www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issue 1 Jan 2016, Page No. 15415-15420

I^{2}mapreduce: Fine-Grain Incremental Processing In Big Data Mining

Miss Mugdha A. Kulkarni [1], Prof I.R.Shaikh [2]
[1]Computer Engineering Department, Savitri Bai Phule University,Pune,India,
mugdhakulkarni966@gmail.com
[2]Computer Engineering Department, Savitri Bai Phule University,Pune,India,
imran.shaikh22@gmail.com

Abstract: I^{2}MAPREDUCE: Fine-Grain Incremental Processing in big data mining a novel incremental processing extension to Map Reduce, the most widely used framework for mining big data. As compare with the high-tech work on In coop, I^{2}MapReduce has its own advantages (i) It prefers key-value pair level incremental processing to perform instead of task level re-computation, (ii) It supports one-step computation along with more sophisticated iterative computation, which is extensively used in data mining applications, and (iii) It reduce I/O overhead for accessing preserved fine-grain computation states by incorporating the set of novel techniques. I evaluate I^{2}MAPREDUCE: Fine-Grain Incremental processing in big data mining using a one-step algorithm and four iterative algorithms with assorted computation characteristics. Experimental results on Amazon EC2 show significant performance improvements of I^{2}MapReduce compared to both plain and iterative MapReduce performing re-computation.

Keywords: Incremental processing, MapReduce, iterative computation, big data .

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 Map Reduce and supporting file system known as Hadoop Distributed File System (HDFS) processes the data and with the help of this i can analyze unstructured data into structured data. The Map-Reduce based approach is used for data cube materialization and mining over massive 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.

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 [5] 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.

Literature survey

I^2MapReduce: Incremental Iterative MapReduce

Cloud intelligence applications often perform iterative computations (e.g., PageRank) on constantly changing data sets (e.g. Web graph). While previous studies extend MapReduce for efficient iterative computations, it is too expensive to perform an entirely new large-scale MapReduce iterative job to timely accommodate new changes to the underlying data sets. In this paper, i propose $I^2$to support incremental iterative computation. I observe that in many cases, the changes impact only a very s- mall fraction of the data sets, and the newly iteratively converged state is quite close to the previously converged state.

Parallel Data Processing with MapReduce

A prominent parallel data processing tool MapReduce is gaining significant momentum from both industry and academia as the volume of data to analyze grows rapidly. While MapReduce is used in many areas where massive data analysis is required, there are still debates on its performance, efficiency per node, and simple abstraction. This survey intends to assist the database and open source communities in understanding various technical aspects of the MapReduce framework. In this survey, i characterize the MapReduce framework and discuss its inherent pros and cons.

MapReduce: Simplified Data Processing on Large Clusters

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model

A Model of Computation for MapReduce

In recent years the MapReduce framework has emerged as one of the most widely used parallel computing platforms for processing data on terabyte and petabyte scales. Used daily at companies such as Yahoo!, Google, Amazon, and Facebook, and adopted more recently by several universities, it allows for easy parallelization of data intensive computations over many machines. One key feature of MapReduce that di_erentiates it from previous models of parallel computation is that it interleaves sequential and parallel computation. I propose a model of effcient computation using the MapReduce paradigm

Big Data Mining using Map Reduce

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

Large-scale data processing using mapreduce in cloud computing environment

The computer industry is being challenged to develop methods and techniques for affordable data processing on large datasets at optimum response times. The technical challenges in dealing with the increasing demand to handle vast quantities of data is daunting and on the rise. One of the recent processing models with a more efficient and intuitive solution to rapidly process large amount of data in parallel is called MapReduce. It is a framework defining a template approach of programming to perform large-scale data computation on clusters of machines in a cloud computing environment. MapReduce provides automatic parallelization and distribution of computation based on several processors. It hides the complexity of writing parallel and distributed programming code. This paper provides a comprehensive systematic review and analysis of large-scale dataset processing and dataset handling challenges and requirements in a cloud computing environment by using the MapReduce framework and its open-source implementation Hadoop.

