4th

SUID-AFRIKAANSE REKENAARSIMPOSIUM
SOUTH AFRICAN COMPUTER SYMPOSIUM

HOLIDAY INN PRETORIA
JULIE 1 – 3 JULY 1987
Proceedings
of the
4th South African Computer Symposium

Holiday Inn, Pretoria
1 – 3 July 1987

edited by

Pieter Kritzinger
Computer Science Department
University of Cape Town
PREFACE

Computer science is an emerging discipline which is having difficulty in being recognised as a worthy member of the sciences. I will paraphrase John Hopcroft, co-winner of the 1986 Turing Award, when, during a recent interview, he said that the primary reason for the lack of recognition, is the age of our researchers. Probably not one of the researchers who presented their work at this symposium is older than 45. I know of no computer scientist in South Africa who is in a position where (s)he can affect funding priorities. As far as I know we have no representation on any of the committees of the Foundation for Research Development and for our Afrikaans speaking fraternity, none who is a member of the Akademie vir Wetenskap en Kuns. It will take time and conscious effort to establish our presence. The same is true of course for our universities. Again, with one exception, I know of no dean of a science faculty, vice-principal or principal who is a computer scientist. We consequently spend an enormous amount of time trying to explain the needs of computer science and its difficulties. I believe this symposium is a further step towards accreditation by our peers and superiors from the other sciences.

The total number of papers submitted to the Programme Committee for consideration was 34. Each paper was reviewed by three persons knowledgeable in the field it represents. Of those submitted, 23 were finally selected for inclusion in the symposium. As a result the overall quality of the papers is high and as a computer science community in Africa we can be justly proud of the final programme.

This is the fourth in the series of South African computer symposia. This year the symposium is sponsored by the Computer Society of South Africa (CSSA), the South African Institute for Computer Scientists and the local IFIP Committee. The executive director of the CSSA and his staff deserve warm thanks for handling the organisation as well as they have, while the Organising Committee provided Derrick and I with very valuable advice.

Finally I would like to express my sincere appreciation to the authors, to the members of the Programme Committee and particularly the reviewers. Without the kind cooperation of everyone, this symposium would not have taken place.

Pieter Kritzinger
SYMPOSIUM CHAIRMAN: PS Kritzinger, University of Cape Town.
SYMPOSIUM CO-CHAIRMAN: D Kourie, University of Pretoria.
MEMBERS OF THE PROGRAMME COMMITTEE

Judy Bishop, Witwatersrand University
Chris Bornman, UNISA
Hannes de Beer, Potchefstroom University
Gideon de Kock, Port Elizabeth University
Jaap Kies, Western Cape University
Derrick Kourie, Pretoria University
Pieter Kritzinger, Cape Town University
Tony Krzesinski, Stellenbosch University
Michael Laidlaw, Durban Westville University
Peter Lay, Cape Town University
Ken MacGregor, Cape Town University
Theo McDonald, Orange Free State University
Jan Oosthuizen, University of the North
Dennis Riordan, Rhodes University
Alan Sartori-Angus, Natal University
John Shochot, Witwatersrand University
Theuns Smith, Rand Afrikaans University
Trevor Turton, ISM (Pty) Ltd
Gerrit Wiechers, Infoplan.
<table>
<thead>
<tr>
<th>REVIEWER</th>
<th>REVIEWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Berman Sonia</td>
<td>MacGregor Ken</td>
</tr>
<tr>
<td>Bishop Judy</td>
<td>Mattison Keith</td>
</tr>
<tr>
<td>Bornman Chris</td>
<td>McDonald Theo</td>
</tr>
<tr>
<td>Carey Chris</td>
<td>Rennhackkamp Martin</td>
</tr>
<tr>
<td>Cherenack Paul</td>
<td>Riordan Denis</td>
</tr>
<tr>
<td>De Beer Hannes</td>
<td>Satori-Angus Alan</td>
</tr>
<tr>
<td>De Villiers Pieter</td>
<td>Schochet John</td>
</tr>
<tr>
<td>Gorringe Pen</td>
<td>Smith Theuns</td>
</tr>
<tr>
<td>Kies Jaap</td>
<td>Turton Trevor</td>
</tr>
<tr>
<td>Kourie Derrick</td>
<td>Van Den Heever Roelf</td>
</tr>
<tr>
<td>Kriztinger Pieter</td>
<td>Van Rooyen Hester</td>
</tr>
<tr>
<td>Krzesinski Tony</td>
<td>Von Solms Basie</td>
</tr>
<tr>
<td>Laidlaw Michael</td>
<td>Vos Koos</td>
</tr>
<tr>
<td>Lay Peter</td>
<td></td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS

Keynote Address

"An Extensible System and Programming Tool for Workstation Computers." .............................................. 1
Niklaus Wirth, ETH, Zurich

Invited Lectures

"The Relationship of Natural and Artificial Intelligence." ..................not included in Proceedings.
G Lasker, University of Windsor, Ontario.

"Software Engineering: What Can We Expect in the Future?" ..................not included in Proceedings.
D Teichrow, University of Michigan, U.S.A.

Computer Languages I

"SPS-Algol: Semantic Constructs for a Persistent Programming Language." ................................. 13
S Berman, University of Cape Town.

"Petri Net Topologies for a Specification Language." .... 25
R Watson, University of the Witwatersrand.

