High Availability is not Enough

Peter Triantafillou
triantaf@cs.sfu.ca
School of Computing Science
Simon Fraser University
Burnaby, B.C. V5A 1S6

1. Introduction

Our work mainly concentrates on transactional distributed systems. Most previous research on replication in such environments has concentrated in employing replication to achieve high availability. Our position is that high availability alone is not enough. First, it is important to consider the cost of providing high availability through replication. Second, we must exploit the potential of replication as a means of improving performance. The performance issues (in addition to availability) in which we are mostly interested are: transaction latency, bottlenecks and throughput, and scalability (in particular as it affects the former issues).

In this position paper we briefly outline our related research efforts which can be classified in the following areas: replication-control protocols, recovery strategies, and studying availability in large-scale distributed systems.

2. Replication-control protocols

Replication-control protocols in the past [6,8] and more recent ones [4, 7, 9], in order to achieve one-copy serializability (1-sr), have required costly synchronization between the nodes that store a replica of an accessed data object, for replicated lock acquisition and/or replicated execution of transaction operations. We argue that such synchronous collaboration between the replicas of an object is unnecessary.

2.1 The Location-based paradigm

The location-based replication paradigm [13] is based on the following motivations and principles. A client of a replicated server/object should only observe the delays incurred in contacting any one replica of the object. If we achieve this goal, we have managed to introduce replication into a distributed system without increasing transaction latency. (Indeed, the views-tamped replication method [11] has managed to achieve this goal by introducing a protocol which is based on a primary-copy technique.) Actually, we should expect that latency will be shorter than one-copy systems since a client may now access a local/nearby replica instead of the single copy (which may be located many hops away from the client). Our concerns, however, about latency, bottlenecks, and scalability suggest that we should avoid the use of the primary-copy replication paradigm. With primary copies the clients cannot access local/nearby replicas (this becomes particularly crucial in large systems spanning large geographical areas). Furthermore, primaries may become severe bottlenecks in the system.

The location-based replication paradigm introduces the concept of a leader. Leaders are replicas which are up-to-date with a very high probability. Each client may choose the closest replica as the leader, and it is only the leaders that perform operations synchronously with the execution of the client—the other replicas participate asynchronously. The mechanism that helps clients choose as leaders up-to-date replicas is the extended location information servers (which are themselves replicated). A detailed description (can be found in [13] and [16]) is beyond the scope of this paper. Here we mention that in distributed systems no operations can be executed unless the named object can be located. Our extended location information service is really the traditional location service (and thus there is no extra overhead in consulting it) but with augmented semantics. It provides clients with a list (actually a hint) of the up-to-date replicas for a named object. Using this list the client can pick an appropriate leader. Assuming that failure events are relatively infrequent, maintaining this extended location service is inexpensive since it is failures that may result in an update of the location information for objects.

2.2 Multiple replica classes

An important difficult problem is how to properly address replication problems in growing systems. Large-scale systems introduce, for example, latency problems (partly because of the distances involved). Should we continuously increase the number of replicas to alleviate these problems? Recently there has been increasing focus on these problems. A noteworthy trend is based on increasing the number of replicas of an object and devise quorum-based protocols that require small quorum sizes. The hierarchical quorum consensus protocol [10], the tree quorum protocol [1], and the multidimensional voting protocol [2] are ingenious approaches to this goal.
Our position is that alternative approaches are desirable for three reasons. First, the storage costs of a large number of replicas would become prohibitive. Second, the gains in latency are questionable since more replicas would have to take part in operation execution and in the two-phase commit protocol. Third, the sites from which access requests originate change with time and one cannot predict where the additional replicas will be needed. In such environments it is important to separately address the two motivations for replication.

We address availability problems with one replica class, which we call the true replicas. (One-copy serializability is ensured between true replicas). We address latency problems using two additional classes. The class of pseudoreplicas represents non-permanent replicas which can be cached at clients nodes and can thus dramatically increase the probability of finding local leaders in environments which exhibit locality properties. The class of metareplicas represents also cached data whose main aim is to provide extended location information of the sort described in the location-based paradigm, except that now it can point to up-to-date pseudoreplicas and true replicas.