Matchmaking: A New MapReduce Scheduling Technique

MapReduce is a powerful platform for large-scale data processing. To achieve good performance, a MapReduce scheduler must avoid unnecessary data transmission by enhancing the data locality (placing tasks on nodes that contain their input data). This paper develops a new MapReduce scheduling technique to enhance map task's data locality. I have integrated this technique into Hadoop default FIFO scheduler and Hadoop fair scheduler. i compare MapReduce scheduling algorithms with an existing data locality enhancement technique (i.e., the delay algorithm developed by Facebook).

Twister: A Runtime for Iterative MapReduce

MapReduce programming model has simplified the implementation of many data parallel applications. The simplicity of the programming model and the quality of services provided by many implementations of MapReduce attract a lot of enthusiasm among distributed computing communities. From the years of experience in applying MapReduce to various scientific applications i identified a set of extensions to the programming model and improvements to its architecture that will expand the applicability of MapReduce to more classes of applications. In this paper, i present the programming model and the architecture of Twister an enhanced MapReduce runtime that supports iterative MapReduce computations efficiently.

Existing System:

In the epoch of "big data", iterative computations take more time to complete the processing of large amount of data. As new 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 [10]. the PageRank algorithm computes ranking scores of web pages based on the web graph structure for supporting web search. However, the web graph structure evolves Web pages, creation, deletion and updation of hyperlinks. As the web graph evolves, the result of PageRanking algorithm gradually become stale and potentially lowers the quality of web search. Hence, it is necessary to refresh the PageRank computation regularly. Nowadays MapReduce [6] is the most famous platform for big data analysis in the cloud. Numerous previous studies enlarge MapReduce concept for efficient iterative computation [5, 7, 20]. However, our groundwork results have shown that for starting a new iterative computation from scratch will be extremely expensive McSherry et al. 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

Disadvantages of Existing system:

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

Proposed System:

Iterative processing. A number of distributed frameworks have recently emerged for big data processing. I discuss the frameworks that improve MapReduce. HaLoop, a modified version of Hadoop, improves the efficiency of iterative computation by making the task scheduler loop aware and by employing caching mechanisms. Twister employs a lightweight iterative mapReduce runtime system by logically constructing a Reduce-to-Map loop. I^2 MapReduce [10] supports iterative processing by directly passing the Reduce outputs to Map and by distinguishing variant state data from the static data. I^{2}MapReduce improves upon these previous proposals by supporting an efficient general-purpose iterative model. Unlike the above MapReduce-based systems, Spark uses a new programming model that is optimized for memory- resident read-only objects. Spark will produce a large amount of intermediate data in memory during iterative computation. When input is small, Spark exhibits much better performance than Hadoop because of in-memory processing. However, its performance suffers when input and intermediate data cannot fit into memory.

I experimentally compare Spark and I^{2}MapReduce in, and see that I^{2}MapReduce achieves better performance when input data is large. Pregel follows the Bulk Synchronous Processing (BSP) model. The computation is broken down into a sequence of super steps. In each super step, a Compute function is invoked on each vertex. It communicates with other vertices by sending and receiving messages and performs computation for the current vertex. This model can efficiently support a large number of iterative graph algorithms. Open source implementations of Pregel include Giraph, Hama , and Pregelix. Compared to I^{2}MapReduce, the BSP model in Pregel is quite different from the MapReduce programming paradigm. It would be interesting future work to exploit similar ideas in this paper to support incremental processing in Pregellike systems. Incremental processing for one-step application. Besides Incoop, several recent studies aim at supporting incremental processing for one-step applications. Stateful Bulk Processing addresses the need for stateful dataflow programs It provides a group wise processing operator Translate that takes state as an explicit input to support incremental analysis. But it adopts a new programming model that is very different from MapReduce. In addition, several research studies, support incremental processing by task-level re-computation, but they require users to manipulate the states on their own. In contrast, I^{2}MapReduce exploits a fine-grain kv-pair level re-computation that are more advantageous. Incremental processing for iterative application. Naiad proposes a timely dataflow paradigm that allows stateful computation and arbitrary nested iterations. To support incremental iterative computation, programmers have to completely rewrite their MapReduce programs for Naiad. In comparison, i extend the widely used MapReduce model for incremental iterative computation. Existing MapReduce programs can be slightly changed to run on I^{2}MapReduce for incremental processing.

Modules

I 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.

