CONCURRENCY CONTROL AND RELIABILITY IN DISTRIBUTED SYSTEMS

Edited by

Bharat K. Bhargava
Department of Computer Science
Purdue University
West Lafayette, Indiana

VNR VAN NOSTRAND REINHOLD COMPANY
New York

Library of Congress Catalog Card Number 86-19121
ISBN 0-442-21148-1

Printed in the United States of America

Van Nostrand Reinhold Company Inc.
115 Fifth Avenue
New York, New York 10003

Van Nostrand Reinhold Company Limited
Molly Millars Lane
Wokingham, Berkshire RG11 2PY, England

Van Nostrand Reinhold
480 La Trobe Street
Melbourne, Victoria 3000, Australia

Macmillan of Canada
Division of Canada Publishing Corporation
164 Commander Boulevard
Agincourt, Ontario M1S 3C7, Canada

16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data

Concurrency control and reliability in distributed
systems.

"May 1986."
Bibliography: p.
Includes index.
1. Electronic data processing—Distributed processing.
I. Bhargava, Bharat, 1948-
QA76.9.D5C67 1987 004'.36 86-19121
ISBN 0-442-21148-1

CONCURRENCY CONTROL AND RELIABILITY IN DISTRIBUTED SYSTEMS

To All the Contributors

Contributors

Bharat K. Bhargava, *Purdue University*
David Butterfield, *LOCUS Computing Corporation*
Bo-Shoe Chen, *Bell Laboratories*
Douglas E. Comer, *Purdue University*
Dean Daniels, *Carnegie-Mellon University*
Danny Dolev, *Hebrew University, Israel*
Daniel Duchamp, *Carnegie-Mellon University*
Robert English, *University of California at Los Angeles*
Jeffrey L. Eppinger, *Carnegie-Mellon University*
Hector Garcia-Molina, *Princeton University*
Cecil T. Hua, *Honeywell*
Jack Kent, *Xerox PARC*
Kane H. Kim, *University of California at Irvine*
Charles Kline, *LOCUS Computing Corporation*
Walter H. Kohler, *University of Massachusetts*
Leslie Lamport, *DEC Research Center (WRL)*
Leszek Lilien, *University of Illinois at Chicago*
Barbara Liskov, *Massachusetts Institute of Technology*
Toshimi Minoura, *Oregon State University*
J. Eliot B. Moss, *University of Massachusetts*
Susan S. Owicki, *Stanford University*
Thomas Page, *University of California at Los Angeles*
F. Panzieri, *The University of Pisa, Italy*
Randy Pausch, *Carnegie-Mellon University*
Marshall Pease, *SRI International*
Larry L. Peterson, *University of Arizona*
Gerald Popek, *University of California at Los Angeles*
Krithi Ramamritham, *University of Massachusetts*
Joylyn N. Reed, *Oxford University*
P. Rolin, *INRIA, France*
Zuwang Ruan, *Purdue University*
Robert Scheifler, *Massachusetts Institute of Technology*
Fred B. Schneider, *Cornell University*
Robert Shostak, *ANSA Software*
Santosh K. Shrivastava, *The University of Newcastle upon Tyne, UK*

Dale Skeen, *TEKNEKRON, Inc.*
Alfred Z. Spector, *Carnegie-Mellon University*
John A. Stankovic, *University of Massachusetts*
Michael Stonebraker, *University of California at Berkeley*
Greg Thiel, *LOCUS Computing Corporation*
Bruce Walker, *LOCUS Computing Corporation*
Gio Wiederhold, *Stanford University*
Raymond T. Yeh, *SYSCORP International*

Foreword

Computer users have been building distributed systems for years now, though each one has been built as a special case. Until recently, there have been few principles available to help in the design of such systems. Largely, they have been built by trial and error—mostly error.

Based on many experimental and production distributed systems, several approaches have been discarded, while a few have survived. For some issues, elegant and powerful abstractions have emerged—notably, the concepts of transaction, recovery, locking, atomicity, fail-stop, Byzantine agreement, distributed naming, distributed execution, and remote procedure calls.

This book collects the best current work on these topics, presented by active workers in the field. Taken as a whole, it presents an exciting and dynamic experimental discipline. Using the ideas presented here, one can understand and solve many of the problems encountered in the construction of distributed computer systems.

