ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 12, December 2017

Scalable Map Reduce Model for Aggregating the Data in the Cloud

[1] S.Rajan, [2] Dr.D.S.Mahendran, [3] Dr.S.John Peter
[1] Assistant Professor and Head, Dept of Computer Science [SF], Kamaraj College, Tuticorin,Tamil Nadu, India.
[2] Associate Professor, Dept of Computer Science, Aditanar College, Tiruchendur, Tamil Nadu, India
[3] Associate Professor & Head of Research Centre, Dept of Computer Science, St.Xavier College, Palayamkottai, Tamil Nadu, India.

Abstract: - **Scalability depends upon the nature of the problem and the algorithm which gives the solution to the problem. Load balancing balances the load between different virtual machines so that performance of the cloud can be improved by removing the load disparity among the Virtual Machines. Load balancing works well for the scalable Application. An improperly Scaled application will behave in an unpredicted manner. The resources such as RAM, CPU, Disk etc., may increase or decrease depending upon the need of the application so the resources should be allocated previously. This can be achieved by effectively scaling the application. Scalable Application decreases the costs as the costs are calculated based on the usage of the resource. To achieve the best scalable application in the cloud, there is a need for a programming model which is highly fault-tolerant for the distributed file system such as GFS or HDFS. In this paper, we analyze the selected problem and the solution is framed based on the Map Reduce Programming Model. The efficiency of the scalability is studied through Parallel efficiency calculation. Based on the Parallel efficiency value effective Scalable Map Reduce Model is proposed.**

Keywords: **Scalability, Parallel Efficiency, GFS, HDFS, Map Reduce, cloud.**

I. INTRODUCTION

The advancement in Cloud computing made the Database to be processed through parallel computing. Parallel computing dates back to 1970s. To perform scalable parallel computing there is a need for Map Reduce Programming model and the HDFS which effectively handles large volume of data and processes data in parallel. The Architecture of Parallel Database, the Architecture of HDFS and the Reading and Writing procedures of the HDFS should be considered before Calculation is done on Parallel Efficiency.

Map Reduce

Map Reduce is one of the important programming models in cloud. The four forms of Map Reduce are Map only, Classic Map Reduce, iterative Map Reduce and loosely synchronous. It changes the way the programming is done. Major draw back in cloud is its security. [10] By effectively managing the key we can improve the security to some extend in cloud.

Architecture of Parallel Database

The different parallel database architectures are as follows
Shared Memory: It suites for machines with many CPUs. It may also works well for CPU with many cores. All the CPUs are sharing the common Memory address. [9]This architecture will be handled well by Symmetric Multi-Processing operating system.
Shared Disk: Its existence can be traced back to 1980s. Here [1][9]Disk storage is shared by the cluster of servers which are connected through high speed network cable such as fiber channel to the storage either through Network attached Storage (NAS) or Storage Area Network (SAN).
Shared – Nothing: Here each disk will be having separate memory. Tables are partitioned across many processing computers. SQL optimizer performs computations through distributed joins from database portioned in different computers.

Hadoop Distributed File System

HDFS works effectively in Map Reduce programming Model. It supports parallel reads, writes and appends by multiple client Programs. It’s highly fault tolerant and manages large files provided cluster of distributed servers are connected by high-speed network. [2] In HDFS Large files are broken up into chunks which are of size of about 125MB. These are stored in Data Nodes. Each Data Node is replicated at least three times on different physical rack with different network segment to obtain high fault tolerant capability.

Read Operation in HDFS

Step 1: The client will be sending full path of the file and offset to the Name Node which it needs to read.

Step 2: The Name Node will send the meta-data about one of the Data Node where the file need to be read is present to the client. The client caches the meta-data and directly reads data from the designated Data Node.

Write Operation in HDFS

Step 1: The client will be sending full path of the file to the Name Node

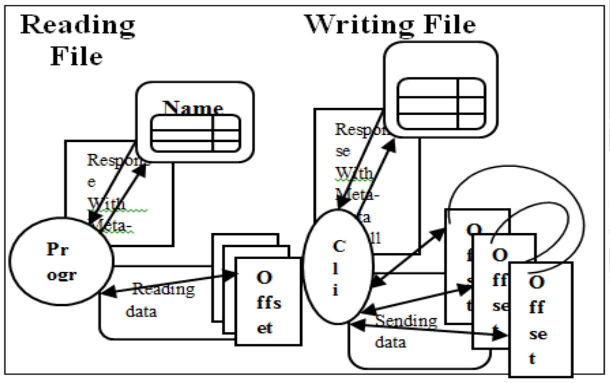
Step 2: The Name Node will send the meta-data about all Data Node (especially three) where the file need to be written are present.

