The South African Computer Journal
An official publication of the South African Computer Society and the South African Institute of Computer Scientists

Editor
Professor Derrick G Kourie
Department of Computer Science
University of Pretoria
Hatfield 0083
Email: dkourie@dos-lan.cs.up.ac.za

Subeditor: Information Systems
Prof John Schochot
University of the Witwatersrand
Private Bag 3
WITS 2050
Email: 035ebrse@witsvma.wits.ac.za

Production Editor
Prof G de V Smit
Department of Computer Science
University of Cape Town
Rondebosch 7700
Email: gds@cs.uct.ac.za

Editorial Board

Professor Gerhard Barth
Director: German AI Research Institute

Professor Judy Bishop
University of Pretoria

Professor Donald Cowan
University of Waterloo

Professor Jürg Gutknecht
ETH, Zürich

Professor Pieter Kritzinger
University of Cape Town

Professor F H Lochovsky
University of Toronto

Professor Stephen R Schach
Vanderbilt University

Professor S H von Solms
Rand Afrikaanse Universiteit

Subscriptions
Annual Single copy
Southern Africa: R45.00 R15.00
Elsewhere $45.00 $15.00

to be sent to:
Computer Society of South Africa
Box 1714 Halfway House 1685
Guest Contribution

This guest contribution is a slightly edited report to the Foundation for Research and Development (FRD) drawn up by Ed Coffman. Ed was an FRD-sponsored guest at the 6th South African Computer Research Symposium. The report was not originally intended for general distribution. Rather, it was specifically compiled for the FRD and its staff. I am therefore grateful to both the FRD and the author for agreeing to its publication in SACJ. I believe that it contains several incisive observations that merit further thought and discussion amongst South African computer scientists. (Editor)

Impressions of Computer Science Research in South Africa

E. G. Coffman, Jr.
AT&T Bell Laboratories, USA

In commenting on the cross section of computer science research in South Africa, I will use the classification in the table of contents of the "Summary of Awards: Fiscal Year 1989," a document published recently by the US National Science Foundation. Of the 5 categories, I will treat Numeric and Symbolic Computation as inappropriate for the discussion below. In this category I noted no research in the computer science setting in South Africa. It is also common in the US and elsewhere to place this effort in other departments, e.g., departments of mathematics or applied mathematics.

Of the remaining categories I found South Africa to be strongest in software systems and engineering, to have a substantial investment in computer systems and architecture, and to be weakest in computer and computation theory.

The coverage in software systems and engineering (SSE) was broad, topical, and similar in scope to that in US universities. Technology transfer and the corresponding relations with industry seemed to be in place or developing along promising lines. I comment in passing that this was rather surprising to me. In the US the development of SSE within university departments has lagged behind almost all other disciplines of computer science. A primary problem has been the insatiable appetite of industry for all Ph.D. graduates in the SSE field.

The investment in parallel processing, computer networks, and distributed computing appears sound, although I expected to see a greater emphasis on mathematical foundations (see my remarks below), particularly in the parallel algorithms area. Given current resources, South African institutions are doing remarkably well in computer science research. But computer science is a fundamentally important course of study, beginning at an early age and extending through graduate Ph.D. research; I take this as sufficiently obvious that I need not dwell on justifications. With this in mind, and with the necessary resources in hand, South Africa should, in my opinion, expand and consolidate its computer science research effort, increase its visibility in the international arena, and correct the rather thin distribution of graduate research among universities.

I can see much of this proceeding along present lines, but I would strongly recommend a concerted development in computer and computation theory (CCT), education and research; this is mainstream computer science and forms the basis for virtually all other fields of study within computer science. It is by no means absent in South Africa curricula, but it appears to be under-represented in advanced studies and Ph.D. level research.

