From Fast to Lightweight Atomic Memory in Large-Scale Dynamic Distributed Systems

Vincent Gramoli

IRISA - INRIA Rennes, CNRS
Campus de Beaulieu 35042 Rennes, France
vgramoli@irisa.fr

Abstract

Failure resilience is a fundamental problem. If failures are frequent, it is reasonable to envisage the replication on multiple servers. Although replicating an object (i.e., data) over multiple network locations copes with failures, this makes object modification costly in term of message exchanges to guarantee consistency. In large-scale dynamic systems, nodes join or leave the system arbitrarily often, while the tremendous number of participant prevents nodes from gathering global information. Therefore there is an inherent tradeoff in dynamic distributed systems between the operation time-complexity and message complexity required for consistency maintenance in atomic memory. This paper addresses this trade-off by presenting and classifying existing solutions.

1. Introduction

Replicating a read/write object over a set of distant network locations leads to consistency problem. Assume that an object is modified at some place. From this point on, changes must appear from any location: If a particular client reads the value, then it must return the lastly written value of the object. This simple expression describes the idea at the core of atomic consistency also known as linearizability [12, 11, 14].

Since late seventies [8, 19], *quorums* (mutually intersecting sets) have been proposed as a building block to provide availability in distributed systems. The simplest representation of quorums among a set of n elements is the majority sets. Quorums systems are fundamental in distributed shared memory (DSM), because

they represent the minimal set of locations where the object must be replicated. That is, reading or writing the object consists simply in accessing a constant number of quorums among the existing ones. The quorum system represents all nodes owning a copy of the object. In the remaining we refer to n as its size. The quorum intersection property guarantees that at least one contacted element can testify of the last value written when contacted through a read operation. Therefore quorums are at the core of atomicity and minimizing quorum access time boils down to speed-up read/write operations.

Even though increasing quorum size can tolerate more failures in static systems, dynamic systems can not afford such a solution. Indeed, in static systems where the amount of failures is upper bounded, the intersection between quorums should contain enough elements to guarantee that at least one of it is correct. In dynamic systems where the amount of failures is unbounded, consistency requires additional mechanisms. Among those mechanisms are *reconfiguration* introducing new active quorums, *adaptive* quorum access, and *probabilistic quorum* probing.

This paper discusses the tradeoff relying between operation time and message complexity induced by consistency requirements.

The paper is divided as follows. Section 2 investigates approaches that cope with infinite amount of failures. Section 3 presents solutions to the problem induced by large-scale systems with limited bandwidth capacity. Finally Section 4 concludes the paper.

2. Facing Dynamism

Several existing papers focus on the emulation of shared memory in message passing systems, also known

as Distributed Shared Memory (DSM). For instance, Attiya et al. [4] propose a robust single-writer multi-reader (SWMR) shared memory algorithm. This algorithm specifies a two-phase read operation and a single phase write operation, each phase consisting in an exchange of messages between the client and a majority of nodes. More recently, single-phase read operation, namely *fast read*, appeared in [7, 20].

2.1. Reconfigurable Atomic Memory

The robustness in dynamic systems is more challenging. Although using majority of nodes is robust in face of a bounded number of failures, in dynamic systems the number of failures might grow infinitely. That is, reconfiguration has been introduced in RAMBO [15] to cope with accumulated failures. The reconfiguration process replaces the sets of contacted nodes by active ones. The quorum replication is sufficient to cope with failures occurring between subsequent reconfigurations.

Fast reconfiguration is at the core of fault tolerance in dynamic systems. More precisely, the faster the system switches to the new active configuration the more fault-tolerant the system is. RDS [6] proposes an accelerated reconfiguration process coupled with fast read operations. The RDS algorithm proposes fast reconfiguration using "Fast Paxos" [13, 5] consensus algorithm as a fully-integrated part of the shared memory. RDS implements a leader-based protocol that makes the reconfiguration three message delays long when the leader stabilizes. To conclude, RDS achieves a fault-tolerant MWMR shared memory especially suited for dynamic networks.

2.2. Restricting the Gossip

Aforementioned algorithms suffer from a potential drawback when applied in large-scale systems where the bandwidth capacity is bounded. Indeed, operation execution at the core of those algorithms relies on the fact that clients know the currently used configuration (i.e., quorum system). Since any node is a potential client, an approach is to maintain global knowledge at each node despite reconfiguration. For this purpose, one should use all-to-all *gossip* (message exchange). Therefore, the number of messages produced after a reconfiguration is n^2.

In [10] the authors propose a substantial improvement in order to reduce the communication complexity from quadratic to linear. The key idea is roughly to differentiate replicas from common nodes. In this paper,
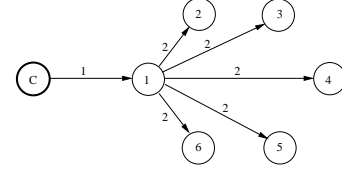

Figure 1. The global knowledge approach

replicas are called *owners* and they are considered locally by other nodes as active configuration members. Restricting gossip among owners makes reconfiguration possible provided that at least one member of any configuration is active. However, maintaining such knowledge at any node remains unachievable when the number of owners is large, the reconfiguration is frequent, and a configuration member must remain active. These reconfigurable approach requires $O(n)$ during a reconfiguration while operations are executed in a constant number of message delays as presented in Figure 1 (the client is represented by a C, and the quorum has size 6).