"Towards a Programming Environment Standard in LISP." ... 45
R Mori, University of Cape Town

"ADA for Multiprocessors: Some Problems and Solutions." .. 63
J Bishop, University of the Witwatersrand.

Computer Graphics

"Polygon Shading on Vector Type Devices." ...................... 75
C F Scheepers, CSIR.

"Hidden Surface Elimination in Raster Graphics Using Visigrams." ................................. 97
P Gorringe, CSIR.

Database Systems I

"On Syntax and Semantics Related to Incomplete Information Databases." .............................................. 109
M E Orlowska, UNISA.

"Modelling Distributed Database Concurrency Control Overheads." ................................................. 131
M H Rennhackkamp, University of Stellenbosch.

Operating Systems

"The Development of a Fault Tolerant System for a Real-time Environment." .............................................. 149
M Morris, CSIR.

"A New General-purpose Operating System." ...................... 161
B H Venter, CSIR.
Computer Languages II

"The Representation of Chemical Structures by Random Context Structure Grammars." ......................... 175
E M Ehlers and B von Solms, RAU.

"A Generalised Expression Structure." ...................... 189
W van Biljon, CSIR.

Computer Networks and Protocols I

"An Approximate Solution Method for Multiclass Queueing Networks with State Dependent Routing and Window Row Control." ................................. 203
A E Krzesinski, University of Stellenbosch.

"A Protocol Validation System." ......................... 227
J Punt, University of Cape Town.

Computer Networks and Protocols II

"Protocol Performance Using Image Protocols." ........ 251
P S Kritzinger, University of Cape Town.

Artificial Intelligence

"A Data Structure for Exchanging Geographic Information." ............................................. 267
A Cooper, CSIR.

"The Design and Use of a Prolog Trace Generator for CSP." ........................... 279
D G Kourie, University of Pretoria.

Database Systems II

"An Approach to Direct End-user Usage of Multiple Databases." ........................................ 297
M J Phillips, CSIR.

"A Semantic Data Model Approach to Logical Data Independence." ......................... 329
S Berman, University of Cape Town.

Information Systems

"The ELSIM Language: an FSM-based Language for the ELSIM SEE." ......................... 343
L du Plessis and C Bornman, UNISA.

"Three Packaging Rules for Information System Design." 363
J Mende, University of the Witwatersrand.
Computer Languages III

"Experience with a Pattern-matching Code Generator." ... 371
M A Mulders, D A Sewry and W R van Biljon, CSIR.

"Set-oriented Functional Style of Programming." ........ 385
C Mueller, University of the Witwatersrand.

Tutorial

The use of Modula-2 in Software Engineering." ........... 399
N Wirth, ETH, Zurich.
## DAY 1

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>07h30</td>
<td>Registration and Coffee.</td>
</tr>
<tr>
<td>08h45</td>
<td>Welcoming address, President of the South African Institute of Computer Scientists, Dr. G. Wiechers.</td>
</tr>
<tr>
<td>09h00</td>
<td>Invited Lecture, Professor D. Teichrow, University of Michigan.</td>
</tr>
<tr>
<td></td>
<td><em>Software Engineering, ... What Can We Expect in the Future.</em></td>
</tr>
<tr>
<td>10h00</td>
<td><strong>Computer Languages I.</strong> Chairman: G. Wiechers.</td>
</tr>
<tr>
<td>10h15</td>
<td>S. Berman, University of Cape Town.</td>
</tr>
<tr>
<td></td>
<td><em>SPS-Algo: Semantic Constructs for a Persistent Programming Language.</em></td>
</tr>
<tr>
<td>10h50</td>
<td>A. Watson, University of the Witwatersrand.</td>
</tr>
<tr>
<td></td>
<td><em>Petri Net Topologies for a Specification Language.</em></td>
</tr>
<tr>
<td>11h25</td>
<td>A. Mori, University of Cape Town.</td>
</tr>
<tr>
<td></td>
<td><em>Towards a Programming Environment Standard in U.S.P.</em></td>
</tr>
<tr>
<td>11h50</td>
<td>J. Bishop, University of the Witwatersrand.</td>
</tr>
<tr>
<td></td>
<td><em>ADA for Multiprocessors: Some Problems and Solutions.</em></td>
</tr>
<tr>
<td>12h30</td>
<td>LUNCH</td>
</tr>
<tr>
<td>14h00</td>
<td><strong>Computer Graphics.</strong> Chairman: D. Hoorie</td>
</tr>
<tr>
<td></td>
<td>C. F. Scheepers, CSIR.</td>
</tr>
<tr>
<td></td>
<td><em>Polygon Shading on Vector Type Devices.</em></td>
</tr>
<tr>
<td>14h35</td>
<td>P. Gorringe, CSIR.</td>
</tr>
<tr>
<td></td>
<td><em>Hidden Surface Elimination in Raster Graphics Using Visigraphics.</em></td>
</tr>
<tr>
<td>15h15</td>
<td><strong>Database Systems I.</strong> Chairman: B. von Solms.</td>
</tr>
<tr>
<td></td>
<td>COFFEE</td>
</tr>
<tr>
<td>15h30</td>
<td>M.E. Orlowska, UNISA.</td>
</tr>
<tr>
<td></td>
<td><em>On Syntax and Semantics Related to Incomplete Information Databases.</em></td>
</tr>
<tr>
<td>16h05</td>
<td>M.H. Rennhakkkamp, Stellenbosch University.</td>
</tr>
<tr>
<td></td>
<td><em>Modelling Distributed Database Concurrency Control Overheads</em></td>
</tr>
<tr>
<td>18h00</td>
<td><strong>Operating Systems.</strong> Chairman: K. MacGregor.</td>
</tr>
<tr>
<td></td>
<td>M. Morris, UNISA.</td>
</tr>
<tr>
<td></td>
<td><em>The Development of a Fault Tolerant System for a Real-time Environment.</em></td>
</tr>
<tr>
<td></td>
<td>B. H. Venter, CSIR.</td>
</tr>
<tr>
<td></td>
<td><em>A New General-purpose Operating System.</em></td>
</tr>
<tr>
<td>18h00</td>
<td><strong>Computer Languages II.</strong> Chairman: J. Bishop.</td>
</tr>
<tr>
<td></td>
<td>COFFEE</td>
</tr>
<tr>
<td></td>
<td>W. van Biljon, CSIR.</td>
</tr>
<tr>
<td></td>
<td><em>A Generalised Expression Structure.</em></td>
</tr>
<tr>
<td>18h00</td>
<td><strong>Database Systems II.</strong> Chairman: J. Bishop.</td>
</tr>
<tr>
<td></td>
<td>W. van Biljon, CSIR.</td>
</tr>
<tr>
<td></td>
<td><em>A Generalised Expression Structure.</em></td>
</tr>
</tbody>
</table>

