Attributed Consistent Hashing for Heterogeneous Storage System

Jiang Zhou, Wei Xie, and Yong Chen

Computer Science Department, Texas Tech University, {jiang.zhou, wei.xie, yong.chen}@ttu.edu

Cloud-scale storage system is an important building block of the cloud infrastructure. It demands the flexibility to distribute data and provide high I/O performance. Consistent hashing algorithm is widely used in large-scale parallel/distributed storage systems for the decentralized design, scalability and adaptability. It can evenly distribute data among nodes but lack efficiency in a heterogeneous environment. In this research, we propose a novel data placement algorithm, which is based on consistent hashing while making it more efficient for heterogeneous storage systems. By considering both the capacity and bandwidth attributes of nodes, our algorithm can make better use of heterogeneous devices. Our current proof-of-concept evaluations with a distributed stroage system, Sheepdog, show promising results.

I. INTRODUCTION

The data explosion in the trending big data era exposes significant challenges to the underlying storage systems. Recently, major Internet service companies like Google and Yahoo! have all built up their cloud-scale data centers to accommodate the massive amount of storage and computing needs of their applications. Cloud-scale storage system is an important building block of the cloud infrastructure, which demands efficient and flexible data distribution for performance improvement.

Many works have focused on the studies of distributed data placement in a large-scale storage system. Consistent hashing algorithm is widely used in parallel/distributed file systems due to its decentralized design, scalability, and adaptability to node membership changes. It provides a key value store and uses hashing functions to assign data to nodes. In addition, only few data will migrate when node addition or removal occurs. Modern distributed storage systems, such as Chord [1] and Sheepdog [2] often use consistent hashing to place data on nodes.

The consistent hashing-based solutions can achieve efficient data placement, but they can not provide balanced data distribution in a heterogeneous storage system environment, such as nodes consisting of hard disk drivers (HDDs) and solid state disks (SSDs). It is well observed that SSDs, have much smaller capacity than HDDs whereas providing significantly better performance. This raises the problem that the capacity-based distribution can compromise the over performance with the SSD under-utilized. On the other hand, the SSD may be overwhelmed for small capacity if only considering the performance requirements. It is a challenge to distinguish heterogeneous device characteristics and make full use of their advantages.

II. RESEARCH OBJECTIVES

Our goal is to design a data placement algorithm for heterogeneous storage systems. We introduce a consistent hashing-base algorithm, which can keep the hash algorithm's inherent features while making better utilization of heterogeneous storage devices. By considering both the capacity and bandwidth attributes of the nodes, our algorithm makes an efficient and scalable data placement.

III. METHODOLOGY

A. Node Attributes

In a heterogeneous system equipped with HDDs and SSDs, the nodes can have different characteristics. To distinguish the merits of HDD and SSD nodes, we add attributes to each node. We mainly consider two attributes, capacity and bandwidth, as they have significant impact on the performance of data storage. Normally, the SSD provides high bandwidth, low latency, and mechanical-component-free characteristics, compared with HDD. But the small capacity, short lifetime, and high prices of SSD still restrict their full utilization in the storage system.

Our algorithm is based on the consistent hashing, which hashes the nodes on a "hash ring" with their hostname or address and places data to the node according to the hash result of the data identification. Different from traditional consistent hashing, our algorithm maintains two attributes for each node on the hashing ring. Given a set of n nodes $V = \{v_1, \ldots, v_n\}$ with attributes of $v_i = < c_i, b_i >$, we can construct an attributed hashing ring. The virtual nodes can also be used in our algorithm. For instance, the hashing ring in Figure 1 contains three nodes A, B, C ($A1$ and $A2$ are the virtual nodes of A). The attributes of different nodes can be described as $a_1 = < 1000, 146 >$, $b_1 = < 256, 540 >$, and so on. It means that node $A1$ has 1TB capacity and 146 MB/s bandwidth, node $B1$ has 256 GB capacity and 540 MB/s bandwidth, respectively. The attributes can be set according to storage device configuration. For virtual nodes, they have the same attributes with the physical node. For instance, the nodes $A1$ and $A2$ are virtual nodes of the same physical node A, which have the same attributes.

