ParSplice Keyspace Locality

1 INTRODUCTION

ParSplice [1]

2 PARSPLICE KEYSPACE ANALYSIS

2.1 Structured Access Regimes

Figures 1, 2 and 3.

- change immediately, different sizes
- monotonically increasing
- random access to a single key
- early keys accessed more

Conclusion: unique patterns of a real HPC application

2.2 Cache Size Analysis

Figure 4.
Conclusion: workload phases (high request rate to a small number of keys and then low request rate to a large number keys) need a dynamic load balancing policy.

2.3 Overfitting Policies

Figures 5 and 6.
Side Idea: we need to figure out what ParSplice memory is sensitive to: max usage or usage over time.

Conclusion: dynamic policies absorb the cost of a high read request rate for a 2.5 hour run, but it is infeasible to do this for every combination of system setup, job lengths, parsplice parameters.

Discussion. Why don't we just an LRU cache:

- we can do better (workload is structured and has locality)
- finding the size of the cache is hard
- hotspots dissipate too quickly

3 MANTLE ENGINE: METHODOLOGY

3.1 Experimental Setup

3.2 Mantle: Dynamic Load Balancing Policies

- motivation: Mochi load balancer microservice
- background: CephFS implementation
- library architecture, callbacks, environment

3.3 Integrating Mantle into ParSplice

- providing environment of metrics
- identifying where policies are made

4 MANTLE BRAINS: TOOLS WE PLUG IN

Requirements: run online, be fast enough to run as often as we want to detect regimes

,
2017. 978-x-xxxx-xxxx-x/YY/MM…$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

4.1 Failed, Overcomplicated Brains

These techniques proliferated more, less transparent knobs

- Statistics
- Calculus
- K-Means
- DBScan
- Anomaly Detection

4.2 System Specific Knowledge

We tried looking at system-specific metrics to design policies, like request rate, unique keys in a sliding window, bandwidth capabilties. For example, we know that LevelDB cannot handle high IO request rates.

4.2.1 when: request rate.

4.2.2 when: Belady's Min.

4.3 Domain Specific Knowledge

We tried looking at domain-specific metrics to design policies, like ParSplice key access locality.

4.3.1 howmuch: regime detection.

4.3.2 when: trajectory length goodness.

4.3.3 howmuch: cache policy from past.

4.4 Cloud Techniques: Elastic Search

What if we re-provision resource in response to events outside the application's control, such as a slow Lustre.

5 RELATION TO FILE SYSTEMS

- Lustre Trace
- LinkedIn Trace
- Nathan's Trace

5.1 Using File System Balancers for ParSplice

5.2 Using ParSplice Balancers for File Systems

5.3 Visualizing File System Traces like ParSplice Keyspace Traces

6 CONCLUSION

REFERENCES

[1] Danny Perez, Ekin D Cubuk, Amos Waterland, Efthimios Kaxiras, and Arthur F Voter. Long-Time Dynamics Through Parallel Trajectory Splicing. *Journal of chemical theory and computation* (????).