Jim Gray
Cupertino, California

Preface

Distributed systems are essential for many real-world applications, ranging from space stations to automatic teller machines. Several design principles necessary to build high performance and reliable distributed systems have evolved from conceptual research and experimentation during the eighties. This book is a compendium of these principles. It contains definitions and introductory material for the beginner, theoretical foundations and results, experiences with implementations of both real and prototype systems, and surveys of many important protocols. This book focuses on the important aspects of transaction processing, including: concurrency, commitment, and recovery as they apply to database systems, operating systems, or programming systems. The manuscript for this book has been used in a graduate course on distributed database systems at Purdue University. The book can be used in a graduate course on distributed systems and can serve as a good reference material for research. The material has been presented to demonstrate the practicality of certain successful designs and implementation choices, so as to benefit the systems programmers and those who have the responsibility for making distributed systems work.

Bharat Bhargava

Introduction

A distributed system consists of a set of computers located in different sites connected by means of a communications network. There are different programs running on each of these computers and the programs are accessing local or remote resources such as databases. The programs can be viewed as transactions which consist of atomic actions. The resources may be partially replicated or partitioned.

The major objective of a distributed system is to provide low cost availability of the resources of the system by localizing access and providing insulation against failures of individual components. Since many users can be concurrently accessing the system, it is essential that a distributed system also provide a high degree of concurrency.

In the last decade much research has been conducted to develop algorithms that provide:

- A high degree of concurrency and consistency
- Transparency to failures of individual sites and communication systems
- Treatment of Byzantine failures
- Management of replicated copy
- Commitment and termination of transactions ensuring atomicity

Many of these algorithms have been implemented in experimental prototype systems such as Argus,[1] Distributed INGRES,[2] EDEN,[3] LOCUS,[4] RAID,[5] SDD-1,[6] SYSTEM R*,[7] TABS,[8] etc. In addition, several important concepts such as transaction,[9] nested transaction,[10] and naming,[11] have been studied in depth to support the development of distributed systems.

This book is a compilation of a subset of the research contribution in the area of concurrency control and reliability in distributed systems. An attempt has been made to cover all interesting areas, including the theoretical and experimental efforts. Thirty-nine computer scientists contributed their research papers for the nineteen chapters that constitute this book.

SUMMARY OF CHAPTERS

The first chapter contains a review of various reliability issues in distributed database systems. It contains definitions of most terms used in this area of research, and identifies the various problems that arise in the operations of a distributed database system.

First of all, the integrity of the database must be maintained. A transaction may be incorrect and may violate this integrity. Concurrent transactions may not be serializable or may produce different updates on different sites. The messages from one site may not reach the other sites in the order they were sent or may arrive too late. In addition, due to a hardware/software malfunction or failure, a site may crash or a set of sites may not be able to communicate with other sites. A failure may cause the loss of certain actions of a transaction, or destroy the contents of the memory and other storage media. Each of these problems can be handled by a subsystem. Six subsystems have been identified: external data integrity control, program (transaction) correctness control, atomicity control, concurrency control, site crash/partition treatment, and internal data integrity control. A survey of algorithms for each subsystem is given and approximately one hundred and seventy-five references are given for further studies.

The second chapter contains the specifications and analysis of concurrency control algorithms for distributed database systems. An event order based model has been used to represent casual relations among actions of a concurrency controller. The model is used as a tool to study the correctness, degree of concurrency, freedom from deadlocks, and robustness properties of a concurrency control algorithm.

The third chapter presents the fundamental concepts used in the design of the experimental system, Argus, that is being developed at MIT. The system provides support for constructing and maintaining distributed programs. A module called **Guardian** is introduced for dealing with concurrency and site failures.

The fourth chapter presents the architecture of the experimental distributed operating system LOCUS implemented at UCLA. LOCUS is UNIX compatible and connects 17 VAX/750's via Ethernet. It provides a distributed file system, distributed (remote) process execution, support for nested transactions and distributed database systems, and high reliability features to deal with recovery of files and network partitioning. The system supports a certain level of heterogeneity.[12]

The fifth chapter describes an experimental distributed database system called SIRIUS-Delta that has been implemented in INRIA, France. The use of principles such as two phase locking, two phase commit, and maintenance of journals to deal with failures are included.

The sixth chapter presents an experimental facility being developed at Carnegie-Mellon University, TABS, that provides operating system-level support for distributed transactions that operate on shared abstract types. The objects in this system are instances of abstract data types and are encapsulated in processes called *data servers*. Data servers also support the synchronization requirements of transactions by using type-specific locking. The system provides extensive support for atomic objects via write-ahead logging.

There are two basic types of communication primitives: message passing and remote procedure call (RPC). While LOCUS and TABS use primitives such as send and receive, Argus uses the remote procedure call (RPC) as the basic means of interprocess communications. Chapter 7 discusses the remote procedure call (RPC) and the functionality of the underlying interprocess communication facility. It discusses the call semantics and exceptions handling, orphans handling, and robust atomic action. Performance results of initial tests for data transfer rates as seen by the user for local and remote operations are compared.

