Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 12, December 2014, pg.574 – 577

RESEARCH ARTICLE

Improvisation of Incremental Computing In Hadoop Architecture- A Literature

Mr. Alhad V. Alsi[1], Prof. A P.Bodkhe[2]
[1]Prof. Ram Meghe Institute of Technology & Research, India
[2]Prof. Ram Meghe Institute of Technology & Research, India
[1] alhad.v.a@gmail.com; [2] principal@mitra.ac.in

Abstract— Automatic increment in data is a difficult problem, as it requires the development of well-defined algorithms and a runtime system to support performance code. Many online data sets grow incrementally over time as new entries are slowly added and existing entries are deleted or updated to manage the dataset. The Hadoop is a dedicated distributed paradigm used to manipulate the large amount of distributed data. This manipulation contains not only storage but also the computation and processing of the data. Hadoop is normally used for data centric applications. Systems for incremental bulk data processing and computation can efficiently use for the updates but are not compatible with the non-incremental systems such as e.g., MapReduce, and more importantly and requires the programmer to implement application-specific dynamic/ incremental algorithms, ultimately increasing algorithm and code complexity. Thus this paper discusses about the various aspects of the incremental computation.

Keywords— Incremental data computing, Hadoop, MapReduce, parallel processing, distributed data systems.

I. INTRODUCTION

Traditional methods for data mining typically make the assumption that the data is centralized and static. This assumption is no longer valid. Such methods waste processing and input/output (I/O) resources when data is dynamic and they impose excessive communication overhead when data is distributed. The efficient implementation of incremental data mining methods is thus becoming crucial for ensuring system scalability and facilitating knowledge discovery when data is dynamic and distributed.

MapReduce is emerging as an important programming model for data-intensive applications [9]. The model proposed by Google for bulk data processing is very attractive for ad-hoc parallel processing of arbitrary data. It shows good performance for batch parallel data processing. MapReduce enables easy development of scalable parallel applications to process vast amount of data on large clusters of commodity machines [3]. Its popular open-source implementation, Hadoop, developed primarily by Yahoo and Apache, runs jobs on hundreds of terabytes of data. Hadoop is also used at Facebook, Amazon, and Last.fm. Because of its high efficiency, high scalability, and high reliability, MapReduce framework is used in many fields, such as life science computing, text processing, web searching, The work presented in this paper is supported in part by National Science Foundation (grants CCF-1128805, OCI-0904938, and CNS-0709329).graph processing, relational data processing, data mining, machine learning and video analysis. Additionally, the framework is not only used in traditional clusters, multi-core and multi-processor systems and heterogeneous environments, but also used in systems with GPU and FPGA, and mobile systems. Such advantages attract researchers to apply it on incremental data processing. Most studies tend to modify its framework as little as possible to achieve their requirements [10].

A. HDFS

Hadoop in fact have two core components, HDFS (Hadoop Distributed file system) use for storing the data and MapReduce to process the data which is stored on its file system. Hadoop is pretty different from other distributed file systems like it has high fault tolerance and its design in fact made to commodity computers via simple programming model. If some of the machine faces the failure or any problem, backup is available to avoid the pause in service [10].

HDFS is purely a distributed file system provides the high throughput and access the data in efficient manner. It has number of replicas to get data without any issue and quickly return to the user. One of the major objectives to create these replicas is to provide the availability all the time and also if some node is failed, the system should continue its operation.

B. MapReduce and the Incremental Datasets

As computer systems produce and collect elastic datasets, analyzing it becomes an integral part of improving the services provided by Internet companies. In this context, the MapReduce framework offers techniques for convenient distributed processing of dataset by enabling a simple programming model that eliminates the excess load of implementing a complex logic or infrastructure for parallel processing, data transfer, scalability, fault tolerance and scheduling. An important feature of the workloads processed by MapReduce applications is that they are often incremental by nature [3]. MapReduce jobs often run repeatedly with small changes in their input. In MapReduce, if process been mapped and after reduce its output must be saved in local storage or we can say it is just materialized approach [9]. If they are not stored, the result cannot be used further until we save it on disk. For small change MapReduce have to run whole MapReduce application, this is very costly in time and resources [1].

II. LITERATURE SURVEY

Frank McSherry and Rebecca Isaacs reported on the design and implementation of Naiad [6]. It is a set of declarative data-parallel language extensions and an associated run time supporting computation which is efficient and incremental and iterative. This combination is enabled by a new computational model we call differential dataflow. Naiad extends standard batch data-parallel processing models like MapReduce, Hadoop so as to support efficient incremental updates to the inputs in the manner of a stream processing system, while at the same time enabling arbitrarily nested fixed-point iteration. . In this technique incremental computation can be performed using a partial, rather than total, order on time. In their paper, they evaluated a prototype of Naiad that uses shared memory on a single multi-core computer. They applied Naiad to various computations, including several graph algorithms also and observe good scaling properties and efficient incremental re-computation.

