Journal of High Performance Computing, Vol. 1, Issue 1, 2010, pp-01-05

FAST DATA RETRIEVAL USING MAP REDUCE: A CASE STUDY

Ravin Ahuja[1], Anindya Lahiri[2], Nitesh Jain[3], Aditya Gabrani[4]

[1]*Corresponding Author PhD scholar with the Department of Computer Engineering, Delhi College of Engineering, Bawana Road, New Delhi-110042, India. ravin_ahuja@yahoo.co.in, +91 9810549168*
[2]*UnderGraduateStudent, Ambedkar Institute of Technology, Delhi-110031, India.*
[3]*UnderGraduateStudent, Ambedkar Institute of Technology Delhi-110031, India.*
[4]*UnderGraduateStudent, Delhi College of Engineering, Bawana Road, New Delhi-110042, India.*

Abstract

With the exponential growth in the volume of data, the need for its fast retrieval has increasingly become imminent. This requires heavy amount of searching that leads to computational complexities. The parallelized scatter and gather processing techniques over a distributed backbone has emerged as an effective solution for this. The scatter technique employs parallelizing the processes that concurrently operate on data in order to provide an efficient search. On the other hand, the gather technique collects data in parallel that leads to fast retrieval of data. In this paper, authors explore the use of MapReduce framework based on scatter and gather processing technique by exploiting it in the document searching techniques. This is based on assimilating occurrence count of a keyword and number of times it is referred to in the system. The authors suggest how MapReduce can be used in conjunction with the existing traditional methods and how effectively it improves the fast retrieval of data. The authors also discuss the advantages of the MapReduce in prioritizing various operations involving assorting and consorting of geographically distributed data.

1. Introduction

In the recent times we have witnessed heavy growth in size of data. Mining, storage and retrieval of such voluminous data have become a challenging task. Web is also emerging as a source of huge pool of information. Retrieval of accurate information from this pool requires heavy processing and efficient retrieval algorithms, which have to be scalable as well. The traditional techniques applicable to the single system fail to exhibit efficient and time bound retrieval over large scattered data in a distributed environment.

Distributed platforms have enabled faster processing and higher availability of information with the benefit of scalability in applications. Hence the focus is now on developing the best suitable techniques to work on distributed data. Breaking a given task into smaller sets and executing it, using a parallel algorithm helps efficient data management with ready availability and fast analysis. This process can be exploited by using the existing distributed programming standard like MapReduce [1].

With MapReduce, working on larger scattered datasets is easier as compared to traditional database programming languages such as SQL. MapReduce scores well as it does not ask a programmer to define database schema to manage, retrieve and operate on datasets [2]. Yet it remains efficient and utmost fault tolerant ensuring unaffected continual computing. The programming model provides an abstraction and transparency of the complexity involved in implementation of networks, distribution of processes and their execution over distributed environment in a convenient manner to the end user. Apache Hadoop project [3] is a prominent open source counterpart of MapReduce. Hadoop provides all basic minimal requirements for MapReduce framework. It uses Hadoop distributed file system inspired by Google file system. Hadoop is used by various universities and organizations for teaching and commercial benefits. Yahoo, Facebook, Amazon and IBM are few organizations to name that have large-scale Hadoop implementations.

The paper presents a case study for application of MapReduce in data retrieval techniques over geographically distributed data. Authors further analyze the role and advantages of MapReduce in prioritizing various steps of data aggregation over traditional existing methods with simplified and enhanced data retrieval. Rest of the paper follows. Section 2 deals with description of various search and data retrieval techniques. The MapReduce

parallel programming framework has been explained in Section 3. Section 4 illustrates the details of Weighted Index data retrieval using MapReduce. We finally conclude the paper in Section 5.

2. Various Search and Data Retrieval Techniques

Improvisation of information retrieval model started with Boolean search technique. The technique provided users higher control over query using Boolean operators to define terms to be included and excluded in search. Since this technique did not have a document ranking mechanism, it showed poor precision results to search queries. More refined models assigned a numeric score or rank to documents on the basis of relevance to a particular search keyword. These models include vector space model, probabilistic model and inference network model [4]**.** Vector space model considers text as vectors. This vector assumes a non-zero value in case it belongs to the document. Model checks for similarity between query vector and document vector by finding cosine of angle between two vectors [5]. Probabilistic model is based on probabilistic ranking principle that ranks document on the basis of decreasing probability of relevance for a search query key. Various algorithms are proposed for estimating this probability.

The use of web oriented, distributed computing platforms have become popular, and information retrieval confronts a major challenge. Under conventional techniques used for search and retrieval, documents were grouped together by categories as suggested by different algorithms. But due to huge data and poor performance of retrieval algorithms, the document clustering technique was slow and proved to be inefficient for precision searching [6]. Advance techniques based on the link structure of web have been developed to bring efficiency in prioritizing relevant data with increase in the size of information pool. Pagerank and Hyperlink Induced Text Search are algorithms using this concept.

