Computer Communications 35 (2012) 1809–1818

Contents lists available at SciVerse ScienceDirect

Computer Communications

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

Cloud-based image processing system with priority-based data distribution mechanism

Tin-Yu Wu [a], Chi-Yuan Chen [b], Ling-Shang Kuo [c], Wei-Tsong Lee [c], Han-Chieh Chao [a,b,d,*]

[a] Institute of Computer Science & Information Engineering, National Ilan University, Taiwan, ROC
[b] Department of Electrical Engineering, National Dong Hwa University, Taiwan, ROC
[c] Department of Electrical Engineering, Tamkang University, Taiwan, ROC
[d] Department of Electronic Engineering, National Ilan University, Taiwan, ROC

ARTICLE INFO

Article history:
Available online 20 July 2012

Keywords:
3D image
Cloud system
Multicast streaming
Image processing

ABSTRACT

Most users process short tasks using MapReduce. In other words, most tasks handled by the Map and Reduce functions require low response time. Currently, quite few users use MapReduce for 2D to 3D image processing, which is highly complicated and requires long execution time. However, in our opinion, MapReduce is exactly suitable for processing applications of high complexity and high computation. This paper implements MapReduce on an integrated 2D to 3D multi-user system, in which Map is responsible for image processing procedures of high complexity and high computation, and Reduce is responsible for integrating the intermediate data processed by Map for the final output. Different from short tasks, when several users compete simultaneously to acquire data from MapReduce for 2D to 3D applications, data that waits to be processed by Map will be delayed by the current user and Reduce has to wait until the completion of all Map tasks to generate the final result. Therefore, a novel scheduling scheme, Dynamic Switch of Reduce Function (DSRF) Algorithm, is proposed in this paper for MapReduce to switch dynamically to the next task according to the achieved percentage of tasks and reduce the idle time of Reduce. By using Hadoop to implement our MapReduce platform, we compare the performance of traditional Hadoop with our proposed scheme. The experimental results reveal that our proposed scheduling scheme efficiently enhances MapReduce performance in running 2D to 3D applications.

1. Introduction

Cloud computing is composed of servers that provide high storage capacity, high flexibility and high-computing performance [1–5]. Because a cloud computing system has several computing servers, users are able to execute procedures that require large amounts of computing resources, process a great deal of data on the cloud, and complete the tasks by low-cost and low-power devices. To process the procedures requested by users with cloud computing, users can simultaneously enhance the performance of the procedures. By using cloud computing for knowledge discovery and data integration [6,7], Google thus proposed the MapReduce programming model.

As a distributed parallel computing architecture presented by Google, MapReduce automatically executes parallel computing and partitions the data input by the developer. To simply write Map and Reduce functions, the developer can execute a full set of computing procedures. MapReduce mainly include three functions: Master, Map function and Reduce function. The Master periodically obtains the status of the servers that run Map and Reduce functions. The Master next divides the input data equally based on the number of the Map function and distributes the sub-tasks to the designated servers. The Map function converts the data designated by the Master to the intermediate data while the Reduce function merges the intermediate data processed by the Map function together and creates the final result.

Derived from an implementation of Google's MapReduce [8], Hadoop consists of two chief components: Hadoop MapReduce and Hadoop Distributed File System (HDFS). Hadoop MapReduce is a distributed computing platform that includes JobNode, which is responsible for allocating the developer's tasks to the TaskNode and is thus called the Master, and TaskNode, which is responsible for executing the tasks designated by the JobNode, including the Map and Reduce functions written by the developer. HDFS mainly comprises NameNode and DataNode. The DataNodes are responsible for storing the results completed by the Map and Reduce functions while the NameNode is responsible for establishing the index stored in the DataNodes and recording the positions of the data needed for the Map and Reduce functions to execute tasks.

* Corresponding author at: Institute of Computer Science & Information Engineering, National Ilan University, Taiwan, ROC. Tel.: +886 39357400x251.
E-mail address: hcc@niu.edu.tw (H.-C. Chao).

0140-3664/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comcom.2012.06.015

At present, most applications built on Hadoop are low-complexity and high-density computing, like WordCount, Sort and so on [9–11]. While running such applications, the Map function can get the processed result within a very small amount of time and the Reduce function mainly spends time waiting for the data processed by the Map function and integrating the data. However, using MapReduce to implement high-complexity and high-density 2D to 3D image processing process [12,13] may cost lots of time for the Map function to analyze images and to convert the images to intermediate data for the Reduce function to generate 3D images. This paper implements MapReduce on an integrated 2D to 3D multi-user system, in which the Map function is responsible for processing 2D effects, like gray scale [14], sobel [15], Gaussian blur and corner detection [16], and converting the images to intermediate data, while the Reduce function is responsible for integrating the intermediate data generated by the Map function, using 3D model construction algorithm [17] to construct the model of the object, and presenting 3D images according to user requirements.

