Parallel/Distributed Query Processing

Graph Matching in Parallel

Minyang Tie
Computer and Data Science
Case Western Reserve University
Cleveland OH, USA
mxt497@case.edu

Wei Du
Computer and Data Science
Case Western Reserve University
Cleveland OH,USA
wxd163@case.edu

ABSTRACT

Before we knew the MapReduce algorithm, the sequential algorithm was widely used in data processing, such as in Query Processing, which is a function that converts high-level queries into low-level expressions. The sequential algorithm is very simple for us to use and has excellent performance in processing simple data, but it takes too long to process complex datasets or huge datasets. Here, we address a MapReduce algorithm, which is based on the sequential algorithm and can process complex datasets more easily. This article aims to separately use a MapReduce algorithm and a sequential algorithm in Graph Query Processing, and then compare the performance of their results.

KEYWORDS

Sequential algorithm, MapReduce algorithm, Query Processing, Parallel/Distributed Query Processing, RDF, Hadoop, Benchmarks

KEY CONCEPTS

Sequential algorithm: is an algorithm that is executed sequentially-from start to end, no other processing is performed.

MapReduce algorithm: is a programming model and related implementations for processing and generating large data sets using parallel distributed algorithms on the cluster.

Parallel/Distributed Query Processing: transforms advanced queries into execution plans that can be efficiently executed in parallel, on multiprocessor computers.

RDF: a short for Resource Description Framework.

1 Methods

1.1 Sequential algorithm

The Sequential algorithm is a linear algorithm that iterates through the dataset to see whether the tuples in the data fulfills the query. If there is a match find,it will add to the list. After the whole dataset has been iterated, the result list will output as the query result.

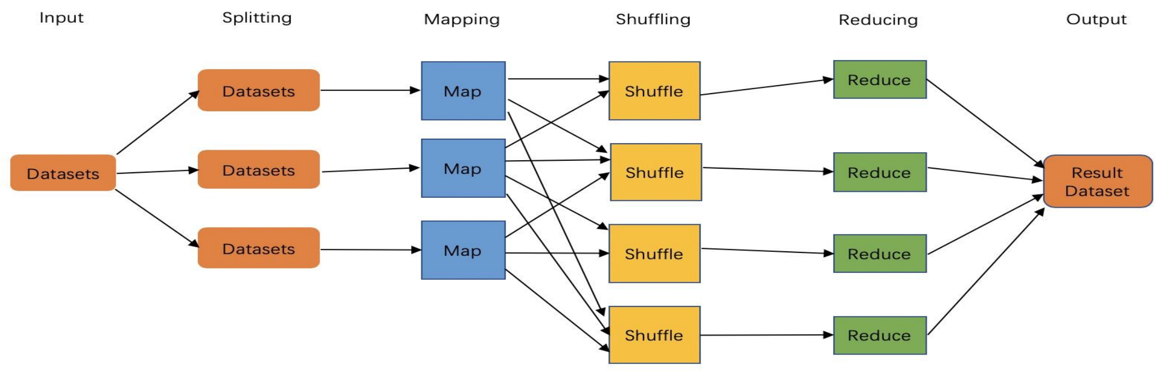

Figure 1: MapReduce Flowchart

1.2 MapReduce algorithm

Nowadays, all kinds of data that need to be analyzed are getting larger and more complex, and often from multiple data sources, which poses challenges to traditional algorithms, and new algorithms come into being. The MapReduce algorithm-new algorithm- is mainly to deal with the larger datasets and complete data analysis more efficiently, which makes it one of the most popular frameworks.

The Mapreduce algorithm has some advantages: developers don't need to write any code for the tedious tasks of distributed programming; it can be used in both companies and academia; availability and resilient nature, etc..

Furthermore, the MapReduce algorithm contains two important parts: Map and Reduce. Map takes a set of data and converts it to another set of data, where each element is decomposed into tuples (key / value pairs).

Figure 1 demonstrates how Mapreduce works.

2 Tools

2.1 Hadoop

Hadoop is a distributed system infrastructure developed by the Apache Foundation and mainly solves the problem of storage of massive data and analysis of massive data. Broadly speaking, HADOOP usually refers to a broader concept-HADOOP ecosystem.

