1-Safe Algorithms for Symmetric Site Configurations

Rune Humborstad, Maitrayi Sabaratnam, Svein-Olaf Hvasshovd
Dept. of Computer and Information Science
Norwegian University of Science & Technology
N-7034 Trondheim, Norway
humbor,maitrayi,sophus@idi.ntnu.no

Øystein Torbjørnsen
Telenor R&D
N-7005 Trondheim, Norway
oytor@fou.telenor.no

Abstract

In order to provide database availability in the presence of node and site failures, traditional 1-safe algorithms disallow primary and hot standby replicas to be located at the same site. This means that the failure of a single primary node must be handled like a failure of the entire primary site. Furthermore, this excludes symmetric site configurations, where the primary replicas are located at the site closest to the accessing clients. In this paper, we present three novel 1-safe algorithms that allow the above restrictions to be removed. The relative performance of these and the traditional algorithms are evaluated by means of simulation studies. Our main conclusion is that the restrictions of the traditional algorithms can be removed without significantly increasing the processing overhead, during normal operation. From an evaluation based on performance, availability, and transaction durability, the novel dependency tracking algorithm provides the best overall solution.

1 Introduction

The high availability requirement of critical database applications is typically met by keeping a hot standby database at a geographically remote site. The hot standby takes over and continues the service, in case a disaster hits the primary. Updates made at the primary must be reflected at the hot standby, in order to keep it consistent. This can be achieved by 1-safe or 2-safe algorithms [GR93, Lyo88]. In 2-safe algorithms, the changes made by a transaction are reflected at both the primary and hot standby before commit. This guarantees the durability of a committed transaction, even in case of disasters. The price paid is increased response time by at least one round trip delay. Long round trip delays will cause problems for the applications demanding stringent response times. The 1-safe algorithms solve this problem at the cost of some committed transactions lost in a disaster. In 1-safe algorithms, transactions commit at the primary without consulting the hot standby. Therefore, if a disaster hits the primary before the updates are propagated to the hot standby, the committed transaction will be lost, referred to as a *missing transaction.* Some bank, travel [PGM92], and telecom [Hva94] applications adopt a 1-safe strategy.

Masking a primary node failure is more costly in the traditional 1-safe than in the 2-safe algorithm. A primary node failure in the 2-safe algorithm is masked by its hot standby *node* taking over, but in the traditional 1-safe algorithms, a *global* take-over is required, i.e., all the hot standby nodes must take over in order to maintain the database consistency. A global take-over causes unnecessarily many active transactions to abort at the primaries. Besides, the unavailability of the database is higher in a global take-over. This is not acceptable for many operationally critical database applications in like telecommunication and process control.

The traditional 1-safe algorithms have an inherent limitation. In order to mask primary failures, they require a *uniform site* configuration, i.e., a site can contain only the primary nodes or only the hot standby nodes. A site failure cannot be masked in *symmetric* configurations, where a site contains both primary and hot standby nodes. However, node failures can be masked in both uniform and symmetric site configurations.

The uniform site configuration is inefficient for the applications having *symmetric load,* i.e., most of the

Proceedings of the 23rd VLDB Conference
Athens, Greece, 1997

transactions originating from a site access the same primary data set. For example, assume a bank that has customers in London and Paris. London is kept as the primary site and Paris as the hot standby site. Most of the transactions that originate from London access the data belonging to London customers and the same applies to those from Paris. Since London is kept as the primary site, even 1-safe transactions originating from Paris suffer a round-trip delay. This may be unacceptable for the Paris customers. A better solution will be: The London site contains the primary data for London customers and the hot standby data for Paris customers, and Paris vice versa. But such a mix will violate the uniform site restriction of the traditional 1-safe algorithms.

In this paper, we introduce three 1-safe algorithms that relax the uniform site restriction and reduce the take-over cost: Extended Locking Algorithm (XL), Dependency Tracking Algorithm (DT), and Extended Epoch Algorithm (EE).

The paper is organized as follows: Section 2 summarizes related work. Section 3 describes the architecture, and motivation for our contribution. Section 4 describes the three novel 1-safe algorithms. Section 5 compares these algorithms with 2-safe and traditional 1-safe algorithms on the basis of a simulation. Section 6 summarizes the main ideas presented in the paper.

2 Related Work

2-Safe and traditional 1-safe algorithms are discussed in [GR93, Lyo88]. A traditional 1-safe algorithm, called *dependency reconstruction algorithm* (DR), is described in [PGM92, KHGMP91]. It can be used for bundle and stream model architectures. In the bundle model, all log records belonging to a transaction are bundled together and sent to the hot standby site. The stream model is similar to the one described in Section 3.1, where, each primary node sends the log records to its hot standby. This is opposed to the single log stream for the whole site, adopted by Tandem [Tan87]. A single log stream is not scalable to the number of nodes at a site. To handle the bundle model, and log records arriving out of order at the hot standby in both models, a ticket system is introduced. This is used to execute the transactions at the hot standby in an order equivalent to the execution order at the primary.

A group commit approach at the hot standby, named the Epoch algorithm (E), is described in [GMP90b]. This algorithm allows for trading efficiency for *minimum divergence.* Minimum divergence means that no transactions other than those missing or depending on missing transactions should be aborted. 2-Safe and 1-safe hybrid schemes are discussed in [GMP90a, MTO93]. The advantages and disadvantages of the 1-safe and 2-safe algorithms are discussed in [PGM92, KHGMP91] in detail.

