농업 빅데이터 처리를 위한 하둡과 스파크 성능 비교

반퀴엣뉘엔, 신녹뉘엔, 둑티엡부, 김경백
전자컴퓨터공학부
전남대학교

The Performance Comparison of Hadoop and Spark in Agriculture Big Data Processing

Van-Quyet Nguyen, Sinh Ngoc Nguyen, Duc Tiep Vu, Kyungbaek Kim
Dept of Electronics and Computer Engineering,
Chonnam National University
e-mail : quyetict@utehy.edu.vn, sinhngoc.nguyen@gmail.com, ductiep91@gmail.com, kyungbaekkim@jnu.ac.kr

요 약

Nowadays, massive data are generated in every minute through multi-devices such as sensors and smartphones, and it leads to the challenges of big data. Meanwhile, most traditional data processing systems are designed for local computation and they do not scale well to big data problems which have large requirements of computational resources and storage. In order to deal with big data, many researchers have been designed the big data platforms using Hadoop and Spark framework, and simple comparison has been done in past. However, no comprehensive study related to the performance of these two frameworks in the aspect of the volume of data and the complexity of computation. In this paper, we highlight the performance comparison between Hadoop and Spark in two cases: one is to conduct arithmetic calculation on massive agriculture data with various size of data and another is to perform K-mean clustering algorithm on the soil data with different number of iterations. We also present how to tune environment parameters such as memory limitation and data chunk size in Hadoop and Spark to achieve better performance. Through the evaluation with practical agricultural data, we experimentally verify that Spark is faster than Hadoop in several times.

1. Introduction

In recent years, data has become ubiquitous. The analysis of these data provides many benefits in many aspects of our daily life. For example, in agriculture sector, farmers have to measure and understand the impact of a huge amount and variety of data which drive overall quality and yield of their fields such as weather data and soil data; however, these data are being generated in every second that leads to the challenges of big data. Therefore, developing a big data platform for handling massive data in real-time is particularly evident and plays an extremely important role.

Apache Hadoop [1] has been the most popular framework for big data processing. It provides a parallel computation model MapReduce [2] and Hadoop distributed file system (HDFS) [3] module. Recently, Apache Spark [4] has emerged as a leading distributed computing framework for real-time analytics with its memory-oriented architecture and flexible processing libraries. These two frameworks are being widely used in many big data applications. There have been a number of studies for the performance comparison of Hadoop and Spark [5][6]. However, no comprehensive study has been done to analyze the impact factors to the performance of these two frameworks. In this paper, we highlight the performance comparison between Hadoop and Spark in two cases: (1) processing massive agriculture data with the growth of data size and (2) clustering soil data using K-means algorithm which has many iterations in computing.

Our work makes the following contributions:

- Firstly, we implemented two algorithms in both of two frameworks Hadoop and Spark for performance comparison. The first algorithm is to compute the farms' field area. The second algorithm is to cluster soil data based on its chemical characteristics.
- Secondly, we experimented and highlighted the performance comparison between Hadoop and Spark.

Therein, Spark is faster than Hadoop in several times.

- Finally, we presented how to tune environment parameters in Hadoop and Spark to achieve higher performance.

2. Background

In this section, we describe an overview of Hadoop and Spark that frameworks we are going to compare the performance in this paper.

2.1 Hadoop and MapReduce

Apache Hadoop is a framework that allows for the distributed processing of large data sets across clusters of computerss [1]. The current Hadoop version consists of four main components: (1) HDFS (Hadoop Distributed File System) that provides high-throughput access to application data, (2) YARN (Yet Another Resource Negotiator) that is a framework for job scheduling and cluster resource management, (3) MapReduce is a YARN-based system for parallel processing of huge datasets, and (4) Hadoop Utilities that provides common utilities to support the other Hadoop modules.

MapReduce is a programming model that supports to run programs in parallel on large distributed system. This model uses a map function that processes a key/value to generate a set of intermediate key/value pair and a reduce function that gathers all values with the same intermediate key to process and returns the results.

2.2. Overview of Spark

Apache Spark is a new cluster computing framework that is designed for fast computation. It uses the concept of RDD (Resilient Distributed Datasets) that lets users store data in memory across queries. This lets RDDs be read and written up faster than typical distributed file systems (e.g., HDFS). There are five main components in Apache Spark: (1) Spark Core, (2) Spark SQL, (3) Spark Streaming, (4) MLlib (Machine Learning library), and Graphx. In this paper, we evaluate the performance of Spark using MLlib that is a distributed machine learning framework.

3. The Performance Comparison

3.1 Case studies of evaluation

We present a couple of case studies for performance evaluation of Hadoop and Spark. In the first case, the performance is evaluated by an algorithm computing the total field area of each farm. The second case study will evaluate the performance of running a parallel K-means algorithm for clustering the soil data based on soil chemical characteristics.

Field Area Computation.

At the current time, we have a 2GB of agriculture data with more than 11 millions of records that contain the information about the farm and its related information such as field products and the area over each field product. There are many farms in our dataset and each farm has a lot of field products. In this case, we implement a simple algorithm on both Hadoop and Spark to compute the total field area of each farm. The purpose of this case study is to evaluate the performance of Hadoop and Spark in processing huge amount of data without iterative computation.

