International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 03 | Mar-2016 www.irjet.net p-ISSN: 2395-0072

STRATEGIC ANALYSIS WITH BIG DATA

Pooja G. Joshi [1], Om P. Bankar [2], Ajay K. Ghode [3],Neeta Patil [4]

[1234] *Student, Information Technology Department, MMIT, Maharashtra, INDIA*

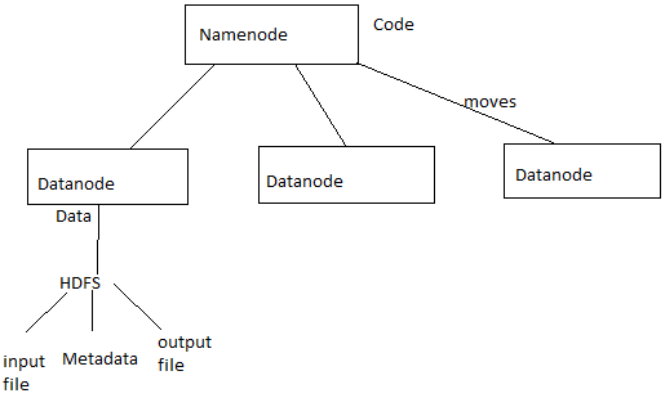
Abstract - In this paper, We introduce a generic application, a robust analytic system that we developed for collecting and delivering high volumes of data. Our system incorporates ideas from existing centralized system and shifted to distributed system. Using hadoop framework which stores large amount of unstructured data. To develop a scalable approach to this system, MapReduce Distributed processing framework where the application is divided into many small fragments of work, each of which may be executed or re-executed on any node in the cluster ,the analysed data can be send without any data loss. Our expected results will produce superior performance depend on criteria of application. Result will be display in any format as per organizations requirement like string,Graph,etc.

Key Words: *Hadoop MapReduce , custom partitioner*

1. INTRODUCTION

The proposed application for big data analytic, where the traditional way of processing huge datasets has been shifted from centralized architecture to distributed architecture. In this generic application ,multiple data cluster data nodes are produces one output after implementing MapReduce job ,finally the best result is provided by name node which monitor whole flow. Hadoop provide scalable approach to this application .MapReduce Distributed processing framework where the application is divided into many small fragments of work, each of which may be executed or re-executed on any node in the cluster.

Hadoop is a combination of two domain that is distributed system and bigdata. Hadoop has 100% guarantee to secure of data and availability. In hadoop echo system data at one place and code will move to the data.



1.1 HDFS

The Hadoop Distributed File System is based on the Google File System and provides a distributed file system that is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However,
the differences from other distributed file systems are significant. It has to be low cost hardware and fault tolerant. It provides high throughput access to application data and is suitable for applications having large datasets.

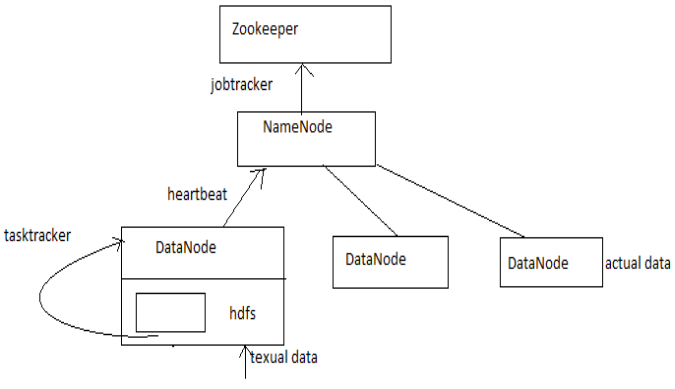


fig. HDFS Architecture

Fig-1 : HDFS architecture

1.2 Zookeeper

Zookeeper is a thread and also demon process. DataNode:actual work done by it.

NameNode If namenode will dead then it will give task on one of the datanode which is nearer to that namenode.

1.3 Hbase

It is combination of hadoop distributed file system. It is used to store all the information about entire the echo system.

2. PARSER

Parser, which Parse multi-structured and industry-standard data on Hadoop, including industry standards documents, log files, and complex file formats.

Here, multiple heterogeneous file format like xml file, text file,xls file,xlxs file,etc. are converted into standard CSV file format. CSV file extension has become a kind of legal industry standard. The information is organized with one record on each line and each field is separated by comma.CSV file is a set of database rows and columns stored in a text file such that the rows are separated by a new line while the columns are separated by a semicolon or a comma.