At the graduate level CCT is heavily mathematical. I understand that mathematical foundations are supplied by mathematics departments in certain cases. This is not ideal, but workable and it is justified by limited resources. However, it is important that mathematics departments not regard this as a mere service; faculty will have to make a major commitment to theoretical computer science, publishing in its leading journals (e.g. SIAM Journal of Computing, Journal of the ACM, Journal of Algorithms, Algorithmica, Journal of Computer and Systems Sciences, Theoretical Computer Science, etc.), and providing the supervision of theses sponsored by computer science departments and leading to degrees in computer science. I would also encourage active participation in the international computer science "theory" societies and their meetings; two highly prestigious examples of the latter are the annual Symposium on the Theory of Computing and the Foundations of Computer Science conference.

Returning to the thin distribution of computer science research, I would make the following point. If the current situation is only a stage of development - i.e., if further resources (both human and financial) can be counted on to bring at least a few of the departments to a critical mass - then little needs to be said beyond the earlier remarks. Critical mass is hard to define, but calls for adequate, expert coverage of mainstream computer science research. In view of the breadth of this research, 8-10 Ph.D. full-time-equivalent faculty would seem to be barely adequate; with the usual clumping of faculty in specific research areas, more would be expected. South Africa has a talent base such that there is little doubt that such departments would achieve a much wider international recognition.
On the other hand, if resources remain fixed at current or even slightly retrenched levels, then I would recommend consolidation to achieve the same goals on a smaller scale. Within a university this can often be done by establishing interdisciplinary, degree-granting laboratories or institutes of computer science, which bring together the computer science efforts located in various departments other than computer science, such as electrical engineering, industrial engineering, business/management science, mathematics, and operations research. The idea is to enjoy the advantages (opportunity, synergy, awareness, etc.) to both students and faculty of reasonably large computer science programs. There are many examples of such intramural laboratories in North America and Europe.

This approach could also be considered among journals on this list are regarded as equally meritorious, and all of them are more meritorious than any conference proceedings. What does all of this mean?

The momentous implication of the committee's deliberations is that the State will not give your institution a single cent for anything that you publish in SACJ. Instead, the State and your institution will scrupulously keep a score of the annual number of publications that count - but actually don't - because someday they might! And to encourage your enthusiastic participation in this Alice in Wonderland exercise, your institution might actually give you some of the standard subsidy funding that the State should have provided according to its own formulae, but didn't.

You will not be allowed to use this money to buy yourself a car - not even a casual meal. You may only use it to finance activities that are provably directed towards producing more papers in approved journals. The great consolation, of course, is that you will not be required to pay income tax on this money. The only tax involved will be the VAT component when you spend it in an approved manner. As a good computer scientist who enjoys recursion, my vote would be that all such revenue collected by the State should be earmarked to be placed in the pay packets of committee members who decided that SACJ should be approved.

If you publish in these approved journals with sufficient regularity and enthusiasm you will almost deserve to be regarded as a researcher. What you additionally need to do, is to ensure that you befriend and impress at least three overseas referees. You then apply to the FRD for official recognition as a researcher, and if they are sufficiently impressed, they will give you more of the non-taxable kind of money that you need to spend on research to publish in approved journals.

Derrick Kourie
Editor

Editor's Notes

Prof John Schochot has graciously accepted to be SACJ's subeditor for papers relating to Information Systems. Authors wishing to submit papers in this general area should please contact him directly. I look forward working with John, and to a significant increase in IS contributions in future.

The hand of the new production editor, Riel Smit, will be clearly evident in this issue. Those papers not prepared in camera-ready format by the authors themselves were prepared by him in TEX. He will be announcing revised guidelines for camera-ready format in a future issue. If you use TEX or one of its variations, Riel would be happy to provide you with a styles document to SACJ format.

At last some Department of National Education committee has decided that SACJ should now be on the list of approved journals. This places it amongst the ranks of some 6800 other journals. These include not merely a number of ACM and IEEE Transactions but also such journals as Ostrich, Trivium, Crane Bag, Koers, Mosquito News, Police Chief, Connoisseur, Lion and the Unicorn, About the House and Ohio Agricultural Research and Development Center Department Series ESS. You will recall that in 1990 this same committee decided that, if judged on its own merits, SACJ did not deserve to be on the illustrious list. In the absence of other evidence, we must assume that the sole reason for its revised decision is that SACJ's predecessor, Quæstiones Informaticæ, was there. (I have a secret suspicion that the committee liked that name.)

