Consistent hashing distance metrics for large-scale object storage

Philip Carns, Kevin Harms, John Jenkins,
Misbah Mubarak, Robert B. Ross
Argonne National Laboratory
Email: carns@mcs.anl.gov

Christopher Carothers
Rensselaer Polytechnic Institute

I. Background

Large-scale storage systems often use object-level replication to protect against server failures. This calls for the development of efficient algorithms to map replicas to servers at scale. Consistent hashes are ideal for this purpose: they map each object to the "K closest" servers based on a numeric distance metric and require minimal data movement on membership change. If membership is stable, then consistent hashes can be calculated locally with no communication.

Consistent hashes are often conceptually represented as a ring, as shown in Figure 1, with proximity measured in terms of numerical difference between IDs [1]. Alternative distance metrics, such as the hash function in Ceph's straw bucket CRUSH algorithm [2], can be used in place of numerical distance. Virtual nodes can also be added to improve load balancing. The multiring hashing algorithm [3] further deviates from the basic conceptual model by placing each virtual node for a given server on a separate ring.

We introduce a modular, open-source consistent hashing library called **libch-placement** and use it to investigate distance metrics for consistent hashing. The choice of distance metric for consistent hashing determines:

- Replica declustering → rebuild time and MTTDL
- CPU efficiency → latency and bulk calculation cost
- Object set sharding → aggregate HPC bandwidth

An ideal distance metric would strike a balance between these properties.

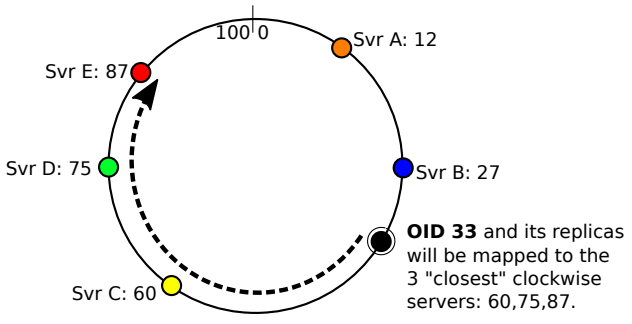
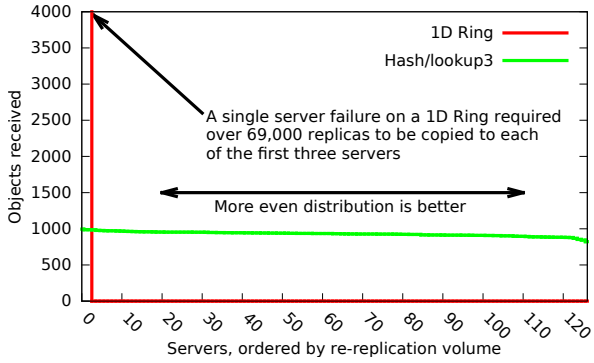
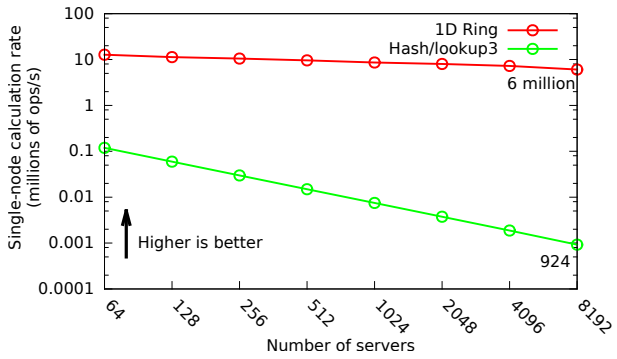


Fig. 1. 1D ring consistent hashing example with five servers randomly dispersed in a numerical range of 0 to 100.



(a) declustering



(b) computational efficiency

Fig. 2. Comparison of simple 1D ring and hash-based distance metrics.

II. Empirical methodology

Figure 2 illustrates how consistent hashing distance metrics can differ radically in key properties. Figure 2(a) shows how many replicas would be rebuilt on each server in a 128-server storage system, assuming 5 million random objects, 3-way replication, and a single server fault. Figure 2(b) shows the placement calculation rate on a 2 GHz AMD Opteron as the system size increases. The 1D ring exhibits poor declustering but excellent CPU efficiency. The hash metric exhibits excellent declustering but poor CPU efficiency, bound by an $O(n)$ per-object calculation.

Consistent hashing properties can be altered not only by changing the distance metric itself, but also by applying a va-

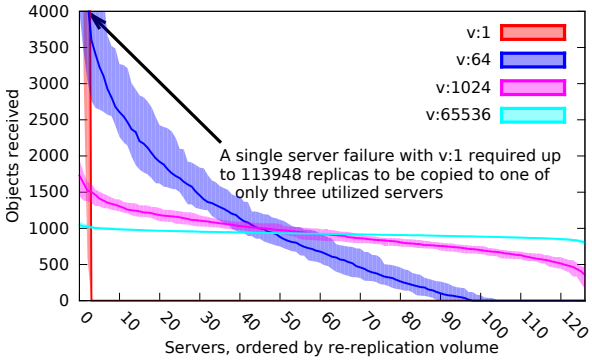


Fig. 3. Declustering in a 1D ring distance metric as the ratio of virtual nodes to physical nodes is increased.

riety of metric-agnostic optimizations. Figure 3, for example, illustrates how virtual nodes can be used to improve declustering. Other optimizations, such as OpenMP directives, can be used to improve CPU efficiency. In this poster we examine the interaction between distance metrics and consistent hashing optimizations to identify the most promising overall algorithms to improve performance, improve reliability, and manage data placement with per-object granularity in large-scale storage systems.

III. Summary of Results

The findings of our study include the following:

- Virtual nodes can dramatically increase declustering, but it may take as many as 65,536 virtual nodes to achieve near-maximal declustering at scale.
- The multiring hashing algorithm balances declustering and CPU efficiency while also supporting optimal object set sharding.
- Consistent hashes are highly amenable to parallelism for bulk calculation.
- Large-scale systems can recalculate placement at an aggregate rate of tens of billions of objects per second to enable placement with per-object granularity.

The libch-placement library, including additional modular distance metric implementations not shown in this work, is freely available at https://xgitlab.cels.anl.gov/codes/ch-placement/.

References

[1] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[2] S. A. Weil, S. A. Brandt, E. L. Miller, and C. Maltzahn, "CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data," in *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*. New York, NY, USA: ACM, 2006.

[3] J. Darcy, "HekaFS: Multi ring hashing." [Online]. Available: http://pl.atyp.us/hekafs.org/index.php/2012/07/multi-ring-hashing/