**COCKTAIL PARTY IN CULLINAN ROOM A.**
## DAY 2

<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>08h30</td>
<td>Keynote Address by Professor Niklaus Wirth, Swiss Federal Institute</td>
</tr>
<tr>
<td></td>
<td>for Technology, Zurich.</td>
</tr>
<tr>
<td></td>
<td><strong>An Extensible System and a Programming Tool for</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Workstation Computers.</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Computer Networks and Protocols I.</strong> Chairman: P.S. Kritzinger.</td>
</tr>
<tr>
<td>09h30</td>
<td>A.E. Krzesinski, University of Stellenbosch.</td>
</tr>
<tr>
<td></td>
<td><strong>An Approximate Solution Method for Multiclass Queueing Networks</strong></td>
</tr>
<tr>
<td></td>
<td><strong>with State Dependent Routing and Window Flow Control.</strong></td>
</tr>
<tr>
<td>10h05</td>
<td>J. Punt, University of Cape Town.</td>
</tr>
<tr>
<td></td>
<td><strong>A Protocol Validation System.</strong></td>
</tr>
<tr>
<td>10h30</td>
<td><strong>COFFEE</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Computer Networks and Protocols II.</strong> Chairman: R. van der Heever.</td>
</tr>
<tr>
<td>11h00</td>
<td>P.S. Kritzinger, University of Cape Town.</td>
</tr>
<tr>
<td></td>
<td><strong>Protocol Performance using Image Protocols.</strong></td>
</tr>
<tr>
<td>11h35</td>
<td>Invited Lecture by Professor G. Lasker, University of Windsor, Ontario.</td>
</tr>
<tr>
<td></td>
<td><strong>The Relationship of Natural and Artificial Intelligence.</strong></td>
</tr>
<tr>
<td>12h30</td>
<td><strong>LUNCH</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Artificial Intelligence.</strong></td>
</tr>
<tr>
<td></td>
<td>Chairman: G. Lasker.</td>
</tr>
<tr>
<td>14h00</td>
<td>A. Cooper, CSIR</td>
</tr>
<tr>
<td></td>
<td><strong>A Data Structure for Exchanging Geographic Information.</strong></td>
</tr>
<tr>
<td>14h35</td>
<td>A. I. Newcombe, University of Cape Town and A. Rado, National Library</td>
</tr>
<tr>
<td></td>
<td><strong>of Medicine, Maryland.</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Strategies for Automatic Indexing and Thesaurus Building.</strong></td>
</tr>
<tr>
<td>15h15</td>
<td><strong>COFFEE</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Database Systems II.</strong></td>
</tr>
<tr>
<td></td>
<td>Chairman: C. Bornman.</td>
</tr>
<tr>
<td>15h30</td>
<td>M.J. Philips, CSIR</td>
</tr>
<tr>
<td></td>
<td><strong>An Approach to Direct End-user Usage of Multiple Databases.</strong></td>
</tr>
<tr>
<td>16h05</td>
<td>S. Berman, University of Cape Town.</td>
</tr>
<tr>
<td></td>
<td><strong>A Semantic Data Model Approach to Logical Data Independence.</strong></td>
</tr>
<tr>
<td>16h45</td>
<td><strong>Open Forum with professors G. Lasker, D. Teichrow and N. Wirth.</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Moderator: Dr. D. Jacobson.</strong></td>
</tr>
<tr>
<td>19h30</td>
<td>Symposium Banquet in Cullinan Room.</td>
</tr>
<tr>
<td></td>
<td><strong>Guest speaker, Dr. D. Jacobson, Group Executive: Technology,</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Allied Technologies Limited.</strong></td>
</tr>
</tbody>
</table>

**Information Systems.**

Chairman: D. Teichrow.

L. du Plessis and C. Bornman, UNISA.

The ELSIM Language: an FSM-based Language for the ELSIM SEE.

J. Mende, University of the Witwatersrand.

Three Packaging Rules for Information System Design.

**COFFEE**

**Computer Languages III.**

Chairman: N. Wirth.

