International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 05
April 2017

PERFORMANCE ANALYSIS OF MAPREDUCE WTH LARGE DATASETS USING HADOOP

V.Sri Divya[1], M.Tejaswini[2], Sk.Parveen[3], B.Triveni[4]

[1] B.Tech C.S.E TKREC Hyderabad E-mail: divyavelugaleti@gmail.com

[2] B.Tech C.S.E TKREC Hyderabad E-mail: madhavaramtejaswini@gmail.com

[3] B.Tech C.S.E TKREC Hyderabad E-mail: pparveen1295@gmail.com

[4] Associate Professor, TKREC, Hyderabad, TS-India, E-mail: triveni.banavatu@gmail.com

ABSTRACT

Big data[1] is a huge amount of data that cannot be managed by traditional data management systems. Hadoop[2] is a tool that is used to handle this big data. For storing and retrieving the bigdata hadoop distributed file[4] system(HDFS) and mapreduce[3] are used respectively. Even petabytes or terabytes of data can be stored and retrieved easily using these techniques. This paper provides introduction to hadoop HDFS and Mapreduce. In this paper we have used large datasets to analyse the performance of mapreduce technique. Number of bytes read and written while performing mapreduce task on input given is also observed. We have analysed the behaviour of mapreduce task by varying the amount of input given. Also the pattern of number of bytes read and written when given input is varied is also analysed.

Keywords— Big data, Hadoop, MapReduce, Distributed file system,

1. INTRODUCTION

MapReduce[1] is a widely used computing model for large scale data processing in cloud computing. A MapReduce job consists of a set of map and reduce tasks, where reduce tasks are performed after the map tasks. Hadoop[2] an open source implementation of Map Reduce has been deployed in large clusters containing thousands of machines by companies such as Amazon and Facebook. In those cluster and data centre environments, Map Reduce and Hadoop[3] are used to support batch processing for jobs submitted from multiple users Map Reduce workloads. Despite many research efforts devoted to improve the performance of a single Map Reduce job there is relatively little attention paid to the system performance of Map Reduce workloads. Therefore, we tried to improve the performance of Map Reduce workloads.

The emerging big-data paradigm, owing to its broader impact, has profoundly transformed our society and will continue to attract diverse attentions from both technological experts and the public in general. It is obvious that we are living a data deluge era, evidenced by the sheer

volume of data from a variety of sources and its growing rate of generation. The exponential growth of data first presented challenges to cutting-edge businesses such as Google, Yahoo, Amazon, Microsoft, Facebook, Twitter etc. Data volumes to be processed by cloud applications are growing much faster than computing power. For instance, an IDC report predicts that, from 2005 to 2020, the global data volume will grow by a factor of 300, from 130 Exabyte to 40,000 Exabyte, representing a double growth every two years. The term of "big-data" was coined to capture the profound meaning of this data-explosion trend and indeed the data has been touted as the new oil, which is expected to transform our society. The huge potential associated with big-data has led to an emerging research that has quickly attracted tremendous interest from diverse sectors, for example, industry, government and research community. Government has also played a major role in creating new programs to accelerate the progress of tackling the bigdata challenges. This growth demands new strategies for processing and analyzing information. Hadoop has become a powerful Computation Model addresses to these problems. Hadoop HDFS became more popular amongst all the Big Data tools as it is open source with flexible scalability, less total cost of ownership and allows data stores of any form without the need to have data types or schemas defined. Hadoop MapReduce is a programming model and software framework for writing applications that rapidly process vast amounts of data in parallel on large clusters of compute nodes. Map reduce is a software frame work[4] introduced by Google in 2004 to support distributed computing on large data sets on clusters of computers. The original MapReduce implementation by Google, as well as its open-source counterpart, Hadoop, is aimed for parallelizing computing in large clusters of commodity machines. MapReduce model advantage is the easy scaling of data processing over multiple computing nodes.

2. EXISTING SYSTEM

SRB is a logical distributed file system based on a client-server architecture which presents users with a single global logical namespace or file hierarchy. SRB provides a uniform interface to heterogeneous computer data storage resources over a network. As part of this, it implements a logical namespace (distinct from physical file names) and maintains metadata on data-objects (files), users, groups, resources, collections, and other items in an SRB metadata catalog (MCAT) stored in a relational database management system. System and user-defined metadata can be queried to locate files based on attributes as well as by name. SRB runs on various versions of Unix, Linux, and Microsoft Windows

3. PROPOSED SYSTEM

For processing and generate MapReduce is a programming model and an associated implementation big datasets with a parallel, distributed algorithm on a cluster. A MapReduce program is composed of a map() procedure (method) that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a Reduce() method that performs a

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 05
April 2017

summary operation (such as counting the number of students in each queue, yielding name frequencies). The "MapReduce System" (also called "infrastructure" or "framework") orchestrates the processing by marshalling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance. The model is a specialization of the split-apply-combine strategy for data analysis.