1-Safe products are provided by: Tandem [Tan87], IBM [ibm95, MTO93], Oracle [ora95], Sybase [syb95], Informix [inf94], and other vendors.

3 Context

3.1 System Architecture

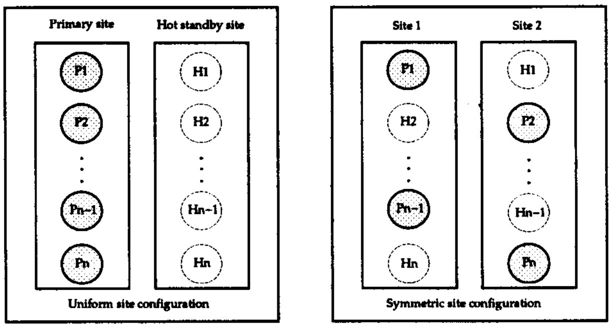

Figure 1: System architecture of uniform and symmetric configurations.

The shared-nothing system architecture used in the following sections is described here. A site contains one or more nodes. Nodes at a site are situated in the same vicinity, and are interconnected by a replicated LAN to mask single communication failures. Sites are connected by high bandwidth, replicated communication lines. An *I'm alive protocol* combined with a *virtual partition management protocol* is used to maintain a consistent set of available nodes [ASC85]. Network partitioning is not considered in this paper. Sites are placed far apart, and this inflicts considerable communication cost.

The system uses an asymmetric replication scheme where one set of replicas is designated as the primary and the other as the hot standby. Each of the primary nodes contain a disjoint partition of a single database and its hot standby node contains a mirrored copy. The union of all nodes at a site gives the complete database.

A two phase commit (2PC) protocol is employed for each transaction execution. Primary and hot standby participants (PP and HP) are coordinated by a primary and a hot standby coordinator (PC and HC). The transactions are run at the primaries with strict concurrency control, and log records are propagated to the hot standbys via multiple log streams (one stream for each primary-hot standby pair). The communication protocol is reliable, delivering log records to the hot standby in the same order as sent, without loosing any, and logs are redone sequentially in the same order at the hot standby. The log records contain enough information to recreate the dependencies between the transactions at the hot standby. The failures occurring in the system are of node or site granularity.

For simplicity, this paper considers a two-site configuration, see Figure 1. The first configuration is uniform. The primary site contains nodes which accommodate only the primary replica, and the hot standby site contains the hot standby replica. This configuration is relevant for the traditional 1-safe algorithms. The second one is a symmetric site configuration, where each site contains a mix of primary and hot standby nodes. This configuration is relevant for the 2-safe algorithm and the algorithms of Section 4.

3.2 Inadequacy of Traditional 1-Safe Algorithms

2-Safe transactions are atomic with respect to both primary and hot standby. A transaction does not commit at the primary without consulting the hot standby. This is referred to as the **cross atomicity** property. 1-Safe transactions lack this property. They commit at the primaries without consulting the hot standbys. Thus, if the primaries fail before the updates are propagated to the hot standbys, then the transaction will be lost.

Lacking the cross atomicity causes the traditional 1-safe algorithms to use an expensive global take-over to mask a single primary node failure. This applies to both uniform and symmetric configurations. This is explained by the following example. Assume transaction T_1 involves primary nodes P_1 and P_2, and their hot standbys H_1 and H_2. The transaction commits at the primary nodes, but P_2 crashes before the update has been propagated to its hot standby H_2. Now, the transaction is partially reflected and thus violates the atomicity property. Take-over by P_2's counterpart H_2 will not help in this situation because it is ignorant of the transaction. There are two ways to restore the database consistency at this point: 1) **Global take-over:** All hot standby nodes takes over from the database state prior to T_1's commitment; 2) **Compensation at the primary:** Effects of partially reflected T_1 and all the transactions that depend on it are compensated at the primaries. The traditional 1-safe algorithms cannot adopt the latter approach, since the primaries have no knowledge of the transactions missed at the hot standbys.

The traditional 1-safe algorithms have an inherent problem when they are executed on symmetric configurations, namely, a site failure before the log propagation will lead to an inconsistent database. The following example illustrates this. Assume the symmetric configuration of Figure 1. Transaction T_1 involves primary nodes P_1 and P_2, and their hot standbys H_1 and H_2. The transaction commits at the primaries, but $site_2$ failes before P_2's update has been propagated to H_2. Now the transaction is partially reflected and thus violates the atomicity property. Global take-over is impossible, since some hot standbys, like H_1, are unreachable due to the failure of $site_2$. To avoid this inconsistency, the traditional 1-safe algorithms adopt the uniform site configuration, such that a primary site failure can be masked by a global take-over, as mentioned in the previous example.

If the traditional 1-safe algorithms can be adapted to incorporate the cross atomicity property, i.e., primary waits to release the resources until the transaction's fate at the hot standby is known, without compromising the response time requirement, then we can solve both problems of the traditional 1-safe algorithms. One way to achieve this is to imitate the 2-safe algorithm, by keeping the resources at the primary until the hot standby's decision is known. The XL algorithm (Section 4.1) adopts this approach. Keeping resources longer reduces concurrency, and hence, throughput. To remedy this, we recommend the DT algorithm (Section 4.2), which uses a less conflicting lock mode to track missing transactions and the ones depending on them at the primary. If a disaster hits, these transactions are aborted at the primary. Thus, the expensive global take-over is avoided. Since these algorithms incorporate the cross atomicity, they can be run on symmetric configurations. They satisfy the minimum divergence criteria, as 2-safe and 1-safe dependency reconstruction algorithms do.