W. van Biljon, CSIR.

Experience with a Pattern-matching Code Generator.

C. Mueller, University of the Witwatersrand.

Set-oriented Functional Style of Programming.
08h00  Registration (Tutorial only).

08h30  Tutorial.
The Tutorial will be given by professor Niklaus Wirth, Division of Computer Science, Swiss Federal Institute of Technology, Zurich.

The use of Modula-2 in Software Engineering.
Topics to be covered include:

What is Software Engineering?
Data types and structures.
Modularization and information hiding.
Definition and implementation parts.
Separate compilation with type checking.
Facilities to express concurrency.
Pompous programming style.
What could be excluded?

12h15  Close of Symposium.

12h30  LUNCH
MODELING DISTRIBUTED DATABASE
CONCURRENCY CONTROL OVERHEADS

M.H. Rennhackkamp
Department of Computer Science
University of Stellenbosch

ABSTRACT

Numerous concurrency control methods have been proposed for distributed databases. Various criteria are used to compare these methods. The comparisons range from qualitative overviews through quantitative analyses to theoretical studies. A quantitative study based on an abstract model of concurrency control methods is presented, where the overheads of the methods are analytically compared using a set of evaluation parameters.

After a overview of the model's development, it is presented in detail. As an example it is applied to two-phase locking as it can be used in a fully-redundant distributed environment. It is concluded that although the model has shortcomings, it does provide a framework according to which distributed database concurrency controls can be compared.
INTRODUCTION

Many distributed database concurrency control methods have been proposed to coordinate simultaneous database accesses at dispersed sites by multiple users; in an attempt to exploit parallelism maximally to provide efficient data accesses. It is not clear which of these methods are superior; due to the diversity of the methods and the mechanisms of locking, (timestamp) ordering and optimism they are based on.

Numerous evaluations and comparative studies have been made. The criteria for these comparisons cover a variety of aspects and the actual evaluations range from qualitative overviews, through simulations and analytical models to detailed theoretical studies.

A quantitative evaluation of concurrency control overheads is presented, based on uniform representations of the methods, using an abstract relational database model. The analytical evaluation uses parameters to depict the CPU, storage and message overheads.

After an overview of the model's development, it is presented in detail. A simple two-phase locking (2PL) method with the two-phase commit (2PC) protocol incorporated, as it can be used in a fully-redundant distributed database, is used as an example. After noting some of the virtues and shortcomings of the model, it is concluded that the model does facilitate conclusive comparisons.

1. BACKGROUND

M.J. Carey [CAR83] presented an abstract model according to which centralized database concurrency control algorithms can be represented and analytically evaluated. The algorithms are depicted uniformly in implementation independent terms to evaluate the overheads incurred by each method. He concluded his presentation with suggestions for extensions to the model, to consider the methods used in multiple copy and distributed databases.

These extensions have been applied, as well as other changes to the model to depict the methods more clearly [REN86]. Carey's evaluations did not consider conflicting transactions, but this was done in the subsequent study by extending the set of parameters. Conflicts are only considered once, meaning the conflicts of re-executed transactions are not considered. Overheads due to abnormal conditions, such as transmission failures, are neither taken account of.
2. THE MODEL

The model functions on the premise that concurrency control methods can be described terms of the information they need, the conditions when conflicts occur and the way in which transactions are processed. The overhead costs of each method can then be determined in implementation independent units of cost by analyzing the uniform representations.

The model does not take recovery into consideration; eg when a site indicates its readiness to commit, it is assumed that the site will commit when commanded to do so. However, it does take conflicts into consideration, where two transactions wish to access the same data item simultaneously.

2.1 TRANSACTION STRUCTURE

A transaction is represented by the following primitives:

BEGIN(t-id)
READ-REQ(t-id, obj-id) or
WRITE-REQ(t-id, obj-id)

COMMIT(t-id)

There may be many read and write operations in a transaction.

The primitive components of transaction requests are sent to the remote sites where the local operations are executed. The global part (at the transaction's site of origin) controls the synchronization with the other sites, while the local parts maintain the local concurrency control.

2.2 CONCURRENCY CONTROL DATABASE

The information about transactions, requests and data objects are stored in a collection of relations, called the concurrency control database (CC-DB). It represents all the information any concurrency control algorithm may need and forms the cornerstone of the model, with which all the methods can be described uniformly; the differences between methods are lessened by representing all the information in this structure. All the algorithms do not necessarily use all the fields and relations of the CC-DB. A description of the relations follow:
TSI(t-id, state, timestamp)
The transaction state information relation contains fields for the transaction identifier, its state (ready, blocked, committed or aborted) and a possible transaction timestamp. It documents all active transactions and their states; an optional timestamp can be associated with each transaction.

ACC(obj-id, t-id, mode, timestamp)
This relation documents accesses to data objects, using fields for the object identifier, transaction identifier, mode of access and a possible object timestamp. The object and transaction identifiers document accesses; the mode and optional timestamp fields are used for exclusion purposes.

BLKD(t-id, cause-id)
The blocked relation documents blocked transactions. It contains the identifiers of the blocked transactions and the transactions they wait for. It is used in the locking based methods to depict the deadlock resolution information.

HIST(obj-id, t-id, mode, timestamp)
The history relation stores information about conditionally granted accesses, mostly in the optimistic methods. It contains similar fields to the ACC relation.