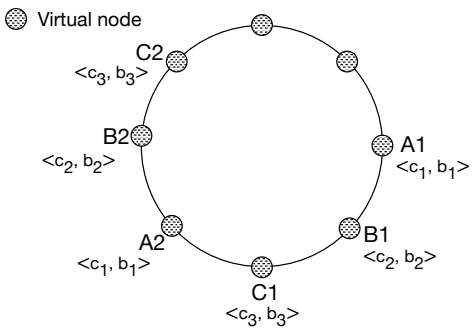

Fig. 1: Consistent hashing ring with node attributes

B. Virtual nodes adjustment

The virtual nodes are often used for data balance in consistent hashing algorithm. Some algorithm adjust the number of virtual nodes to distinguish different node capacity [3]. It is efficient to some extent as the method can proportionally distribute data on different nodes according to their capacity. With the similar idea, our algorithm adjusts virtual node number by the capacity attribute but is more flexible than current method.

The naive approach is to use $\left\lceil \frac{c_i}{min_{j \in V\{c_j\}}} \right\rceil$ virtual nodes for each node, where c_i is the capacity attribute of node v_i. This is not feasible, if $max_{j \in V\{c_j\}}/min_{j \in V\{c_j\}}$ is too large [3]. For example, SSD nodes with small capacities will significantly increase the number of virtual nodes of all nodes. The evaluation test shows that data movement will occur on more nodes when a node fails if the virtual node number increases [4]. Though the total amount of data migration is constant, the declustering of data movement on various nodes will result in much network communication and data synchronization. To avoid this issue, our algorithm will proportionally adjust virtual node numbers if they are too large. It can take advantage of the capacity attributes while reducing data movement for nodes.

C. Data placement algorithm

With the virtual node adjustment, our algorithm can use the capacity attribute of each node. To address the bandwidth attribute, we define a range for each virtual node according to the node bandwidth. Each node is then assigned to a range in order from the start hash value 0 on the hashing ring. The range length is calculated via dividing the node bandwidth by a predefined parameter p. The virtual node also has the same range length with its physical node.

Thus, our algorithm places data with two steps. First, the data is hashed to a position on the ring with hash value s. Second, data is assigned to a node with pseudo-number algorithm $val = f(x, s, max)$, where x is data ID, s and max are lower and upper range limits of the algorithm. The data will be mapped to a node when the pseudo number val matches the value of the node's range. Figure 2 illustrates the data placement with our algorithm. Given the parameter $p = 146MB/s$, the four virtual nodes (A1, B1, A2, C1) are assigned to different ranges. Node A has two virtual nodes with two ranges, which are $[0, 1)$ and $[2.5, 3.5)$. For instance, the data x will be placed on node $B1$ if the random number generated $val = 1.8$. With the replica number r, there is r random numbers generated for mapping r nodes on the hashing ring. This way, our algorithm both considers the capacity (with virtual node number) and bandwidth (with virtual node range) for data placement.

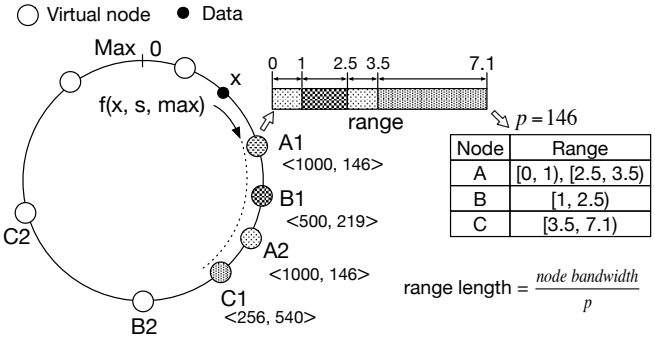

Fig. 2: Illustration of data placement

IV. Summary of Results

Our simulation tests show that the proposed algorithm achieves similar calculation time and memory usage with traditional consistent hashing. The overhead is negligible for the adding of node attributes. Performance evaluation on Sheepdog storage system shows that our algorithm can significant improve the read performance.

V. Conclusion and Future Work

In this study, we propose a consistent hashing based data placement algorithm to promote better utilization of heterogeneous storage devices. We consider both the capacity and bandwidth attributes when distributing data on the nodes of hashing ring. The evaluation results proves the efficiency of our algorithm. We plan to further investigate data distribution and replica management in heterogeneous storage systems and improve the productivity of advanced computing systems.

References

[1] I. Stoica, R. Morris, D. Nowell, D. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[2] "Sheepdog Project," 2015. [Online]. Available: http://www.sheepdog-project.org/.

[3] C. Schindelhauer and G. Schomaker, "Weighted distributed hash tables," in *Proc. of the 17th annual ACM symposium on Parallelism in algorithms and architectures (SPAA)*, 2005, pp. 218–227.

[4] "The libch-placement library." [Online]. Available: https://xgitlab.cels.anl.gov/codes/ch-placement/