Hyperlink Induced Text Search is an iterative link based algorithm that rates web document by its hub and authority value. Hub value is calculated using number of in links to a webpage and authority the value finds relevance of content in document. Google's PageRank is another algorithm based on the link structure of web-data. It assumes that more informative document on web would be highly linked and referred. Each page has same vote at the start and gives fraction of its votes to all out linked pages and receives votes of in linked pages. This process is iterated till total of vote is received by a page attaining a constant value. At the end, votes of page with high PageRank have higher weight.

A weighted index [7] document searching technique is influenced by the term frequency mechanism and the referenced nature of academic documents similar to the link nature of web-documents. When a search query is made for a keyword, a weight rank is calculated not only on the basis of frequency of occurrence of that keyword in the document but also on the basis of number of references made to the document by other documents.

Processes were earlier executed sequentially. With parallel programming, concurrent execution of process over different blocks of data came into picture. Google developed MapReduce framework for data intensive task on distributed systems. MapReduce is framework which allows inexpensive utilization of large distributed networks using non-local resources to compute parallel processes. The highly scalable, economical and simple programming model has encouraged various applications of MapReduce.

3. MapReduce Parallel Programming Framework

MapReduce is a programming model developed initially by Google to process large distributed data efficiently without compromising the scalability of applications that work over such data. Functional programming model of MapReduce allows data portability to distributed framework. Among various application of the framework, use in fields such as searching, sorting, indexing, machine learning, artificial intelligence and graph based computations is more prominent.

3.1 Scatter and Gather Technique

A processing logic that acts on data can be simplified as a transformation on a group of data. The MapReduce model works by scattering [8] the transformation by splitting the process into sub-tasks to act concurrently on chunks of data. The fragmentation of data is done by formatting it into a distributed file system [9]. Each set of data is applied a 'mapping' function to perform transformations through parallel processing. Later the results of independent transformations can be gathered [8] using a 'reduce' function to produce output.

3.2 Execution model

MapReduce execution starts with splitting data and storing it in a distributed file system. Input reader reads data from distributed file system and produces initial key value pairs [10]. Map is a user-defined function, which processes these pairs and produce intermediate pairs. In MapReduce execution there are independent map and reduce tasks that may be running simultaneously on different nodes.

Master node keeps track of these tasks and monitors other worker nodes. Worker nodes inform master about location of output by map phase. Reduce worker execute reduce function on intermediate value from each map to produce zero or user defined output corresponding to each key. Reduce worker iterate through intermediate key value pairs by remote procedure calls and merge values of same key. On completion of all map and reduce functions, master is informed which returns the result to user program.

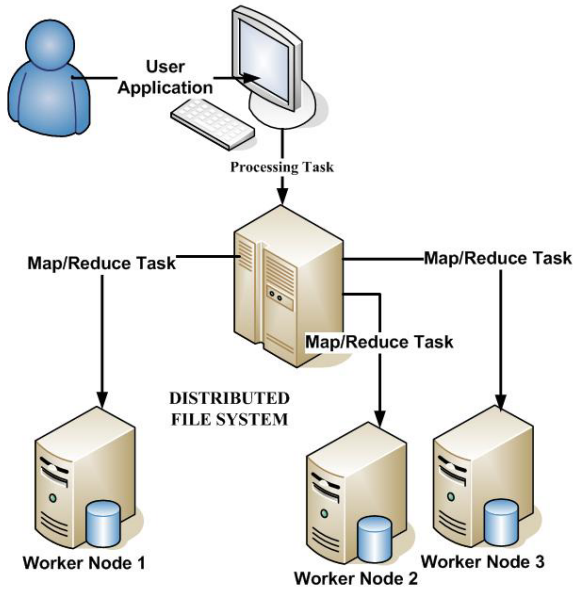

Figure 1: MapReduce model working over a cluster environment

3.3 Merits and Demerits of MapReduce

The strength of MapReduce lies in its ability to hide complexities of implementation and scalability from programmer. Programmers do not need knowledge of parallel programming; they only have to make use of MapReduce libraries. The MapReduce framework does not require redesigning of algorithms for application over large distributed systems; single system code can be used for parallel execution.

MapReduce built upon Google file system helps to provide a system for parallel processing of large scale data distributed over clusters. Cluster systems usually have centralized storage servers accessible by all other nodes in the cluster but the Google file system master server acts differently. Google file system operates in a manner where each cluster acts as chunk servers for storage and computation powerhouse, hence it eliminates the need for data transfer between a central data storage thus removing bottlenecks for execution of application. The worker node directly handles data broken into chunks for easy execution. The processing over different nodes is independent of each other hence it leads to high aggregate throughput by concurrent multiple mappings and reduces. The distributed file system relieves the master server of data transfers to only distribution of task among chunk servers thus networks issues do not become bottleneck. Fault tolerance is achieved by replicating data to increase data availability and through fast and automatic recovery of failed processes [9].

Initial model of MapReduce was inefficient in processing relational data operating on heterogeneous data sets. The problem was analyzed and solved by proposing extended merge phase in existing model in [11]. The problem of replication of data over large heterogeneous environment such as grid still remains an issue in the programming model.