Soil data clustering with Kmeans algorithm

The purpose of this case is to evaluate the performance of Hadoop and Spark in processing a huge amount of data with an iterative algorithm, and because of that, we chose K-means algorithm. K-means algorithm is the most well-known used clustering method. It groups objects (data points) based on features into k number of groups. K-means algorithm performs the following steps:

Step 1: Select k data points from dataset to be used as cluster centroids (random)

Step 2: Assign data points to clusters according to their distance to the cluster centroids.

Step 3: For each cluster, recompute the cluster centroid using the newly computed cluster members.

Step 4: Go to step 2 until the process converges.

In k-means algorithm, the computation cost is made in the calculation of distances step (Step 2) in which each iteration require a total of (n*k) distance computations. Therefore, the performance of the distance calculation is the key for improving the time performance of the algorithm. To do this, we can process the distance calculation in parallel by using MapReduce or Spark because the execution orders of distance calculation of data points will not affect the final result of clustering.

For running the algorithm on Spark, we implement a Java program to set up the parameters and call functions of K-means algorithm in Spark MLlib library.

3.3. Experimental Evaluation

Experimental setting. Our experiments were run on the distributed system, in which, both Hadoop and Spark are deployed on five machines: one machine for master node, and four others for compute nodes. Each compute node has 4 CPUs and 8GB of RAM. All algorithms are implemented in Java.

Experimental Results.

For field area computation case, to evaluate the

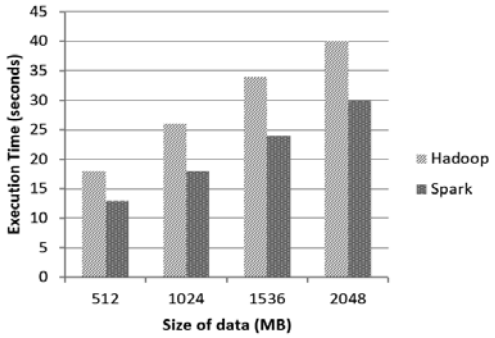

Figure 1. Performance comparison of Hadoop and Spark with varying of data size.

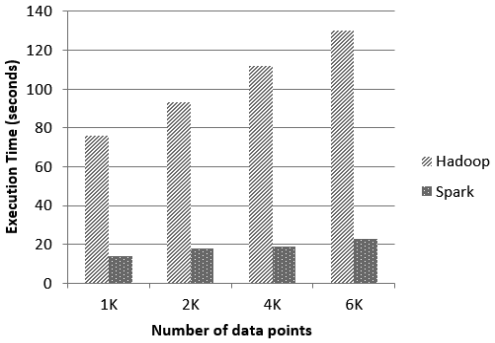

Figure 2. Hadoop and Spark with varying number of data points in K-means algorithm

performance of Hadoop and Spark on various size of dataset, we varied the data size from 0.5GB to 2.0GB. The result of this case is shown in Figure 1. We can see that the algorithm running on Spark outperforms Hadoop in all cases of data size. Thus, in this case, Spark is faster than Hadoop 1.4 times in an average of execution time.

For soil data clustering with K-means algorithm, we experimented with four cases of varying number data points, the number of data partitions, the number of iterations of K-means algorithm, and the number of clusters.

Figure 2 and Figure 3 show the results of running K-means with 4 clusters on Hadoop and Spark, in which the convergence is achieved. In the result, we observed that Spark is 5.5 and 5.7 faster than Hadoop for varying number of data points and varying number of data partitions, respectively. While Figure 4 shows the execution time of Hadoop and Spark with varying the number of iterations, that is, Spark proved effective than Hadoop when the number of iterations of the algorithm increases. We can see that after 2 iterations Spark is faster than Hadoop about 2 times, but it is approximate 6 times after 8 iterations.

The result in Figure 5 shows the performance comparison of Hadoop and Spark of running K-means

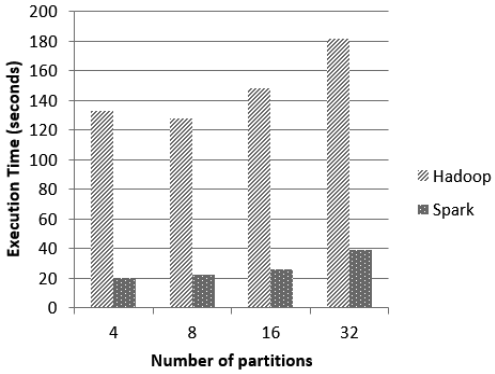

Figure 3. Hadoop and Spark with varying number of data partitions in K-means algorithm

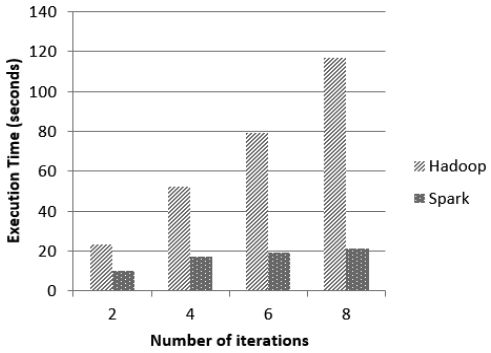

Figure 4. Hadoop and Spark with varying number of iterations in K-means algorithm