The advantage of using CSV file format for data exchange is that the CSV file is relatively easy to process by any application and data extraction can be achieved with the help of a simple program. In the earlier years when database technologies were still in their infancy, the CSV was the most standard portable format. A CSV file is a way to collect the data from any table so that it can be conveyed as input to another table-oriented application such as a relational database application. Microsoft Excel, a leading spreadsheet or relational database application, can read CSV files. A CSV file is sometimes referred to as a flat file.

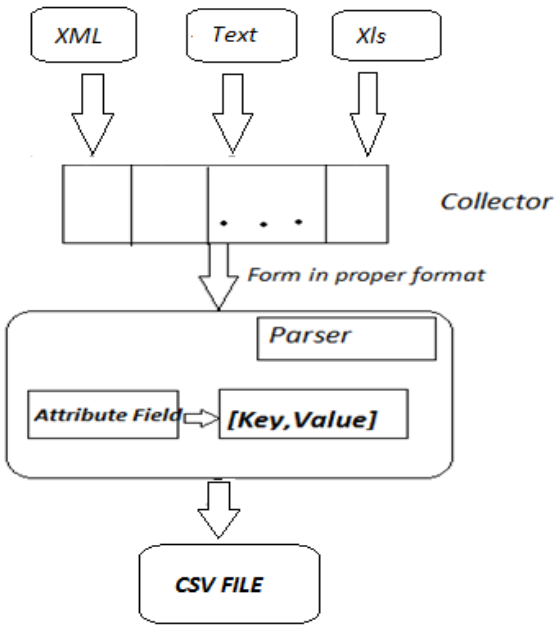


Fig: Working of parser

3. MAP PHASE

The purpose of the **map** phase is to organize the data in preparation for the processing done in the **reduce** phase. The input to the map function is in the form of key-value pairs, even though the input to a MapReduce program is a file or file(s). By default, the value is a data record and the key is generally the offset of the data record from the beginning of the data file.

map (*InputKeyType* inputKey, *InputValueType* inputValue):

- do processing on the *inputKey* and *inputValue* and form *intermediateKey* , *intermediateValue* pair
- Emit(intermediateKey, intermediateValue);
- map can emit more than one intermediate key-value pairs

4. CUSTOM PARTITIONING

The partitioning phase takes place after the map phase and before the reduce phase. The number of partitions is equal to the number of reducers. The data gets partitioned across the reducers according to the partitioning function. The difference between a partitioner and a combiner is that the partitioner divides the data according to the number of reducers so that all the data in a single partition gets executed by a single reducer. However, the combiner

functions similar to the reducer and processes the data in each partition. The combiner is an optimization to the reducer. The default partitioning function is the hash partitioning function where the hashing is done on the key. However it might be useful to partition the data according to some other function of the key or the value.By default the partitioner implementation is called HashPartitioner.

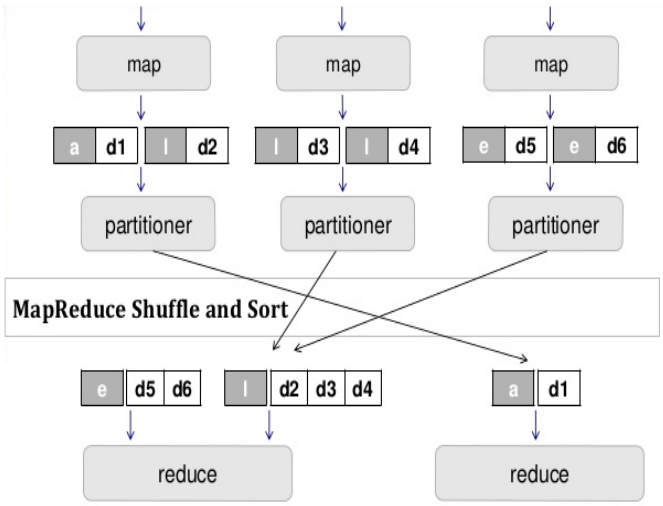


Fig:-Partitioner Data Flow

4. REDUCER

Each **reduce** function processes the intermediate values for a particular key generated by the **map** function and generates the output. Essentially there exists a one-one mapping between keys and reducers. Several reducers can run in parallel, since they are independent of one another. The number of reducers is decided by the user. By default, the number of reducers is 1.

