Future Generation Computer Systems 78 (2018) 920–930

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

An efficient load-balancing mechanism for heterogeneous range-queriable cloud storage☆

Xun Shao *, Masahiro Jibiki, Yuuichi Teranishi, Nozomu Nishinaga

National Institute of Information and Communications Technology, 4-2-1, Nukui–Kitamachi, Koganei, Tokyo, 184–8795, Japan

HIGHLIGHTS

- We identified the fundamental differences between range queriable systems and DHT systems in load balancing.
- We introduced a novel distributed virtual node-based load balancing method for range queriable Cloud storage systems.
- Our method is completely distributed, so that it has good scalability for cloud storage networking systems.
- As our method is based on virtual nodes, it can be applied to heterogeneous environment inherently.

ARTICLE INFO

Article history:
Received 9 November 2016
Received in revised form 29 May 2017
Accepted 21 July 2017
Available online 1 August 2017

Keywords:
Cloud storage
Key–value store
Range query
Load balancing
Virtual node

ABSTRACT

The rising popularity of big data processing for semantically rich applications such as social networks and IoT (Internet of Things) has made the range-queriable cloud storage increasingly important. To support range queries, the data locality is preserved strictly, which makes the load balancing among nodes a challenging task. Currently, most of the range-queriable cloud storage adopts the combination of neighbor item exchange and neighbor migration methods, which incurs large overhead, and suffers from slow convergence. In this work, we present a novel virtual node based decentralized load-balancing method for range-queriable cloud storage. In our method, each physical node is partitioned into multiple virtual nodes, and all the virtual nodes are organized with range-queriable P2P network. Load balancing is conducted in both overlay level (between neighboring virtual nodes) without global knowledge and physical level (among physical nodes) with limited global knowledge. Both theoretical analysis and simulations show that our method can significantly reduce the overhead and shorten the convergence time.

1. Introduction

The cloud storage is becoming increasingly more important for cloud applications, as it can seamlessly handle huge amount of data efficiently [2–14]. To achieve high scalability required by cloud applications, most of the cloud storage systems employ the non-relational data structures such as key–value and BigTable-like table-based data structure to simplify the relations among data items. This kind of data structure enables cloud service providers to build cloud storage systems by managing a large number of connected commodity machines. In such kind of systems, distributing data items to storage nodes according to their capacities dynamically is very important. Skewed data distribution will decrease the performance and even cause system failure. For example, the total execution time of a MapReduce task mostly depends on the execution time of the node with the heaviest workload, even though most of the other nodes have only a little workload.

Due to the excellent load-balancing characteristics, DHT-based key–value stores become one of the most popular cloud storage technologies. DHT-based key–value store employs consistent Hashing function to transform both the keys of data items and the addresses of storage devices into hash values without semantic meanings, and then distributes data items into different storage devices if their hash values are "close". Although DHT-based key–value store can balance the workload among nodes simply, range queries cannot be supported efficiently because the data locality is destroyed by consistent Hashing. Specifically, in DHT-based key–value stores, data items with similar semantic meaning are always far away from each other physically (i.e. stored in different physical machines), and therefore, it is difficult to execute range queries which are important for social network and IoT services. Range query is an important functionality as it allows flexible query

☆ This paper is an extended version of the work originally to be presented at IEEE CloudCom 2015 Shao et al. (2015) [1].

* Corresponding author.
E-mail addresses: x-shao@nict.go.jp (X. Shao), jibiki@nict.go.jp (M. Jibiki), teranisi@nict.go.jp (Y. Teranishi), nisinaga@nict.go.jp (N. Nishinaga).

http://dx.doi.org/10.1016/j.future.2017.07.053
0167-739X/

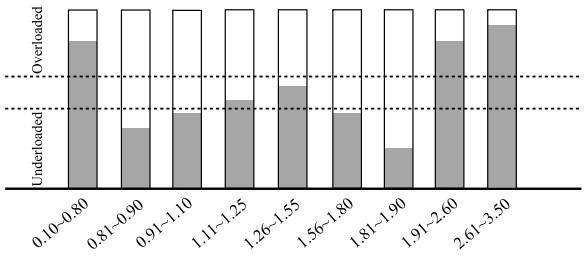

Fig. 1. An example of range-queriable cloud storage system. The range below each node is its responsible range.

conditions without specifying exact keys of the data required. Consider that in a key–value store of environment monitoring sensor information, the key is location and the data is sensor description. If a user wants to know the sensor information in a certain region, she can query the system with a fuzzy condition such as "sensors within 1 km from 45° northern latitude, 40° north latitude".