We introduce the EE algorithm (Section 4.3), which incorporates cross atomicity property into the epoch algorithm. It keeps track of the epochs that can not commit at the hot standby. These are compensated at the primary nodes. This algorithm violates the minimum divergence, as much as the epoch does, by abandoning all transactions belonging to the unsuccessful epoch, even though they do not miss or depend on a missing transaction. Our novel algorithms adopt equally well to finer replication granularities than nodes like, e.g., fragments. This aspect is not further investigated in this paper.

4 1-Safe Cross Atomicity Algorithms

4.1 The Extended Locking Algorithm

The XL algorithm can be viewed as a slightly modified variant of the 2-safe algorithm. The main modification is as follows: In the absence of failures, in the 2-safe algorithm, the votes from the HPs and HC will not differ from the corresponding votes of the PPs and PC. In the XL algorithm we exploit this observation to reduce the response time of transactions. The steps involved in normal transaction execution, are illustrated in Figure 2. Once the PC has received ready messages from all PPs, and itself agrees to commit the transaction, the transaction has reached its **1-safe commit point**. At this point the client is informed about the outcome of the transaction. When the PC, in addition, has received ready messages from the HPs, it logs the commit decision at the HC. When this is completed, the transaction has reached the **2-safe commit point**. Like in the 2-safe algorithm, all exclusive primary locks are kept past the 2-safe commit point. In the event of failure, the algorithm will not differentiate between 1-safe committed and active transactions. Thus, the system only guarantees durability of 1-safe committed transactions in the absence of failure. Furthermore, the correctness of the XL algorithm, rests on the correctness of the 2-safe algorithm, and requires no additional proof.

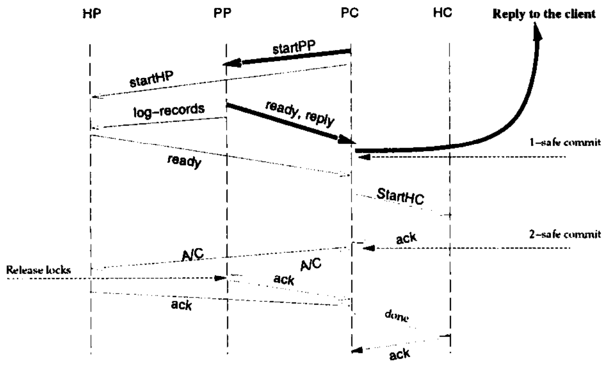

Figure 2: Transaction execution of the extended locking algorithm.

The XL algorithm is simple, compared to the other 1-safe algorithms, and reduces the transaction response time significantly, compared to the 2-safe algorithm. However, locks are kept for the same duration as in the 2-safe algorithm. Compared to other 1-safe algorithms, this tends to increase lock contention and reduce throughput for transactions accessing hot-spot data.

4.2 The Dependency Tracking Algorithm

To remove the disadvantages of the XL algorithm, primary locks must be downgraded prior to the 2-safe commit point to track transactions past 1-safe commit and to reduce lock contention. To support fine granularity take-overs, and symmetric configurations, such an algorithm must: 1) After a primary node failure, remove the effects of 1-safe committed transactions that cannot be installed at the hot standbys, from the remaining primaries; 2) After a site failure, recover a transaction consistent state from a mix of primary and hot standby nodes. In addition, the hot standby commit protocol must be able to provide sufficient throughput to keep up with the 1-safe commit rate of the primaries.

The DT algorithm is designed to meet these requirements, while still providing minimum divergence in the event of a failure. The DT algorithm is based on the stream model dependency reconstruction algorithm (DR) of [KHGMP91, PGM92].

Transactions are first serialized, executed, and atomically 1-safe committed at the primaries. Log records are propagated to the hot standbys outside the time critical path. Based on the log records received from the primary nodes, the hot standbys reconstruct the dependencies between transactions. The transactions are then atomically 2-safe committed and redone at the hot standbys. To be able to reconstruct all dependencies between transactions from the log sequence, all operations performed by write transactions including reads and coarse granularity locks must be reflected in the log stream. Strict two-phase locking is employed by all nodes.

In order to meet the extended recovery requirements, the DT algorithm maintains dependency lock information at the primaries, and increases the synchronization between the primary and hot standby commit protocols, as compared to DR. Furthermore, the recovery subsystems at all primaries, have been extended to handle abortion of 1-safe committed transactions.

Normal execution: The main steps performed by the DT algorithm during normal transaction execution are illustrated in Figure 3. Note that the 1-safe commit decision is taken by the PC, while the 2-safe commit decision is taken by the HC. After the execution phase of the transaction is completed, the PC makes a 1-safe commit decision based on its own vote and the votes received from the PPs and informs the client of the transaction's outcome. It then sends 1-safe commit messages to all involved PPs and the HC. In case of failure, it coordinates the abortion of the transaction.

The lock managers at the PPs are extended to track the dependencies between 1-safe committed transactions. For each normal lock mode, a corresponding 1-safe lock mode is provided. 1-Safe locks are compatible with all normal locks. The compatibilities between different 1-safe locks correspond to those of normal lock modes. When a PP receives a 1-safe commit message, it requests a set of 1-safe locks corresponding to its current locks, and then releases all the normal locks. This allows other transactions to obtain normal locks on the same data items.