Elastic Replica Management System introduces an active/standby storage model which takes advantage of a high performance complex event processing engine to distinguish the real time data types and brings an elastic replication policy for the different types of data [4]. Based on the data access pattern, data in Hadoop can be classified into different types. Hot data - data having a large number of concurrent access and high intensity of access, cold data- unpopular and rarely accessed data, normal data - rest of the data other than hot and cold. ERMS introduces active/standby storage model which classifies the storage nodes into active nodes and stand by nodes. Efficiency depends on a number of threshold values and hence the careful selection of threshold values is needed and also memory requirement is high.

HadoopDB stores data in the local RDBMS's using ACID conforming DBMS engines which will affect the dynamic scheduling and fault tolerance of Hadoop [8]. The HadoopDB changes the interface to SQL and this is why the programming model is not as simple as Map/Reduce. Also changes are required to make Hadoop and Hive frameworks work together in HadoopDB. It can perform like parallel databases along with high fault tolerance, ability to run in heterogeneous environments and software license cost as Hadoop and can reduce the data processing time. Moreover, it can yield scalability, fault tolerance and flexibility of Map/Reduce systems.

Clydesdale is a research prototype built on top of Hadoop for structured data processing. It provides the key feature of MapReduce i.e.; the scalability, fault tolerance and elasticity [5]. Moreover, it enhances the performance without making modifications to the underlying platform. For storing large amounts of dataset which further undergo processing, Clydesdale uses HOFS. Column oriented layout is used to store data and tries to collocate the columns of a given row to the specified node. For the operations in datasets Map tasks are carefully designed. The SQL queries are presented as Map/Reduce programs in Java. The data structures for query processing can be shared by multiple threads and allows multiple tasks to execute successively on any node. Scheduling of Clydesdale workloads along with Map/Reduce loads pose network load management problems and updates to dimension tables are not permitted. Also, Parallelism may be limited for small data sets.

Single Pass Clustering processes the documents one by one [3]. It compares each document to the existing clusters. A feed similarity threshold value is used and if the similarity between the document and any cluster is above the similarity threshold, it will add the document to the same cluster. If not, it will form the new cluster.

Matthew Hayes (LinkedIn) and Sam Shah (LinkedIn) introduced Hourglass, a library for developing incremental monoid computations on Hadoop [11]. Hourglass runs on unmodified Hadoop. The framework ensures that only the necessary sub computations are performed. It is successfully used at one of the largest online social networks LinkedIn, for many use cases in dash boarding and machine learning. In this work, they described Hourglass, LinkedIn's incremental processing library for Hadoop. An accumulator-based interface for programmers to store is provided by library and use state across successive runs. This way, only the necessary sub computations need to be performed and incremental state management and its complexity is hidden from the programmer. Using public datasets and internal workloads at LinkedIn, they showed that Hourglass yields a 50–98% reduction in total task time and a 25–50% reduction in wall clock time compared to non-incremental implementations. And the most important thing about Hourglass is, it is an open source.

Cairong Yan proposed the IncMR framework to deal with incrementally processing elastic dataset, which takes state as implicit input and combines it with new data which has been added later [12]. Map tasks are created according to new splits instead of entire splits in existing dataset while reduce tasks fetch their inputs including the states of Map task which saved and the intermediate results of new map tasks from designate nodes or local nodes. Data locality is the important optimization mean for scheduling the iterative jobs. It is implemented based on Hadoop. It is compatible with the original MapReduce interfaces and transparent to users. Moreover, while processing the datasets the non-iterative algorithms of MapReduce framework can also be used in IncMR directly to get efficient incremental and continuous processing without any modification. IncMR has many overcome many lacunas of previous studies and in all studied cases runs faster than that processing the entire data set.

Pramod Bhatotia and Alexander Wiederwe described the architecture, implementation of a generic MapReduce framework, named Incoop, for incremental processing. It identifies the changes in the input data and enables the automatic modifications of the outputs by employing a fine-grained and efficient result re-use mechanism [13]. The efficiency is achieved by adopting recent advances in the area of programming languages to identify systematically the shortcomings of task-level memorization approaches, and address them using several novel techniques such as a storage system to store the input of consecutive iterations, a contraction phase that make the incremental computation of the reduce tasks more efficient and the scheduling algorithm for Hadoop that is aware of the location of previously computed results. Incoop was implemented by extending the Hadoop and further it was evaluated with a variety of applications, including case studies of higher-level services: incremental query (based on Pig) and log processing systems. The results showed significant performance improvements without changing a single line of application code.