SITE(obj-id, site-id)
This relation documents the sites of data objects, which are normally kept in the data dictionary. It can contain an entry for each site of each data object.

2.3 MACROS

Sequences of operations on the CC-DB are called macros; these are used to depict the primitive operations executed during the processing a transaction.

The CC-DB operations are presented as query language statements; using decision (if) and iteration (while) structures. Comments are delimited by curly brackets. Pseudocode is also used liberally.

Some general macros, namely to block, restart and entirely remove (expunge) transactions are used; a boolean macro CYCLE(t-id) is also used to detect whether a transaction is involved in a deadlock cycle depicted in the blocked relation.
BLOCKING
When a transaction is blocked, its state is changed and the cause of its blocking is documented in the BLKD relation.

\[
\text{BLOCK}(t-id, \text{cause-id}): \\
\quad \text{replace } \text{TSI}(\text{state} = \text{"blocked"}) \\
\quad \quad \text{where } \text{TSI}.t-id = t-id \\
\quad \text{append BLKD}(t-id, \text{cause-id})
\]

EXPUNGING
When a transaction is expunged, it is removed from the CC-DB.

\[
\text{EXPUNGE}(t-id): \\
\quad \text{delete } \text{TSI} \\
\quad \quad \text{where } \text{TSI}.t-id = t-id \\
\quad \text{delete } \text{ACC} \\
\quad \quad \text{where } \text{ACC}.t-id = t-id \\
\quad \text{delete } \text{BLKD} \\
\quad \quad \text{where } \text{BLKD}.t-id = t-id \quad \text{or } \text{BLKD}.\text{cause} = t-id \\
\quad \text{delete } \text{HIST} \\
\quad \quad \text{where } \text{HIST}.t-id = t-id
\]

RESTARTING
When a transaction is restarted, its state is changed and it is removed to nullify its effects. This macro is also used for aborting a transaction.

\[
\text{RESTART}(t-id): \\
\quad \text{replace } \text{TSI}(\text{state} = \text{"aborted"}) \\
\quad \quad \text{where } \text{TSI}.t-id = t-id \\
\quad \quad \{\text{Undo all the DB changes}\} \\
\quad \text{EXPUNGE}(t-id)
\]

COMMUNICATION
Messages must be sent to the indicated sites. A command or macro is not explicitly received; it is merely executed at the indicated sites.

\[
\text{SEND}(\text{message}, \text{sites}) : \\
\quad \{\text{Send the message or command to the indicated site(s)}\}
\]
A macro must be used where a returned message must be explicitly received.

RECEIVE(message, sites):
   {Receive message(s) from the indicated site(s)}

2.4 CONCURRENCY CONTROL MODELING

The concurrency control methods are described by the following:

CONCURRENCY CONTROL DATABASE

Each concurrency control method makes use of different structures to represent concurrency control information. The CC-DB fields and relations used must therefore be specified for evaluation purposes.

CONFLICT SITUATIONS

The conditions under which blocking and restarting take place vary for each method; it must be specified which of these situations occur, as well as under which conditions. Boolean typed macro functions are used for this purpose.

MACROS

Different CC-DB operations are executed in each method to represent the execution of the transaction initiation, access request and commitment primitives. A major component of the model is to describe these operations in a uniform manner using macros.

2.5 PARAMETERS

Transaction volumes are quantified by the following parameters:

$T_a$: The average number of transactions in the system.

$F_0$: The average fraction of conflicting transactions. The meaning of this parameter depends on the context where it is used. In the 2PL cases, it means the fraction of blocked transactions which wait for locked data items. In the timestamp ordering (TSO) methods it enumerates the fraction of the transactions which must be restarted due to conflicts with transactions with more recent timestamps. In the optimistic methods it depicts the fraction of the transactions restarted because they did not satisfy the validation criteria.
\(F_0\): The average fraction of fatally conflicting transactions. In the 2PL methods, it represents the fraction of transactions which become blocked and cause deadlock cycles; in the TSO methods it depicts transactions restarted during commitment.

\(F_{rc}\): The average fraction of recently committed transactions. It is used to account for overheads incurred after commitment. In the optimistic methods it represents the transactions which have committed since the startup time of the oldest active transaction.

Different access requests are executed in different ways, causing different overhead costs. The following parameters are used to identify fractions of the requests:

\(R_r\): The average number of read requests per transaction.

\(R_w\): The average number of write requests per transaction.

\(R_a\): The average number of general access requests per transaction. Where equal, \(R_r + R_w = R_a\) is used to simplify evaluation expressions.

\(D\): The cost of deadlock detection at a single site. Distributed deadlock detection cost is not easily assessed; it depends greatly on the method used. It will be assumed that it is proportional to \(D\) at every site.

The following parameters are used to depict the sites:

\(S\): The total number of sites in the system.

\(F_s\): The average fraction of the sites where data objects reside. It augments the previous parameter in methods where data is not necessarily fully-redundantly duplicated; it represents the average duplication factor.

\(O_d\): The average number of data objects in the system. It represents the size of the data dictionary, where there is an entry for each data item. It is used in some TSO or optimistic methods where timestamps are stored with the data items, independent of transaction executions.

### 2.6 OVERHEADS

The total amount of storage a concurrency control method utilizes during its execution is determined, based on the sizes of the CC-DB relations used. The unit of storage overhead is taken as one field of one tuple of one CC-DB relation.
The analysis of CPU overhead is based on the total number of CC-DB accesses when executing the macros. The unit of CPU overhead is taken as one tuple access, insertion or replacement in one CC-DB relation.