4. ARCHITECTURE

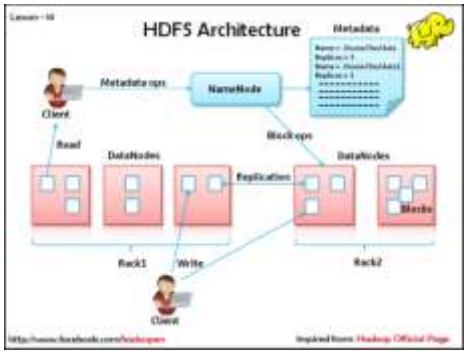


Figure 5.1 HDFS Architecture

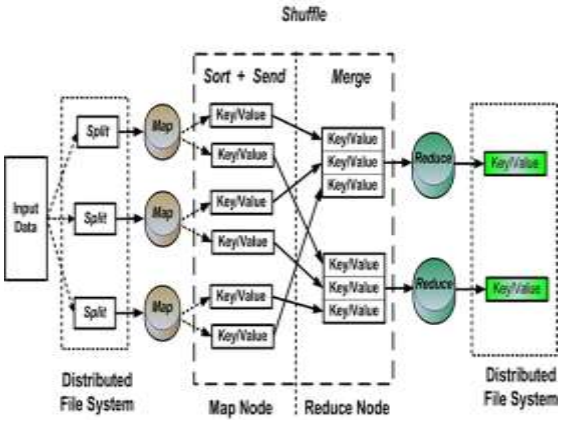


Figure 5.2 mapreduce architecture

The mapReduce is master slave architecture as in HDFS. There are two types of nodes Task-tracker and Jobtracker. Tasktracker act as master node and jobtracker as slave. The tasktracker divide the entire program into a number of individual programs and give it to the workers. The worker computes each program individually and results are give back to Tasktracker. Job Tracker runs with the namenode, receives the user's job, decides on how many tasks will run (number of mappers) and decides on where to run each mapper (concept of locality) Master pings workers periodically to detect failures.

Master can distribute a map task or a reduce task to an idle worker. If a worker is assigned a Map task, it will parse the input data partition and output the key/value pairs, then pass the pair to a user defined Map function. The map function will buffer the temporary key/value pairs in memory. The pairs will periodically be written to local disk and partitioned into P pieces. After that, the local machine will inform the master of the location of these pairs. If a worker is assigned a Reduce task and is informed about the location of these pairs, the Reducer will read the entire buffer by using remote procedure calls. After that, it will sort the temporary data based on the key. Then, the reducer will deal with all of the records. For each key and according set of values, the reducer passes key/value pairs to a user defined Reduce function. The output is the final output of this partition. After all of the mappers and reducers have finished their work, the master will return the result to users' programs. The output is stored in F individual files.

5. ALGORITHM

The mapreduce algorithm happens in the following sequence.

"Map" step: Each worker node applies the "map()" function to the local data, and writes the output to a temporary storage. A

master node ensures that only one copy of redundant input data is processed.

"Shuffle" step: Worker nodes redistribute data based on the output keys (produced by the "map()" function), such that all data belonging to one key is located on the same worker node.

"Reduce" step: Worker nodes now process each group of output data, per key, in parallel.

6. MODULES

i. (i)Map Module
ii. (ii)Reduce Module

(i) Map Module

The Mapper class defines the Map job. Maps input key-value pairs to a set of intermediate key-value pairs. Maps are the individual tasks that transform the input records into intermediate records. The transformed intermediate records need not be of the same type as the input records. A given input pair may map to zero or many output pairs.

(ii) **Reduce Module**

The Reducer class defines the Reduce job in MapReduce. It reduces a set of intermediate values that share a key to a smaller set of values. Reducer implementations can access the Configuration for a job via the JobContext.getConfiguration() method. A Reducer has three primary phases – Shuffle, Sort, and Reduce.

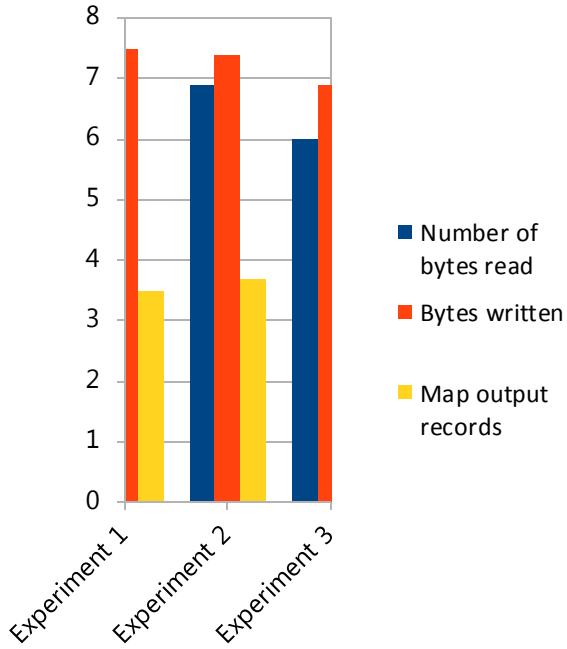
i. Shuffle − The Reducer copies the sorted output from each Mapper using HTTP across the network.

ii. Sort −The framework merge-sorts the Reducer inputs by keys (since different Mappers may have output the same key). The shuffle and sort phases occur simultaneously, i.e., while outputs are being fetched, they are merged.

iii. Reduce − In this phase the reduce (Object, Iterable, Context) method is called for each <key, (collection of values)> in the sorted inputs.