It is my understanding that for official purposes, all universities within a confined geographical area, admit-tedly with greater difficulty perhaps. The Institute of Discrete Mathematics and Computer Science connecting Princeton University, Rutgers University, AT&T Bell Laboratories, and Bell Communications Research is a possible model. Examples in South Africa might consist of universities and research institutions on the Reef or those in the Western Cape (just to mention those with which I'm a little familiar).

As a final comment, I should note that my impressions have been based on limited information which may not give a representative picture. I am sure that my reactions will be appropriately discounted where I have been off target.
The Design and Analysis of Distributed Virtual Memory Consistency Protocols in an Object Orientated Operating System

K J MacGregor* and R H Campbell†

*Department of Computer Science, University of Cape Town, Rondebosch 7700, South Africa
†Department 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.

Received October 1991, Accepted November 1991

1 Introduction

Distributed virtual memory (DVM) exploits traditional virtual memory mechanisms to support a shared memory style of multiprocessing on a network of computers [5]. 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[1, 6] 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.

Virtual Memory in Choices

A Choices process [7] executes in a Domain [2] 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 diagram-
mation 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.

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.

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.

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

The Basic Choices Protocol
The basic Choices protocol [4, 5] 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.
- AckInvalid: 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.

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:

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

Where
Figure 1: Memory sharing between domains

Figure 2: A peer group of DMOC's
of time, to satisfy the page requirements of an application
different Domains. 

Such usage, however, is restricted to processes executing in
ciently large so that the page is available in the next request.

Processes receiving this message can either sleep or send
Retry can be handled differently on each requester. Net­
work loading is minimal if the retry interval is made suffi­

This protocol guarantees atomic update by denying requests
on copy holders.

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: GetRead request (1 packet) + HereRead (3 packets) = 4N + 3M + V
Write: GetWrite (1 packet) + Invalidates ((m - 1) packets)
+ AckInvalidates ((m - 1) packets) + Domain flush
(Fixed cost) + HereWrite (3 packets) + AckHereWrite
(1 packet) = 5N + 3M + F + (m - 2)(V + 2N)
UpgradeO: Invalidates ((m - 1) packets) = (m - 1)(V + 2N)
UpgradeNO: GetUpgrade (1 packet) + Invalidates ((m - 1)
packets) + AckInvalidates ((m - 1) packets) + Do­
main flush (Fixed cost) + HereUpgrade (1 packet) +
AckHereUpgrade (1 packet) = 3N + F + (m - 2)(V + 2N)

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 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. Net­
work loading is minimal if the retry interval is made suffi­
ciently 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.

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 immedi­
ately. 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 en­
forced 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.

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 [3, 5], 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 Con­
sumer, were chosen as they exhibit extremes of page con­
flicts. The applications were run on wo Encore Multimaxes
each with four processors connected by Ethernet.
Table 1: Comparative timings in sec, for matrix multiplication, estimated accuracy ±1sec

<table>
<thead>
<tr>
<th># Processes</th>
<th>Shared Basic</th>
<th>Distributed Locking</th>
<th>ijk</th>
<th>Basic Locking</th>
<th>ijk</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>89.2</td>
<td>92.0</td>
<td>91.0</td>
<td>176.9</td>
<td>186.7</td>
</tr>
<tr>
<td>4</td>
<td>44.6</td>
<td>47.4</td>
<td>47.4</td>
<td>88.1</td>
<td>96.5</td>
</tr>
<tr>
<td>8</td>
<td>45.5</td>
<td>25.7</td>
<td>25.0</td>
<td>90.2</td>
<td>51.9</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>Time(ms)</th>
<th># Packets</th>
<th>Time(ms)</th>
<th># Packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>219.8</td>
<td>28</td>
<td>83.0</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>325.1</td>
<td>40</td>
<td>181.7</td>
<td>30</td>
</tr>
<tr>
<td>10</td>
<td>690.3</td>
<td>82</td>
<td>551.6</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 3: Basic operation timings (ms) for varying system component speeds (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>18.2</td>
</tr>
<tr>
<td>Write</td>
<td>48.2</td>
<td>36.2</td>
<td>36.4</td>
<td>41.2</td>
<td>21.2</td>
</tr>
<tr>
<td>UpgradeO</td>
<td>19.4</td>
<td>19.4</td>
<td>12.3</td>
<td>18.4</td>
<td>11.2</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>

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.