When the HC receives a 1-safe commit message, it starts executing the 2-safe commit protocol. After the HPs have received all the transaction's log records, and obtained all locks, they inform the HC of their willingness to commit the transaction. When HC has

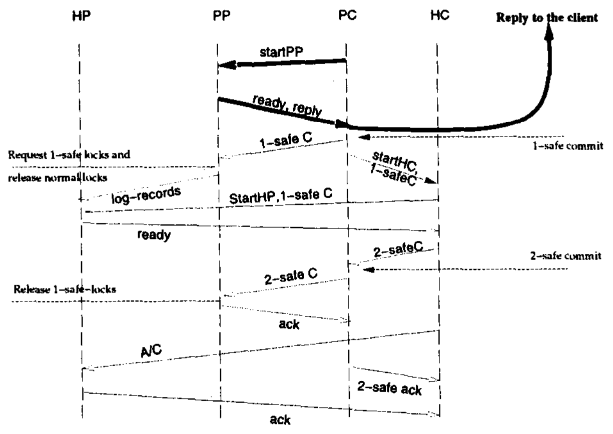

Figure 3: Transaction execution of the dependency tracking algorithm.

received ready messages from all HPs, it makes the 2-safe commit decision, and informs the PC and the HPs of this. The PC forwards the 2-safe commit message to the involved PPs. When a PP receives a 2-safe commit message, it releases all 1-safe locks, and acknowledges the completion of the transaction execution.

Compared to the DR algorithm, the DT algorithm requires some additional message transfers during normal operation: A 2-safe commit message to each PP involved in the transaction, and one 2-safe acknowledgment from the PC to the HC. However, these message transfers are outside the critical path of the transactions. Delaying these messages will only affect the duration of 1-safe locks. They can therefore be piggy-backed on other messages.

Recovering from failures: The dependency lock information, maintained by the PPs are utilized for three purposes: 1) Enable efficient identification of a transaction consistent state after a site failure in the symmetric configuration; 2) Provide immediate partial database availability after a node or site failure; and 3) Simplify identification of 1-safe committed transactions to be rolled back after a failure.[1] To minimize the number of 1-safe committed transactions rolled back due to failures, they are rolled forward whenever possible. We assume that local commit status information for all transactions that may be involved in the recovery after a failure is available at all involved nodes.

Local recovery: When a PP learns about the failure of another PP, it immediately blocks new and active transactions from becoming dependent on 1-safe committed transactions that may have to be aborted, by acquiring normal exclusive locks on all data items the transactions 1-safe lock. Unaffected parts of the database remain available to new transactions. During recovery processing, a PP might have to respond to two types of messages in addition to those received during normal operation: 1) **1-safe abort** from the PC after a PP failure indicating that a transaction in the 1-safe commit state, must be rolled back. The PP will inspect the 1-safe and normal lock queues, and abort all transactions both active and 1-safe committed, that depend on the transaction. It will also send abort messages to the PCs for these transactions. 2) **2-safe ready requests** from HC after an HP failure. The HC might have to determine whether a transaction no longer depends on other 1-safe committed transactions, at the PP corresponding to the failed HP. In these situations the HC sends a 2-safe ready request to the PP. After receiving such a request, the PP waits until the transaction has obtained all requested 1-safe locks, before sending a **2-safe ready** message to the HC. If the transaction has to be aborted, due to being dependent on another aborting 1-safe committed transaction, the PP sends an abort message to the HC. When the recovery processing at the PP is completed, i.e., all 1-safe locks requested prior to the detection of the failure have been released, the PP logs the completion of recovery, and releases the exclusive locks acquired at the beginning of the recovery processing.

When an HP learns about the failure of its counterpart, it acquires shield locks for all data items affected by operations for which it has received log records prior to the failure, which are unprocessed. These shield locks are incompatible with all locks at primaries, but compatible with all locks at hot standbys. When this is completed the HP assumes the role of PP for new transactions, making the remaining parts of the database available to these. It then continues its HP processing of log records received from the failed node with some modifications to the processing rules. If it receives an abort message for a transaction from the HC, it converts the transaction's locks to dirty locks, and aborts the transaction. Subsequent transactions that attempt to acquire locks for dirty-locked data items, will also be aborted. If an HP under recovery, receives a 1-safe commit message for a transaction for which it has received none or only some of the log records prior to the failure, it responds by sending an abort message to the HC. When the recovery processing is completed, i.e., all the log received prior to the failure has been processed, the HP releases the shield locks and all dirty locks, making the entire replica available to new transactions.

No take-over is required for HPs whose counterparts are not affected by the failure. However, data items ac-

[1]An alternative would be to reconstruct the dependency information at the primary nodes at recovery time. However, this would require extensive analysis of potentially large log volumes, which would increase the recovery time substantially. This option is therefore excluded from further discussion due to the consequences for system availability.

cessed by 1-safe committed transactions that must be aborted, are still dirty-locked, leading to the abortion of subsequent transactions that tries to acquire locks on them. When the HP receives an end of recovery log record from the PP, all dirty-locks are released.