The analysis of message overhead is based on the total number of messages required when executing the macros. The unit of message overhead is taken as one message sent to a single site.

3. MODELING DISTRIBUTED FULLY-REDUNDANT TWO-PHASE LOCKING

The use of 2PL as a distributed concurrency control method was discussed in detail by Bernstein & Goodman [BER81], Ceri & Pelagatti [CER85] and many others.

A simple scenario for 2PL is where locking is applied to all the copies of data items duplicated at all the sites. The locking primitives issued by the site of origin are sent to all the sites, where local locks are applied to the data item copies. Transactions discover their conflicts at all the sites where they request locks.

2PL as such is sufficient to provide serializability, but it does not provide atomicity. If a transaction releases its exclusive locks during the shrinking phase, any transaction can observe its results prior to its commitment. To guarantee isolation, a transaction must satisfy a stronger form of 2PL where all exclusive locks are held until commitment.

In order to guarantee global consistency, 2PC must also be incorporated in the protocol to ensure that all the updates on the local data item copies are either all committed or aborted.

3.1 CONCURRENCY CONTROL DATABASE

The following CC-DB fields and relations are used:

- TSI(t-id, state)
- ACC(obj-id, t-id, mode)
- BLKD(t-id, cause-id)

Timestamps are not used; neither are transactions conditionally granted, thus the HIST relation is not used. Although it is a distributed method, the site relation is not used as the data objects are duplicated and locked at all sites. An ACC relation entry models the lock set on the particular data object.
3.2 CONFLICT SITUATIONS

As locking takes place globally, block and restart conflicts will be detected locally at all the sites.

BLOCKING

A blocking conflict occurs if there is an access entry for a requested data object, with a conflicting access mode. It depicts the conflict where a transaction wishes to write or read a data object already locked by another transaction. This macro is in liberal pseudocode; the identifier of the conflicting transaction is also returned in the cause-id parameter; the calling macro must document the blocking conflict.

\[
\text{BLOCKCONFLICT}(t\text{-id}, \text{obj}\text{-id}, \text{mode}, \text{VAR cause-id}): \text{BOOLEAN}
\]

\[
\text{cause-id} := -
\]

\[
\text{cause-id} := \text{any ACC.t-id}
\]

of the conflicting transaction is also returned in the cause-id parameter; the calling macro must document the blocking conflict.

\[
\text{BLOCKCONFLICT}(t\text{-id}, \text{obj}\text{-id}, \text{mode}, \text{VAR cause-id}): \text{BOOLEAN}
\]

\[
\text{cause-id} := -
\]

\[
\text{cause-id} := \text{any ACC.t-id}
\]

where (not (t\text{-id} = ACC.t\text{-id})
and obj\text{-id} = ACC.obj\text{-id}
and (mode = "write"
or (mode = "read"
and ACC.mode = "write")))

\[
\text{BLOCKCONFLICT} := \text{not (cause-id = -)}
\]

RESTARTING

A restarting conflict occurs if the requested transaction is involved in a deadlock cycle. This exists if two or more transactions are waiting for each other in a circular fashion in such a manner that neither can proceed; depicted by entries in the blocked relation forming a cyclic path containing the transaction identifier.

\[
\text{RESTARTCONFLICT}(t\text{-id}): \text{BOOLEAN}
\]

\[
\text{RESTARTCONFLICT} := \text{CYCLE}(t\text{-id})
\]

and (BLKD.cause-id = t\text{-id}
or BLKD.t\text{-id} = t\text{-id})

3.3 TRANSACTION INITIATION

On transaction initiation, the root agent (RA) at the transaction's site of origin must broadcast the begin operation to all the remote sites.
On BEGIN(t-id):
    SEND("L-BEGIN(t-id)", all-other-sites)
L-BEGIN(t-id)

A local begin records the active transaction at the site.

On L-BEGIN(t-id):
    append TSI(t-id, "ready")

3.4 ACCESS REQUESTS

In this fully-redundant, overly strong 2PL implementation, a global read access request is broadcasted to all the sites where it is executed locally. All the sites must apply a local lock to the data object before accessing it.

On READ-REQ(t-id, obj-id):
    SEND("L-REQ(t-id, obj-id, "read")", all-other-sites)
    L-REQ(t-id, obj-id, "read")

The macro for global write requests is similar to the read request macro. The only difference between modeling read and write operations is in the conflict situations; these appear in the blocking conflict macro.

A local access operation is executed if the necessary lock can be granted; otherwise the transaction is blocked and even restarted if its blocking causes a deadlock cycle.

On L-REQ(t-id, obj-id, mode):
    If not BLOCKCONFLICT(t-id, obj-id, mode, -)
    and (TSI.t-id = t-id
    and TSI.state = "ready") Do
    {Set the lock and access the requested data item}
    append ACC(obj-id, t-id, mode)
      where not (ACC.obj-id = obj-id
      and ACC.t-id = t-id)
    Else BLOCKCONFLICT(t-id, obj-id, mode, cause-id)
      or (TSI.t-id = t-id
      and not (TSI.state = "ready")) Do
      {Block the transaction}
    BLOCK(t-id, cause-id)
If RESTARTCONFLICT(t-id) Do
  {Unblock the transactions blocked by the restarted transaction}
  replace TSI(state := "ready")
  where (TSI.state = "blocked"
  and TSI.t-id = BLKD.t-id
  and BLKD.cause-id = t-id)
  {Restart the transaction causing the deadlock}
  RESTART(t-id)
Endif
Endif

3.5 TRANSACTION COMMITMENT

Transaction commitment includes 2PC, with the RA as coordinator.