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.

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.

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 transfers.

\(^2\)The current processors were 16Mhz NS32332a.
ber of memory copies. Increases in processor speed, and network transmission speed would only provide marginal improvements.

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


Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research papers. However, non-refereed review and exploratory articles of interest to the journal’s readers will be considered for publication under sections marked as a Communications or Viewpoints. While English is the preferred language of the journal papers, Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

Form of Manuscript
Manuscripts for review should be prepared according to the following guidelines.

- Use double-space typing on one side only of A4 paper, and provide wide margins.
- The first page should include:
  - title (as brief as possible);
  - author’s initials and surname;
  - author’s affiliation and address;
  - an abstract of less than 200 words;
  - an appropriate keyword list;
  - a list of relevant Computing Review Categories.
- Tables and figures should be on separate sheets of A4 paper, and should be numbered and titled. Figures should be submitted as original line drawings, and not photocopies.
- Mathematical and other symbols may be either handwritten or typed. Greek letters and unusual symbols should be identified in the margin, if they are not clear in the text.
- References should be listed at the end of the text in alphabetic order of the (first) author’s surname, and should be cited in the text in square brackets. Corrected proofs should be returned to the production editor within three days.
- In the case of camera-ready submissions, it is the author’s responsibility to ensure that such submissions are error-free. However, the editor may recommend minor typesetting changes to be made before publication.

Letters and Communications
Letters to the editor are welcomed. They should be signed, and should be limited to about 500 words.

Advertisements
Placement of advertisements at R1000-00 per full page per issue and R500-00 per half page per issue will be considered. These charges exclude specialized production costs which will be borne by the advertiser. Enquiries should be directed to the editor.
## Contents

**GUEST CONTRIBUTION**

Impressions of Computer Science Research in South Africa  
E.G. Coffman, Jr. .................................................. 1

**RESEARCH ARTICLES**

An Implementation of the Linda Tuple Space under the Helios Operating System  
PC Clayton, EP Wentworth, GC Wells & FK de-Heer-Menlah ........................................... 3

Modelling the Algebra of Weakest Preconditions  
C Brink and I Rewitzky ............................................. 11

The Design and Analysis of Distributed Virtual Memory Consistency Protocols in an Object Oriented OS  
KJ McGregor and RH Cambell ..................................... 21

An Object Oriented Framework for Optimistic Parallel Simulation on Shared-Memory Computers  
P Machanik .............................................................. 27

Analysing Routing Strategies in Sporadic Networks  
SW Melville ............................................................. 37

Using Statecharts to Design and Specify a Direct-Manipulation User Interface  
L Van Zijl & D Mitton ................................................. 44

Extending Local Recovery Techniques for Distributed Databases  
HL Viktor & MH Rennhackkamp ..................................... 59

Efficient Evaluation of Regular Path Programs  
PT Wood ................................................................. 67

Integrating Similarity-Based and Explanation-Based Learning  
GD Oosthuizen & C Avenant ........................................ 72

Evaluating the Motivating Environment For IS Personnel in SA Compared to the USA. (Part I)  
JD Cougar & DC Smith ................................................. 79

**TECHNICAL NOTE**

An Implementation of the Parallel Conditional  
U Jayasekera and NCK Philips ...................................... 85

**COMMUNICATIONS AND REPORTS**

Book Review .............................................................. 87

The CSP Notation and its Application to Parallel Processing  
PG Clayton ............................................................ 90