td>
</tr>
<tr>
<td>Kritzinger, Pieter</td>
<td>UCT</td>
</tr>
<tr>
<td>Krzesinski, Tony</td>
<td>Stellenbosch</td>
</tr>
<tr>
<td>Laing, Doug</td>
<td>ISM</td>
</tr>
<tr>
<td>Labuschagne, Willem</td>
<td>UNISA</td>
</tr>
<tr>
<td>Levy, Dave</td>
<td>Natal</td>
</tr>
<tr>
<td>Name</td>
<td>Institution</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>MacGregor, Ken</td>
<td>UCT</td>
</tr>
<tr>
<td>Machanick, Philip</td>
<td>Wits</td>
</tr>
<tr>
<td>Mattison Keith</td>
<td>UCT</td>
</tr>
<tr>
<td>Messerschmidt, Hans</td>
<td>UOFS</td>
</tr>
<tr>
<td>Mutch, Laurie</td>
<td>Shell</td>
</tr>
<tr>
<td>Neishlos, N</td>
<td>Wits</td>
</tr>
<tr>
<td>Oosthuizen, Deon</td>
<td>Pretoria</td>
</tr>
<tr>
<td>Peters Joseph</td>
<td>Simon Fraser</td>
</tr>
<tr>
<td>Ram, V</td>
<td>Natal, Pmb.</td>
</tr>
<tr>
<td>Postma, Stef</td>
<td>Natal, Pmb</td>
</tr>
<tr>
<td>Rennhackkamp, Martin</td>
<td>Stellenbosch</td>
</tr>
<tr>
<td>Shochot, John</td>
<td>Wits</td>
</tr>
<tr>
<td>Silverberg, Roger</td>
<td>Council for Mineral Technology</td>
</tr>
<tr>
<td>Smit, Riel</td>
<td>UCT</td>
</tr>
<tr>
<td>Smith, Dereck</td>
<td>UCT</td>
</tr>
<tr>
<td>Terry, Pat</td>
<td>Rhodes</td>
</tr>
<tr>
<td>van den Heever, Roelf</td>
<td>UP</td>
</tr>
<tr>
<td>van Zijl, Lynette</td>
<td>Stellenbosch</td>
</tr>
<tr>
<td>Venter, Herman</td>
<td>Fort Hare</td>
</tr>
<tr>
<td>Victor, Hema</td>
<td>Stellenbosch</td>
</tr>
<tr>
<td>von Solms, Basie</td>
<td>RAU</td>
</tr>
<tr>
<td>Wagenaar, M</td>
<td>UPE</td>
</tr>
<tr>
<td>Wentworth, Peter</td>
<td>Rhodes</td>
</tr>
<tr>
<td>Wheeler, Graham</td>
<td>UCT</td>
</tr>
<tr>
<td>Wood, Peter</td>
<td>UCT</td>
</tr>
</tbody>
</table>
6TH RESEARCH SYMPOSIUM - 1991

FINAL PROGRAM

TUESDAY 2nd July 1991

10h00 - 13h00  Registration
13h00 - 13h50  PUB LUNCH

14h00 - 15h30  SESSION 1A
Venue: Hassner
Chairman: Prof Basie von Solms

14h00 - 14h30  "A value can belong to many types."
B H Venter, University of Fort Hare

14h30 - 15h00  "A Transputer Based Embedded Controller Development System"
M R Webster, R G Harley, D C Levy & D R Woodward, University of Natal

15h00 - 15h30  "Improving a Control and Sequencing Language"
G Smit and C Fair, University of Cape Town

15h30 - 16h00  TEA

SESSION 1B

Venue: Hassner C
Chairman: Prof Roelf v d Heever

14h00 - 14h30  "Design of an Object Orientated Framework for Optimistic Parallel Simulation on Shared-Memory Computers" P Machanick, University of Witwatersand

14h30 - 15h00  "Using Statecharts to Design and Specify the GMA Direct-Manipulation User Interface" L van Zijl & D Mitton, University of Stellenbosch

15h00 - 15h30  "Product Form Solutions for Multiserver Centres with Hierarchical Classes of Customers" A Krzesinski, University of Stellenbosch and R Schassberger, Technische Universität Braunschweig
16h00 - 17h30 SESSION 2A

Venue: Hassner

Chairman: Prof Derrick Kourie

16h00 - 16h30
"A Reusable Kernel for the Development of Control Software" W Fouché and P de Villiers, University of Stellenbosch