Traditionally, the Map function converts the images to intermediate data so that the Reduce function can combine the images together for the final output. However, if the computing speed of the Map function is too slow, the data will not be completed and the Reduce function has to wait until the data is completed for further combination. In such a manner, when the computing speed of the Map function cannot cope with the Reduce function, the tasks of the Reduce function will be greatly delayed, which moreover increases the operation time.

In this paper, because large amounts of users simultaneously uses MapReduce for 2D to 3D image processing, we present a solution to the above-mentioned problem to enhance the performance of MapReduce in executing 2D to 3D image processing: Dynamic Switch of Reduce Function (DSRF) algorithm, a scheduling scheme on the Reduce function for users who compete simultaneously to acquire Reduce resources to finish the tasks efficiently. With a priority mechanism, DSRF algorithm allows the Reduce function to switch to different tasks according to the achieved percentage of the tasks. In other words, when a combination task is not finished and the Reduce function is waiting for the Map function to generate intermediate data, the Reduce function can switch to another task to combine the image data first. For example, when the data needed for the first task is not completed yet but the image data of the second combination task is ready, the Reduce function processes the second task first. After finishing the second combination task, supposing the image data of the first and the third tasks is both ready, the Reduce function processes the first task according to the priority values. With our proposed scheme, the time spent waiting for data can efficiently utilized by the Reduce function and the delayed tasks also can be decreased to enhance the performance of MapReduce in 2D to 3D image processing.

The rest of the paper is organized as follows. Section 2 introduces the background and Section 3 presents our proposed system architecture and DSRF algorithm. Section 4 includes the implementation and performance analysis awhile Section concludes this paper.

2. Background

2.1. MapReduce

Proposed by Google, MapReduce is a distributed parallel computing architecture designed for processing large amounts of intensive data and its storage system is Google File System (GFS). For users to process large amounts of data while searching for data by the Google search engine, Google MapReduce provides a simple programming interface for the developer to develop search engine and processing searching procedures. MapReduce is composed of three major components: Master, Map function and Reduce function. The Master is responsible for managing the back-end Map and Reduce functions and offering data and procedures to them. The Map function converts the data to intermediate data so that the Reduce function can combine the intermediate data together to obtain the final output.

Whenever the data is received and processed by the Map and Reduce functions, there will be a record that includes a key and a value. The key generated by the input processed by the Map function refers to the correlation between the data while the value on the other hand means the processed message. When a task is divided into several small tasks and handled by the Map function, their correlations are not marked especially. But, one or several records are generated, including the key and the value, after the Map function finishes processing the designated data. According to the key produced by the Map function, the Reduce function collects the data with the same key together and generates a value to form a new set of key/value.

As shown in Fig. 1, the input file is cut into M chunks before the MapReduce operation and the file size of each chunk ranges between 16 MB to 64 MB. The file chunks are copied to the file system and one of the chunks is especially designated to the Master to find the suitable idle server as the computing server. Next, after the Master determines the servers responsible for the Map and Reduce functions, M Map functions and R Reduce functions are designated to the servers. According to the key/value of the input data, servers responsible for the Map function deliver the processed data to the developer's Map function and store the Map result in the file system. The intermediate data output by the Map function is cut into R pieces and their locations are returned to the Master. Once receiving the locations of the data processed by the Map function from the Master, servers responsible for the Reduce function read the data remotely, sort the keys and group the values with the same key. The Reduce function calculates the number of the single key and forwards the data to the developer's Reduce function to generate the final output. After the data is completely processed, the Master notifies users to access the result.

Because of the high-speed network and a storage system that can store a large volume of data, MapReduce allows most search engines to operate. The hardware of MapReduce belongs to an open-source operating system and is controlled and maintained by managers to reduce not only the cost of purchasing and maintaining equipments but also the cost of developing MapReduce. Different from centralized RAID-based SAN and NAS storage

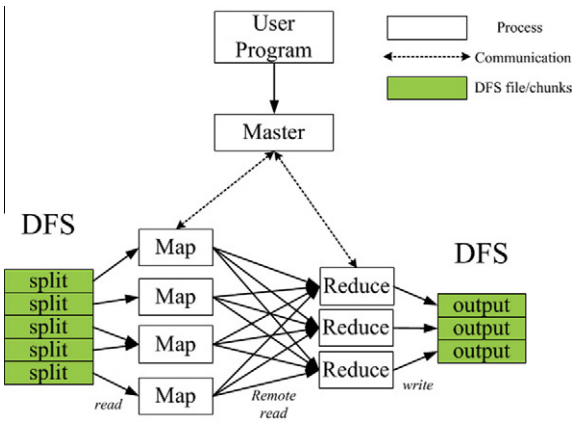

Fig. 1. MapReduce framework.