Step 3: The client cache the meta-data of the entire Data Node and sends the data to be appended to all the Data Node.

Step 4: After getting the Acknowledgement for the received data from all Data Nodes, the client informs primary Data Node about this.

Step 5: The Primary Data Node appends its copy of data at an offset of its choice and forward to all Data Node to write in the same offset.

Step 6: If failure occurs step 5 is repeated with different offset until writing is done for all Data Node.



II. RELATED RESEARCH

Not much research is done on the scalability of Map Reduce Framework for aggregating the data. We see some research Paper on Map Reduce . [3] Manisha sahane analyzed the research data using the word count algorithm of Map Reduce Model. His research domain mainly focused on zoology and Botany. He used the existing models to study and analyze the Research data. The trend of Research in this Area can be studied by using this method. [4] To improve efficiency, Yingyi Bu et al made changes in Map Reduce Framework to support repetitive program. [5] Kumar et al designed a method to improve efficiency of Hadoop and process data on memory. [6] Lam et al proposed the solution for managing data generated at high speed in Map Reduce. [7] Emad et al combined SDN and adaptive combiner to improve the efficiency of Map Reduce. [8] Anca Vulpe proposed cloud enabled astrophysics algorithm that obtained sustainable Data scalability by increasing the number of sub clusters to tens of millions of comparisons.

Statement of the Problem

Here we choose the Banking problem where the bank wants to know the number of Bank Accounts linked to Each Aadhaar Number for each customer in the same or different branches. The splitter Reads the Customer Data from different branches and tokenizes into <Aadhaar key, value> pairs and Mapper shuffles and counts Bank Accounts linked to Aadhaar in a single file and stores the result in HDFS. The Reducer Reads <Aadhaar Key, Value> pair from the GFS files and calculates the aggregated data. The input document is the text file with different fields such as account number, aadhaar number, Customer name, Customer Street, Customer city stored as record of each customer in each line of the text file.

Proposed Model

The banking problem of linking different Bank accounts to Aadhaar number is studied based on Parallel Efficiency calculated by different Mapper-Reducer Processors. Based on the value of Parallel efficiency we are designing the Map Reduce Model. Assuming the Bank's Customer Data is stored in Cloud for different branches.

The document contains r number of lines called rows. There are n numbers of documents. Each row gives details about each Bank accounts and Aadhaar number linked to that Bank Account. The tokens are as follows

(Dz, [Ac1, Aa1, ……] -> [(Aa1, Count)]) And Map: $(K1, v1) -> [(K2, v2)]$

Reduce: $(K2, [v2]) -> (K[illegible], f([v2]))$

$(Aaj, [Countj]) -> (Aaj, \sum_j Countj)$

Document is indexed by its ID. The Document is tokenized to give Aadhaar count for different Bank accounts Linked to Aadhaar number. The Reducer sums the Aadhaar count for each account.

$Ep = T1 / pTp$; Parallel Efficiency is the time required to compute the task by single processor to the time required to do the same task using p number of processors. [2] This is calculated to find the scalability and parallel efficiency of the task with the given number of processors.

$$TD = cD + CmD + Cr\sigma D + c\sigma\mu D \ldots\ldots 1$$

$$\epsilon MR = \frac{TD}{P\left((TD)/P + 2c\frac{(\sigma D)}{P}\right)} = \frac{1}{\left(1+\frac{2c}{w}\sigma\right)} \ldots.2$$

We are slightly altering the above formula to find solution for our problem. The volume of data D =n2r; where n is the number of documents and r is the no of lines in each document. 2r is the number of Aadhaar and Account number present in a single document. We are assuming each line

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 12, December 2017

contains information about each customer. Cr=Cm=0 as adding Aadhaar count takes negligible amount of time. As we are using Shared Disk and the cluster of servers are sharing storage through High-Speed Network, the time to read/write locally and remotely are same. As reading and writing takes the same time c=w. Substituting in σ =2P/n2r = P/nr in equation 2 we get

$$EMR = \frac{1}{1+\frac{2P}{nr}} \qquad -----3$$

The scalable Parallel implementation can be obtained if it satisfies the following conditions

a) Parallel efficiency (EMR) remains constant as the size of data increases with a corresponding increase in processors.

b) The parallel efficiency (EMR) increases and approaches one with the size of data for a fixed number of processors.

