Performance Analysis using Large-scale Parallel Collaborative Filtering on Spark and Hadoop

Kundjanasith Thonglek
High Performance Computing and Network Center
Deparment of Computer Engineering, Kasetsart University
Bangkok, Thailand
kundjanasith.t@ku.th

Kohei Ichikawa
Software Design and Analysis Laboratory
Graduate School of Information Science, NAIST
Nara, Japan
ichikawa@is.naist.jp

***Abstract*—Spark is a cluster computing technology for large-scale data processing that provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. Following the SparkContext can connect to several types of cluster manager which allocate resource across applications. Once connected, Spark acquires executors on nodes in the cluster, which are processes that run computations and store data for the application. Next, it sends application code to the executors. SparkContext represents the connection to a Spark cluster and can be used to create resilient distributed datasets (RDDs). it also accumulators and broadcast variables on the cluster. This research approaches the problem by comparing Spark cluster and Hadoop cluster performance are used the parallel collaborative filtering program?s execution time as criteria to find when the application should execute on Spark or Hadoop. This research is primarily intended for cluster computing benchmark.**

***Index Terms*—Large scale data processing; Resilient Distributed Datasets; Collaborative Filtering; Spark;**

I. Introduction

Large scale data processing is the process of applying data analysis techniques to a large amount of data. Typically, large scale data analysis is performed through two popular techniques: parallel database management system (DBMS) or MapReduce powered systems. The parallel database management system requires that the data be in a DBMS supported schema, whereas the MapReduce option supports data in any form. Moreover, the data extracted or analyzed in large-scale data analysis can be displayed in various different forms, such as tables, graphs, figures and statistical analysis, depending on the analysis system. One of the popular engines for large scale data processing based on the MapReduce model is Spark.

In Spark cluster, there are two types of nodes: driver node and worker node. The driver node is one of the critical components of the cluster which maintains state information of all worker nodes attached to the cluster and also responsible for maintaining the SparkContext. The worker node runs the Spark executors and other critical services required for the proper functioning of the cluster when distribute workload with Spark, all the distributed processing happens on these workers.

In Hadoop cluster, there are three types of node: client node ,master node and slave node. The client node is neither master or slave, rather play a role of loading the data into the cluster, submit MapReduce jobs describing how to the data should be processed and then retrieve the data to see the response after job completion. The master node consists of three components name node that oversees the health of data node and coordinates access to the data stored in a data node, a secondary node that is to contact name node in a periodic manner after certain time interval and job tracker that coordinates the parallel processing of data using MapReduce. The slave node consists of two components data node and task tracker. The data node that stores the data and process the computation. The task tracker that communicates to their masters.

II. Literature review

Parallel Collaborative Filtering technique on Spark cluster and Hadoop cluster

A. Large-scale Parallel Collaborative Filtering for Netflix

This paper describe about why they use parallel collaborative filtering to creating recommendation system for Netflix dataset by using Matlab library and Linux cluster.

This paper doesn?t describe about when we execute parallel collaborative filtering using Spark MLlib on Spark cluster and Mahout on Hadoop cluster

B. Comparing Apache Spark and MapReduce with Performance Analysis using K-means

This paper describe about the performance comparison between Spark and MapReduce using K-means clustering algorithm on 1 node and 2 nodes.

This paper doesn?t describe about when we execute alternating least square algorithm on Spark and Hadoop cluster so we will conduct the experiment to execute the algorithm on both cluster

III. Design&Implementation

IV. Experimental result

The experimental result is divided into 2 sections :

First section is the experimental result of executing parallel collaborative filtering using alternating least square algorithm with Spark ML library on Spark cluster and Mahout library on Hadoop cluster by changing the number of CPU cores per

TABLE I
RESULT OF TESTING ON 5 WORKERS NODE
BY EACH NODE HAS CPU 2 CORES AND MEMORY 8 GB

Size of dataset (MB)	Execution time on Hadoop (seconds)	Execution time on Spark (seconds)
139.90	463.88	1018.69
279.80	920.81	1061.27
419.71	1437.65	1099.85
559.61	1900.33	1154.24
699.51	2337.72	1208.77
839.42	2753.91	1338.82

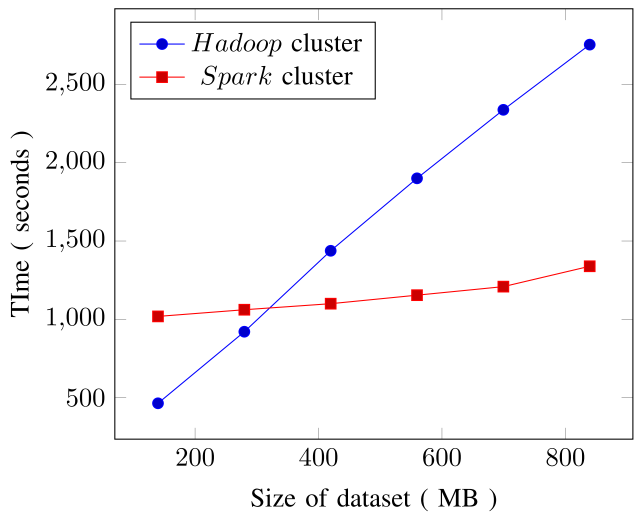

Fig. 1. Graph of testing on 5 workers node by each node has CPU 2 cores and memory 8 GB

each worker node in the cluster which has 1 master node and 5 worker nodes for observe the impact of parallel computing

