Attributed Consistent Hashing for Heterogeneous Storage System

Jiang Zhou, Wei Xie, Yong Chen
Department of Computer Science
Texas Tech University

Abstract

- **Objectives**
  - To design a data placement algorithm for heterogeneous storage composing with hard disk driver (HDD) and solid state disk (SSD)
  - Based on consistent hashing to keep its inherent features while making better utilization of heterogeneous storage devices

- **Background**
  - Cloud-scale storage system is an important building block of the cloud infrastructure, which demands efficient and flexible data distribution for performance improvement
  - Heterogeneous storage system is a very promising trend in data centers

- **Contributions**
  - A novel data distributed placement algorithm based on consistent hashing for heterogeneous storage systems
  - Improve the I/O performance while keeping storage space load balanced and well utilized

Motivation

**Heterogeneous device characteristics**

A heterogeneous storage system, which contains nodes with both HDDs and SSDs, is often desired to combine merits of different storage devices.

- I/O bandwidth comparison of the HDD (Seagate ST950620NS) and SSD (Intel SSDSC2BA200G3T)
- SSD favors workloads with small I/O size
  - The bandwidth of SSD is 65 times higher than that of HDD when the request size is 4KB
- SSD excels in random I/Os generally
  - The SSD achieves nearly 2 to 9 times higher throughput than that of HDD when the request size varies from 64KB to 4MB

**Consistent hashing algorithm**

Consistent hashing (used in Dynamo, Chord, and Sheepdog) conceptually assigns nodes and data to a ring. The data is assigned to the “k closest" nodes with the hashing value. Virtual nodes can be added to improve load balance.

Data x and its replicas will be mapped to 3 “closest" nodes (A, B, C) with clockwise way. The virtual node (A2) will be skipped for data reliability.

**Challenge**

Consistent hashing is mainly designed for a homogeneous environment and lacks efficient method to distinguish heterogeneous device characteristics.

Methodology

- **Attributed Consistent Hashing Ring**
  - Add two attributes to nodes on the hashing ring
  - Capacity: the overall data capacity of a node
  - Bandwidth: the maximum bandwidth of a node

- **Consider Capacity Attribute**
  - Making virtual nodes according to node capacity
    - Making $c_i = \min_{v_j \in V} v_j$ virtual nodes for each physical node, where $c_i$ is the capacity of node $v_i$.
    - Proportionally adjust virtual node numbers if they are too large
    - To avoid the declustering of data movement on each node

- **Consider Bandwidth Attribute**
  - Define a range for each virtual node according to node bandwidth
  - Each node has a range on the hashing ring. The virtual node has the same range with the physical node. Range length is computed with a predefined parameter $p$ with node configuration

- **Data Placement algorithm**
  - Step 1: data is hashed to a position on the ring with hash value $s$
  - Step 2: data is assigned to a node with pseudo-number algorithm $val = hash(s, x, max)$, where $x$ is data ID, $s$ and $max$ are lower and upper range limits of the hashing ring.

Performance Results

- **Simulation tests**
  - Make tests in a modular consistent hashing library libch-placement
  - Results show our algorithm achieves similar calculation time and memory usage with traditional consistent hashing

| Table 1 The calculation time of different algorithms (s) |
|---|---|---|
| Node number | Our algorithm | Consistent hashing | CRUSH |
| 64 | 0.0131 | 0.0124 | 1.4591 |
| 512 | 0.0218 | 0.0204 | 9.1039 |
| 4096 | 0.0361 | 0.0332 | 70.2185 |
| 32768 | 0.1783 | 0.1948 | 58.24153 |

| Table 2 The memory usage in different configs (MB) |
|---|---|---|
| Node No. * X Virtual No. | Our algorithm | Consistent hashing |
| 100 * 256 | 0.98 | 0.78 |
| 1000 * 256 | 8.77 | 7.81 |
| 10000 * 256 | 81.65 | 78.13 |
| 10000 * 1024 | 39.06 | 31.25 |
| 10000 * 4096 | 1953.0 | 1562.4 |

**Performance Evaluation on Sheepdog storage system**

- On a cluster of 30 nodes (half SSDs) with FIO and Filebench
- Use FNV-1 hash function for consistent hashing and SIMD-oriented Fast Mersenne Twister (SFMT) for generate pseudo random numbers
- Results show the performance can be improved more than 3 times

Conclusion

We propose a consistent hashing based data placement algorithm to promote better utilization of heterogeneous storage devices. This work is supported by the National Science Foundation under grant CNS-1338078 and IIP-1362134 through the Nimboxx membership contribution.