Volume 3, Issue 11, November 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Survey on Text Based Indexing Techniques in Hadoop

Priyabrat Pattnaik
MTech, Final Year
Department of Computer Science & Engineering, Nit Calicut, India

Abstract— The explosion of unstructured data by social media and other web-based applications represents the biggest opportunity and challenge facing the database community today so modern analytics platform must support operations on unstructured text. Although there are number of indexing techniques currently available but none of them provide optimized text based selection operation so there is a need of an effective index framework on HDFS and MapReduce systems without altering the underlying Hadoop framework which provides better performance for text based analytics.

Keywords— Hadoop, MapReduce, Lucene, full-text indexing, Distributed File Systems.

I. INTRODUCTION

Cloud computing is an emerging area within the field of information technology (IT). In a pure cloud computing model, this means having all the software and data hosted on a server or a pool of servers, and accessing them through the internet without the need for very much (if any) local hard disk, memory, or processor capacity, allowing the use of very light weight client computers by the end user. Current Cloud systems rely on underlying Distributed File Systems (DFS) to manage data. Examples include Google's GFS [1] and Hadoop's HDFS [2]. The challenges here lie in how to partition data among nodes and how to have nodes collaborate for a specific job. To simplify implementation a simple query processing strategy is followed, i.e., parallel scanning the whole data set and given enough processing nodes, even the simple strategy can provide good performance. In an open service Cloud system Data management becomes more complicated. Therefore, instead of scanning, a more efficient data access service is required. To access data effectively it is required to implement indexing techniques in HDFS part through which we can selectively access the data of interest and skipping the unnecessary data hence resulting in better performance.

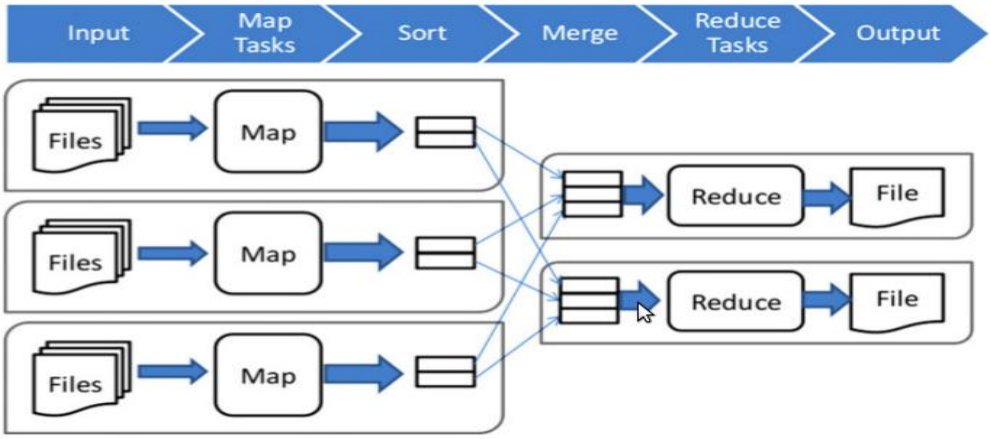

Fig 1. A graphical illustration of the phases of a typical MapReduce job.

MapReduce, programming model of Hadoop open-source implementation, has recently emerged as a popular framework for large-scale data analytics. Several research works have focused on supporting index access in MapReduce systems. These works have allowed users to significantly speed up selective MapReduce jobs by orders of magnitude. However, all these proposals require users to create indexes up-front, which might be a difficult task in certain applications (such as in scientific and social applications) where workloads are evolving or hard to predict.

The original MapReduce paper by Dean and Ghemawat [3] presents a short description for performing indexing in MapReduce, which is directly quoted below:

''The map function parses each document, and emits a sequence of <word, document ID> pairs. The reduce function accepts all pairs for a given word, sorts the corresponding document-IDs and emits a <word, list (document ID)> pair. The set of all output pairs forms a simple inverted index. It is easy to augment this computation to keep track of word positions.''

The biggest issue arise when creating an indexing framework in Hadoop ecosystem is key-value pairs, in RDBMS the data are structured so indexing based on some attribute is easier to implement where as in HDFS data are not structured and no concept of candidate key is present. Still researchers have come up with some indexing techniques in Hadoop through which we can enhance performance. However, all these indexing approaches have three main weaknesses. First, they require a high upfront cost or long idle times for index creation. Second, they can support only one physical sort order (and hence one clustered index) per dataset. Third, they require users to have a good knowledge of the workload in order to choose the indexes to create. The growth of semi-structured and unstructured data far out-paces the growth of relational data. As a result, approach to large-data analysis that integrates structured and unstructured data processing are becoming increasingly critical, so the optimization on text operations (such as search queries, item descriptions, emails, and other user-generated content) are important.

