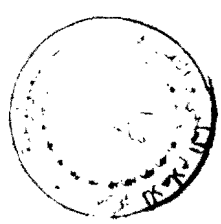
A LOCKING PROTOCOL FOR RESOURCE COORDINATION IN DISTRIBUTED DATABASES.

DANIEL A. MENASCET.



A LOCKING PROTOCOL FOR RESOURCE COORDINATION IN DISTRIBUTED DATABASES*

Daniel A. Menasce†, Gerald J. Popek and Richard R. Muntz

Computer Science Department
University of California
Los Angeles, California 90024

ABSTRACT: A locking protocol to coordinate access to a distributed database and to maintain system consistency throughout normal and abnormal conditions is presented in this paper. The protocol is robust in the face of failures of any participating site and in the face of network partitioning. The proposed protocol supports the integration of virtually any locking discipline including predicate locking methods. A cost and delay analysis of the protocol as well as a proof of its correctness is included in this work. The paper concludes with a proposal for an extension aimed at optimizing operation of the protocol to adapt to highly skewed distributions of activity.

KEYWORDS AND PHRASES: concurrency, crash recovery, distributed databases, locking protocol, consistency.

Introduction

This paper is concerned with issues of resource coordination in distributed systems, and the maintenance of system consistency throughout normal and abnormal conditions. A database is said to be in a consistent state if all the data items satisfy a set of established assertions or consistency constraints. A database subject to multiple access requires that accesses to it be properly coordinated in order to preserve consistency. Coordination of resources in a distributed environment exhibits additional complexity over resource coordination in centralized environments due to:

1. possibility of crashes of participating sites and or communication links. Occurrence of such failures can render the database inconsistent if not appropriately handled by the coordination algorithm.

2. network partitioning: in general, it is not possible to distinguish between messages which could not be delivered due to a crash of the recipient site and undelivered messages due to network partitioning. Therefore, network partitioning in the more general sense considered here is not simply a matter of proper network topology design. It turns out that detection of network partitioning can only occur at network reconnection time.

3. inherent communication delay: the time to get a message through a computer communication network may be arbitrarily long, although finite. Therefore any proposed solution should operate correctly regardless of the delay experienced by any message, and in general should be efficient.

A protocol to coordinate concurrent access to a distributed database using locking is presented in this paper. The algorithm has as its core a centralized locking protocol with distributed recovery procedures. A centralized controller with local appendages at each site coordinates all resource control, with requests initiated by application programs at any site. Recovery is broken down into three disjoint mechanisms; for single node recovery, merge of partitions and reconstruction of the centralized controller and tables.

Among the properties of the proposed protocol we have:

a. robustness in the face of crashes of any participating site, as well as communication failures, is provided. The protocol can recover from any number of failures which occur either during normal operation or during any of the three recovery processes.

---

* This research was supported by the Advanced Research Projects Agency of the Department of Defense under Contract MDA 903-77-C-0211.

† Partially supported by the Conselho Nacional de Desenvolvimento Cientifico e Tecnologico, CNPq, Brazil.

b. deadlock prevention and or detection methods can be easily integrated given the centralized control characteristic of the proposed algorithm.

c. straightforward integration of predicate locking methods [1] is permitted. Value dependent lock specification at the logical level is necessary to avoid the problems of "phantom tuples" discussed by Eswaran et al [1]. Other locking disciplines may also be easily supported.

d. continued local operation in the face of network partitioning is supported. The locking algorithm operates, and operates correctly, when the network is partitioned, either intentionally or by failure of communication lines. Each partition is able to continue with work local to it, and operation merges gracefully when the partitions are reconnected.

e. performance of the algorithm does not degrade operations. It is shown in this paper that for many topologies of interest, the delay introduced by the protocol is not a direct function of the size of the network. The communication cost is shown to grow in a relatively slow, linear fashion with the number of sites in the network.

f. the correct operation of the protocol in the face of the failures mentioned before can be proven in a straightforward way.

Several other approaches for synchronization in distributed databases have been suggested in the literature, but none deal satisfactorily with all of these issues.

The Majority Consensus protocol proposed by Thomas[8] requires the sites involved in a transaction to agree by majority vote for it to proceed. Timestamps on data items at each site indicate whether the item is current and therefore whether a transaction based on it can be approved.

This protocol is quite elegant, with attractive behavior in the face of failures, especially for fully replicated databases. Unfortunately, for the cases considered in this paper, it presents several drawbacks. The locking discipline and scheduling of transactions are fixed by the nature of the algorithm itself, limiting flexibility (predicate locking cannot be supported for example). Performance can degrade severely with increasing system load in a thrashing like manner, since several partially complete transactions which conflict lead to multiple resubmission of each.

Synchronization in SDD-1 [4] is handled by several different protocols designed to co-exist with one another. The simpler ones can be used for certain restricted classes of transactions known in advance of system generation. In such cases significant improvements in cost and delay over more general protocols results. Otherwise however, we recommend our protocol since its performance is absolutely better and issues such as robustness and crash recovery, not handled by SDD-1, are considered fully.

A ring structured solution is proposed by Ellis[6]. It uses sequential propagation of synchronization and update messages along a statically determined circular ordering of the nodes. Two round trips are required for each update. This protocol, while in general much slower than the others mentioned above, is quite simple and Ellis has employed formal verification procedures to show its correctness. Unfortunately however, failures and error recovery are not addressed by the protocol.

Other proposed schemes, called primary copy strategies have been suggested in [3], [5] and [7]. Alsberg in [3] introduced some techniques aimed at providing a certain degree of resiliency to the single primary, multiple backup strategies discussed in [5] and [7]. The primary copy scheme is primarily designed to maintain mutual consistency of databases subject to somewhat limited types of update operations, but it does not address explicitly the problem of internal consistency of a distributed system supporting general transactions.

The protocol presented in this paper is described in an intuitive manner in section one, followed by a more detailed description in the two subsequent sections. An algorithmic specification of this locking protocol can be found in [2]. An informal proof of the correctness of the algorithm is presented here. The proof is decomposed into five major parts, one for normal operation, three for the recovery phases, and a last part that shows the parts actually can be proved disjointly.

The paper concludes with a proposal for an extension aimed at optimizing operation of the algorithm to adapt to highly skewed distributions of activity. The extension applies nicely to interconnected computer networks.

1 - Centralized Lock Controller Protocol - Intuitive Description

The database we are considering here is distributed among n nodes of a computer network, numbered from 1 to n. We assume that the network protocols are such that a copy of a message is kept by its sender until an acknowledgment for it is received. In other words, there are no lost messages. However, messages may have to be retransmitted many times until they get through the net. An implication of these assumptions is that messages may be delayed by an arbitrary but finite amount of time. We also assume that messages from a source site A are delivered to a destination site B in the same order they were generated. However, we make no assumptions about the order in which messages from two distinct sources are received by a third one. We require that the network routing procedures be such that every pair of nodes can communicate with each other if the necessary physical connection is available.

User interaction with the database is done