Global recovery processing: The global recovery processing after a failure is coordinated by the active PCs and HCs. The PCs and HCs act as process pairs. If one of them is affected by a node failure, the other takes over its responsibilities. In these situations, participants will send messages to the acting coordinator instead of to the failed. All transactions in the active state, whose PPs or PC are affected by the failure, are aborted. Transactions in the 1-safe committed state are 2-safe committed whenever possible. The HC, or acting HC if the original HC has failed, will base its decision on the votes received from the HPs, whenever these are available. If one or more of the HPs are unavailable, the HC will send 2-safe ready requests to the corresponding PPs, and use the responses from these instead of the missing HP votes. The 2-safe commit state of the primary nodes lags behind that of the hot standbys. Thus, the dependencies represented by the 1-safe lock queue, constitute a superset of the dependencies currently available at the HPs. When a PP sends a 2-safe ready message to the HC, i.e., when the transaction has obtained all requested 1-safe locks, it is guaranteed that it is no longer dependent on any 1-safe committed transaction at this PP/HP pair of nodes. In situations where either the PC or HC of a transaction have failed, the remaining coordinator might have to poll one or more of the participants, to obtain information lost due to the failure. One particular failure scenario requires special attention: If the PC fails while all involved PPs are in the ready or 1-safe commit state, but before the HC has been started, i.e., the node where the HC is to be executing does not receive the 1-safe commit message from the PC, neither the PC nor the HC will be available. In this case, the involved PPs or HPs, will start a dedicated coordinator at one of the remaining nodes, which will identify the set of participants, and abort the transaction.

Correctness: The correctness of the DT algorithm rests on the correctness of the dependency reconstruction algorithm [GMP90b, KHGMP91]. The recovered state after a failure, will be based on the dependencies reconstructed by the HPs whenever these are available. Dependencies between transactions lost at failed HPs can be recovered from the dependency information kept at the PPs. In the event of a site failure in the symmetric configuration, there will always be sufficient information available at the remaining site to recover a transaction consistent database state. 1-Safe committed transactions are only aborted if affected by PP or PC failures, where log records or messages are lost, or if they are dependent on other aborted 1-safe committed transactions. Thus, the minimum divergence criteria is fulfilled.

4.3 The Extended Epoch Algorithm

In the DT algorithm, dependencies between transactions are traced at a fine granularity, i.e., at the granularity of locks. This minimizes the loss of 1-safe committed transactions due to failures. However, reconstruction of dependencies and the logging of read operations mean that the hot standby load will only be slightly lower than that of the primaries. In the symmetric configuration, it would be possible to increase system performance by utilizing resources at hot standbys for execution of PCs. Thus, it would be desirable to keep the HP resource consumption as low as possible. In the EE algorithm, we reduce the CPU resource consumption of the HPs, by tracking dependencies between transactions at the granularity of epochs. The EE algorithm is a simple adaption of the single mark epoch algorithm presented in [GMP90b], to the symmetric site configuration. Like in the original algorithm, epoch identifiers (EIDs) are distributed periodically and synchronously by an epoch coordinator. When a node is informed of the change of epoch, it includes an end of epoch record in its log. By utilizing the properties of the 2PC protocol and strict concurrency control mechanisms, each transaction is assigned a single commit EID, i.e., the current EID known by the PC at the commit point. It is ensured that a transaction with a given commit EID is only dependent on transactions that committed in the same or earlier epochs. Epochs are atomically committed at the hot standby nodes. The epoch commit processing is coordinated by an epoch commit coordinator. Compared to the original epoch algorithm, the following extensions are made: 1) The epoch coordinator distributes the new EIDs to all nodes in the system; 2) To allow PCs to run at hot standby nodes, these nodes must also keep a primary transaction log. Log records generated at these nodes, must be propagated to the corresponding nodes at the opposite site; 3) The epoch commit coordinator must include all nodes in the system in the epoch commit protocol; 4) The PPs keep track of all locks acquired by data modifying operations, for all transactions that committed in an epoch, i.e., whose PCs reached the 1-safe commit point in the epoch; 5) Epoch commit decisions are logged at both sites before the epoch is committed. After a primary node or a site failure, the epoch commit coordinator determines the EID of the last epoch that can be committed, and broadcasts it to all remaining nodes. If the epoch commit coordinator has failed, its backup process broadcasts the last committed EID. When a PP learns about the failure, it stops processing new

Table 1: Simulation parameters.

SPECint	269
CPU: send msg: intra-node	2500 instr.
CPU: send msg: intra, inter-site	32500 instr.
Intra-site line delay	0.01 ms
Inter-site line delay	2.5 ms
Log generation	850 instr.
Data access	12550 instr.
Piggyback interval	20ms
Epoch length	30 ms

transactions, and waits for the broadcast of the last committed EID. When this is received, it reacquires all locks held by the transactions of uncommitted epochs, and resumes the execution of new transactions. The rollback of 1-safe committed and active transactions are performed based on the log.

After a failure, the EE algorithm recovers the same state as the original epoch algorithm. Thus, no additional proof of correctness is provided. Due to the use of epochs as the unit for 2-safe commitment of transactions, the minimum divergence criteria is not fulfilled.

5 Performance Evaluation

In order to evaluate the performance and transaction durability properties of the cross atomicity algorithms vs. traditional 1-safe and 2-safe algorithms, we perform a set of simulation studies. We obtain estimates for throughput, mean response time, and CPU usage at primary and hot standby nodes during normal operation, and for the numbers of active and 1-safe committed transactions aborted after a primary node failure. The former metrics are estimated for two different loads: 1) Asymmetric load, where all clients are located at the same site; 2) Symmetric load, where half of the clients are located at each site. The fraction of write transactions ranges from 0.1 to 1.0. For all estimated values, 95%, ±2.5% confidence intervals are established.