In the preparation phase, all the sites are prompted by a message to determine their willingness to commit. The sites determine if the transaction is not blocked by a locked data item or if it has not been restarted to resolve a deadlock.

In the implementation phase the sites indicate their readiness to commit by returning ready or not ready messages. Only if all the sites answer positive, will the RA decide to commit, which is activated by broadcasting local commits to all the sites; otherwise an abort is broadcasted to all the sites. The timeout mechanism is used to detect sites which are not ready to commit due to their inactivity.

The RA's site identifier is included in the preparation, commit and abort messages to indicate where the replies must be returned to.

On COMMIT(t-id):
  VAR msg
  {1st preparation phase}
  SEND("L-PREPARE(t-id, site)", all-other-sites)
  L-PREPARE(t-id, site)
  Activate timeout
  {2nd implementation phase}
  RECEIVE(msg, all-other-sites)
If timeout (at least one site not ready)
or any (msg = "not-ready") Do
SEND("L-ABORT(t-id, site)", all-other-sites)
L-ABORT(t-id, site)
{Restart the transaction globally}
RESTART(t-id)
Else all (msg = "ready") Do
SEND("L-COMMIT(t-id, site)", all-other-sites)
L-COMMIT(t-id, site)
Endif
RECEIVE(acks)
Return status

During preparation the sites control the state of the transaction and make sure that it is not still
blocked by another transaction. The site's readiness or not to commit is returned to the RA.

On L-PREPARE(t-id, site-id):
\[
\text{If (any (BLKD.t-id = t-id) or (not TSI.state = "ready" \and TSI.t-id = t-id)) Do}
\]
\{not willing to commit\}
SEND ("not ready", site-id)
\[
\text{ELSE (not any (BLKD.t-id = t-id) and TSI.state = "ready" \and TSI.t-id \neq t-id) Do}
\]
\{willing to commit\}
SEND ("ready", site-id)
Endif

Local transaction commitment entails changing the transaction's state and making all the DB
changes permanent. All the blocked transactions waiting for locks set by the transaction must
be reactivated when the locks are released. The sites indicate their completion by returning
acknowledgements to the RA.

On L-Commit(t-id, site-id):
\[
\text{replace TSI(state := "committed")}
\]
\{where TSI.t-id = t-id\}
During a local abort, apart from unblocking the transactions blocked by this transaction, the effects of the transaction must be nullified. After expunging the transaction, the RA is informed by an acknowledgement.

On L-ABORT(t-id, site-id):

{Unlock the transactions blocked by the restarted transaction}
replace TSI(state := "ready")
where (TSI.state = "blocked"
and TSI.t-id = BLKD.t-id
and BLKD.cause-id = t-id)
{Undo all the DB changes}
RESTART(t-id)
SEND("ack", site-id)

3.6 OVERHEADS

STORAGE OVERHEAD

All the relations are replicated at every site.

The TSI relation has 2 fields. It contains the states of all active and all re-initiated fatally conflicting transactions. It represents a cost of $2T^a_S + 2T^a_Frc_S$.

The ACC relation has 3 fields. It documents the read locks of each transaction and each fatally conflicting transaction; numerous read locks can be set on every data item. It represents a cost of $3T^a_R_S + 3T^a_R_Frc_S$.

The ACC relation also documents all the write locks of each active and each restarted transaction; only one lock can be set on each data item. It represents a cost of $3T^a_W_S + 3T^a_W_Frc_S$ in the worst case (where the transactions have write requests to
different data items) to \(3S + 3F_{rc}S\) in the best case (where all the transactions have a write request to one data item).

The BLKD relation has 2 fields. It contains an entry for every blocked access request; a transaction can only be blocked at one access. Fatally conflicting transactions must also be blocked before deadlocks can be detected. It represents a cost of \(2T_aF_{rc}S\) plus \(2T_aF_{rc}S\).

Thus, the total storage overhead is given by the expression \(2T_aS + 2T_aF_{rc}S + 4T_aF_{rc}S + 3T_aR_{a}S + 3T_aR_{a}F_{rc}S + 3S + 3F_{rc}S < \text{STO}_{DZPL} < 2T_aS + 2T_aF_{rc}S + 4T_aF_{rc}S + 3T_aR_{a}S + 3T_aR_{a}F_{rc}S.\)

**CPU OVERHEAD**

All the operations are executed at every site.

A BEGIN operation involves a single access to document new and restarted transactions. It costs \(1T_aS\) plus \(1T_aF_{rc}S\).

Any non-blocking access request involves one access to determine that the data item is not locked, one access to determine that the transaction is active and one access to lock the data item. It costs \(3T_aR_{a}S\), plus \(3T_aR_{a}F_{rc}S\) for restarted transactions.

Every blocked access request, not involved in a deadlock cycle, involves one access to determine that the data item is locked, two accesses to change the transaction's state and note the deadlock detection information, then \(D\) accesses to determine it is not involved in a deadlock cycle. It costs \((3 + D)T_aR_{a}F_{rc}S\).