Second section is the experimental result of executing parallel collaborative filtering using alternating least square algorithm with Spark ML library on Spark cluster and Mahout library on Hadoop cluster by changing the number of worker nodes in the cluster has the total CPU 20 cores and memory 40 GB for observe the impact of parallel I/O

TABLE II
RESULT OF TESTING ON 5 WORKERS NODE
BY EACH NODE HAS CPU 4 CORES AND MEMORY 8 GB

Size of dataset (MB)	Execution time on Hadoop (seconds)	Execution time on Spark (seconds)
139.90	254.23	813.04
279.80	593.71	907.73
419.71	906.54	959.61
559.61	1253.17	1042.10
699.51	1468.55	1112.97
839.42	1831.36	1196.68

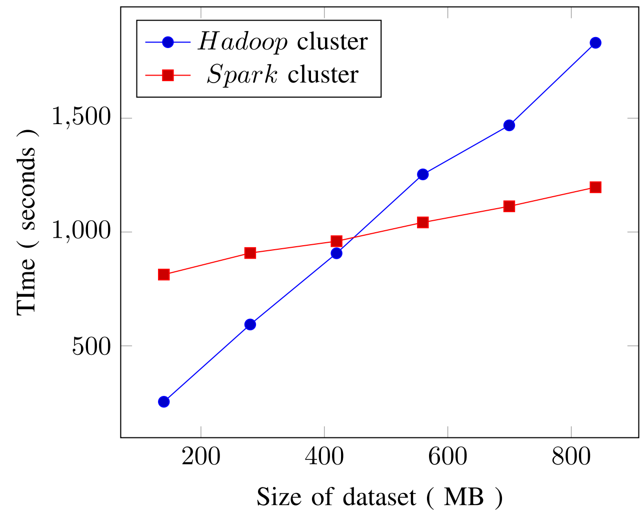

Fig. 2. Graph of testing on 5 workers node by each node has CPU 4 cores and memory 8 GB

TABLE III
RESULT OF TESTING ON 5 WORKERS NODE
BY EACH NODE HAS CPU 8 CORES AND MEMORY 8 GB

Size of dataset (MB)	Execution time on Hadoop (seconds)	Execution time on Spark (seconds)
139.90	174.88	613.34
279.80	351.64	663.38
419.71	550.53	741.04
559.61	727.93	833.40
699.51	921.13	903.04
839.42	1092.84	993.54

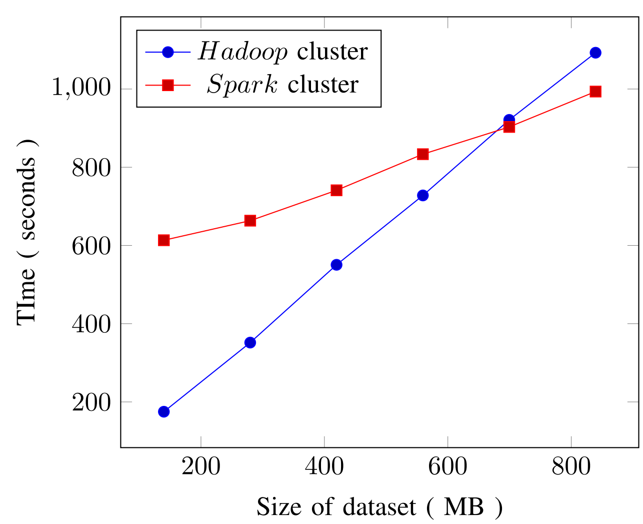

Fig. 3. Graph of testing on 5 workers node by each node has CPU 8 cores and memory 8 GB

TABLE IV
RESULT OF TESTING ON 5 WORKERS NODE
BY EACH NODE HAS CPU 16 CORES AND MEMORY 8 GB

Size of dataset (MB)	Execution time on Hadoop (seconds)	Execution time on Spark (seconds)
139.90	160.47	409.36
279.80	323.76	475.68
419.71	438.17	543.79
559.61	549.12	612.68
699.51	678.46	708.02
839.42	818.09	798.31

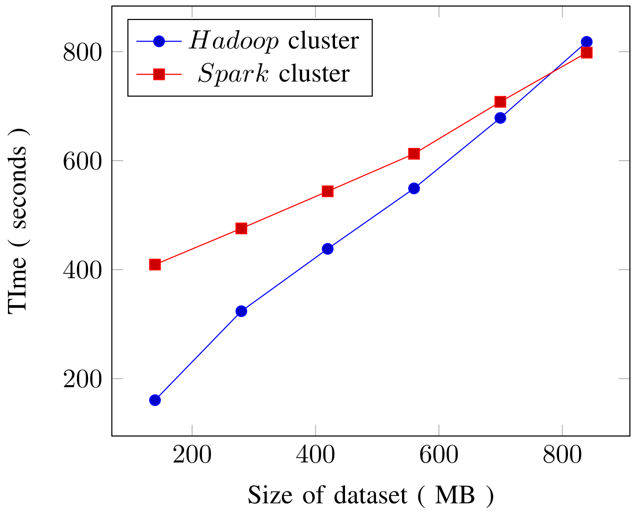

Fig. 4. Graph of testing on 5 workers node by each node has CPU 16 cores and memory 8 GB

V. DISCUSSION

VI. CONCLUSION

APPENDIX A
PROOF OF THE FIRST ZONKLAR EQUATION

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.