Chapter 8 discusses practical mechanizations based on the monitor approach to interprocess communications. The coordination of the detection and recovery activities of cooperating processes is discussed. This work contributes towards the design of fault-tolerant concurrent programming.

A formal model for commit protocols for a distributed database systems is given in Chapter 9. The commit protocol is specified as a collection of nondeterministic finite state automata—one for each site. A local state transition consists of reading one or more messages, performing some local processing, and sending zero or more messages. Independent site recovery and network partitions are modelled and conditions are specified for correct recovery and termination of transactions. It is concluded that there exists no protocol using independent recovery that is resilient to arbitrary failures by two sites and there exists no protocol resilient to multiple partitions.

It is essential that concurrency control and reliability mechanisms support each other. Chapter 10 discusses an optimistic approach to concurrency control and demonstrates how it lends itself naturally towards the design of a reliable distributed database system. Optimistic concurrency control is based on the validation of conflicting transactions just before commit rather than the locking of database objects. The optimistic approach has been found to provide good performance when there is a mix of large and small transactions.[13] The optimistic approach is also good for network partition treatment protocols.

In a distributed system, sites may be up, down, or recovering. The basic idea for correct recovery is that a user transaction must have a consistent view of the status (up or down) of the sites at the time of reading and writing any physical copy. This problem becomes very interesting when the databases are replicated over several sites. Chapter 11 deals with the problem of replicated copy control and transaction processing when failed sites are recovering. A global view of the status is represented by a nominal session vector which contains the session number (incarnation number) of each operational site. The identification of out-of-date data items for the recovering site is done by using fail-locks on the operational sites.

The Byzantine Generals problem has drawn much attention from many researchers since it deals with not only clean failures of components of a distributed system but also malfunctions that give conflicting information to different parts of the system. The fundamental problem is the agreement on a piece of data based on the cooperation among several processes. Chapter 12 presents algorithms to ensure that all correctly functioning processors reach an agreement. It is shown that, with unforgeable written messages, the algorithms are possible for any number of processors and possible malfunctioning components. Applications of the solutions to the Byzantine Generals problem to reliable computer systems are discussed.

Many applications do not require a perfect system that must tolerate an arbitrary number of failures within a certain time interval. In the fail-stop processor approach presented in Chapter 13, it is possible to build systems that can tolerate well-defined failures. The operating characteristics of a fail-stop processor, programming fail-stop processors, illustrative examples, and the implementation issues have been presented in this chapter. Convincing arguments for studying the fail-stop processor are given.

Nested transactions can model user's programs in a much more natural way. A typical transaction for travel plans may involve three subtransactions: get a seat on a plane, reserve a rental car, and book a room in a hotel. Similarly, an update on an object with several copies may be issued as a nested transaction with three subactions corresponding to each update. Chapter 14 presents the model of a nested transaction and gives a nested transaction management algorithm for distributed systems. Convincing arguments are given to justify the concept of nested transactions for reliable distributed software. Algorithms for deadlock detection, avoidance, and resolution are given.

Chapter 15 presents the true-copy token scheme for concurrency and reliability control in a distributed database system. True-copy tokens are used to designate the most up-to-date physical copies. This chapter presents the regeneration of lost tokens (or true copies). This research contributed towards the management of replicated copies in the presence of site failures and network partitions.

Chapter 16 deals with the issue of performance evaluation of the reliability of a distributed system. A collection of models and measures have been proposed and then used to obtain sample results on commit protocols, numbers of blocked sites, numbers of backed out transactions, etc.

A distributed system provides users with access to a variety of objects such as files, processing agents, devices, etc. The users identify objects in the same manner, whether they are available locally or remotely. This transparency to location allows software to migrate both before and during its execution. The name resolution mechanism performs the mapping of names (as defined by users) to objects as they exist in the system. Chapter 17 presents a survey of naming mechanisms supported by several experimental distributed systems. It presents a model that describes the underlying principles and concepts of naming and investigates the role of name servers in name resolution. The model views names to be purely syntactic entities and name resolution to be a syntax-directed operation.

Chapter 18 describes a specification and verification technique for concurrent distributed systems. Specification refers to a concise mathematical characterization of the functionality of programs actually implemented with code. The technique is used to study liveness properties of concurrent programs. The technique is illustrated with communicating sequential processes. Illustrative examples that use this technique have been included.

The last chapter contains a survey of current efforts in distributed systems software and in distributed database systems.