16h30 - 17h00
"An Implementation of Linda Tuple Space under the Helios Operating System" P G Clayton, E P Wentworth, G C Wells and F de-Heer-Menlah, Rhodes University

17h00 - 17h30
"The Design and Analysis of Distributed Virtual Memory Consistency Protocols in an Object Orientated Operating System" K MacGregor, University of Cape Town & R Campbell, University of Illinois at Urbana-Champaign

19h30
PRE-DINNER DRINKS

20h00
GALA CAPE DINNER
(Men: Jackets & ties)
WEDNESDAY 3rd July 1991

7h00 - 8h15 BREAKFAST

8h15 - 9h45 SESSION 3A

Venue: Hassner

Chairman: Assoc Prof P Wood

8h15 - 8h45
"Concurrency Control Mechanisms for Multidatabase Systems" A Deacon, University of Stellenbosch

8h45 - 9h15
"Extending Local Recovery Techniques for Distributed Databases" H L Victor & M H Rennhackkamp, University of Stellenbosch

9h15 - 9h45
"Analysing Routing Strategies in Sporadic Networks" S Melville, University of Natal

9h45 - 10h15 TEA

10h15 - 11h00 SESSION 4

Venue: Hassner

Chairman: Prof P S Kritzinger
Invited paper: E Coffman

11h00 - 11h10 BREAK

SESSION 3B

Venue: Hassner C

Chairman: Prof G Finnie

8h15 - 8h45
The Design of a Speech Synthesis System for Afrikaans" M J Wagener, University of Port Elizabeth

8h45 - 9h15
"Expert Systems for Management Control: A Multiexpert Architecture" V Ram, University of Natal

9h15 - 9h45
"Integrating Similarity-Based and Explanation-Based Learning" G D Oosthuizen and C Avenant, University of Pretoria
11h10 - 12h40 SESSION 5A

Venue: Hassner

Chairman: Prof C Bornman

11h10 - 11h40
"Efficient Evaluation of Regular Path Programs"
P Wood, University of Cape Town

11h40 - 12h10
"Object Orientation in Relational Databases"
M Rennhackkamp, University of Stellenbosch

12h10 - 12h40
"Building a secure database using self-protecting objects"
M Olivier and S H von Solms, Rand Afrikaans University

SESSION 5B

Venue: Hassner C

Chairman: Prof A Krzesinski

11h10 - 11h40
"Modelling the Algebra of Weakest Preconditions"
C Brink & I Rewitsky, University of Cape Town

11h40 - 12h10
"A Model Checker for Transition Systems"
P de Villiers, University of Stellenbosch

12h10 - 12h40
"A New Algorithm for Finding an Upper Bound of the Genus of a Graph"
D I Carson and O R Oellermann, University of Natal

12h45-12h55 GENERAL MEETING of RESEARCH SYMPOSIUM ATTENDEES

Venue: Hassner

Chairman: Dr M H Linck

13h00 - 14h00 LUNCH

FINIS 6th COMPUTER SYMPOSIUM
PAPERS

of the

6TH RESEARCH SYMPOSIUM
The Design and Analysis of Distributed Virtual Memory Consistency Protocols in an Object Orientated Operating System.

K. J. MacGregor
Dept of Computer Science, University of Cape Town, Private Bag, Rondebosch 8000, South Africa
and
R. H. Campbell
Dept. of Computer Science, University of Illinois at Urbana Champaign, Urbana Illinois 61801, U.S.A.

Abstract
An object-orientated framework for the design of distributed virtual memory consistency protocols is presented. It is shown that custom designed protocols for different types of applications are easy to construct and use within this framework. Consistency protocols are shown to be useful in implementing atomic updates, and in controlling the assignment of pages to process.

An analysis of the performance of the Choices distributed virtual memory implementation is presented. It is shown that distributed virtual memory improves the performance of many applications by separating, and distributing the tasks of process management, synchronisation and paging. This performance improvement occurs despite the overhead in setting up and maintaining the coherence across the network. This is particularly noticeable when more processes than processors are active on a node. The results presented illustrate the effective improvements that can be achieved using correct coherence protocol and the current hardware restrictions limiting the performance of distributed virtual memory. In this it suggests further areas for research.

1 Introduction

Distributed virtual memory (DVM) exploits traditional virtual memory mechanisms to support a shared memory style of multiprocessing on a network of computers [Li86]. Shared memory parallel programs can be ported to a loosely coupled environment using DVM without change.