The motivation behind this work is to illustrate the importance of text processing capabilities in a large-data analysis pipeline. The rest of this paper is organized as follows. We provide a survey on the need of indexing in section II by comparing the performance of hadoop with relational databases and then a brief on different indexing technique currently available in Section III, Finally, Section IV concludes this paper with acknowledgement to follow.

II. HADOOP VS. PARALLEL DATABASES

A. It has been pointed out that Hadoop lacks many optimizations that are common in relational databases, and therefore suffers from poor performance on certain analytics tasks. Andrew Pavlo et al [4] have compared the Map Reduce Framework with parallel SQL database management systems (DBMS) in terms of performance and development complexity. The experiment shows that parallel database systems displayed a significant performance advantage over Hadoop MapReduce in executing a variety of data intensive analysis benchmarks. Some of the analytical task includes data loading, selection task, aggregation task, Join task etc.

B. Erik Paulson et al [5], in a similar theme argued that using MR systems to perform tasks that are best suited for DBMSs yields less than satisfactory results, concluding that MapReduce is more like an extract-transform-load (ETL) system than a DBMS, as it quickly loads and processes large amounts of data in an ad hoc manner. Hence after experiments concluded that neither is good at what the other does well, MapReduce complements DBMSs since databases are not designed for extract-transform-load tasks, a MapReduce specialty.

III. EXISTING INDEXING FRAMEWORKS

Kun-Lung-Wu et al [6], proposes an indexing framework for the Cloud system based on structured overlay. The framework reduces the amount of data transferred inside the Cloud and facilitates the development of database back-end applications. The indexing framework supports the existing index structures and two commonly used indexes hash index and B+ index are employed to demonstrate the effectiveness of the framework. Each processing node builds its local index to speed up data access. The global index is distributed over the network, and each node is responsible for maintaining a subset of the global index.

Disadvantages with the framework are:-

1) The network cost dominates the index lookup cost.
2) Concurrent Access cannot be avoided.

Distributed Lucene [7] is based on two Apache open source projects, Lucene and Hadoop and may be said as the first of its kind which focuses on text based indexing. Lucene, a free/open source information retrieval software library, originally created in Java. Lucene is suitable for application which requires full text indexing and searching capability.

Manimal [8], a framework for applying relational optimizations to MapReduce programs. This paper proposes Manimal, which uses static code analysis to detect MapReduce program semantics and thereby enable wholly-automatic optimization of MapReduce programs. Maximal will process the map function, which is similar to a MapReduce program described by both Dean and Ghemawat [3] in their original MapReduce paper, and by Pavlo, et al. [4], in their comparison of MapReduce and RDBMS performance then they have injected a UDF in existing framework which encodes a simple grep count program, counting the number of lines in a text file that match the regular expression given by pattern. By employing static analysis to automatically discern some of the semantics of user code, Manimal can operate on unchanged compiled MapReduce programs. In [9], they continue on the same theme of Manimal [8] by addressing one inefficient aspect of Hadoop based processing: the need to perform a full scan of the entire dataset, even in cases where not necessary to do so. To overcome this drawback they proposed that a full-text index informs the Hadoop execution engine which compressed data blocks contain query terms of interest blocks are referenced by byte offset positions, and only those data blocks are decompressed and scanned.

Advantage of this approach:-

1) Speed up selection procedure hence enhances performance.

Disadvantages with this framework are:-

1) Indexing will store the indexes in a tabular format hence access time is higher which can be further improvised.
2) b) Input splits are computed from the job submission node, it fails to address features like fault tolerance and scalability.

In a related work of Hadoop++ [10], they talk on indexing and joining techniques known as Trojan Index and Trojan Join. Hadoop++ injects Trojan indexes into Hadoop input splits at data loading time. These indexes make it possible to execute relational operation efficiently but these indexes are not designed for full-text processing.

Advantage:-

1) Non-Invasive i.e., do not change the existing framework.
2) Provides optional index access paths which can be used for selective MapReduce jobs.
3) Outperform Hadoop framework in terms of index and join operation.

Disadvantages:-

1) Not designed for full-text processing.

Chandrasekhar S et al [11], proposes a Nobel indexing scheme for reliable storage and management of large number of small files. Storing large number of small files into HDFS becomes an overhead in terms of memory usage by metadata stored in NameNode. In such scenarios, a single NameNode becomes a bottleneck for handling metadata requests, when an application accesses a larger set of these small files. The size of main memory in NameNode restricts the number of files that can be stored into HDFS hence prevents HDFS from being used as a primary data store for scientific and many other applications that produce large amount of small files. Advantage of this approach lies with the I/O performance (Read, Write operation) of system which improves significantly. This work doesn't really talk about text processing but it can be effective for small files.

