J. Parallel Distrib. Comput. 74 (2014) 2934–2940

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

Optimal metadata replications and request balancing strategy on cloud data centers

Zeng Zeng, Bharadwaj Veeravalli *

Computer Networks and Distributed Systems Laboratory, Department of Electrical and Computer Engineering, The National University of Singapore, 10 Kent Ridge Crescent, Singapore 117576, Singapore

HIGHLIGHTS

- In this paper we address an important problem pertaining to Metadata Server Clusters.
- We propose strategies to handle metadata replication and a load balancing strategy to minimize mean response time.
- We present rigorous theoretical analysis of the strategies proposed and present a practical algorithm.
- We compare our findings with most commonly used strategies that use hashing functions and show a significant gain.
- We demonstrate a trade-off relationship between makespan and the monetary cost.

ARTICLE INFO

Article history:
Received 11 September 2013
Received in revised form
28 February 2014
Accepted 23 June 2014
Available online 5 July 2014

Keywords:
Mean response time
Request balancing
Distributed system
Metadata server
Queueing theory

ABSTRACT

In large-scale cloud data centers, metadata accesses will very likely become a severe performance bottleneck as metadata-based transactions account for over 50% of all file system operations. Clusters of Metadata Servers (MDS) that provide metadata searching service can improve the system performance significantly. For a data stored in cloud data centers, there may be several MDS storing the metadata replicas. Therefore, when a data request arrives, it has many potential metadata paths, one of which shall be chosen to obtain the best performance. In this paper, we attempt to determine the number of MDS that each data object in the system shall have and the request rates that each MDS shall serve, in order to achieve the minimum mean response time (MRT) of all the metadata requests. The target optimal constrained function has been formulated and a novel metadata request balancing algorithm based on request arrival rates has been proposed, which can find near-optimal solutions by a theoretical proof. In our experiments, we compare our algorithm with widely used hashing functions that have 0, 1, 2, 3 replicas, respectively. We validate our findings via simulations with respect to several influencing factors and prove that our proposed strategy is scalable, flexible and efficient for the real-life applications. Some interesting perspectives of the work are also presented at the end of this paper.

1. Introduction

The emergence of cloud computing poses a grand challenge for modern data centers. More and more large commercial web-based services, such as Google, Amazon, and Alibaba, serve millions of people every day, which are extremely important services but also very expensive to host. A web index, e.g., some search functions within a single cloud, contains billions of documents that are partitioned and managed in thousands of search servers. Furthermore, data-intensive scientific applications, which mainly aim at answering some of the most fundamental questions facing human beings, are becoming increasingly prevalent in a wide range of scientific and engineering research domains, such as high-energy particle physics and astronomy [10], and climate change modeling [15]. In such applications, large amounts of data sets (files) are generated, accessed, and analyzed by scientists and researchers worldwide. One distinct feature of cloud data centers is that they manage very large amount of data sets, in the order of terabytes and petabytes. For example, the Large Hadron Collider (LHC) at the European Organization for Nuclear Research (CERN) is the largest scientific instrument on the planet. Since it began operation in August, 2008, it was expected to produce roughly 15 petabytes of data annually, which are accessed and analyzed by thousands of scientists and researchers around the world [13].

Cloud data centers have service-oriented architectures, in which services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS) that includes equipments such

* Corresponding author.
E-mail addresses: elebv@nus.edu.sg, elezengz@nus.edu.sg (B. Veeravalli).

http://dx.doi.org/10.1016/j.jpdc.2014.06.010
0743-7315/

as hardware, storage, servers, and networking components are made accessible over the Internet; Platform-as-a-Service (PaaS) that includes hardware and software platforms such as visualized storage servers, operating systems, and the like; and Software-as-a-Service (SaaS) that includes software applications and other hosted services such as network disks [6].

In cloud data centers, any arrival request is serviced within a suitable metadata server (MDS), in order to retrieve the metadata of the target data, and then the request will be forwarded to the raw data server (RDS) for the target data retrieval [5,17]. An MDS may contain different computing, memory, harddisk, bandwidth resources, and others. In order to provide the best QoS of web-based services, storage service providers normally have multiple replicas within data centers and hope the Mean Response Time (MRT) of all the requests can be reduced by load balancing among these replicas existing on different MDS. Normally, large-scale file systems, such as Ext3 [5] and Luster [17], adopt hashing functions to balance the requests for the same data objects among MDS. However, a cloud data center can have a large number of MDS, typically of the order of tens or hundreds and hashing functions can just balance few of the MDS at a time [6]. From the system aspect, although the set of MDS that have a replica of the metadata is well balanced, the entire system is still un-balanced.

