

Metadata Load Balancing Policies and Key-Value Stores

MICHAEL SEVILLA*

ABSTRACT

Enter the text of your abstract here.

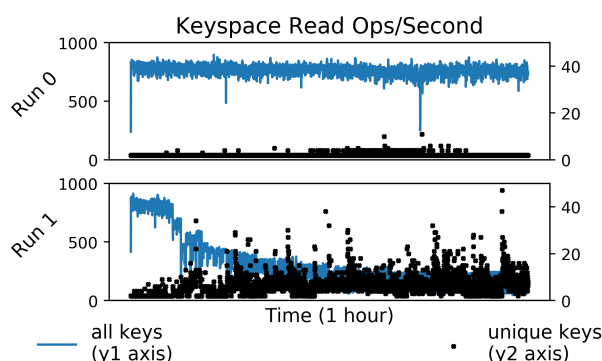


FIG. 1. The keyspace for run 1 has a larger but less active keyspace. Caching the 10 most recently accessed keys is sufficient for run 0 but but a more complicated keyspace like run 1 needs dynamic load balancing policies.

1. Introduction

- key-value stores scale because they support
 1. fine scale annotation
 2. flexible, extensible formats
- science apps are structured, entropy increases over time (e.g., Figure 1 shows key distribution and key popularity changing over time)

Hypothesis: re-distributing keys requires dynamic load balancing policies [1], similar to distributed file systems. A one-size-fits all policy is not sufficient.

* Corresponding author address: Los Alamos National Laboratory
E-mail: msevilla@ucsc.edu

2. Background

3. Methodology

ParsplICE is an HPC application with distinct workload phases and a well-known keyspace (Figure 2).

Part 1: ParsplICE architecture uses a backend KV store

- Single Node DB (LevelDB, BerkeleyDB) is insufficient
- Distributed KV store solves sync problem and enables load balancing
- HXHIM

Part 2: As ParsplICE simulates, entropy increases resulting in keyspace imbalance (Figure 1 shows how imbalance controlled by “growth rate”)

- Mantle (approach/API) to explore dynamic load balancing policies
 - quantifies effect of load balancing
 - formalized effective FS balancers
 - debugging tool
- HXHIM is a good fit because it has migration mechanisms for load balancing
 - bulk operations (put/get())
 - key partitioners
 - secondary indices
- we need a way to learn these regimes
 - Figure ?? shows the same phases as 100K but that the timestamps affected by delay

Results should show, In order from most likely to least likely:

1. HPC key-value store workloads are structured (because they are mostly workflows and simulations)

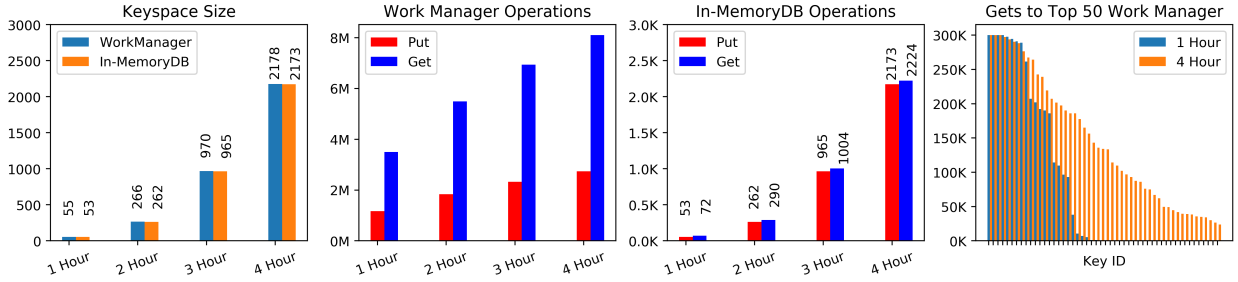


FIG. 2. The keyspace grows in a structured way, so we can predict the number and popularity of hot keys

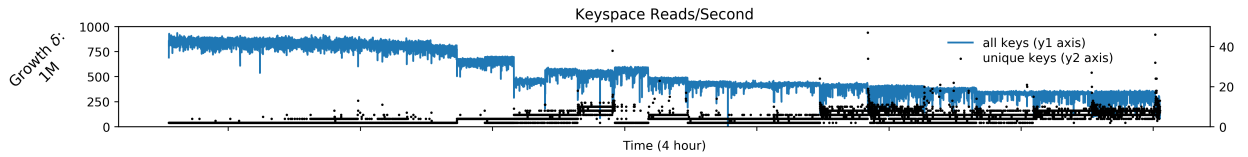


FIG. 3. Access patterns for a growth δ of 1M shows similar behavior to 100K but on different time scales.

that their job phases can be learned and exploited using dynamic load balancing policies.

2. HPC key-value store workloads are so structured that one policy-fits-all
3. HPC key-value store workloads are not structured enough to be learned
4. HPC key-value store workload hotspots/flash crowds are too fast to be exploited

4. Conclusion

1. analysis of Parsplice keyspace
2. using a modern distributed kv store
3. positive effects of Mantle

Acknowledgments. Start acknowledgments here.

References

- [1] D. Perez, E. D. Cubuk, A. Waterland, E. Kaxiras, and A. F. Voter. Long-Time Dynamics Through Parallel Trajectory Splicing. *Journal of chemical theory and computation*.