To support range queries, the order of keys must be preserved, which makes load balancing a very challenging task. Fig. 1 shows a simple image of range-queriable cloud storage. We can see that each storage node has a responsible range, and the responsible ranges of different nodes do not overlap. A data item is stored in a specific storage node if and only if its key falls into the responsible range of that node. Due to the skewed data distribution, skewed data insertion and deletion, the responsible ranges of storage nodes have to be adjusted to sustain the load balancing. Some of the existing range-queriable cloud storage systems employ a central management node to manage the metadata information of physical nodes and data items, and make load-balancing decisions [3]. There are also systems that employ decentralized load-balancing methods. In these systems, the storage nodes form a range-queriable P2P network to realize self-organizing [2,4]. The storage nodes exchange load information and make load-balancing decisions independently. For both centralized and decentralized load-balancing methods, most of the existing research employs the combination of NBRADJUST and REORDER as the basic mechanism.

Fig. 2(a) shows an example of NBRADJUST. With NBRADJUST, an overloaded node can move a portion of its load to a neighbor. In a system with N nodes, the time complexity of load balancing with NBRADJUST is O (N) because NBRADJUST can only be carried out between neighboring nodes. Due to performance reasons, REORDER is rarely employed as the single load-balancing mechanism, but is always used as complement of the NBRADJUST method. With REORDER method, an underloaded node first executes a NBRADJUST operation to move its load to a neighbor (no matter whether the neighbor is overloaded or underloaded), then leaves the system and joins again as a neighbor of an overloaded node. After the re-joining, the overloaded node can move a portion of its load to this new empty neighbor with a NBRADJUST operation (Fig. 2(b)). This method suffers mainly from the data movement overhead. Specifically, to share the load with an overloaded node, an underloaded node has to move all its load to a neighbor first. This data movement can be seen as overhead and "unnecessary". Correspondingly, the time consumed by the "unnecessary" data movement can be considered as time overhead. Even worse, after the underloaded node moves its load to a neighbor, the neighbor may become a new overloaded node, and the load-balancing process has to be carried out iteratively, which may take a lot of time to converge.

In this work, we introduce a novel decentralized virtual node-based load-balancing method, while avoiding NBRADJUST and REORDER. The virtual node-based load-balancing method is widely used in DHT systems, however, the fundamental difference between range-queriable system and DHT system makes our method quite different from the existing work. In our method, we divide each physical node into multiple virtual nodes with identical capacity, and construct range-queriable P2P network in virtual node level rather than physical node level. We keep some of the virtual nodes in each physical node filled with data items, and the others empty. By maintaining the load of each non-empty virtual node in a certain range, the load of physical nodes can be approximated as the ratio of non-empty virtual nodes to the total virtual nodes within them. With this approximation, we can consider that a physical node is overloaded if it contains too many non-empty virtual nodes and only a few empty-virtual nodes. To balance the load between an overloaded physical node and an underloaded physical node, we can simply move the load from a non-empty virtual node in the overloaded physical node to an empty virtual node in the underloaded physical node. This mechanism incurs much less data movement overhead than the NBRADJUST and REORDER-based methods in that data movement only happens between overloaded node and underloaded node, and new overloaded node will not be produced as NBRADUST and REORDER-based methods do. Correspondingly, the proposed mechanism can also converge fast. To realize this mechanism, we addressed many challenging problems in this work, including constructing overlay network with virtual nodes to realize self-organization while preserving the order of data items, maintaining the load of non-empty virtual nodes in a certain range locally and dynamically, evaluating the average load of the system in realtime for physical nodes, discovering underloaded physical nodes to share the workload for the overloaded nodes and so on. Our method is completely decentralized, so that it has good scalability for cloud storage systems. Moreover, as our method is based on virtual nodes, it can be applied to heterogeneous environment inherently. Although the research objective of this work is to distribute data items among storage devices to adapt to their capacities, the proposed method can also be extended to be applied to distributing other kinds of workload such as query accessing workload. The proposed method can be applied if only the system has a unique kind of bottleneck.

The rest of the paper is organized as follows. In Section 2, we briefly introduce the related research about load balancing for range-queriable systems, then in Section 3, we present our method in detail, including the load-balancing framework, the key technologies to realize the framework and some theoretical analysis. In Section 4, we present extensive simulation results to show the effect of our method. Some implementation issues are discussed in Section 5, and in Section 6, we conclude the work and discussed our future plan.

2. Related work

In this section, we give a brief introduction of the load-balancing technologies for range-queriable systems. Different from DHT systems, the load-balancing problem of range-queriable system is very challenging because the order of data items has to be preserved. One of the earliest research was conducted by Aspnes et al. [15]. In their work, the authors described a load-balancing mechanism for skip graphs and similar distributed data structures. The mechanism is based on a global threshold, where nodes with load below the threshold continue to accept new data items and nodes with load above the threshold attempt to shed data items. Although the performance is good according to experiments, the authors did not prove that it works in theory. In the promising work of Ganesan et al. [16], the authors presented asymptotically optimal online load-balancing algorithm based on the combination of NBRADJUST and REORDER, which becomes the basic mechanism for the load-balancing of range-queriable systems. Based on their