In this paper, we attempt to provide a mathematical formulation that captures and demonstrates the ability to evaluate the potential data flow paths within cloud data centers. Based on this formulation, we devise distributed strategies to achieve minimum MRT of all requests arriving on MDS within cloud data centers. We attempt to formulate the load balancing problem of MDS as a non-linear constrained optimization problem. As with the principle of load balancing, requests are allowed to migrate from heavily loaded MDS to lightly loaded MDS for minimizing MRT of the requests. Based on the request arrival rates, our strategy can determine the number of replicas among MDS and their locations. Furthermore, our strategy can balance the metadata requests among the MDS with a metadata replica, in order to achieve the minimum MRT of all the requests. We validate our findings via simulations compared with hashing functions of 0, 1, 2, 3 replicas and prove that our proposed strategy is scalable, flexible, and efficient.

The rest of the paper is organized as follows. Section 2 discusses on relevant research work. Section 3 describes the system model, notation and problem definition. In Section 4, we propose our strategy to place and balance the requests among the MDS within cloud data centers. Section 5 shows the results of our simulation experiments. We conclude our work in Section 6.

2. Related work

The software architectures of cloud data centers are based on the file systems, which can be categorized in general in local file systems and distributed (or cluster) file systems. Classical local file systems, such as NTFS [20] and ext3 [5], which are wildly used in a single machine, consider the storage devices as locally attached to a given host. Storage devices are not shared; therefore, no device-sharing semantics is required in the file system design. The actual goal in designing a local file system is to increase performance by reducing the number of disk accesses through caching and bundling file system operations as much as possible. However, local file systems lack the scalability and cannot meet the requirements in High Performance Computing (HPC) platforms, e.g., cloud data centers, that need the adoption of the distributed memory paradigm by employing multiple machines for peak floating point performances. Compared with the local file systems, cluster file systems, such as Luster [17], Google File System [6], and AFS [7], provide transparent parallel access to the storage devices keeping standard file system semantics. Users only see one logical device through the standard I/O primitives. Cluster file systems emphasize on concepts such as sharing and connectivity as well as client side caching. Unlike local file systems, cluster file systems distribute their resources across the whole storage subsystem, thus allowing simultaneous access from multiple machines. Furthermore, the distributed file system shall address two important issues: provide a large bandwidth for data access from multiple concurrent tasks and can scale to many millions or billions of files among thousands of raw data servers (RDS).

The Google File System (GFS) [6] was proposed to meet the addressed issues and was cloned in open source projects like Hadoop Distributed File System (HDFS) [3] and Kosmos File System (KFS) [12] that are used by companies like Yahoo, Facebook, Amazon, and Baidu. The GFS architecture comprises of a single GFS master server that stores the metadata of the file system and multiple slaves known as chunkservers that store the data. Files are divided into chunks (usually 64 MB in size) and the GFS master manages the placement and data-layout among the various chunkservers. The GFS master also stores the metadata like filenames, size, directory structure and information about the location, and placement of data in memory. One of the direct implications of this design is that the size of metadata is limited by the memory available at the GFS master. This architecture is chosen for its simplicity and works well for hundreds of terabytes with few millions of files [14]. With storage requirements growing to petabytes, there is a need for distributing the metadata storage to more than one server. In Ceph, a dynamic distributed metadata cluster provides extremely efficient metadata management and seamlessly adapts to a wide range of general purpose and scientific computing file system workloads. Ceph has excellent I/O performance and scalable metadata management, supporting more than 250,000 metadata operations per second [18].

Applications communicate with a GFS master only while opening a file to find out the location of the data and then directly communicate with the chunkservers (RDS) to reduce the load on the single master. A typical GFS master can handle a few thousand operations per second [14]. However, when massively parallel applications like a MapReduce [4] job with many thousand mappers need to open a number of files, the GFS master becomes overloaded and turns to the system bottleneck. As cloud data centers grow to accommodate many thousands of machines in one location, distributing the metadata operations among multiple MDS would be necessary to increase the throughput in a single data center. Handling metadata operations efficiently is an important aspect of the file system as they constitute up to half of file system workloads [16]. In order to meet the scalability and functionality requirements for exponentially growing data sets and increasingly complex metadata queries in large-scale, exabyte-level file systems with billions of files, Yu Hua et al. proposed SmartStore in [8]. SmartStore can exploit semantics of files' metadata to judiciously aggregate correlated files into semantic-aware groups by using information retrieval tools. The basic idea of SmartStore is to limit the search scope of a complex metadata query to a single or a minimal number of semantically correlated groups and hence, avoid or alleviate brute-force search in the entire system. While I/O bandwidth available for a distributed file system can be increased by adding more data storage servers, scaling metadata management involves dealing with many issues, such as data consistency across replicated servers, metadata migration and replications, and load balancing [17].

When a request of file is generated by application users, from the view of file systems, there is a looped data path across three layers: file system portal, metadata layer, and data storage layer. It is very interesting to see that the development of file systems follows a bottom-up style, from the data storage layer to other up layers, in order to provide exponentially increasing data storage capacities.