The simulation model is based on the ClustRa DBMS [HTBH95, BGH$^+$96], which keeps real-time data in main memory, and performs main memory logging at two sites, to meet sub 15 ms response time requirements. This reflects a typical environment of, e.g., operational telecom applications, where 1-safe solutions are applied. The simulation parameters used throughout the simulations, listed in Table 1, are obtained from measurements on ClustRa. Each of the two sites, consists of eight ultra SPARC nodes, interconnected by ATM networks. Message transfers are heavily CPU-bound operations, and the overhead pr. message is substantial. This corresponds to the current state of the Internet/ATM communication technology. To reduce the number of message transfers, non time critical messages are piggy-backed on other messages. The workload consists of a mix of read-only and write transactions, accessing 4 tuples each. The number of writes performed by write transactions, is binomially distributed, and ranges from 1 to 4 with mean 2.5. In the performance experiments, transactions are generated by eight clients in a back-to-back manner. In the symmetric load case, 80% of the transactions for cross atomicity algorithms only access data items whose primary replicas are located at the originating site. In the simulations estimating the consequences of failures, transactions are generated at a rate of 400 tps with exponentially distributed inter arrival times.

Performance estimates: The performance of the six algorithms under the asymmetric load, are presented in Figure 4. As would be expected, all 1-safe algorithms show significantly higher performance than the 2-safe algorithm, due to their avoidance of inter-site communication inside time critical paths. However, there is no significant difference in performance between the DT and EE cross atomicity algorithms, and their traditional 1-safe counterparts. From this, we conclude that the cross atomicity property can be provided, without significantly increasing the processing overhead during normal operation. The E and EE algorithms are less sensitive to increasing write-ratios, than the other algorithms. Epoch management overheads at the primary nodes are only dependent on the epoch length. Increasing the epoch length will, thus, reduce the overhead and increase the throughput, but will also increase the consequences of primary node failures. Furthermore, these algorithms avoid the logging of read operations. The throughput for the DR and DT algorithms follow slightly steeper curves. This is due to logging of read operations performed by write transactions and to more complex commit protocols for write transactions. Furthermore, the reconstruction of dependencies increases the CPU usage at the hot standby nodes, as compared to the other algorithms. The poor performance of the XL algorithm is primarily due to high lock contention at the primary nodes. All exclusive locks are kept past the 2-safe commit point. Thus, two round trips of inter-site communication are completed before these locks are released. The performance impact of this depends on the length of the inter-site communication delay, and on the access pattern of the transactions.

The performance results obtained under the symmetric load are presented in Figure 5. Like in the asymmetric load case, the performance of the E and EE algorithms is dependent on the epoch length. Thus, we have excluded the EE algorithm from this experiment. Due to their cross atomicity properties, the

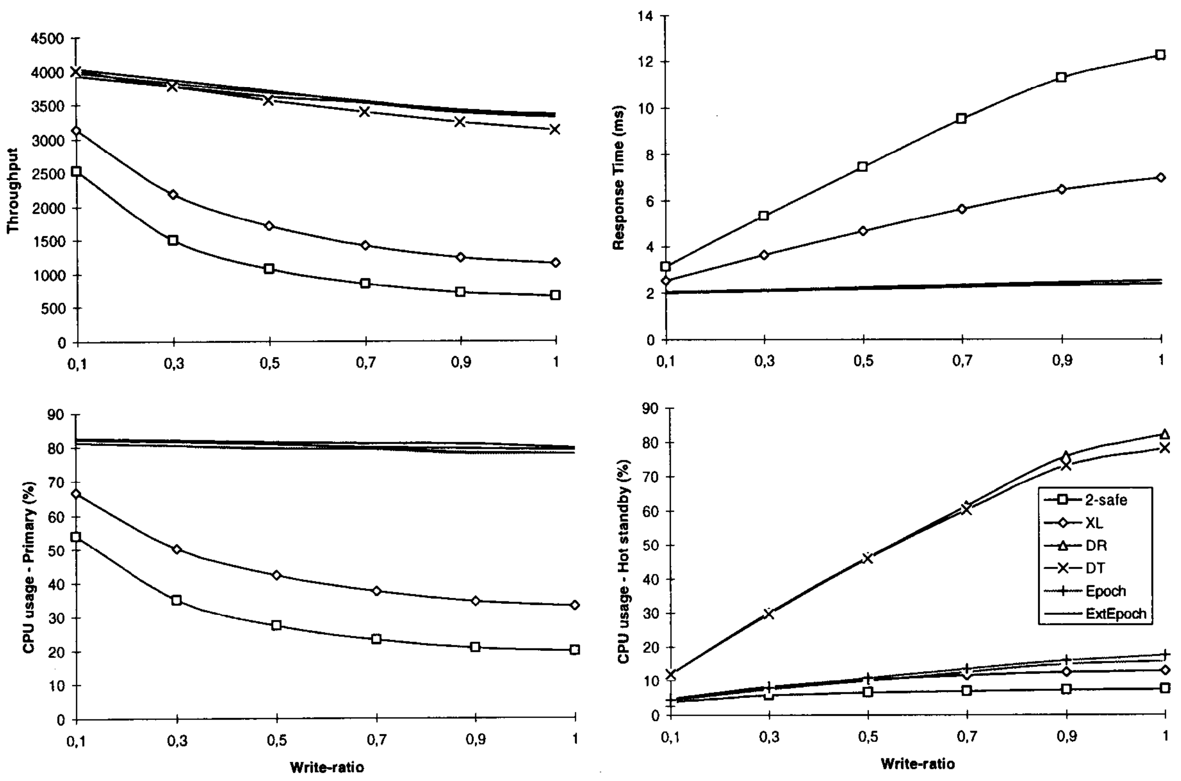

Figure 4: Performance of the uniform configuration.

