IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727,
PP 24-27
www.iosrjournals.org

I2mapreduce: Incremental Map reduce For Mining Big Data

S.S.Pophale[1], Shubham S. Lenekar[2], Pooja D. Mache[3], Pradnya N. Husale[4]
Snehal N.Shinde[5]
[1,2,3,4,5]*(Dept of Information Technology, P.D.V.V.P.C.O.E./ SPP University, India)*

Abstract : *Data is continuously arriving by different data generating factors like social network, online shopping, sensors, e-commerce etc. Because of this Big Data the results of data mining applications getting stale and disused over time. Fine-grain Incremental processing is a promising approach to reviving data mining results. I^2MAPREDUCE: Fine-Grain Incremental Processing in big data mining a new incremental processing extension to Map Reduce, the most widely used framework for mining big data. As compare to the high-tech work on In coop System, I^2MapReduce has its own advantages (i) To perform instead of task level re-computation, uses key/value pair level incremental processing (ii) It supports one-step computation along with sophisticated iterative computation, which is extensively used in data mining applications, and (iii) It reduces Input and output overhead for accessing preserved fine-grain computation states by incorporating the set of novel techniques. We evaluate I^2MapReduce: Fine-Grain Incremental processing in mining the big data using a one-step and iterative algorithms with assorted computation characteristics.*

Keywords: *Incremental processing, Map Reduce, iterative computation, big data.*

I. Introduction

Healthcare contain huge amount of data which becomes essential for performance and planning. For every organization data is important for gaining the knowledge, annotation, research. So for healthcare big data is one of the solutions for potential impact. A programming model MapReduce, supporting file system known as Hadoop Distributed File System (HDFS) processes the data and with the help of this we can analyze unstructured data into structured data. The Map-Reduce based approach is used for data cube materialization, mining over huge datasets using non algebraic measures.

Data Provider which will provide a high Dimensional patient data, provider can collect all the data and store it into structured database. Another authorized person in our system who can access the patient data allocated from provider only.

II. Prerequisite

Data Cube: - The multi-dimensional views in data warehousing is provided by data cube model. If n size given in relation, then there are 2^n cuboids and this cuboids computed in the cube materialization using algorithm [2] which is able to support the feature in MapReduce for efficient cube computation.

MapReduce:- MapReduce programming model processes large volumes of data in parallel manner by dividing the work into a set of autonomous tasks. This model explains the nature of programming model and how it can be used to write programs which run in the Hadoop environment.

MR-Cube:- MapReduce based algorithm uses MR-Cube that introduce efficient cube computation [3] and identifies cube sets/groups on non algebraic measures. Complexity of the cubing task depends on two things which are size of data and size of cube lattice.

III. Literature Survey

3.1 MapReduce: Simplified Data Processing on Large Cluster

From the last five years, many authors and others at Google have implemented lots of special purpose computations that processes large amount of data such as web request logs, crawl data, etc. To compute various types of derived data such as various representation of graph structure of web documents, most frequent query in a day, etc. Many computations are straightforward. Most of the time input data is large. This data is distributed across many machines. In this system, design a new abstraction that allows to express the simple computation and trying to perform but hides the unstructured details of parallelism, data distribution and load balancing in library and fault tolerance. This abstraction is inspired by the map and reduce in many functional languages. In this map function key/value pairs are used. Apply reduce operation to all the values that shared same key, in order to combine the derived data appropriately.

3.2 Incoop: MapReduce for Incremental Computation

A computer system produces and collects increasing amounts of data. Services of Internet companies analyzing to improve services. Incoop system is generic framework which is based on Hadoop and use for incremental computation. Incoop can detects changes to the input data and enable the automatic updates of the output by reusing mechanism of fine-grain incremental processing. There are two case studies of higher-level services that are: i) Incremental query ii) Log Processing System without changing a single line of code of application input data it improves the significant performance of results.

3.3 Big Data Mining using Map Reduce

Big data is large amount of data. Big data applications where data collection has grown continuously, it is expensive to capture, extract and manage and process data using existing software tools. For example, Forecasting of weather, Electricity Supply, Social media. With increasing size of data in data warehouse it is expensive to perform data analysis. Data cube commonly abstract and summarize databases. It is way of structuring data in different n dimensions for analysis on some measure of interest. For data processing Big data processing framework relay on cluster computers and parallel execution framework provided by Map-Reduce.

3.4 iMapReduce: A Distributed Computing Framework for Iterative Computation

Relational data pervasive in most of the applications such as a social network analysis and data mining. These relational data containing at least millions and hundreds of relations. This need distributed computing frameworks for processing these data on large cluster. Example of such a framework is MapReduce. This paper presents iMap Reduce, a framework that supports iterative processing. Users are getting allow by specified the iterative operations with map and reduce functions.

IV. Existing System

In the age of “Big Data”, iterative computation takes more time to complete the processing of large amount of data. As novel changes occur in data the previous iterative computation results become decayed and outdated over time. So periodically refreshment of iterative computation is very advantageous. For example, The PageRank algorithm in web search engines, incline descent algorithm for optimization, and many other iterative algorithms for applications including re-commander systems [2] and link prediction [3]. The PageRank algorithm computes ranking scores of web pages for supporting web search based on the web graph structure. However, the web graph structure evolves Web pages, creation, deletion and updation of hyperlinks. As the web graph evolves, the result of Page Ranking algorithm gradually become stale and potentially lowers the quality of web search. Hence, it is necessary to refresh the PageRank computation regularly. Nowadays MapReduce [2] is the most famous platform for big data analysis in the cloud. Numerous previous studies enlarge MapReduce concept for efficient iterative computation [3]. However, our groundwork results have shown that for starting a new iterative computation from scratch will be extremely expensive McSherry. proposed a new computational model for incremental iterative computations based on differential dataflow that is hugely different from the MapReduce programming model. To the best of our knowledge, for popular platform, MapReduce, no solution has so far been demonstrated so as to able to efficiently handle incremental data changes for complex iterative computations. A new MapReduce well-suited model which supports incremental iterative computation is preferred.

