# Brief Announcement: Atomic Consistency and Partition Tolerance in Scalable Key-Value Stores

Cosmin Arad, Tallat M. Shafaat, and Seif Haridi

KTH Royal Institute of Technology, Sweden

**Abstract.** We propose *consistent quorums* to achieve linearizability in scalable and self-organizing key-value stores based on consistent hashing.

Key-value stores based on consistent hashing [5] provide scalable and self-organizing storage for modern web applications. Simply applying quorum-based read and write operations [3] within replication groups dictated by consistent hashing, fails to achieve atomic consistency in a partially synchronous system prone to network partitions [2]. Given the advantages of consistent hashing (simplicity, incremental scalability, self-organization) and the realities of data-center environments (partial synchrony, node dynamism, and the possibility of network partitions) we set out to achieve linearizability in a dynamic and scalable key-value storage system governed by consistent hashing. We apply results from reconfigurable atomic storage, within dynamic replication groups where node membership is automatically dictated by the principle of consistent hashing.

A naïve approach to achieving consistency is to use quorum-based *read/write* operations within every replication group. This will not work as false failure suspicions, along with consistent hashing, may lead to non-intersecting quorums [2]. Any quorum-based algorithm, such as ABD [3] and Paxos [6], will suffer from the problem of non-intersecting quorums when used with consistent hashing. We propose *consistent quorums* as a solution. Each node has a *view* $\langle v, i \rangle$, where $v$ is the set of nodes in the replication group and $i$ is the version number of the view. A consistent quorum is a quorum of nodes that are in the same view when the quorum is assembled. When a node replies to a request it stamps its reply with its current view. A quorum-based operation (e.g. ABD, Paxos) will succeed only if it finds a quorum of nodes with the same view, i.e., a consistent quorum.

Changes to replication group $\langle v, i \rangle$ are proposed as a new view $\langle v', i+1 \rangle$ via Paxos augmented with consistent quorums. The decision is *installed* on all nodes in $v \cup v'$. In spite of reconfigurations, it can be shown that any two consistent quorums will always intersect [2]. This implies that ABD writes and reads will satisfy linearizability in the presence of reconfigurations and concurrent reconfiguration operations will be applied in a total order [2].

Paxos and ABD are intrinsically partition-tolerant; since they depend on quorums, operations in any partition that contains a quorum will succeed. To maintain partition tolerance when Paxos and ABD are applied within consistent hashing replication groups, we use consistent quorums. We employ a ring unification algorithm [7] that repairs the ring topology of consistent hashing, hence fixing node responsibilities, after a transient network partition. This enables our overall key-value store to be tolerant to network partitions.

A linearizable *read/write* register in a fully asynchronous system model was implemented by the ABD algorithm [3] which used majority replication within a static set of nodes. Atomic registers were extended to dynamic networks with protocols like RAMBO and RDS [4] which used consensus to coordinate the sequence of system reconfigurations. DynaStore [1] showed that reconfigurable atomic registers can be implemented without consensus in a fully asynchronous system. To enable linearizability at large scale, we turn every consistent hashing replication group into a set of reconfigurable atomic registers, one for each key-value object, maintaining a consistent mapping of objects to replication groups.

Similar to previous work on reconfigurable atomic storage [4,1], our approach decouples reconfiguration from register operations, allowing operations to execute concurrently with reconfigurations. While in RDS and DynaStore all operations are forced to contact quorums in all active configurations, with consistent quorums, operations optimistically contact only a single quorum. Only operations that get different consistent quorums between their read and write phases need to repeat the read phase in the new configuration.

Like in RDS [4], we use consensus only for reconfiguration, but more generally, consistent quorums could be used to transform any static quorum-based protocol to be dynamically reconfigurable, while paying for consensus only on reconfiguration, and otherwise maintaining the original protocol's complexity.

The design, implementation, and evaluation of a scalable key-value store based on consistent hashing and consistent quorums, as well as the algorithms and correctness proofs are available in a separate technical report [2].

## References

1. Aguilera, M.K., Keidar, I., Malkhi, D., Shraer, A.: Dynamic atomic storage without consensus. In: Proceedings of the 28th ACM Symposium on Principles of Distributed Computing, PODC 2009, pp. 17–25. ACM, New York (2009)
2. Arad, C., Shafaat, T., Haridi, S.: CATS: Linearizability and partition tolerance in scalable and self-organizing key-value stores. SICS Technical Report T2012:04
3. Attiya, H., Bar-Noy, A., Dolev, D.: Sharing memory robustly in message-passing systems. J. ACM 42, 124–142 (1995)
4. Chockler, G., Gilbert, S., Gramoli, V., Musial, P.M., Shvartsman, A.A.: Reconfigurable distributed storage for dynamic networks. J. Parallel Distrib. Comput. 69, 100–116 (2009)
5. Karger, D., Lehman, E., Leighton, T., Panigrahy, R., Levine, M., Lewin, D.: Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web. In: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, STOC 1997, pp. 654–663. ACM, New York (1997)
6. Lamport, L.: The part-time parliament. ACM Trans. Comput. Syst. 16, 133–169 (1998)
7. Shafaat, T., Ghodsi, A., Haridi, S.: Dealing with bootstrapping, maintenance, and network partitions and mergers in structured overlay networks. In: Sixth IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2012. IEEE Computer Society, Washington, DC (2012)