Scattering of processes and gathering of their results may often overrun the computational cost of these processes. Transferring intermediate results for parallel processing may lead to performance degradation. Heavy parallel computation involving multiple map and reduce phases require synchronization, fault tolerance, scheduling and re-scheduling of failed tasks. Synchronization is a bottleneck as data is often modified during execution thus data validation and maintaining concurrency are important challenges.

4. Weighted Index data retrieval using MapReduce

For each queried keyword, documents are searched and results, sorted on the basis of relevance are returned. The relevance of document is judged by their weight index. The weight index of document is affected by both word frequency and number of referrals made to that document. With the large amount of geographically distributed data available, scanning each document by this technique and

calculating its weight one at a time becomes cumbersome and requires heavy processing, thus consumes time. Solution to heavy computing involved is parallelization of task. MapReduce provides parallel execution of sorting and collecting phase processing.

The weighted index data retrieval technique can be expressed as two processes. The first process involves calculation of weight index based on number of occurrences of keyword and the document referrals and the second processes is sorting of document list generated as output from the previous process in descending order of weight for providing most relevant query results to the user. The two processes can be implemented in parallel over large data by further breaking each process into sub tasks to be processed by multiple processors. The map and the reduce functions allow configuring the sub tasks to act on each document concurrently.

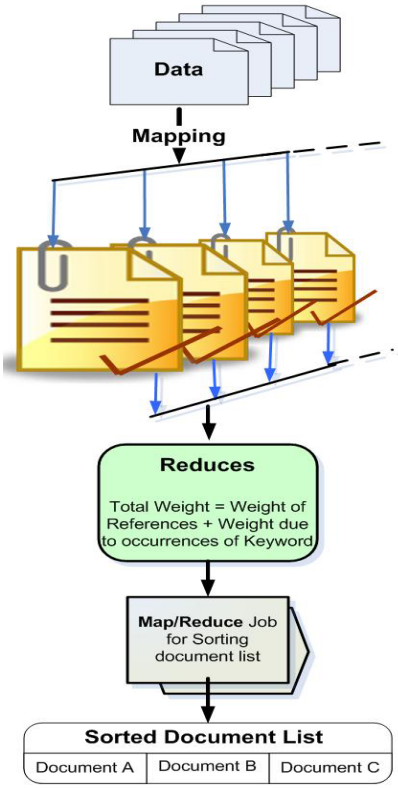

Figure 2: MapReduce model for weighted index search

As the algorithm involves not only counting the occurrences of a search keyword in particular document but also scanning all other documents for calculating its weight due to references, searching relevant documents would require faster processing to return query result in specific time bound. Calculating weight of each data set and finally merging into a single ranked structure becomes difficult when the data is distributed and also consumes time. The difficulty can be easily resolved by the operation style of MapReduce.

5. Conclusion

MapReduce provides the benefit of scattering computing processes into sub-tasks to be executed as separate processes concurrently over data. The authors discuss a weighted index document-searching algorithm and highlight how MapReduce is used over scattered data. The operations involved in scanning of data, merging intermediate output and finally sorting the retrieved documents, prioritized and indexed according to their respective weights invite a lot of complexities. With accomplished study of MapReduce framework we now target implementation of the weighted index data retrieval algorithm using Hadoop in a distributed environment.

6. Acknowledgement

The authors would also like to express their gratitude to Prof. Asok De, Principal AIT and Professor Delhi College of Engineering and Dr. G. Gabrani, Director TSL and former HOD, computer engineering department, Delhi College of Engineering for their constant support and encouragement of our effort that have made possible for us to complete the work described in this paper.

7. References

[1]. Jeffery Dean and Sanjay Ghemawat, “Mapreduce: Simplified data processing on large clusters,” in Proceedings of OSDI’04: Sixth Symposium on Operating System Design and Implementation, December 2004.

[2]. Joe Hellerstein, "Programming a Parallel Future", white paper, UC Berkeley Computer Science.

[3]. Hadoop. http://lucene.apache.org/hadoop/

[4]. A. Singhal, "Modern information retrieval: a brief overview", IEEE Data Engineering Bulletin, Special Issue on Text and Databases, 24(4), Dec. 2001.

[5]. Gerard Salton, A.Wong, and C. S. Yang. A vector space model for information retrieval. Communications of the ACM, 18(11): 613–620, November 1975.

[6]. C. J. Van Rijsbergen and W. B. Croft. Document Clustering: An evaluation of some experiments with the Cranfield 1400 collection. Information Processing & Management, 11:171-182, 1975.
[7]. Hye-Jin Jeong and Yon-Sung Kim, "Index weight decision technique for searching reliable documents", Second International Conference on Future Generation Communication and Networking, 2008.
[8]. http://www.openspaces.org/display/DSG
[9]. S. Ghemawat, H. Gobioff, and S. T. Leung, "The Google file system," in Proceedings of 19th ACM Symposium on Operating Systems Principles, October 2003.
[10]. Ralf Lämmel, "Google's MapReduce Programming Model Revisited. http://www.cs.vu.nl/~ralf/MapReduce/paper.pdf
[11]. H.-C. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker, Jr.—Map-Reduce-Merge: Simplified relational data processing on large clusters. In SIGMOD, pages 1029–1040, 2007.