, ,

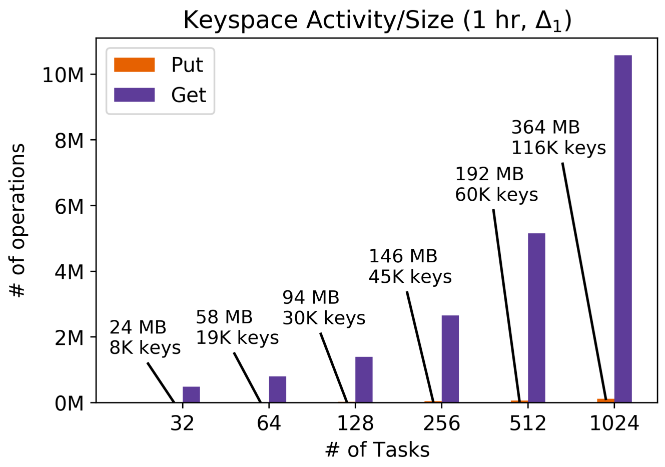

Figure 1

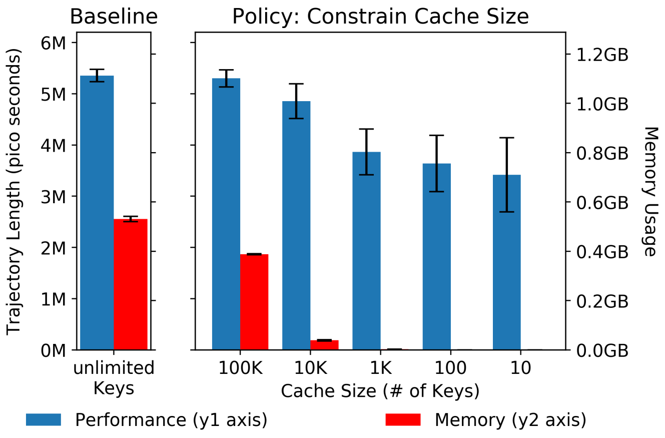

Figure 4

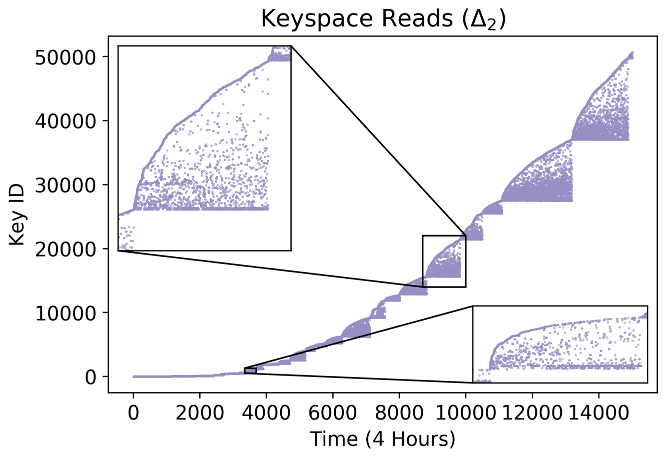

Figure 2

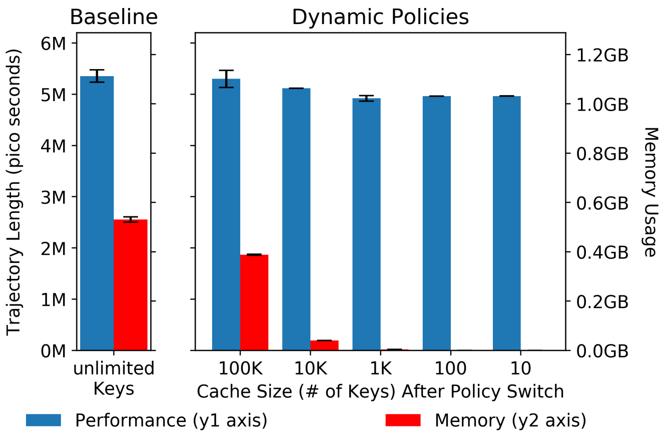

Figure 5

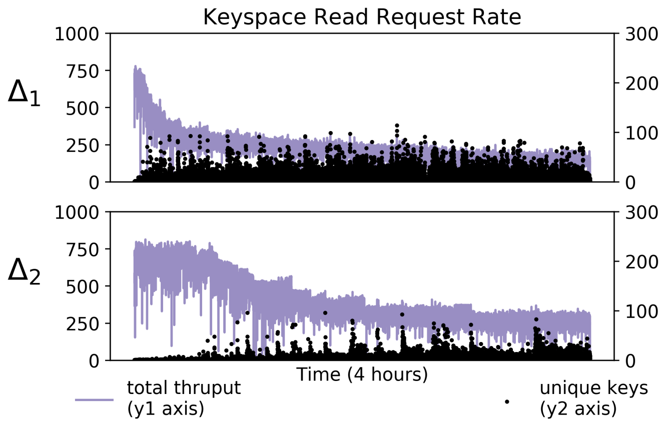

Figure 3

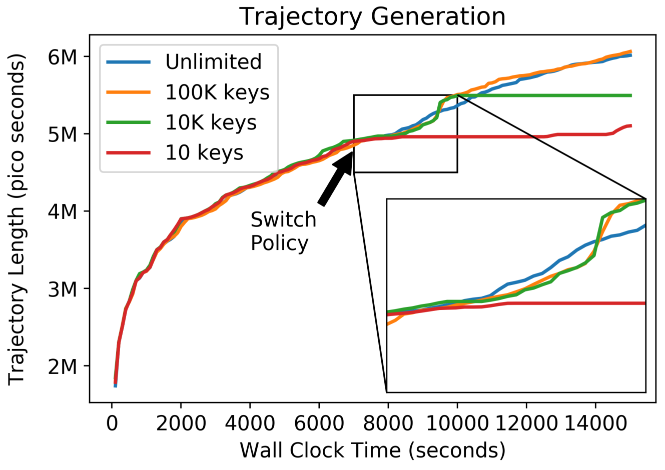

Figure 6