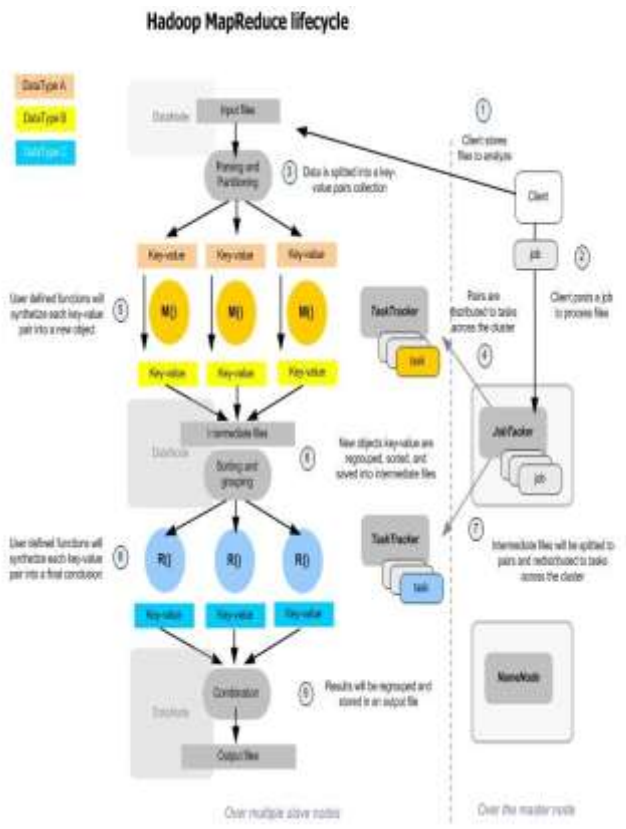
2: Doctor

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, I^{2}Map reduce, Materialize view etc)

3: Admin

Admin who can access everything from system or he have all credential like create alter delete also.

Architecture:


Hadoop MapReduce lifecycle
DataType A
DataType B
DataType C
DataNode
Input files
Parsing and Partitioning
3 Data is splitted into a key-value pairs collection
1 Client stores files to analyze
Client
job
2 Client posts a job to process files
Key-value
M()
5 User defined functions will synthetize each key-value pair into a new object
TaskTracker
task
Pairs are distributed to tasks across the cluster
4
JobTracker
Intermediate files
Sorting and grouping
6 New objects key-value are regrouped, sorted, and saved into intermediate files
R()
8 User defined functions will synthetize each key-value pair into a final conclusion
7 Intermediate files will be splitted to pairs and redistributed to tasks across the cluster
NameNode
Combination
9 Results will be regrouped and stored in an output file
Output files
Over multiple slave nodes
Over the master node

Algorithm 1. Query Algorithm in MRBG-Store
Input: queried key: k; the list of queried keys: L
Output: chunk k

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

CONCLUSION

I have described I^{2}MapReduce- Fine Grain Incremental Processing based on MapReduce framework. That supports kv-pair level fine-grain incremental processing to minimize the amount of re-computation and MRBG-Store to support efficient quires to retrieve fine-grain states for incremental processing and to preserve the fine-grain states in MRBGraph. This framework combines a fine-grain incremental 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 for refreshing big data mining results compared to re-computation on both plain and iterative MapReduce.

REFERENCES

[1] J. Dean and S. Ghemawat, Mapreduce: Simplified data process- ing on large clusters, in Proc. 6th Conf. Symp. Opear. Syst. Des. Implementation, 2004,

[2] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, Resilient distributed datasets: A fault-tolerant abstraction for, in-memory cluster computing, in Proc. 9th USENIX Conf. Netw. Syst. Des. Implementation, 2012.

[3] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski,

Pregel: A system for large-scale graph processing, in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2010,

[4] S. R. Mihaylov, Z. G. Ives, and S. Guha, Rex: Recursive, delta- based data-centric computation, in Proc. VLDB Endowment, 2012,

[5] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, Distributed graphlab: A framework for machine learning and data mining in the cloud, in Proc. VLDB Endowment, 2012,

[6] S. Ewen, K. Tzoumas, M. Kaufmann, and V. Markl, Spinning fast iterative data flows, in Proc. VLDB Endowment, 2012,