Abstract

- 2 new papers completed
  - Tiered-CRUSH
  - Pattern-directed Replication Scheme
- 1 new paper in preparation: version consistent hash
- 2 papers under review: PRS and SUORA
- Simulation code and prototype developed to evaluate the proposed schemes
- Proposed schemes improve the performance of heterogeneous storage systems and maintain the balanced storage utilization

Motivation and Goals

- Most data centers use heterogeneous storage combining hard disk drives HDD with emerging storage class memory SCM (e.g., solid state drives SSD and phase change memory PCM)
- Lack a storage system that unifies the management of heterogeneous storage devices efficiently
- Data are replicated for data availability, but not for performance
- To take full advantage of faster storage devices’ performance and the hard disk drives’ cost-efficiency

Research Paper 1: Tiered-CRUSH

- CRUSH ensures data placed across multiple independent locations to improve data availability
- Tiered-CRUSH integrates storage tiering into the CRUSH data placement
- The virtualized volumes have different access pattern
- Access frequency of object recorded per volume, hotter data more likely to be placed on faster tiers
- Fair storage utilization maintained

Research Paper 2: Pattern-directed Replication Scheme

- Overview of a pattern-directed replication scheme
  - (1) The original objects are placed on hybrid nodes in default layout
  - (2) The runtime object I/O requests are traced by a trace collector
  - (3) The scheme analyzes the trace, reorganizes objects for identified access patterns and makes replications
  - (4) When applications run again, objects are read from replicas in optimized layout
- Object distance calculation
  - (a) Local access pattern
    \[ \text{dist}(o_1, o_2) = 1 - \min \left\{ \frac{\text{count}(o_1, o_2)}{\text{count}(o_1)}, \frac{\text{count}(o_1, o_2)}{\text{count}(o_2)} \right\} \]
  - (b) Global access pattern
    \[ \text{dist}(o_1, o_2) = \frac{\text{count}(o_1) - \text{count}(o_2)}{\text{count}(o_1) + \text{count}(o_2)} \]
- Evaluation
  - FIO with different numbers of processes
  - (a) Random Read
  - (b) Sequential Read
  - (c) Overhead

Research Paper 3: Version Consistent Hashing

- Build versions into the virtual nodes
- Avoid data migration when adding nodes or node fails
- Maintain efficient data lookup
- Example of 3 versions of a consistent hashing ring
- Data lookup algorithm
  - v1: 1, 2
  - v2: 4, 1
  - v3: 4, 6
  - v4: 4, 6
  - Lookup locations: (4, 6, 1, 2)
- Performance Improvement

Acknowledgements

We are grateful to 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.