The Parallel Efficiencies obtained are listed as follows:

No of lines in document (n)	No of Documents (r)	No of Processor (P)	Parallel Efficiency (EMR)
100	1000	10000	.8333
100	1000	1000	.9803921
100	1000	500	.99009900
100	1000	50	.999000999
1000	1000	50	.99990000
100	1000	5	.99990000
1000	10000	5	.999999000000
10000	100000	5	.9999999900000000
1000	10000	50	.9999900000
10000	100000	500	.999999000000

Table1 - Overall

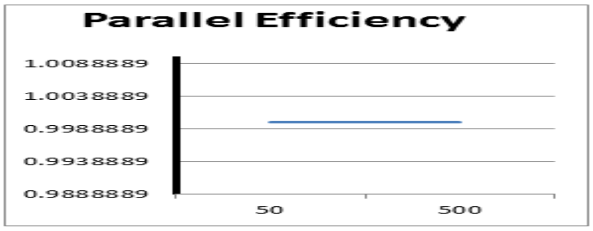
By looking through the table given above, we extracted the following information

The condition a) is met for n=1000, r=10000 and p=50; n=10000, r=100000 and p=500 (EMR is almost constant); for n=1000; r=1000 and p=50; n=100, r=1000 and p=5 (EMR is exactly same constant value)

The condition b) is met for n=100, r=1000; n=1000, r=10000 and n=10000, r=100000 for p=5 (EMR value approaches 1)

But if the no of processor is increased for the fixed value of Data, the Parallel Efficiency decreases and moving towards zero. So the scalability cannot be obtained for n=100, r=1000 P=10000; n=100, r=1000 P= 1000; n=100, r= 1000, P= 500; n=100, r=1000, P=50

Graph1: Y axis EMR Value; X Axis: No of processors



Graph1

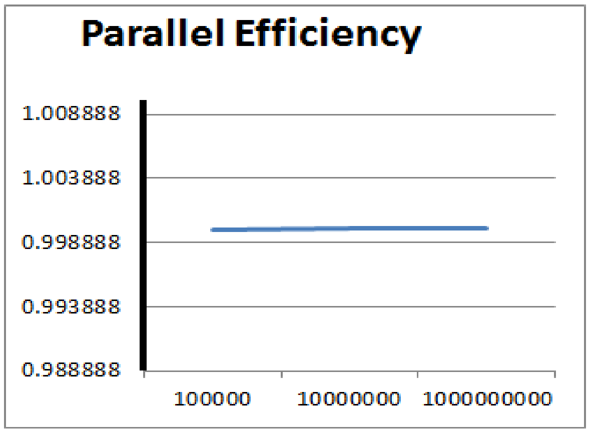
No of Processors (with increased no of Documents)	Parallel efficiency(EMR)
50	.9999900000
500	.999999000000

Table2

Graph2: Y axis EMR; X Axis: No of Documents * No of rows in Document

Table3

No Of rows in document X No of Documents (For 5 processors)	Parallel efficiency(EMR)
100000	.99990000
10000000	.999999000000
1000000000	.9999999900000000



Graph2

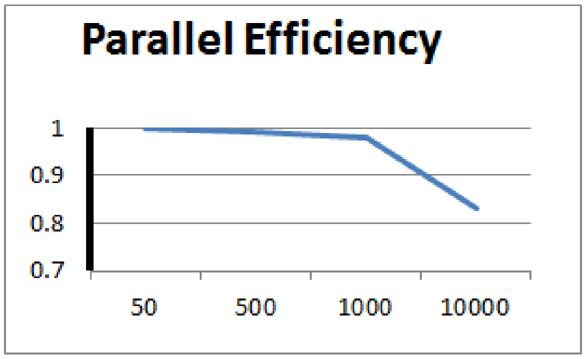
Graph3: Y axis EMR; X Axis: No of Processors

Table4

No of processor (P) (For fixed 1000 documents with 100 rows)	Parallel efficiency(EMR)
10000	.8333
1000	.9803921
500	.99009900
50	.999000999