In [14] and [12] we have presented the multi-class replication protocol which efficiently manipulates these replica classes in order to further improve latency and provide high availability. At the same time, the multi-class protocol identifies and exploits the properties of metareplicas and pseudoreplicas and thus avoids enforcing one-copy serializability for these classes. Therefore, their manipulation becomes inexpensive.

2.3 Efficiently achieving high availability

Another important issue is how should replication protocols react to failures. By examining many protocols in the literature one can observe that a large portion of replication costs can be attributed to handling failure events. View-based protocols [5,7,11] are a case in point. In these protocols failures (as well as recoveries) require the execution of special system transactions which ensure that replicas have consistent views/states. Considering that a site may contain thousands of objects, a single site failure will result in all of these objects (either individually or as part of a global system transaction) exchanging state messages to establish new views and/or perhaps update newly-recovered replicas. In large-scale systems the above strategy will introduce great waste of network bandwidth and CPU cycles—especially if "perceived" failures occur, say, due to network overload.

Our position is that, although view information can significantly improve the availability of objects, requiring special system transactions to react to failures and maintain it can become extremely costly and it is avoidable. In [15] we proposed a new protocol (called VELOS) that offers a novel view-management technique. In particular, each replica stores as part of its state a few additional integer values which identify the replica’s view. VELOS uses this view information to achieve high availability. The view information of each replica is maintained as (user) transaction operations execute on these replicas, thus no additional messages are needed. Also the view data is made part of the replica’s state and it need be updated only in response to failures. In this way VELOS avoids the view-management cost associated with the aforementioned protocols.

3. Efficient recovery strategies

It is a fact that transaction-based distributed systems need atomic commit protocols (ACPs) to ensure transaction atomicity. A well-known theoretical result states that no ACP can avoid dependent recovery and blocking if communication and site failures are possible [3]. When a replica participating in, say, 2PC has voted to commit a transaction but does not receive the decision (commit/abort) either because its site crashes or because of communication failures, the replica is said to be in its uncertain period. Upon recovery, an uncertain replica must contact either other participants who know the decision or the coordinator of 2PC. This is termed dependent recovery. If failures prevent an uncertain replica from reaching sites that know the decision, the replica is said to be blocked. Dependent recovery and blocking are undesirable since they require additional messages and delay access to resources (such as locks on data). In environments with replicated data, dependent recovery becomes significantly costlier because many replication protocols require the recovering replicas to not only resolve their uncertainty periods but to, in addition, bring their states up-to-date.

Our position is that the need for dependent recovery should be eliminated—especially in large systems where at any point in time there may be one or more sites that are recovering. Furthermore, we have recognized that, in systems with replicated data, both the construction of a non-blocking 2PC protocol and independent recovery are possible. We are currently developing recovery strategies [17] whose central ideas are the following. A recovering replica raises a flag to mark its state as uncertain. It then immediately exits its uncertainty period (if it was uncertain). Subsequently, it can be used by clients as before, except that a quorum of non-flagged replicas is required before committing a transaction that accessed objects with one or more flagged replicas. The simplicity of this approach to recovery make it very attractive and indeed we were able to incorporate it into many
quorum-based and primary-copy replication protocols.

There are, however, two potential pitfalls associated with our independent recovery protocol. First, recall that our goal is to allow a client to synchronously access any one replica it chooses. Since recovering replicas do not update their state, clients may access stale replicas and this leads to the abortion and restart of the clients' transactions. Second, since a quorum of non-flagged replicas is required for commitment, availability is worse than requiring any quorum of replicas.

4. High availability in large systems

It is likely that larger systems of the future will consist of previously isolated systems which have been combined in order to allow users access to larger data sets. Such large systems are likely to be heterogeneous with each system wishing to retain its local autonomy. A particular application of such systems is multidatabase systems (MDBS). Our efforts in this area focus on investigating availability issues in this environment.