III. PROPOSED WORK

Hadoop is being used by every big organization in this world and also using in cloud computing but still there are some issues or shortcomings in Hadoop.

- Low level programming paradigm and schema
- Strictly batch processing
- Time Skew
- Incremental Computation

The proposed system is focused on one of Hadoop's major shortcomings e.g. incremental computations [1]. In this, map and reduce processes need to be run again every time even if there is some minute change in input. There is a requirement of some caching at small level only. But it will cache only that user's data or that block's data whose storage is too much or it's incoming and outgoing links are very much. This can be decided when the map function are going to have input via job tracker. Once caching would be done for the data that has big blocks upon a small change we can stop the map and same reduce job any time. This will surely save the time and resources very well.

IV. CONCLUSIONS

The proposed paper addresses the different issues of the incremental data and computational techniques and/or methodologies which have been explored so far. A new approach to incremental data computing is proposed for HDFS which will cluster similar documents in the same set of data nodes with minimal changes to the existing framework. When the systems are distributed, there is a need of iterative computation. Thus, various techniques of iterative computations have been mentioned in this paper. The frameworks like Incoop, Hourglass introduced their approach for the incremental computation. The computations become efficient if the states of the previous Map functions are available. Thus, to solve the issue of incremental computation these different frameworks have been introduced. Also the proposed system address about the caching technique which will be focusing on caching the data to temporary buffer.

ACKNOWLEDGEMENT

I extend my sincere thanks to Prof. A. P. Bodkhe, Principal, Prof. Ram Meghe Institute of Technology & Research ,Badnera India, for his valuable support and guidance in pursuing my research work. My thanks are also to Prof. Ms. V. M. Deshmukh, Head of the department, Prof. Ram Meghe Institute of Technology & Research, Badnera India for her encouragement and continuous supervision.

REFERENCES

[1] Anam Alam and Jamil Ahmed."Hadoop Architecture and Its Issues".2014. International Conference on Computational Science and Computational Intelligence.2014

[2] Yanfeng Zhang,Qixin Gao,Lixin Gao,Cuirong Wang."iMapReduce: A Distributed Computing Framework for Iterative Computation".Springer.2012.

[3] Jeffrey Dean,Sanjay Ghemawat."MapReduce: Simplified Data Processing on Large Clusters".Communications of the ACM - 50th anniversary issue: 1958 - 2008 CACM Homepage archive Volume 51 Issue 1, January 2008.Pages 107-113

[4] Zhendong Cheng, Zhongzhi Luan,You Meng, Yijing Xu, Depei Qian."ERMS : An Elastic Replication Management System for HDFS".IEEE International Conference on Cluster Computing Workshops.2012

[5] Tim Kaldewey,Eugene J. Shekita,Sandeep Tata."Clydesdale: Structured Data Processing on MapReduce".EDBT '12 Proceedings of the 15th International Conference on Extending Database Technology.2012

[6] Frank McSherry Rebecca Isaacs Michael Isard Derek G. Murray."Composable Incremental and Iterative Data-Parallel Computation with Naiad". Microsoft Research Paper,2012.

[7] Matthew Eric Otey, Srinivasan Parthasarathy."Parallel and Distributed Methods for Incremental Frequent Itemset Mining". IEEE Transactions on systems,man and cybernetics-Part B: Cybernetics,Vol.34,No 6. December 2004

[8] Azza Abouzeid, Kamil Bajda-Pawlikowski,Daniel Abadi, Avi Silberschatz, Alexander Rasin."HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads ".Journal Proceedings of the VLDB Endowment VLDB Endowment Hompage archive Volume 2 Issue 1. Pages 922-933. August 2009

[9] MapReduce Design Patterns by Donald Miner and Adam Shook

[10] Hadoop Cluster Deployment by Danil Zburivsky

[11] Matthew Hayes,Sam Shah."Hourglass: a Library for Incremental Processing on Hadoop".2013 IEEE International Conference on Big Data.2013

[12] Cairong Yan,Xin Yang, Ze Yu, Min Li, Xiaolin Li."IncMR: Incremental Data Processing based on MapReduce".2012 IEEE Fifth International Conference on Cloud Computing.2012

[13] Pramod Bhatotia, Alexander Wieder, Rodrigo Rodrigues, Umut A. Acar, Rafael Pasquini."Incoop: MapReduce for Incremental Computations". Proceedings of the 2nd ACM Symposium on Cloud Computing Article No. 7 ACM New York, NY, USA .2011

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