Tiered-CRUSH: A High-Performance Data Placement for Multi-Tiered Storage System

Wei Xie, Jiang Zhou, Mark Reyes, Jason Nobel and Yong Chen
Department of Computer Science and Nimboxx, Inc.

Center for Cloud and Autonomic Computing

Abstract

- Data placement on multi-tiered storage system
- Limitation in consistent hashing and CRUSH
- Integrate workload characterization and storage tier information into the data placement
- Borrow ideas from CRUSH, but with consideration of heterogeneous storage devices and access patterns
- Target on achieving maximized performance and load balance without much data migration overhead

Motivation and Goals

- Requirements for load balancing and maximizing throughput are conflict
- For load balancing, place more data on HDDs, in which contain more data; for maximizing throughput, placing more data on SSDs with higher read and write bandwidth
- A lot of IO workloads are highly skewed: the majority of data accesses concentrate on a small part of data
- Hot and cold data
- Goal: provide load balance while put hotter data in faster storage tier as much as possible
- Should still satisfy the requirement of distributed store: scalability, low migration overhead, fault-tolerance

Methods and Techniques

- Tiered-CRUSH algorithm

Similar to CRUSH algorithm, but assign weights to data objects to represent hotness, assign weights to tiers to represent their throughput

- Tier-map configuration

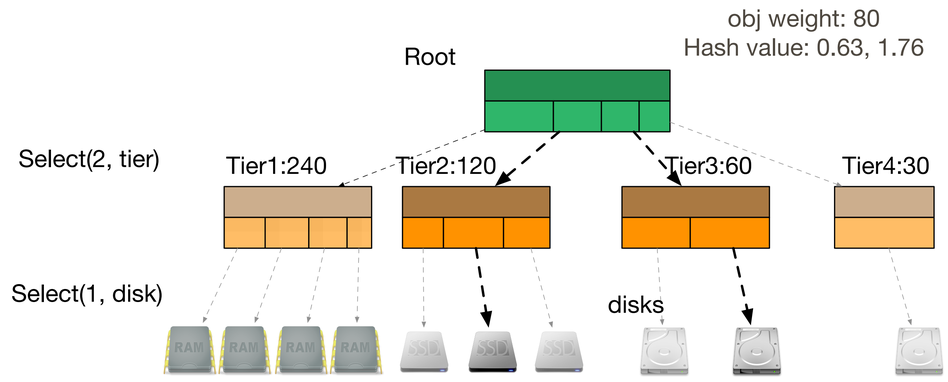
The weight of each tier should be configured to appropriately represent its total performance and capacity so that the resulted data distribution is able to achieve load-balance and maximum throughput

Current Status

- Completing the design details of the algorithm and workload characterization component
- Starting to implement the algorithm on Ceph
- Plan to experiment on the cluster at TTU
- May need to test on a cluster with more storage tiers (Nimboxx might be able to provide)

Project Results or Plans

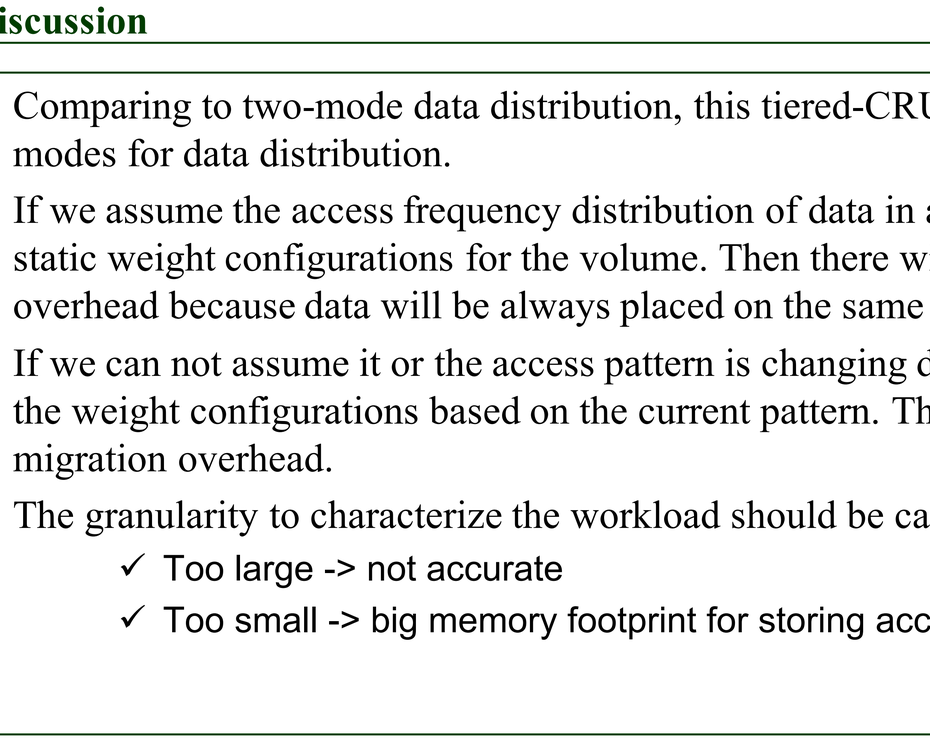
- Tier-map configuration and data placement



$$C_{1i} = |W(id) - W(i)| \times \text{hash}(id, r, i)$$

r - the number of replicas,
id - is the datum id
i – the tier number
Choose the tier i that C_i is smallest

- Each logical volume may be configured with different tier-map, weight numbers or even distribution algorithms



Steps to determine the weight of each tier:

1. Get the percentage of capacity of each storage tier in the whole cluster ($p_1, p_2, \ldots, p_n$)
2. Sort the data segments by the access frequency in descending order
3. Divide the data segments so that the total access frequency of each segment has p_i percent of the whole access frequency in the cluster
4. Set the weight of each tier as the average weight of segments in each divided area in step 3

- Milestones:
  - ✓ Detailed design of the algorithm
  - ✓ Implement the algorithm in Ceph
  - ✓ Develop necessary component in Ceph to support the design
  - ✓ Test the algorithm under benchmarks and real applications

Discussion

- Comparing to two-mode data distribution, this tiered-CRUSH does not need multiple modes for data distribution.
- If we assume the access frequency distribution of data in a volume is known, we can set static weight configurations for the volume. Then there will be no data migration overhead because data will be always placed on the same set of nodes
- If we can not assume it or the access pattern is changing dramatically, we need to adapt the weight configurations based on the current pattern. Then there could be data migration overhead.
- The granularity to characterize the workload should be carefully considered:
  - ✓ Too large -> not accurate
  - ✓ Too small -> big memory footprint for storing access information

Acknowledgements

We are grateful to the Nimboxx and the Cloud and Autonomic Computing site at Texas Tech University for the valuable support for this project. Thank High Performance Computing Center at Texas Tech University for providing the computing resources and support for this project.

The Center for Cloud and Autonomic Computing is supported by the National Science Foundation under Grant No. 1362134.

NIMBOXX

TEXAS TECH UNIVERSITY