3. Facing Scalability

Because of the recent growing interest for dynamic large-scale systems, a radically different approach has been adopted. This approach relies strongly on a locality principle: participants maintain only a piece of the system information depending on their location. Therefore reconfiguration is made locally and involves less messages.

3.1. Restricting Knowledge

Assuming that each node maintains the information about a restricted set of nodes, reconfiguration is executed locally, thus, with low communication complexity. The direct drawback of such a solution is the operation latency. Although reconfiguration message complexity is minimized, the time complexity of an operation is increased. Indeed, minimizing knowledge prevents operation from completing after a single round-trip. Suppose that any replica maintains knowledge of only one other replica in a logical overlay. Then, in order to contact an entire quorum a phase will take as many message delays as the number of the quorum elements.

3.2. Adaptive Atomic Memory

Global/local reconfiguration paradigm is interestingly related to the *proactive/reactive routing* paradigm.

The local reconfiguration process is such that only re-active routing can be adopted by phases: a node (not the first contacted one) belonging to the quorum is not identified before the phase reaches its direct neighbor. Unlike local reconfiguration, global reconfiguration process requires that a phase executes a pro-active routing, where the whole set of nodes is known when the phase starts. Algorithms based on the first technique are called *non-adaptive* while algorithms based on the second one are called *adaptive* according to [18]. This paper extends this notion to atomic memory.

Recently, some authors have focused on dynamic types of quorums [18, 17, 1]. In [1], another node is inserted by adding a vertex in a De Bruijn graph, while vicinity is defined by graph neighboring. For instance, the Dynamic Path [18] defines quorums in a logical overlay using Voronoi cells and the Dynamic And-Or Quorum System [17] defines quorums as a set of leaves.

DSM finds its way into such emergent ideas. SAM [2, 3], is a read/write memory using adpative operations. Quorums are defined as rows and columns of a dynamic torus grid. This solution provides local reconfiguration, that is, the message required to reconfigure after a single departure is constant. In the meanwhile, the adaptive operations require $O(\sqrt{n})$ message delays to complete, where all nodes of a $(\sqrt{n})$-sized quorum are contacted in turn. In the meanwhile reconfiguration requires a constant number of messages, since the number of neighbors is constant. Figure 2 represents the way a quorum is contacted in this example of adaptive approach.
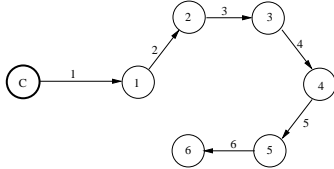

Figure 2. The adaptive approach

3.3. Probabilistic Atomic Memory

In order to guarantee progress and consistency in asynchronous dynamic systems the previous solutions make some assumptions. For instance, in [6] reconfiguration is guaranteed provided that the system stabilizes. Likewise, failure detection is required in [3] to guarantee consistency.

Recently, probabilistic approaches have been investigated to implement quorum systems. Malkhi et al. defined in [16] a probabilistic quorum system (i.e. ϵ-intersecting quorum system) as a quorum system whose quorum intersection is guaranteed with probability $(1 - \epsilon)$. This approach brings a solution for asynchronous dynamic systems where ensuring intersection deterministically remains unachievable without constraining assumptions.
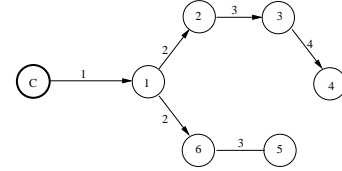

Figure 3. The random walks approach

In [1] the authors propose a logical overlay structure using a De Bruijn graph in order to maintain k-connectivity. They make this structure dynamic by defining join and leave primitives, each requiring local reconfiguration with $O(\log n)$ messages before the structure stabilizes. They define quorums that intersect with high probability. To achieve this result, their approach consists in executing $O(\sqrt{n})$ random walks (cf. Figure 3), each of length $O(\log n)$. That is, a quorum access lasts $O(\log n)$ message delays.

3.4. Structureless Atomic Memory

Differently, it is noteworthy that the structure imposes several constraints due to the maintenance of the overlay. When dynamic events—such as join or leave events—occur, the overlay changes. Therefore, the overlay must be readapted to reflect changes involving message exchanges.

Consequently, providing building blocks that are independent from the underlying overlay minimizes communication complexity. In [9] the authors present a way to preserve object persistence through the use of a set of nodes called a *core*. This core can be used to preserve information persistence, such as quorums, and thus, can serves as DSM building block.