In the area of operating systems software, three classes have been identified: network operating systems, distributed operating systems, and distributed processing operating systems. Most experimental systems have been classified in these classes. The choices for structuring distributed systems have been discussed next. Other issues such as addressing distributed processes, communication primitives, decentralized control, and distributed file systems have been discussed.

In the area of distributed database systems, the transaction model, data replication, deadlock resolution, and recovery have been summarized and several test-beds have been identified. Once again, over two hundred references (some of which overlap with those in the first chapter) have been listed.

FUTURE RESEARCH ISSUES

The research in concurrency control for distributed systems appears to be maturing while there is great activity in the area of reliability. The failure treatment in replicated copy management needs further studies. Site failure and network partition treatment protocols that include detection and recovery from failures need to be developed. There has been good research in the area of clock synchronization, but the incorporation of these algorithms into experimental systems is still awaited. One of the primary applications of distributed systems is high performance transaction processing systems. The system level support to process up to one thousand transactions per second will be needed. Hopefully, the research in parallel processing will help. Still, the distributed systems which appear more like distributed file systems need to incorporate more of such notions as a global process and shared memory. Transparency to location needs to be built into the system itself. We are still waiting for commercially available distributed systems.

Fortunately, the research is proceeding at a fast pace. This is evident from specialized symposiums (e.g., the IEEE Computer Society's Symposiums on

Reliability in Distributed Software and Database Systems), workshops, and special issues of journals devoted to concurrency and reliability problems. We look forward to the work of our colleagues in academia, industry, and government.

ACKNOWLEDGMENTS

This book is possible only due to the brilliant research of the contributors of all the chapters. I am indebted to them for providing me with the opportunity to produce this book. Their research ideas will benefit the research of the readers. Graduate students in particular will find this work of great interest.

REFERENCES

1. Liskov, B. and R. Scheifler, On linguistic support for distributed programs, *IEEE Trans. Software Eng.*, vol. SE-8, pp. 203–210, May 1982.
2. Stonebraker, M. The design and implementation of distributed INGRES, in *The INGRES Papers*, (M. Stonebraker, ed.), Reading, MA: Addison Wesley, 1985.
3. Jessop, W. H., J. D. Noe, D. M. Jacobson, J. L. Baer, and C. Pu, The Eden transaction-based file system, in *Proc. 2nd IEEE Symp. on Reliability in Distributed Software and Database Systems*, Pittsburgh, PA, July 19–21, 1982.
4. Walker, B., G. Popek, R. English, C. Kline and G. Thiel, The LOCUS distributed operating system, in *Proc. 9th ACM Symp. on Operating Systems Principles*, Bretton Woods, NH, Oct. 10–13, 1983.
5. Bhargava, B. and J. Riedl, The design of an adaptable distributed system, *Proc. IEEE COMPSAC*, Chicago, IL, Oct. 1986.
6. Rothnie, J., P. A. Bernstein, S. Fox, N. Goodman, M. Hammer, T. A. Landers, C. Reeve, D. W. Shipman, and E. Wong, Introduction to a system for distributed databases (SDD-1), *ACM Trans. Database Systems*, pp. 1–17, March 1980.
7. Lindsey, B. G., L. M. Haas, C. Mohan, P. F. Wilms, and R. A. Yost, Computation and communication in R*; A distributed database manager, *ACM Trans. Computer Systems*, pp. 24–38, Feb. 1984.
8. Spector, A., J. Butcher, D. S. Daniels, D. J. Duchamp, J. L. Eppinger, C. E. Fineman, A. Heddaya, and P. M. Schwartz, Support for distributed transactions in the TABS prototype, *IEEE Trans. Software Eng.*, vol. SE-11, June 1985.
9. Gray, J. N., *The transaction concept: Virtues and limitations*, Very Large Database Conference, pp. 144–154, Sept. 1981.
10. Moss, J. E. B., Nested transactions and reliable distributed computing, *Proc. 2nd IEEE Symp. on Reliability in Distributed Software and Database Systems*, pp. 33–39, July 1982.
11. Comer, D. and L. Peterson, *A name resolution model for distributed Systems, Proc. 6th IEEE Int. Conference on Distributed Computing Systems*, Cambridge, MA, May 19–23, 1986.
12. Popek, G. and B. Walker, *The LOCUS Distributed System Architecture*, Cambridge, MA: MIT Press, 1986.
13. Bhargava, B., Performance evaluation of the optimistic concurrency control approach to distributed database systems and its comparison with locking, *Proc. 2nd IEEE Int. Conference on Distributed Computing Systems*, Miami, FL, October 18–22, 1982.

CONCURRENCY CONTROL AND RELIABILITY IN DISTRIBUTED SYSTEMS

Contents