7. GRAPH

When the dataset size is increased that is the number of records are increased, number of bytes read and written by the wordcount method are not directly proportional to the dataset size.



8. CONCLUSION

The MapReduce programming model has been successfully used at Google for many different purposes. We attribute this success to several reasons. First, the model is easy to use, even for programmers without experience with parallel and distributed systems, since it hides the details of parallelization, fault-tolerance, locality optimization, and load balancing.

Second, a large variety of problems are easily expressible as MapReduce computations. For example, MapReduce is used for the generation of data for Google's production web search service, for sorting, for data mining, for machine learning, and many other systems. Third, we have developed an implementation of MapReduce that scales to large clusters of machines comprising thousands of machines. The implementation makes efficient use of these machine resources and therefore is suitable for use on many of the large computational problems encountered at Google. We have learned several things from this work. First, restricting the programming model makes it easy to parallelize and distribute computations and to make such computations fault-tolerant. Second, network bandwidth is a scarce resource. A number of optimizations in our system are therefore targeted at reducing the amount of data sent across the network: the locality optimization allows us to read data from local disks, and writing a single copy of the intermediate data to local disk saves network bandwidth. Third, redundant execution can be used to reduce the impact of slow machines, and to handle machine failures and data loss.

9. FUTURE ENHANCEMENT

MapReduce Programming model has been instrumental in data processing and data analysis applications in the field of bigdata. This is accomplished through mapreduce support of scalability, efficiency and fault tolerance. However, this means that framework must make tradeoffs between these features and performance optimization. Improving upon and extending mapreduce model to implement such optimization and improve overall performance while not compromising on the features that have made it successful has been an important area of research and will prove to be so in the future.

10. BIBILOGRAPHY

[1] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.

[2] "Apache hadoop yarn," http://hadoop.apache.org/docs/ current/hadoop-yarn/hadoop-yarn-site/YARN.html.

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

[4] "A Performance Analysis of MapReduce Task with Large Number of Files Dataset in Big Data Using Hadoop" Amrit pal, Kunal Jain, oinkiAgrawal, Sanjay Agrawal.

[5] "The BTWorld Use Case for Big Data Analytics: Description, MapReduce Logical Workflow, and Empirical Evaluation", Tim Hegeman, BogdanGhit,, MihaiCapot˘a, Jan Hidders, Dick Epema, and AlexandruIosup Parallel and Distributed Systems Group, Delft University of Technology, the Netherlands T.M.{B.I.Ghit, M.Capota, A.J.H.Hidders, D.H.J.Epema, A.Iosup}@tudelft.nl.

[6] Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, David E. Culler, Joseph M. Hellerstein, and David A. Patterson. High-performance sorting on networks of workstations. In Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, May 1997.

[7] Remzi H. Arpaci-Dusseau, Eric Anderson, Noah Treuhaft, David E. Culler, Joseph M. Hellerstein, David Patterson, and Kathy Yelick. Cluster I/O with River: Making the fast case common. In Proceedings of the Sixth Workshop on Input/Output in Parallel and Distributed Systems (IOPADS '99), pages 10–22, Atlanta, Georgia, May 1999.

[8] Arash Baratloo, Mehmet Karaul, Zvi Kedem, and Peter Wyckoff. Charlotte: Metacomputing on the web. In Proceedings of the 9th International Conference on Parallel and Distributed Computing Systems, 1996.

[9] Luiz A. Barroso, Jeffrey Dean, and Urs Holzle. ¨ Web search for a planet: The Google cluster architecture. IEEE Micro, 23(2):22–28, April 2003.

[10] Arash Baratloo, Mehmet Karaul, Zvi Kedem, and Peter Wyckoff. Charlotte: Metacomputing on the web. In Proceedings of the 9th International Conference on Parallel and Distributed Computing Systems, 1996.

[11] Luiz A. Barroso, Jeffrey Dean, and Urs Holzle. ¨ Web search for a planet: The Google cluster architecture. IEEE Micro, 23(2):22–28, April 2003.

[12] John Bent, Douglas Thain, Andrea C.Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, and Miron Livny. Explicit control in a batch-aware distributed file system. In Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation NSDI, March 2004.

[13] Guy E. Blelloch. Scans as primitive parallel operations. IEEE Transactions on Computers, C-38(11), November 1989.

[14] Armando Fox, Steven D. Gribble, Yatin Chawathe, Eric A. Brewer, and Paul Gauthier. Cluster-based scalable network services. In Proceedings of the 16th ACM Symposium on Operating System Principles, pages 78– 91, Saint-Malo, France, 1997.

[15] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google file system. In 19th Symposium on Operating Systems Principles, pages 29–43, Lake George, New York, 2003.