Although the reduce phase depends on output from the map phase, map and reduce processing is not necessarily sequential. **The reduce phase** uses results from map tasks as input to a set of parallel reduce tasks. The reduce tasks consolidate the data into final results. By default, the MapReduce framework stores results in HDFS.reduce(IntermediateKeyType intermediate Key, Iterator values):

- the values associated with a particular intermediateKey is iterated and a user-specified operation is performed over the values
- multiple reducers can run in parallel and the number of reducers is specified by the user
- *outputKey* will contain the key output from the reducer and *outputValue* will contain the value that is output for that particular key
- Emit(outputKey, outputValue);
- reduce method can emit more than one output key-value pairs.

5. ALGORITHM

Step 1 : Data Collection from distributed client node.

Step 2: source data will parse into standard single format data.

Step 3 : Data will be stored in hdfs input.

Step 4 : Each node data will Map on distributed field.

Step 5 : Custom Partitioner will produce anlysed data.

Step 6 : Data will stored in hdfs output .

Step 7 : Result will read from hdfs.

6. CONCLUSION

In this paper, we have presented a generic system which successfully complete strategic analysis with big data. And also Hadoop provides an extremly powerful set of tools to slove very big problem. MapReduce is for deal with huge data, but not hard problems. It is best for string parsing kind of operation. Moreover flexibility is increased. This work is the first step to contribute to the real time implementation of the system. The early encouraging results we obtained only aimed to improve the quality of analyzing system. Our present work is limited to the text files based only. In particular future plans are to contribute to different formats. Additionally we plan to desire a more sophisticated approach to when a user should be penetrated into a black lists.

REFERENCES

[1] J. Dean, S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," In Proc. of the 6th Symposium on Operating Systems Design and Implementation, San Francisco CA, Dec. 2004.

[2] A. Gates, O. Natkovich, S. Chopra, P. Kamath, S. Narayanam, C. Olston, B. Reed, S. Srinivasan, U. Srivastava. "Building a High-Level Dataflow System on top of MapReduce: The Pig Experience," In Proc. of Very Large Data Bases, vol 2 no. 2, 2009, pp. 1414–1425

[3] H. Carns, W. B. Ligon III, R. B. Ross, and R. Thakur. "PVFS: A parallel file system for Linux clusters," in

Proc. of 4th Annual Linux Showcase and Conference, 2000, pp. 317–327.

[4] F. P. Junqueira, B. C. Reed. "The life and times of a zookeeper," In Proc. of the 28th ACM Symposium on Principles of Distributed Computing, Calgary, AB, Canada, August 10–12, 2009.

[5] K. V. Shvachko, "HDFS Scalability: The limits to growth," ;login:. April 2010, pp. 6–16.

[6] T. White, Hadoop: The Definitive Guide. O'Reilly Media, Yahoo! Press, June 5, 2009.

[7] F. P. Junqueira, B. C. Reed. "The life and times of a zookeeper," In Proc. of the 28th ACM Symposium on Principles of Distributed Computing, Calgary, AB, Canada, August 10–12, 2009.

[8]" Hadoop Performance Modeling for Job Estimation and Resource Provisioning" Mukhtaj Khan, Yong Jin, Maozhen Li, Yang Xiang and Changjun Jiang,2015,IEEE

[9]"Mammoth: Gearing Hadoop Towards Memory-Intensive MapReduce Applications"
Xuanhua Shi, Ming Chen, Ligang He, Member, IEEE, Xu Xie, Lu Lu, Hai Jin, Senior Member, IEEE, Yong Chen, Member, IEEE, and Song Wu,AUGUST 2015,IEEE

[10] "Dache:" A Data Aware Caching for Big-Data Applications Using the MapReduce Framework" Yaxiong Zhao, Jie Wu, and Cong Liu,TSINGHUA SCIENCE AND TECHNOLOGY ISSNl l1007-0214l l05/10l lpp39-50 Volume 19, Number 1, February 2014

[11]"Toward Scalable Systems for Big Data Analytics: A Technology Tutorial"
HAN HU1, YONGGANG WEN2, (Senior Member, IEEE), TAT-SENG
CHUA1, AND XUELONG LI3, (Fellow, IEEE)July 8, 2014.

[12] Apache Hadoop. http://hadoop.apache.org/

[13] Hadoop File System http://hadoop.apache.org/hdfs/

[14] Zookeeper http://hadoop.apache.org/zookeeper/