To maximise the number of parallel activities possible, local copies of the shared data are maintained on the machines involved. Some form of consistency among the various copies is guaranteed by means of consistency protocols. The protocols control the replication and invalidation of data within the system to support the required definition of consistency. Since they control page movement, they can be used to control the page assignment to network nodes.

In this paper, we study the Choices framework for the design of consistency protocols. This framework uses inheritance
in an object-orientated environment to permit new coherence protocols to be derived from existing ones. Three protocols are presented and the performance of two of them are analysed both with actual and synthetic timings. The timings illustrate the effectiveness of the various protocols for different applications, and in addition show how new protocols can be designed.

2. Overview of Choices.

Choices[Campbell87, Russo90] began as an investigation of the use of class hierarchies and object-orientated design for the construction of multiprocessor operating systems. All operating system concepts and components are implemented within the framework of a class hierarchy. Subclasses are used to encapsulate machine dependencies and to separate mechanisms from policy decisions. Choices is implemented in over 70,000 lines of C++ and includes code for virtual memory management, interrupt and exception handling, parallel processing, file systems, I/O, and networking. C++ supports class hierarchies and object orientated design without sacrificing efficiency.

Choices was designed on the Encore Multimax shared memory multiprocessor and is currently implemented on the Macintosh II, HP Precision, and Intel 386/486 computers.

2.1 Virtual Memory in Choices.

A Choices process [Russo88] executes in a Domain [Campbell88] which is a mapping of virtual address space to MemoryObjects. A MemoryObject is a logical collection of data. It is made accessible to the process by its MemoryObjectCache. A MemoryObjectCache maintains its own machine independent physical memory management information. A MemoryObject can be mapped into multiple Domains providing shared memory.

After a paged fault, the MemoryObject corresponding to the faulting virtual address is determined using the Domain. The RepairFault method is invoked on the MemoryObject to get the data at the virtual address. A local caching strategy, encapsulated entirely within the MemoryObjectCache is used to repair the fault. Subclasses of the MemoryObjectCache embody the ways in which a fault is repaired. This includes traditional paging, simple alteration of access (e.g. to implement copy on write), and getting the page from across the network (for DVM). A diagrammatic description of the sharing of memory in Choices is in Figure 1.

It is important to note that the Domain has no need to know or care how a MemoryObjectCache repairs a fault.
Figure 1: Memory sharing between Domains

Figure 2: A peer group of DMOC’s
3. Distributed Virtual Memory

The DVM implementation consists of a number of parts: DVM setup methods, basic networking and the two most significant extensions to the virtual memory class hierarchy, the logical page information and the consistency mechanism. The two most important classes involved in the implementation of DVM, DistributedMemoryObjectCache and PageRecord are considered below.

3.1 Class DistributedMemoryObjectCache

An instance of DistributedMemoryObjectCache provides a local physical memory cache for the copy of shared data on a networked node. These DistributedMemoryObjectCaches form a peer group (Figure 2). Each DistributedMemoryObjectCache is responsible for locating and retrieving pages from its peers in order to repair virtual memory faults generated by processes on its node. Local copies of data can be paged to backing store on their respective nodes. This activity is also managed by the DistributedMemoryObjectCache.

3.2 Class PageRecord

The PageRecord class is the hub of the DVM implementation. Besides the traditional VM information inherited from the VM page record, page state and other information required for maintaining coherence is kept in instances of the class. The Choices consistency protocols are defined using state machines that are implemented in the PageRecord class. The methods of PageRecord and its subclasses correspond to the generation of and responses to events such as

- messages between nodes
- timeouts
- local read/write accesses
- process terminations

Existing protocols can be combined (using multiple inheritance) or subclassed to create new protocols. Thus new protocols can very easily be generated from the existing basic protocols, with most of the functions being inherited from the basic protocols and only the new features coded.

4. The Consistency Protocols

4.1 The Basic Choices Protocol

The basic Choices protocol [Johnston89, Li86] is designed to avoid the overhead associated with a heavyweight network protocol, therefore, it assumes a low-level, unreliable datagram service, i.e. delivery is not guaranteed. In addition it handles page to packet assembly/disassembly and recovery of lost packets. Consistency is maintained using a single writer/multiple readers discipline.
When writeable, a page resides on one machine. Read requests from other machines make the page read only, and copies of the page are sent to the requesters. A list of copy holders is maintained at the machine that originally had write access. This machine is designated as the owner of the page. A subsequent write request will be serviced by the owner, invalidating the readable copies and giving the writeable page to the requester. The requester then becomes the new owner of that page.