The key idea is temporal in the sense that an object persists if operations are sufficiently frequent regarding to the system dynamism, or *churn*—the rate at which nodes enter and leave the system. Roughly speaking, if the object is written at enough locations with a reasonable frequency then the object persists. This build-
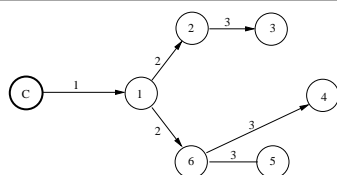
3

Figure 4. The disseminating approach

ing block can be used to provide efficient quorum access with weak maintenance constraint. Figure 4 presents an access through dissemination that is $O(\log n)$ message delays long.

4. Conclusion

This paper classifies a panel of achievements in the context of quorum-based solution for large-scale dynamic DSM implementation. Two major problems arise from large-scale and dynamic DSM: the time complexity required by operation and the communication complexity required to guarantee consistency while participants join or leave and while object state must reflect write operations. This paper enlightens the inherent tradeoff between these two fundamental issues.

We are currently specifying a probabilistic protocol for structureless quorum system with $O(\log n)$ operation time complexity and very low reconfiguration cost.

References

[1] I. Abraham and D. Malkhi. Probabilistic quorum systems for dynamic systems. *Distributed Systems*, 18(2):113–124, 2005.

[2] E. Anceaume, M. Gradinariu, V. Gramoli, and A. Virgillito. P2P architecture for self* atomic memory. In *Proc. of 8th IEEE International Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN'05)*, pages 214–219, Dec. 2005.

[3] E. Anceaume, M. Gradinariu, V. Gramoli, and A. Virgillito. Self-adjusting atomic memory for dynamic systems based on quorums on-the-fly. correctness study. Technical Report 1795, IRISA/INRIA Universite de Rennes 1 - Campus de Beaulieu, Rennes, France, April 2006.

[4] H. Attiya, A. Bar-Noy, and D. Dolev. Sharing memory robustly in message-passing systems. *J. ACM*, 42(1):124–142, 1995.

[5] R. Boichat, P. Dutta, S. Frolund, and R. Guerraoui. Reconstructing paxos. *SIGACT News*, 34(2):42–57, 2003.

[6] G. Chockler, S. Gilbert, V. Gramoli, P. Musial, and A. Shvartsman. Reconfigurable distributed storage for dynamic networks. In *Proc. of 9th International Conference on Principles of Distributed Systems (OPODIS'05)*, Dec. 2005. to appear.

[7] P. Dutta, R. Guerraoui, R. R. Levy, and A. Chakraborty. How fast can a distributed atomic read be? In *Proc. of the 23th annual symposium on Principles of distributed computing (PODC'04)*, pages 236–245, New York, NY, USA, 2004. ACM Press.

[8] D. K. Gifford. Weighted voting for replicated data. In *SOSP '79: Proceedings of the seventh ACM symposium on Operating systems principles*, pages 150–162. ACM Press, 1979.

[9] V. Gramoli, A.-M. Kermarrec, A. Mostefaoui, and M. Raynal. Core persistence in peer-to-peer systems: Relating size to lifetime. Technical Report 1799, IRISA/INRIA Universite de Rennes 1 - Campus de Beaulieu, Rennes, France, April 2006.

[10] V. Gramoli, P. Musial, and A. Shvartsman. Operation liveness and gossip management in a dynamic distributed atomic data service. In *Proc. of the 18th International Conference on Parallel and Distributed Computing Systems (PDCS'05)*, Sept. 2005.

[11] M. P. Herlihy and J. M. Wing. Linearizability: a correctness condition for concurrent objects. *ACM Trans. on Programming Languages and Systems (TOPLAS)*, 12(3):463–492, 1990.

[12] L. Lamport. On interprocess communication, part II: Algorithms. *Distributed Computing*, 1:86–101, 1986.

[13] L. Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, 1998.

[14] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.

[15] N. Lynch and A. Shvartsman. RAMBO: A reconfigurable atomic memory service for dynamic networks. In *Proc. of 16th International Symposium on Distributed Computing (DISC'02)*, pages 173–190, 2002.

[16] D. Malkhi, M. Reiter, A. Wool, and R. Wright. Probabilistic quorum systems. *Information and Computation*, 170(2):184–206, 2001.

[17] U. Nadav and M. Naor. The dynamic and-or quorum system. In Pierre Fraigniaud, editor, *Distributed algorithms*, volume 3724 of *Lecture Notes In Computer Science*, pages 472–486, September 2005.

[18] M. Naor and U. Wieder. Scalable and dynamic quorum systems. In *Proc. of the 22th annual symposium on Principles of distributed computing (PODC'03)*, pages 114–122. ACM Press, 2003.

[19] R. H. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *ACM Trans. Database Syst.*, 4(2):180–209, 1979.

[20] M. Vukolic and R. Guerraoui. How fast can a very robust read be? In *Proceedings of the 25th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing (PODC'06)*, 2006.