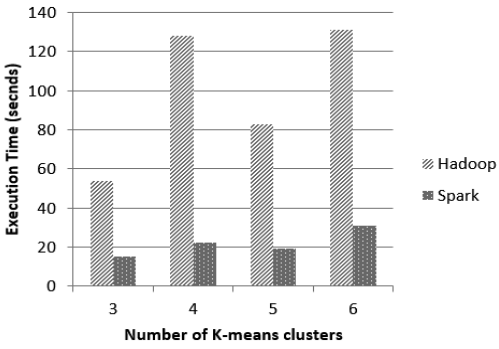

Figure 5. Hadoop and Spark with varying number of clusters in K-means algorithm

algorithm with a different number of clusters. That is, Spark also runs faster than Hadoop a fluctuation from 4 to 6 times.

4. Discussion

From the result in previous section, we can see that the performance of Hadoop and Spark affected by some parameters configuration such as the number of data partitions (see Figure 3). If the size of dataset is the same, a smaller number of data partitions fitting with the maximum the number of containers managed by YARN will obtain better performance. In the case of

running K-means algorithm, 8 data partitions given the best performance; meanwhiles, 32 data partitions given the lowest performance in our experiment. That because, in our system, we configured 2GB for each map and reduce task; therefore, 16 is the maximum number of containers that YARN can allocate for running an application. However, one of them is used to run AppMaster, so 15 map/reduce (in Hadoop) or executor (in Spark) tasks can be run in concurrent to achieve the best performance. If the dataset is split with the number of partitions more than that number, one of slave node/worker node will take more than one tasks, but if the number of partitions too small, it does not utilize the capability of the system.

There are several ways to tune Hadoop and Spark; however, for space constraints, in this paper, we only describe two ways in detail to achieve better performance.

Changing the number of data partitions. To split a huge dataset into small input file, we can change the block size of HDFS thought dfs.blocksize property in hdfs-default.xml file or change the value of property mapreduce.input.fileinputformat.split.minsize in mapred-default.xml file. By default, the value for this property is 128MB. The size of each input file should be set as shown the Equation 1.

$$split_{size} = \frac{S}{k \times (M-1)} \quad (1)$$

where S is the size of dataset in byte, k is a positive integer, and M is the maximum number of contains of the system can be allocated for a job that can be estimated by Equation 2.

$$M = \min(2 \times cores, RAM_{available} / C_{\min-size}) \quad (2)$$

where cores is the number of CPUs cores, $RAM_{available}$ is the total amount of memory (in MB) of the system, $C_{\min-size}$ is the minimum amount of memory (in MB) that YARN allocate for a container. Therefore, in Equation 1, we should choose the minimum value of k such that each map/reduce task can handle a $split_{size}$ depending on the problem. This will maximize the number of map/reduce or executor tasks with a balanced workload.

Changing the number of containers. From Equation 2, we can see that the number of containers that YARN can allocate for a job depending on the minimum amount of memory that YARN allocate for a task. Moreover, it also depending on the amount of memory that we set up for each map/reduce or executor task. To achieve a better performance we can change these properties depending on estimating the require amount of memory per task of the problem. To set up the minimum size of memory for a container we can change the value of yarn.scheduler.minimum-allocation-mb property in yarn-site.xml. And, for modifying the amount of memory per task we can change the value of two properties mapreduce.map.memory.mb and mapreduce.reduce.memory.mb.

5. Conclusion

In this paper, we highlighted the performance comparison between Hadoop and Spark with two case studies: processing a large amount of data with various size of data and handling iterative computations with K-means algorithm. Using practical agriculture dataset, we experimented and shown that Spark is faster than Hadoop in few times.

In the future work, we plan to design and implement the big data platform using both Hadoop and Spark to provide the services and applications for analysis the data in agricultural production and disaster warning notification system.

Acknowledgements

This work was carried out with the support of "Cooperative Research Program for Agriculture Science and Technology Development (Project No. PJ01182302)" Rural Development Administration, Republic of Korea. This work was supported by the National Research Foundation of Korea Grant funded by the korean Government(NRF-2014R1A1A1007734). This research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2016-R2718-16-0011) supervised by the IITP(Institute for Information & communications Technology Promotion).

References

[1] Apache Hadoop, http://hadoop.apache.org (2009).

[2] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.

[3] Borthakur, Dhruba. "HDFS architecture guide." HADOOP APACHE PROJECT http://hadoop. apache. org/common/docs/current/hdfs design. pdf (2008): 39.

[4] Zaharia, Matei, et al. "Spark: Cluster Computing with Working Sets." HotCloud 10 (2010): 10-10.

[5] Gopalani, Satish, and Rohan Arora. "Comparing apache spark and map reduce with performance analysis using K-means." International Journal of Computer Applications 113.1 (2015).

[6] Pan, Shengti. "The Performance Comparison of Hadoop and Spark." (2016).