The message types in the basic protocol are:

- **GetWrite**: Sent to request the writeable copy (and ownership) of a page from the page's owner.
- **HereWrite**: Sent to transfer the writeable copy of a page to a node which has requested it.
- **AckHereWrite**: Sent to acknowledge a HereWrite.
- **GetRead**: Sent to request a read-only copy of a page from the page's owner.
- **HereRead**: Sent to transfer a read-only copy of a page to a node that has requested it.
- **GetUpgrade**: Sent to request write access (and ownership) of a page from the page's owner when the requesting node already has a read-only copy.
- **HereUpgrade**: Sent to transfer the ownership of a page to a node that has requested it.
- **AckHereUpgrade**: Sent to acknowledge a HereUpgrade.
- **Invalidate**: Sent to request that a node with a read-only copy of a page disallow all accesses to it.
- **AckInvalidate**: Sent to acknowledge an Invalidate.
- **OwnerHint**: Sent as a reply to a GetWrite or GetRead message when the recipient is not the owner of the page. The replying node provides "better" page knowledge to the requesting node.

Only the **HereRead** and **HereWrite** messages pass page data between the nodes. The other message types are control messages.

Reader and Writer processes can dynamically join or leave the group of processes using DVM.

This protocol is very general and does not provide any support to:

1. **guarantee page assignment i.e.**
   - retain pages until sufficient work is done with them.
   - order page usage among processes to enforce data dependencies.

2. **guarantee atomic updates.**

Without support within the protocol, the above can be achieved at the additional cost of unnecessary application knowledge, inefficiency and possible fatal susceptibility to untrustworthy processes.
4.2 Characteristics Determining Performance

The time overhead resulting from the protocol is a function of control and data packets generated and data structure processing. This can be represented as:

\[ O = N \cdot p_k + M \cdot p_g + F + V \cdot m \]

Where

\( O \): protocol overhead.

\( p_k \): Total number of packets transferred.

\( p_g \): Total number of data packets.

\( N \): Network transmission cost per packet.

\( M \): Memory usage cost for data transfer per packet.

\( F \): Fixed data structure processing costs independent of protocol events.

\( V \): Data structure processing costs related to maintaining information on copy holders.

\( m \): Number of machines.

Not all of this overhead reflects directly on the total time taken by the application, since some events are handled concurrently with the application. Performance also depends on page assignment. The basic protocol determines the overhead of events.

For the basic protocol events Read, Write, UpgradeO, and UpgradeNO (change from read copy to write when the node is the page owner and not the page owner respectively) the maximum overhead, assuming a 4k page size and using Ethernet as a transport mechanism with a 1.5k packet size, is:

**Read:**

\[ \text{GetRead request (1 packet) + HereRead (3 packets)} \]
\[ = 4N + 3M + V \]

**Write:**

\[ \text{GetWrite (1 packet) + Invalidates ((m-1) packets) + } \]
\[ \text{AckInvalidates ((m - 1) packets) + Domain flush (Fixed cost) + HereWrite (3 packets) + AckHereWrite (1 packet) } \]
\[ = 5N + 3M + F + (m - 2)(V + 2N) \]

**UpgradeO:**

\[ \text{Invalidates ((m - 1) packets)} \]
\[ = (m - 1)(V + 2N) \]

**UpgradeNO:**

\[ \text{GetUpgrade (1 packet) + Invalidates ((m - 1) packets) + AckInvalidates ((m - 1) packets) + Domain flush (Fixed cost) + HereUpgrade (1 packet) + } \]
\[ \text{AckHereUpgrade (1 packet)} \]
\[ = 3N + F + (m - 2)(V + 2N) \]

4.3 A Locking Protocol

This protocol guarantees atomic update by denying requests for a locked page. If the basic protocol and test-and-set locks are used, it is not possible to guarantee that a page will remain; circumvention of the convention by untrustworthy processes is possible.