Hadoop has a long history: Lucene--Open source software created by Doug Cutting, which writes code in java and implements a full-text search function similar to Google. It provides a full-text search engine architecture, including a complete query engine and indexing engine. At the end of 2001, it became the Apache Foundation's sub-project. For a large number of scenes, Lucene faced the same difficulties as Google. By learning and imitating Google's solutions to these problems, the miniature version of Nutch was developed. Moreover, it can be said that Google is the source of ideas for Hadoop (Google 's third in big data Papers): GFS —> HDFS; Map-Reduce —> MR; BigTable —> Hbase. From 2003 to 2004, Google disclosed the details of some GFS and Mapreduce ideas. Based on them, Doug Cutting, etc. took 2 years of spare time to implement the DFS and Mapreduce mechanisms, which made Nutch's performance soar. In 2005, Hadoop was officially introduced to the Apache Foundation as part of Lucene's subproject- Nutch. In March 2006, Map-Reduce and Nutch Distributed File System (NDFS) were included in a project called Hadoop. The name comes from Doug Cutting 's son 's toy elephant. Hadoop was born and developed rapidly, marking the advent of this era of cloud computing.

In addition, Hadoop has four main advantages: High reliability-Hadoop assumes that computing elements and storage will fail, because it maintains multiple copies of working data, and can redistribute failed nodes in the event of a failure; High scalability- Distribute task data between clusters, which can easily expand thousands of nodes; Efficiency- Under the MapReduce idea, Hadoop works in parallel to speed up the task processing speed; High fault tolerance- It can automatically save multiple copies of data and reassign failed tasks

2.2 Benchmarks

Benchmark is the act of running a computer program, a set of programs, or other operations to assess the relative performance of an object, usually by running multiple standard tests.

The Berlin SPARQL Benchmark (BSBM) is chosen to evaluate the performance of the two algorithms. This benchmark is implemented to test the performance of the RDF storage system. This benchmark is built around an e-commerce environment in which the data is majorly about the user's review on different items. This benchmark itself contains the tools which help to generate the data in different sizes and formats.

3 Experimental Studies

With the development of the times, big data processing has become mainstream. Sequential algorithm has many advantages, but as the data continues to increase, its processing time also increases linearly. This experiment aims to introduce the MapReduce algorithm for big data processing, which has a dramatic growth trend as the data in the modern world grows rapidly. So in this experiment, by comparing the runtime of the two algorithms, we will test the performance of the Mapreduce algorithm and a sequential algorithm to see how well that mapreduce can do in different scale of dataset compared to the sequential algorithm.

3.1 Datasets

To evaluate the performance of map reduce algorithm and the sequential algorithm, the scale of the dataset should vary in different magnitudes. Both two algorithms should be tested in the different datasets.

There are different types of RDF data that the tools can generate. What we choose is to use the N-triples data to represent the RDF relationship because the N-triples data are separate in different lines with the same format which consist of the data in Map Reduce where data are read line by line. We use the data generator to generate the N-triples data with the scale of 370 MB, 1.8GB, 3.68GB, 7.35GB and 29.46GB.

3.2 Experimental Setup

To guarantee the consistency of the result, both algorithms are tested in the same environment. Thus, in the same computer, we set a java environment, and a Hadoop single node cluster with pseudo-distributed mode to test the two methods in different environments.

Methods	Dataset1(370M)	Dataset2(1.8G)	Dataset3(3.68G)	Dataset4(7.35G)	Dataset5(29.46G)
Sequential	3342ms	15174ms	29434ms	58930ms	236223ms
MapReduce	9872ms	96785ms	180835ms	358768ms	1345722ms

Table.1 Result of runtime in five datasets

3.3 Results

After running the two algorithms with the 5 scale of dataset respectively, the running time and the dataset size are shown in table 1. When the dataset is 370M, which is the smallest one, the runtime of Sequential algorithm is 3342ms, which is much less than 9872ms of MapReduce algorithm. Even though the dataset is equal to 1.8G, the difference between Sequential algorithm and MapReduce algorithm is 81638ms (=96785-15174), which shows the performance of Sequential algorithm is much better than MapReduce algorithm.

As we discussed in the previous parts, the MapReduce algorithm was developed to solve big datasets, so we thought maybe 1.8G is not a big enough dataset. However, we tried to test different datasets from 370M to 29.46G, and then we found our datasets are not big enough to test these two algorithms. Because until we used a dataset of 29.46G, the runtime of MapReduce is still more than the runtime of a Sequential algorithm. On the one hand, this is the biggest dataset our computers can test; On the other hand, we still believe if the dataset is big enough, the runtime of MapReduce algorithm will be less than the runtime of Sequential algorithm. So we decided to compare the slope between each two different datasets to see what results are.

Figure 2 exhibits the comparison of running time between map reduce and sequential algorithm. And the running time of the map reduce is slower than the sequential algorithm as the figure.2 shown.