Any fatally conflicting request involves the same number of accesses as in the previous case to detect that it is involved in a deadlock cycle and an additional \((2 + R_{a})\) accesses to restart the transaction. It costs \((5 + R_{a} + D)T_aR_{a}F_{rc}S\). In addition it costs three accesses to unblock the blocked transactions, thus \(3T_aF_{rc}S\) assuming one transaction becomes unblocked or \(3T_aF_{rc}(T_aF_{rc})S\) if all the transactions become unblocked.

A committing COMMIT operation involves two accesses during preparation to determine if the transaction is ready to commit; during commitment it involves one access to change the transaction's state, then an additional \((2 + R_{a})\) accesses to remove the transaction. It costs \((5 + R_{a})T_aS\), plus the possible maximum of \(3T_aF_{rc}S\) for unblocking.

A globally restarting COMMIT operation involves two accesses during preparation to determine that the transaction has either already been aborted, or is still blocked; during the abort phase it involves one access to change the transaction's state and an additional \((2 + R_{a})\) accesses to
remove the transaction. Thus it costs \((5 + R_a)T animations_8 F_{fc}S\), plus the possible maximum of \(3T_a F_{fc}S\) for unblocking.

Thus, the total CPU overhead is given by the expression:

\[
6T_a + 9T_a F_{fc}S + 4T_a R_a S + (3 + D)T_a R_a F_{fc}S + (9 + R_a + D)T_a R_a F_{fc}S < CPU_{cpl} < 6T_a S + 3T_a F_{fc}S + 3T_a F_{fc}S + 4T_a R_a S + (3 + D)T_a R_a F_{fc}S + (9 + R_a + D)T_a R_a F_{fc}S + 2T_a F_{fc}S.
\]

**MESSAGE OVERHEAD**

A BEGIN operation involves one message sent to each other site for new and fatally conflicted transactions. It costs \(1T_a (S - 1) + 1T_a F_{fc}(S-1)\).

Any access requests of transactions and restarted transactions involve one message sent to each other site. It costs \(1T_a R_a (S - 1) + 1T_a R_a F_{fc}(S-1)\).

Any COMMIT operation involves one message sent to each other site for preparation, one answer back from each, one commit or abort message sent to each and then one acknowledgement back from each. It costs \(4T_a (S - 1) + 4T_a F_{fc}S(S-1)\) for transactions which were globally restarted.

Thus, the total message overhead is given by the expression:

\[
MES_{cpl} = 5T_a (S - 1) + 5T_a F_{fc}(S - 1) + 1T_a R_a (S - 1) + 1T_a R_a F_{fc}(S - 1).
\]

**4. VIRTUES OF THE MODEL**

The prime advantage of the model is that it provides a framework for representing concurrency control methods. The differences between the methods are standarized by the uniform representations. This leads to meaningful quantitative overhead comparisons.

Another advantage of the model is that it is easy to use. Researchers in database directions use it comfortably.

The SITE relation representing the data dictionary allows a researcher to evaluate the effects of different data dictionary implementations.

An aspect not obvious from this paper, is that the storage overhead of a data object lock (three ACC fields) is represented as being less than a data object timestamp (four HIST fields). The storage of transaction timestamps also result in higher storage costs.
5. SHORTCOMINGS OF THE MODEL

A problem of the model is that it is not always clear how to depict complex concurrency control operations using macros or how to to represent complex data structures using the CC-DB. Although usually representable in some ad hoc manner, it can contribute a greater source of overhead than a realistic representation.

During the analytical comparison of methods based on different mechanisms (eg locks, timestamps and optimism), the conflict parameters are assumed relatable [CAR83],[REN86]. The validness of this assumption has not been proved.

Apart from the fraction of recently committed transactions parameter, no attention has been given to the duration of storage overheads. In the modeling of conflicts, time duration is intuitively implied when re-executing conflicting transactions [REN86]. These aspects can be analyzed in detail. Similarly, the sizes of messages have not been parameterized. In some optimistic methods the entire effect of a transaction is broadcasted as a single message. Such messages incur greater message overheads, but is not accurately depicted by the model.

It is not clear how to rate methods when they differ with respect to different overheads. In the evaluations made, the different overheads have been considered weighing equal [REN86]. This assumption should be justified by analyzing the relative contribution of each factor.

CONCLUSION

Although an actual comparison was not presented here, due to space limitations, it can be concluded that the abstract model of distributed database concurrency control methods can be used for meaningful analytical comparisons, despite its shortcomings.

The model can be used as a framework for future research. It should be modified to eliminate its discrepancies and it can be extended for the evaluation of aspects closely integrated with concurrency control, such as recovery or query decomposition methods.

It should be kept in mind that overheads are not the only criteria for comparisons. Related criteria include the degree of concurrency provided, to determine whether the amount of concurrency achieved relatively merrits the overheads incurred by a method.
REFERENCES

[BER81] P.A. Bernstein & N. Goodman
Concurrency Control in Distributed Database Systems

[CAR83] M.J. Carey
An Abstract Model of Database Concurrency Control Algorithms
ACM SIGMOD Record, Vol 13, No 4, 1983.

[CER85] S. Ceri & G. Pelagatti
Distributed Databases - Principles and Systems

[REN86] M.H. Rennhackkamp
Evaluation of Concurrency Control Methods in Distributed Databases
M.Sc thesis, Department of Computer Science, University of Stellenbosch, 1986.