A response message **Retry** is returned to any requesters. Processes receiving this message can either sleep or send the request again. This defines the lock to be a sleep or

\[ 0 \text{ if } m \leq 2 \]
spin lock respectively. For a sleep lock, a queue of requesters is kept to send a wake-up. Sleep locks minimise network loading but centralise wake-up information and blocking primitives. A spin lock provides non-blocking synchronisation. Response to Retry can be handled differently on each requester. Network loading is minimal if the retry interval is made sufficiently large so that the page is available in the next request. This protocol can also be used as an ordinary lock to guarantee order of page use to satisfy data dependencies. Such usage, however, is restricted to processes executing in different Domains. It is also possible to lock pages for a given amount of time, to satisfy the page requirements of an application when they can be predicted beforehand or determined by some heuristic.

4.4 A Delay protocol

The Delay protocol is a method for retaining pages for a sufficient time without specific knowledge. In the basic protocol a request is serviced immediately. Two, or more, processes writing simultaneously could cause a page to thrash from machine to machine, without any useful work being done. A threshold time, after which external requests for copies of the page are serviced, can improve the ratio of useful time to page transport time in all except pathological cases. Requests arriving during this time are queued and serviced later. The threshold time value is decided by the current activity within the page and the number of requests for that page in order to generate a page assignment that reduces the page movement and increases page usage. Threshold time also reduces the possibility of live-lock, where a page is shuttled between some machines, and others never get access. Whereas the locking protocol is user dependant enforced by lock and unlock system calls the delay protocol is dependant on the transmission speed of the network, the processor speed of the various computers and the level of access. The range of applications that can use this protocol vary from numerical to artificial intelligence which show different characteristics in their pattern of page use, the numerical being the more predictable and as such the easier to optimise. This protocol is still being studied to determine a suitable heuristic to decide the threshold time, particularly for commercial systems usage. As its characteristics are not significantly different from the locking protocol it is not included in the timing analysis.
4.5 Variations

The protocols can be altered to trade off characteristics such as resiliency to packet loss, network loading, etc. For example, in the basic protocol, only an approximate knowledge of the owner is kept on all nodes except those that have read/write access. Thus, requests from a node that does not have a copy of the page can end up at a node that is not the actual owner. One of two actions can be taken; forward the request to the current value of the owner, which is approximate, until the actual owner is reached [Forin89, Li86], or indicate the possible owner to the requester for an explicit send by the requester. In the former strategy, the number of messages is less than in the latter. However, the latter is more resilient to network packet loss.

5. Empirical Results

Two algorithms, matrix multiplication and Producer Consumer, were chosen as they exhibit extremes of page conflicts. The applications were run on 2 Encore Multimaxes each with 4 processors connected by Ethernet.

5.1 Matrix Multiplication

A simple 3 loop matrix multiplication program was used. In the case of the Lock protocol each page was locked when calculating the row on that page. Table 1 shows the timings for the two protocols. The results are summarised below:

- Matrix multiplication time for DVM is comparable to that for shared memory.
- DVM offers access to a greater number of processors, and gives better performance when the number of processes on a single machine exceeds the number of processors.
- Paging behaviour is drastically altered by access patterns.
- The locking algorithm performs slightly better by saving page transfers.

5.2 Producer Consumer

The producer locks the buffer and fills it, then releases the lock. The consumer then acquires the lock and empties the buffer. The processing time is enough to force Retry. Table 2 illustrates the timings for Producer Consumer. The Lock protocol is significantly more efficient than the basic protocol even when spin locks are used - 33.4% of the basic protocol time for a buffer up to 4k in size and 55.4% of the basic protocol time for a buffer of 10k. When the buffer size increases to over 60k the basic protocol is more efficient. The amount of network traffic generated by the Lock protocol is also less. For larger packet sizes and
longer fill/empty processing times, traffic could be further reduced by using a sleep lock.

Table 1: Comparative timings in sec, for matrix multiplication, estimated accuracy + 1sec

<table>
<thead>
<tr>
<th># Processes</th>
<th>kij</th>
<th></th>
<th>ijk</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Shared</td>
<td>Distributed</td>
<td>Shared</td>
<td>Distributed</td>
</tr>
<tr>
<td></td>
<td>Basic</td>
<td>Locking</td>
<td>Basic</td>
<td>Locking</td>
</tr>
<tr>
<td>2</td>
<td>89.2</td>
<td>92.0</td>
<td>91.0</td>
<td>176.9</td>
</tr>
<tr>
<td>4</td>
<td>44.6</td>
<td>47.4</td>
<td>47.4</td>
<td>88.1</td>
</tr>
<tr>
<td>8</td>
<td>45.5</td>
<td>25.7</td>
<td>25.0</td>
<td>90.2</td>
</tr>
</tbody>
</table>

