Brief Announcement: On the Robustness of (Semi)Fast Quorum-Based Implementations of Atomic Shared Memory

Chryssis Georgiou
University of Cyprus
chryssis@cs.ucy.ac.cy

Nicolas C. Nicolaou
University of Connecticut
nicolas@engr.uconn.edu

Alexander A. Shvartsman
University of Connecticut
aas@cse.uconn.edu

ABSTRACT

Atomic (linearizable) read/write memory is a fundamental abstractions in distributed computing. Following a seminal implementation of atomic memory of Attiya *et al.* [6], a folklore belief developed that in messaging-passing atomic memory implementations "reads must write." However, work by Dutta *et al.* [4] established that if the number of readers R is constrained with respect to the number of replicas S and the maximum number of crash-failures t so that $R < \frac{S}{t} - 2$, then single communication round-trip reads are possible. Such an implementation given in [4] is called *fast*. Subsequently, Georgiou *et al.* [3] relaxed the constraint in [4], and proposed *semifast* implementations with unbounded number of readers, where under realistic conditions most reads need only a single communication round-trip to complete. Their approach groups collections of readers into *virtual nodes*. Semifast behavior of their algorithm is preserved as long as the number of virtual nodes V is constrained by $V < \frac{S}{t} - 2$.

Quorum systems are well-known mathematical tools that provide means for achieving coordination between processors in distributed systems. Given that the approach of Attiya *et al.* [6] is readily generalized from majorities to quorums (e.g., [5, 2]), and that the algorithms in [4] and [3] rely on intersections in specific sets of responding servers, one may ask: *Can we characterize the conditions enabling fast implementations in a general quorum-based framework?* This is what we establish in this work.

1. COMPUTATIONAL MODEL

An atomic SWMR implementation is *fast* if all read and write operations complete in a single communication round-trip in any execution. A *semifast* implementation [3] allows one complete *slow* read operation for each write, and all the rest read/write operations must be fast. Lastly an implementation is *non-robust* if it has a single point of failure. A quorum system $\mathbb{Q}$ is a collection of sets Q_i such that, $\forall Q_i, Q_j \in \mathbb{Q} : Q_i \cap Q_j \neq \emptyset$. A quorum $Q_i \in \mathbb{Q}$ is faulty if it contains a crashed process. We assume that at least one quorum in $\mathbb{Q}$ is non-faulty in any execution of the algorithm.

2. OUR RESULTS

Fast Implementations: We show that a quorum-based fast implementation is possible *iff* a *common* intersection exist among all quorums. Since a single failure in the common intersection may collapse the quorum system then this leads to the conclusion that *robust* fast quorum-based implementations are *impossible*.

SemiFast Implementations: The third semifast property as defined in [3] states that only a *single complete* read operation is allowed to perform a second communication round-trip for every write operation. We prove that a single complete slow read is not enough, for any quorum-based implementation without common intersection. Thus *robust semifast* quorum-based implementations are also *impossible*.

Quorum Views and a New Algorithm: Consequently we seek implementations that enable fast reads, but permit multiple slow reads per write. We call such implementations *weak-semifast*. We introduce the notion of *Quorum Views* that refer to the distribution of the maximum timestamp that a read operation ρ_i witnesses after its first communication round. Consider that ρ_i contacts quorum Q_i during its first communication round. For each $s \in Q_i$, ρ_i receives a timestamp $s.ts$ and $maxTS = max(s.ts)$. Quorum views for ρ_i are defined as follows:

1. $[qView(1)]$ $\forall s \in Q_i : s.ts = maxTS$,
2. $[qView(2)]$ $\forall Q_j \in \mathbb{Q}, i \neq j, \exists A \subseteq Q_i \cap Q_j$, s.t. $A \neq \emptyset$ and $\forall s \in A : s.ts < maxTS$, and
3. $[qView(3)]$ $\exists Q_j \in \mathbb{Q}, i \neq j$ and $\forall s \in Q_i \cap Q_j$: $s.ts = maxTS$.

A quorum view may provide "sufficient" information on whether or not a write operation is complete. Our new algorithm makes use of this idea. Briefly the write protocol propagates the value-timestamp pair to a full quorum, increments its timestamp and completes. The read protocol propagates read messages to a full quorum and examines the quorum's $maxTS$ distribution. If either $qView(1)$ or $qView(2)$ are satisfied, then the reader terminates in the first communication round and returns $maxTS$ or $maxTS-1$ respectively. If the view satisfies $qView(3)$, then the reader proceeds to a second communication round where it propagates the maximum timestamp to a full quorum and then returns $maxTS$.

Simulations: We simulated our algorithm and observed that under realistic scenarios less than 13% of the reads need to be slow.

For more details we refer the reader to [1].

3. REFERENCES

[1] C. Georgiou, N. Nicolaou, and A. Shvartsman. On the Robustness of (Semi)Fast Quorum-Based Implementations of Atomic Shared Memory. http://www.cse.uconn.edu/~ncn03001/TRs/GNS08TR.pdf

[2] R. Guerraoui and M. Vukolić. Refined quorum systems. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing (PODC)*, pages 119–128, 2007.

[3] C. Georgiou, N. Nicolaou, and A. Shvartsman. Fault-tolerant semifast implementations for atomic read/write registers. In *Proceedings of the 18th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 281–290, 2006.

[4] P. Dutta, R. Guerraoui, R. R. Levy, and A. Chakraborty. How fast can a distributed atomic read be? In *Proceedings of the 23rd ACM symposium on Principles of Distributed Computing (PODC)*, pages 236–245, 2004.

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

[6] H. Attiya, A. Bar-Noy, and D. Dolev. Sharing memory robustly in message passing systems. *Journal of the ACM*, 42(1):124–142, 1996.

PODC'08, August 18–21, 2008, Toronto, Ontario, Canada.
ACM 978-1-59593-989-0/08/08.