4.1 Disadvantages of Existing system:

1. The existing system cannot gives promising output which enough for working in the Big Data.
2. The update of any data will result in re-run the complete setup.
3. It does support only task-level incremental processing.
4. It does support only one-step computation.

4.2 Modules

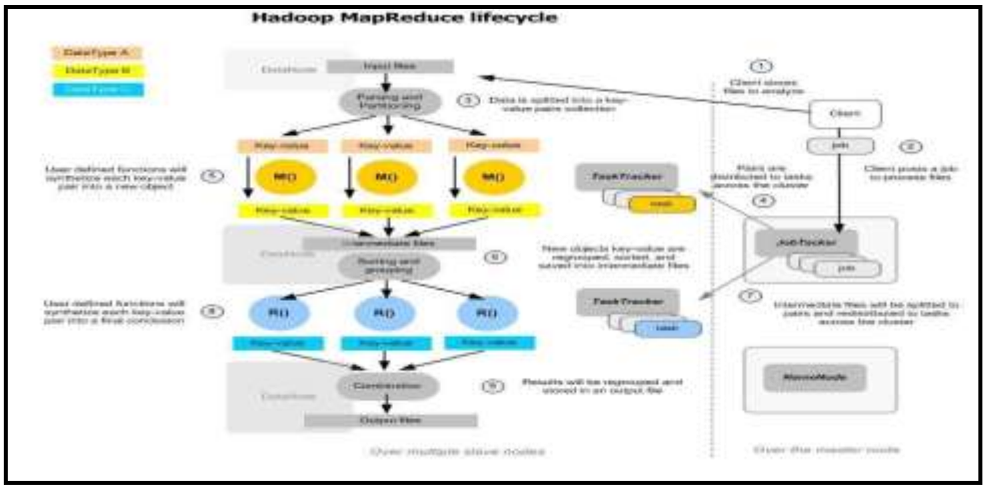
We implement the proposed research work on health care dataset system. The below modules will work in whole system.

1. Data Provider: Which will provide a high Dimensional patient data, provider can collect all the data and store it into structured database.
2. Doctor: Doctor is second module in our system who can access the patient data allocated from provider only. (Same module will work with HDFS architecture with cube query, I2Map reduce, Materialize view etc)
3. Admin: Admin who can access everything from system or he have all credential like create alter delete also.

V. Algorithm

Algorithm: Query Algorithm for MRBG Store Input: Queried key: k1; the list of queried keys: LM
Output: chunk k1

```
1.  if! read cache.contains(k1) then
2.  gap =0, w =0
3.  i=k1's index in M That is, M i = k1
4.  while gap < T and w + gap + length(M i) < readcache:
5.  size do
6.  w =w+gap + length(M i)
7.  gap= pos(M i+1) pos(M i) - length(M i )
8.  i =i + 1
9.  end while
10. starting from pos k1, read w bytes into read cache
11. end if
12. return read cache.getchunk(k1)
```



VI. Conclusion

We have described I^{2}MapReduce- Fine Grain Incremental Processing based on MapReduce framework. That supports k/v pair level fine-grain incremental processing to minimize the amount of re-computation and MRBG-Store to support efficient quires for retrieving fine-grain states for incremental processing and to preserve the fine-grain states in MRBG This framework combines a fine-grain advance engine, a general-purpose iterative model, and a set of effective techniques for fine-grain incremental iterative computation. Real-machine experiments show that I^{2}MapReduce can significantly reduce the run time to refresh big data mining result compared to re-computation on both plain and iterative MapReduce.

References

[1] Y. Zhang, S. Chen, Q. Wang, and G. Yu, "i^{2}mapreduce: Incremental MapReduce for mining evolving big data," CoRR, vol. abs/ 1501.04854, 2015.
[2] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in Proc. 6th Conf. Symp. Opear. Syst. Des. Implementation, 2004, p. 10.
[3] Y. Zhang, Q. Gao, L. Gao, and C. Wang, "imapreduce: A distrib- uted computing framework for iterative computation," J. Grid Comput., vol. 10, no. 1, pp. 47–68, 2012.
[4] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox, "Twister: A runtime for iterative mapreduce," in Proc. 19th ACM Symp. High Performance Distributed Comput., 2010, pp. 810–818.
[5] D. Peng and F. Dabek, "Large-scale incremental processing using distributed transactions and notifications," in Proc. 9th USENIX Conf. Oper. Syst. Des. Implementation, 2010, pp. 1–15.
[6] P. Bhatotia, A. Wieder, R. Rodrigues, U. A. Acar, and R. Pasquin, "Incoop: Mapreduce for incremental computations," in Proc. 2nd ACM Symp. Cloud Comput., 2011, pp. 7:1–7:14.
[7] C. Yan, X. Yang, Z. Yu, M. Li, and X. Li, "IncMR: Incremental data processing based on mapreduce," in Proc. IEEE 5th Int. Conf. Cloud Comput., 2012, pp.pp. 534–541.