DT, XL, and 2-safe algorithms are capable of utilizing the access pattern of this workload, by locating primary replicas, at the same site as the clients, by which they will most frequently be accessed. However, for the 20% of transactions that access data items whose primaries are located at the opposite site, the response times will include an inter-site round trip delay. The other algorithms are not able to utilize this information. Thus, an inter-site round trip delay will be added to the response times of all transactions originating at the hot standby site. The DT algorithm provides higher performance than the other 1-safe algorithms, especially for lower values of write-ratio. This is due to its ability to access local primary replicas for a larger fraction of the transactions. Furthermore, symmetric site configuration allows for utilization of unused resources at the hot standby nodes. While for the DR and E algorithms, lock holding times are unaffected by accesses of remote primaries, this is not the case for the DT algorithm. Thus, the performance benefit of the DT algorithm decreases for high write-ratios due to increased lock contention. The performance of the XL and 2-safe algorithms are slightly decreased, as compared to the asymmetric load experiment. This is due to the fact that some of the read-only transactions require access to primaries located at the opposite site.

Estimates of transaction durability properties: Estimates of the number of active and 1-safe committed transactions to be aborted after a primary node failure are presented in Figure 6. The DT algorithm provides the lowest number of active transactions aborted due to primary node failures. It allows all active transactions not directly affected by the failure to be completed. Furthermore, due to the short response time and the transaction generation rate used in this experiment, the number of active transactions at failure times, is low. For the 2-safe and XL algorithms, their longer active periods increase the number of transactions active at the failing node when the failure occurs. In the DR, E, and EE algorithms, all transactions that are active in the system when the failure is detected, must be aborted.

Of the 1-safe algorithms, XL provides the lowest number of 1-safe aborted transactions. This is due to its policy of disallowing other transactions from accessing data modified by 1-safe committed transactions. The number of lost 1-safe committed transactions of the DR and DT algorithms, are slightly higher than for

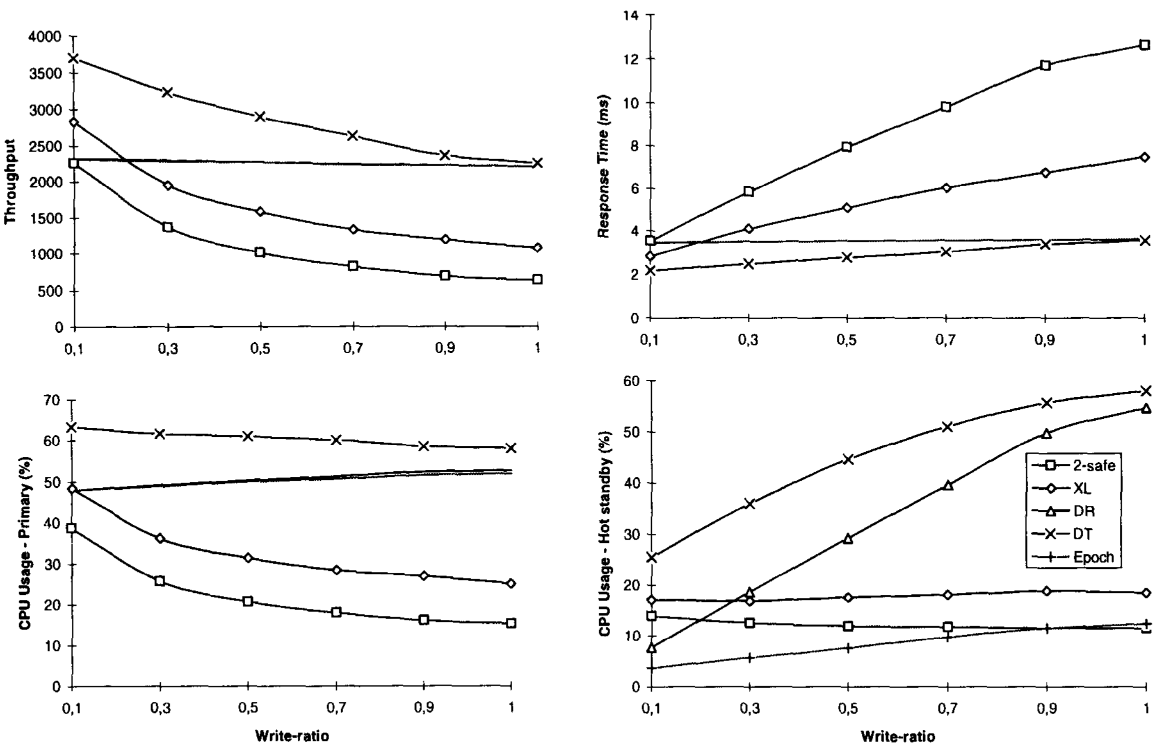

Figure 5: Performance of the symmetric configuration.

XL, but still very low. This is due to their fine granularity reconstruction of dependencies. The number of lost 1-safe committed transactions of the E and EE algorithms exceeds the other algorithms by one order of magnitude due to their coarse granularity tracking of dependencies. The number can be reduced by decreasing the epoch length. It should be noted that the number of 1-safe committed transactions lost due to a failure, in the E and EE algorithms will be independent of the number of actual dependencies between the transactions.

The DT and XL algorithms will be able to restore partial database availability immediately after a primary node failure without any global coordination. The EE algorithm requires identification of the latest epoch that can be committed, before partial database availability can be restored. Both DR and E algorithms require execution of a global take-over which makes the entire database unavailable until the take-over is completed.