Table 2: Comparative timings in ms for Producer Consumer, estimated accuracy + 2ms

<table>
<thead>
<tr>
<th>Buffer Size (pages)</th>
<th>Basic (test and set)</th>
<th>Locking</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (ms)</td>
<td># Packets</td>
</tr>
<tr>
<td>1</td>
<td>219.8</td>
<td>28</td>
</tr>
<tr>
<td>3</td>
<td>325.1</td>
<td>40</td>
</tr>
<tr>
<td>10</td>
<td>690.3</td>
<td>82</td>
</tr>
</tbody>
</table>

6. Analysis of Performance

For DVM to achieve the efficiency required for parallel processing applications, the transfer time for a page over the network should be close to the disk swap time, which on the Encore Multimax in Choices is 20ms. In analysing the relative contributions of processor speed, memory speed and network transmission to the page transfer rate it was found that the memory access time followed by the network setup time were by far the largest components. Some of this resulted from the Encore Ethernet driver performing multiple memory copies per packet transferred, while another contributing factor was the page assembly/disassembly resulting from the 1.5k Ethernet packet restriction.

Table 3 shows present values and the effect of 50% less memory overhead, 50% less network overhead and 50% faster processor\(^2\). Only in the last case is the desired performance achieved. Thus the communication transfer mechanism should have a packet size at least that of a page. This can be achieved by using a different transfer mechanism e.g. hyperchannel or FDDI, or by reducing the page size.

\(^2\) The current processors were 16Mhz NS32332s
The Intel 80386/80486 can have variable page sizes and a 1k page for memory swaps could prove more efficient with an Ethernet transfer medium. This latter case however changes the access characteristics of the applications as there would be fewer page conflicts as the amount of data shared between nodes would be reduced. A 1k page is incompatible with most Unix type file systems. The smaller page size requires investigation if DVM is to be used as a cheap implementation of parallel processing. DMA accesses from the user data area onto the network using scatter-gather hardware would also reduce the number of memory copies. Increases in processor speed, and network transmission speed would only provide marginal improvements.

Table 3: Basic Operation timings, in ms for varying+ system component speeds (for 2 machines) estimated accuracy ± 1ms

<table>
<thead>
<tr>
<th>Event</th>
<th>Measured</th>
<th>Memory</th>
<th>Network</th>
<th>Processor</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>39.9</td>
<td>27.8</td>
<td>30.4</td>
<td>33.7</td>
<td>8.2</td>
</tr>
<tr>
<td>Write</td>
<td>48.2</td>
<td>36.2</td>
<td>36.4</td>
<td>41.2</td>
<td>1.2</td>
</tr>
<tr>
<td>UpgradeO</td>
<td>19.4</td>
<td>19.4</td>
<td>12.3</td>
<td>18.4</td>
<td>1.2</td>
</tr>
<tr>
<td>UpgradeNO</td>
<td>11.1</td>
<td>11.1</td>
<td>6.3</td>
<td>11.0</td>
<td>6.2</td>
</tr>
</tbody>
</table>

7 Conclusion

Performance of applications operating in a DVM environment is sensitive to their locality properties. To achieve the best performance it is necessary to choose an appropriate consistency protocol. This paper demonstrates a versatile framework for the design of such protocols. It is easy to specialise existing protocols or combine them to beat a custom designed protocol. The remainder of the VM system supports the creation of independent policy decisions for the different shared areas.

It has also been shown that consistency protocols can be used to achieve more than consistency. The Lock protocol provides a way of guaranteeing atomic update, by preventing the movement of a page. These page locks can also be used as distributed locks, however their use is restricted to between Domains. When used as distributed locks they are cheaper than test-and-set locks. This protocol can also be used to assign pages, when predictable by the application.

The Delay protocol attempts to optimally assign pages based on page usage, retaining pages on a machine. This is achieved without any knowledge of the application. The protocol also reduces the possibility of livelock problems. Other protocol variations to trade off the characteristics that affect performance are also indicated.
The experimental results are limited to the Encore Multimax however they show the hardware restrictions currently limiting the performance of DVM. The results show adequate promise that further investigation on the effects of page size, transfer medium and application performance should be carried out to achieve the optimal performance of applications in a DVM environment.

References