In [12], Evangelos et al proposes a distributed architecture for indexing and serving large and diverse datasets. In this work, they present a distributed processing platform suitable for indexing, storing and serving large amounts of content data under heavy request loads. It uses Hbase database to serve users request. The content along with some instructions (indexing rules) is fed to the Uploader, which is a MapReduce program. The Uploader creates an HBase table with the content in a record oriented view. A second MapReduce task (Indexer) takes as input the Content table, and extracts the Index, which is also stored as an HBase table to serve objects. In other application to text based indexing is [13], a Hadoop MapReduce framework which performs parallel processing used for an online Content-Based Image Retrieval (CBIR) applications. The idea here is to integrate an image analysis algorithm into the text based image search engine without degrading their response time.

In recent times [14] also focuses on the need of text based indexing. This paper illustrates how the MapReduce model should be used in XML metadata indexing for scientific datasets and the framework scales well for large scale datasets.

IV. CONCLUSIONS

Given the explosion of unstructured data begotten by social media and other web-based applications, we take the position that any modern analytics platform must support operations on free-text fields as first-class citizens. Although these combinations appears to be a workable solution, but the best approach for integrating full-text search capabilities in the Hadoop ecosystem remains an open question. This shows that clearly there is interest in this area and it is the author's belief that as time progresses many business organization and researchers will try to optimize the throughput and reduce response time for large scale analytics.

ACKNOWLEDGMENT

The author would like to thank the Computer Science and Engineering Department, NIT Calicut, for giving him the opportunity and also for providing him with the requisite resources and infrastructure for carrying out the research.

REFERENCES

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, "*The Google file system*," SIGOPS Oper. Syst. Rev., vol. 37, no. 5, pp. 29–43, 2003.

[2] T.A.S. Foundation. (2010) "*Hadoop distributed file system."* [Online]. Available: http://hadoop.apache.org/hdfs/.

[3] J. Dean and S. Ghemawat, "*Mapreduce: simplified data processing on large clusters*," in Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6. San Francisco, CA: USENIX Association, 2004, pp. 10–10.

[4] A. Pavlo, E. Paulson, A. Rasin, D. Abadi, D. DeWitt, S. Madden, and M. Stonebraker. "*A comparison of approaches to large-scale data analysis*," SIGMOD, 2009.

[5] M. Stonebraker, D. Abadi, D. DeWitt, S. Madden, E. Paulson, A. Pavlo, and A. Rasin. "*MapReduce and parallel DBMSs: Friends or foes?*" CACM, 53(1):64–71, 2010.

[6] K.-L. W. Sai Wu, "*An indexing framework for efficient retrieval on the cloud*," IEEE Data Eng. Bull., vol. 32, no. 1, pp. 75–82, 2009.

[7] Butler, Mark H.; Rutherford, James, *"Distributed Lucene: A distributed free text index for Hadoop*," HP Laboratories, HPL-2008-64.

[8] M. Cafarella and C. Ré. *"Manimal: Relational optimization for data-intensive programs." WebDB, 2010.*

[9] Jimmy Lin, Dmitriy Ryaboy, and Kevin Weil*: "Full-text indexing for optimizing selection operations in large-scale data analytics,"* MapReduce'11, June 8, 2011, San Jose, California, USA. *ACM* 978-1-4503-0700-0/11/06.

[10] J. Dittrich, J.-A. Quiané-Ruiz, A. Jindal, Y. Kargin, V. Setty, and J. Schad. *"Hadoop++: Making a yellow elephant run like a cheetah (without it even noticing)."* VLDB, 2010.

[11] Chandrasekar S, Dakshinamurthy R, Seshakumar P G, Prabavathy B, Chitra Babu: *"A Novel Indexing Scheme for Efficient Handling of Small Files in Hadoop Distributed File System*." ICCCI -2013, India.

[12] Ioannis Konstantinou, Evangelos Angelou, Dimitrios Tsoumakos, Nectarios Koziris, "*Distributed Indexing of Web Scale Datasets for the Cloud*," ACM, MDAC '10 Proceedings of the 2010 Workshop on Massive Data Analytics on the Cloud.

[13] Premchaiswadi Wichian, Tungkatsathan Anucha, Intarasema Sarayut , Premchaiswadi Nucharee, *"Improving performance of content-based image retrieval schemes using Hadoop MapReduce,*" High Performance Computing and Simulation (HPCS), 2013, Page(s): 615 – 620.

[14] Dede E., Fadika Z., Gupta C, Govindaraju M, "*Scalable and Distributed Processing of Scientific XML Data*," Grid Computing (GRID), 2011 12th *IEEE/ACM*.

[15] Tom White, Hadoop: *The Definitive Guide.*

[16] http://hadoop.apache.org/