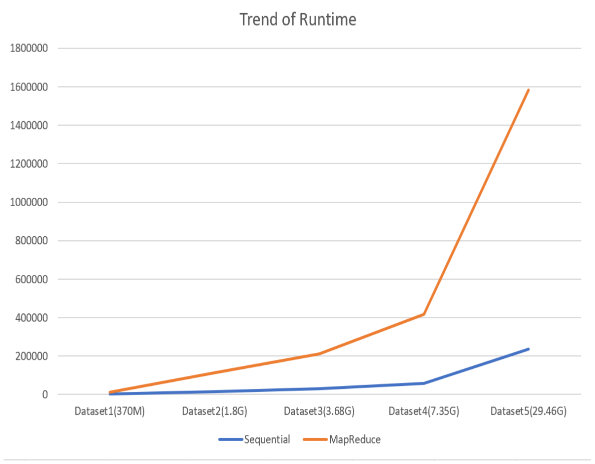

Figure.2. The trend of the runtime

3.4 Comparison

Although the running time of map reduce algorithm are longer than the sequential algorithm, However, after we compute the slope between each running as Figure 3 and Table 2 shown, we found that when handling the small amount of data, the slope of sequential algorithm are lower than the map reduce algorithm, which means the sequential algorithm perform better, when data size trends to small. When we turn the data into a large mount, from 7.35GB to 29.46GB, the slope of the map reduce is lower than the sequential algorithm.

The most obvious result is when we used the dataset of 29.46G, the slope of the Sequential algorithm is around 4.0, but the slope of MapReduce is around 3.75, which clearly

Methods	Dataset2(1.8G)	Dataset3(3.68G)	Dataset4(7.35G)	Dataset5(29.46G)
Sequential	4.540394973	1.939765388	2.002106408	4.008535551
MapReduce	9.803991086	1.868419693	1.983952222	3.750953262

Table.2 Slope of runtime between datasets

shows with the growth of datasets, the growth of runtime for MapReduce decreased. Which means when dataset trends to big, the map reduce works well. Because of the performance limit of our laptop, the upper bound dataset we generate is 29.46GB, but according to slope trend, we can speculate that mapreduce works better when performed on large dataset.

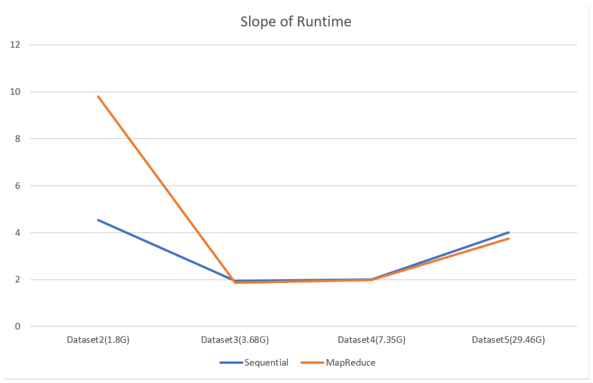

Figure.3 Slope of the Runtime

4 Conclusion

After all the results have been compared, we derive such trends that the map reduces trends to performance well in large datasets. The potential reason that the running time of a map reduces algorithms longer in the sequential algorithm might be in reality, the map reduces algorithm work in multiple clusters that have more resources. And the data is also stored separately in different machines. So when the dataset is dramatically large, different datanode can handle the data independently, thus, the performance of Map reduce is better than a sequential algorithm. In contrast, in pseudo distributed mode, the map reduce algorithm depends on the resources of one computer, and it also needs time to map the data, which might cause a lot of resource and time to handle such a process. Thus sequential algorithms can perform better than map reduce algorithms in such circumstances. So in the future, we can perform this algorithm with multiple clusters, and more larger data to test its performance fully. Compared to the sequential algorithm.

REFERENCES

[1] Sequential algorithm, https://en.wikipedia.org/wiki/Sequential_algorithm

[2] SQL Query Processing, https://www.tutorialride.com/dbms/sql-query-processing.htm

[3] MapReduce. https://en.wikipedia.org/wiki/MapReduce

[4] Parallel Query Processing, https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_1077

[4] Berlin SPARQL Benchmark http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/

[5] Christos Doulkeridis · Kjetil Nørv˚ag, *A Survey of Large-Scale Analytical Query Processing in MapReduce*

[6] Hyunsik Choi, Jihoon Son, *SPIDER : A System for Scalable, Parallel / Distributed Evaluation of large-scale RDF Data*