ISSN (Online) 2394-2320

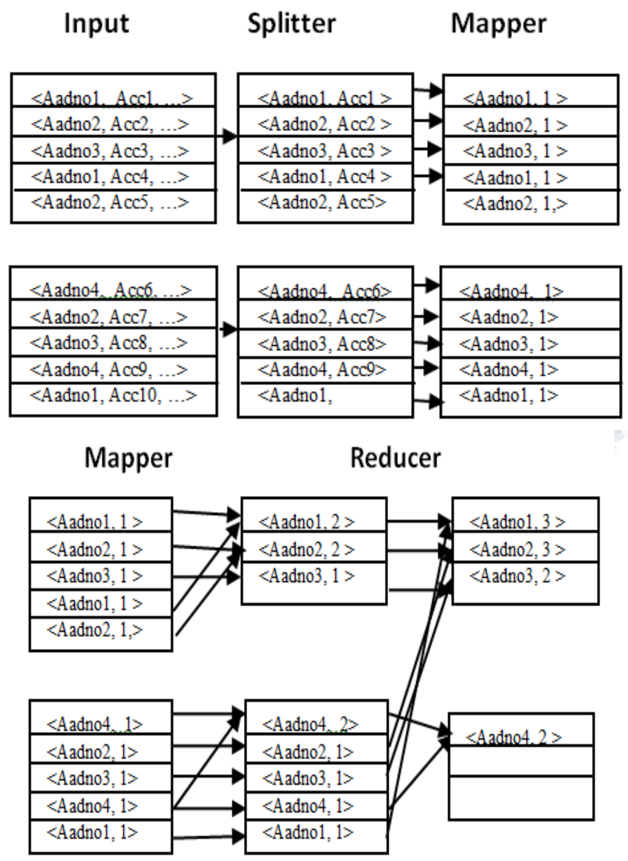
International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 12, December 2017



Graph3

For the conditions mentioned in Graph 1 (Table2) and Graph 2 (Table3) Scalability can be achieved and for the condition mentioned in Graph 3 (Table4) scalability cannot be achieved.

Map Reduce Model for Aadhaar Linked to Bank Account



For simple demonstration we took Mapper = 2 and Reducer =2; n=2 and r =5. We can increase the number of Mapper, Reducer, n and r values depending upon the complexity of the problem.

Split is used for Splitting Aadhaar No and Bank AccNo **Mapper** is used to count the number of accounts Linked to the Aadhaar number in a single file.
Reducer further reduces and counts the no of Bank AccNos Linked to Aadhaar in the single and different files. No of Mapper Processor and No of Reducer Processor depends on the volume of Data.
To attain Scalable parallel implementation, number of Processors (for Mapper and Reducer) and Data (Number of Documents X Number of rows in each document) is chosen in such a way that it meets the criteria specified in Graph1 (Table2) and Graph2 (Table3).

III.CONCLUSION

The effective Scalable Parallel implementation of Map Reduce model can be designed for this problem if P<<nr and a) Parallel Efficiency remains constant if P increases with corresponding increase in nr value. b) Parallel Efficiency approaches 1 for the fixed number of processors. The Scalability and Parallel efficiency cannot be obtained if P >> nr. The Effective Scalable Parallel implementation is thus designed for finding the number of Bank Accounts linked to Aadhaar number.

REFERENCES

[1] Ragu Ramakrishnan and Johanes Gehrke, Database Management system, McGraw-Hill, 2003

[2] S. Ghemawat, H.Gobioff and S.T.Leung. Google File System. In Proceedings of the 19th ACM SOSP Conference, Dec 2003.

[3] Analysis of Research Data using Map Reduce Word Count Algorithm, Manisha Sahane, Sanjay Sirsat, Razaullah Khan, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 5 , May 2015

[4] B.H.M.B.M.D.E.Yingyi Bu, "HaLoop: Efficient Iterative Data Processing on Large Clusters," in 36th international Conference on Very Large Data Bases, Singapore, 2010

[5] J.G.A.D.a.1.LK.Ashwin Kumar, Hone: "Scaling Down Hadoop on Shared-Memory Systems" in 39th international Conference on Very Large Data Bases, Trento, Italy, 2013.

[6] L. L.S.P.A.R.Z.V.A.D.Wang Lam, "Muppet: Map Reduce-Style Processing of Fast Data" in 38th International Conference on Very Large Data Bases, Istanbul, Turkey, 2012.

[7] Emad Soltani Nejad Mohammad Reza Majma "A Modem Method to Improve Efficiency of Hadoop and Map Reduce Cluster Using Software-Defined Networks Technology" in 25th Iranian Conference on Electrical Engineering (ICEE), 2017

[8] Anca Vulpe, "Exploring Scalability in Pattern Finding in Galactic Structure using Map Reduce" in 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, 2016

[9] Gautam shroff, Enterprise Cloud Computing Technology, Architecture, Applications, Cambridge University Press, 2010

[10] S.Rajan, Dr.D.S.Mahendran, Dr.S.John Peter, "A Novel Model for Key Management on Cloud Research Article/Survey Paper/Case Study", IJSRET, ISSN 2278-0882 Volume 6, Issue 9, September 2017.