6 Summary and Conclusions

In this paper we have focused on the properties of 1-safe transaction execution algorithms. In order to provide database availability in the presence of node and site failures, traditional 1-safe algorithms disallow primary and hot standby replicas of different data items to be stored at the same site. This means that a single primary node failure must be handled like a failure of the entire primary site. Furthermore, this requirement disallows symmetric configurations, where the primary replicas are located at the site closest to the accessing clients. We defined the cross atomicity property of transaction execution algorithms, and showed that the above restrictions can be removed for algorithms that fulfill this property. Furthermore, we presented three novel 1-safe algorithms based on the 2-safe, dependency reconstruction and epoch algorithms respectively, and showed that they provide cross atomicity as well as meeting the traditional 1-safe correctness requirements. The relative performance of these algorithms and the traditional 1-safe and 2-safe algorithms were evaluated by means of a set of simulation studies. Our main conclusion is that the cross atomicity property can be provided without significantly increasing the processing overhead during normal operations.

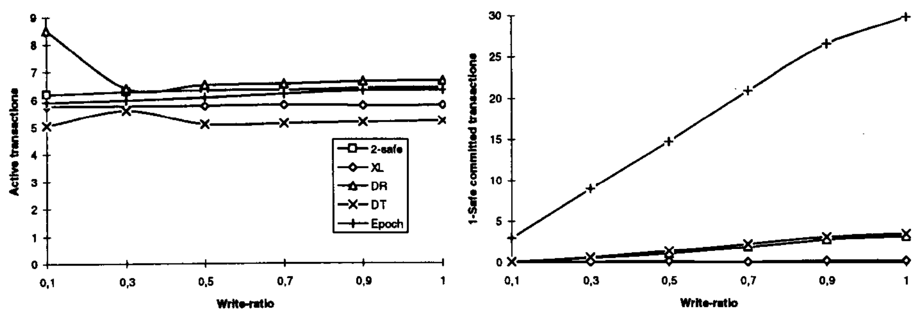

Figure 6: The number of aborted transactions after a primary node failure.

The EE algorithm provides high performance, but less durability for 1-safe committed transactions than the other algorithms. The XL algorithm is very simple, but its performance is very sensitive to long inter-site communication delays in the presence of hot spots. From an evaluation based on performance, transaction durability, and system availability, the DT algorithm provides the best overall solution.

Acknowledgments

The authors would like to thank Svein Erik Bratsberg, Tore Sæter, and Øystein Grøvlen for their valuable comments.

References

[ASC85] A.E. Abbadi, D. Skeen, and F. Christian. An Efficient Fault-Tolerant Protocol for Replicated Data Management. *ACM Distributed Database Systems*, pages 259–73, 1985.

[BGH+96] S.E. Bratsberg, Ø. Grøvlen, S.O. Hvasshovd, B.P. Munch, and Ø. Torbjørnsen. Providing a Highly Available Database by Replication and Online Self-Repair. *Engineering Intelligent Systems*, pages 131–139, 1996.

[GMP90a] H. Garcia-Molina and C.A. Polyzois. Issues in Disaster Recovery. *IEEE Compcon, San Fransisco, CA, September*, 1990.

[GMP90b] H. Garcia-Molina and C.A. Polyzois. Two Epoch Algorithms for Disaster Recovery. *Proceedings of the 16th VLDB Conference*, 1990.

[GR93] J. Gray and A. Reuter. *"Transaction Processing: Concepts and Techniques"*. Morgan Kaufmann Publishers Inc., 1993.

[HTBH95] S.O. Hvasshovd, Ø. Torbjørnsen, S. E. Bratsberg, and P. Holager. The ClustRa Telecom Database: High Availability, High Throughput and Real Time Response. In *Proceedings of the 21st VLDB Conference, Zürich*, 1995.

[Hva94] S.O. Hvasshovd. Global Location Register - High Level Description Study. INMARSAT study, Report TF R 44/94, Norwegian Telecom Research, 1994.

[ibm95] IMS/ESA Administration Guide: System, Version 5, First Edition. Technical Report SC26-8013-00, IBM, April 1995.

[inf94] INFORMIX-Online Dynamic Server, Database Server, Administrator's Guide, Volume 1, Version 7.1. Product Documentation Part No. 000-7778, INFORMIX Software Inc., December 1994.

[KHGMP91] R.P. King, N. Halim, H. Garcia-Molina, and C.A. Polyzois. Management of Remote Backup Copy for Disaster Recovery. *ACM Transactions on Database Systems*, June 1991.

[Lyo88] J. Lyon. Design Considerations in Replicated Database Systems for Disaster Protection. *IEEE, Compcon*, 1988.

[MTO93] C. Mohan, K. Treiber, and R. Obermark. Algorithms for the Management of Remote Backup Data Bases for Disaster Recovery. *IEEE 9th International Conference on Data Engineering*, pages 511–518, 1993.

[ora95] Oracle7 Server Distributed Systems: Replicated Data, Release 7.1. Product Documentation Part No. A21903-2, ORACLE Corporation, 1995.

[PGM92] C.A. Polyzois and H. Garcia-Molina. Evaluation of Remote Backup Algorithms for Transaction-Processing Systems. 1992.

[syb95] Replication Server Technical Publications. Replication Server Administration Guide. Technical Report 32511-01-0101, Sybase Inc., March 1995.

[Tan87] Tandem Computer Corp. *"Remote Duplicate Database Facility (RDF) System Management Manual"*, 1987.