We are currently studying the implications of selecting a consistency criterion for transaction processing on the availability that can be provided. In particular, in [18] we are studying upper bounds on the allowed availability if multidatabases are to enforce the (traditional) one-copy serializability correctness criterion. We have shown a model for replicating data in such settings by designating the original copy of each MDBS object as a primary and introducing additional secondary replicas of data objects at other member DBSs. In our model, we categorize the execution of transaction operations as follows: We say that an operation executes in primary (secondary, hybrid) mode if it accesses only the primary (secondary, both primary and secondary) replica(s). We categorize transactions into primary, secondary, hybrid, and multimode transactions if their operations execute in primary, secondary, hybrid, or in different modes, respectively. Given this model, we characterize the sets of transactions that can be allowed to commit. We have shown that only (local and global) primary transactions and read-only secondary transactions can be accepted by a replication-control protocol in a replicated MDBS. Update secondary transactions, hybrid and multi-mode transactions cannot be accepted without violating 1-sr or local autonomy.

Finally, we are developing replication protocols that can achieve this upper bound. An essential component of such protocols must deal with the data-recency problem, which exists since update transactions can only write to primary copies, leaving the secondaries stale. Thus, although secondary transactions can access secondary replicas, these will be stale unless the replicated MDBS system addresses this problem.

5. Lessons learned

Detailed simulation studies [16] have shown that indeed the location-based replication paradigm achieves smaller transaction latency than one-copy systems, the primary-copy replication paradigm, and the traditional quorum-based protocols. Furthermore, the multi-class protocol can further improve transaction latency in environments which exhibit locality properties [12]. Also, performing view-management on-line, as transaction operations execute, not only reduces the costs of replication, but does not adversely affect the availability provided by the VELOS protocol. VELOS can provide as high (or better) availability as the other view-based protocols. In particular, it allows the concurrent writing of replicas by clients in different partitions and the "running in the past" of read-only transactions, while ensuring one-copy serializability [15].

Our independent recovery protocol can provide all the benefits of independent recovery and non-blocking atomic commitment at negligible costs (which are introduced by the pitfalls mentioned earlier). In [17] we conducted availability analyses which show that the adverse effects of independent recovery on availability are negligible in typical operational environments. For instance, assuming site reliability of 90%, the sacrifice in availability is 0.00237% and 0.00079% with five and ten replicas respectively. More importantly, for large-scale systems with larger number of replicas for each object, the availability losses caused by the independent recovery algorithm are even smaller. We also combined the independent recovery algorithm with the location-based paradigm for replication in an effort to reduce the number of restarts caused by clients contacting stale (newly-recovered) replicas. Simulation studies [17] have shown that such restarts are very rare. For example, in a system of 15 sites with a total of 300,000 data items, with each site contributing 100 transactions per second, with transactions consisting of 10 operations (80% of which were reads), with objects having 5 replicas, and with 80% of the total accesses directed to a 1% of the database, even for largely-exaggerated failure frequencies (the mean of failure-arrival events was set to 1 minute) only about 0.2% - 1.6% of the total transactions were restarted (for site availability varying from 94% to 73%). Thus, our experience shows that we can avoid the pitfalls of independent recovery and at the same time enjoy the associated benefits at very small costs.

Finally, our results concerning availability upper bounds in multidatabases [18] can be briefly summarized as follows: First, the availability limitations inherent in replicated MDBSs enforcing 1-sr are significant. Second, protocols that achieve optimal availability in this environment appear to be costly, since they also have to solve the data-recency problem...
which can only be solved through the employment of global system transactions.

To recapitulate, our position is that simply employing replication to provide high availability is not enough. We have to, in addition, achieve the following two goals. First, we have to eliminate the costs of replication. To this end we have developed protocols that avoid the costly synchronous collaboration between the replicas during operation execution and when handling failures and recoveries. At the same time our protocols do not introduce bottlenecks. Second, we have to exploit replication to improve performance. We have achieved this goal by allowing local/nearby replicas to be accessed by clients, reducingthus execution delays. We have also exploited replication to achieve the independent recovery and non-blocking atomic commitment of replicas—goals which have been long recognized as important, and have been proven to be unattainable in one-copy systems. Finally, we have to investigate availability issues in large-scale systems which integrate previously-isolated, heterogeneous, and autonomous systems. Such systems may become very common in the future.

References
18. P. Triantafillou, One-Copy Serializable Replicated Multidatabases Respecting Local Autonomy, 1992. submitted for publication