International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 06 | June-2016 www.irjet.net p-ISSN: 2395-0072

Design an Acceleration Framework That Optimizes Hadoop

Madhav D.Ingale[1], Vaibhav Pathare[2], Ankita Nirmal[3] , Diksha Wadkar[4]

Assistant Professor, Dept. of Computer Engineering, JSCOE, Hadapsar, Pune, India1

UG Student, Dept. of Computer Engineering, JSCOE, Hadapsar, Pune, India2,3,4

Abstract - *Big data is a collection of large amount of data which cannot be processed using traditional computing techniques. Big data contains data produced from Black box data, social media data, transport data. It also includes huge volume, high velocity and extensible data of the following types as structured, semi structured and unstructured data. Hadoop technology is a popular open source code of the MapReduce programming model for cloud computing. It faces a multiple of issues to achieve the best performance from the particular systems. These include a serialization barrier that delays the reduce phase, cyclic merges, and disk accesses, and the lack of portability to different interconnects. To keep up with the increasing volume of data sets, Hadoop also requires efficient I/O capability from the underlying computer systems to process and analyse data. We describe Hadoop-A, an acceleration framework that optimizes Hadoop with plug-in components for fast data movement, overcoming the existing limitations. A novel network-levitated merge algorithm is introduced to merge data without repetition and disk access. In addition, a full pipeline is designed to overlap the shuffle, merge, and reduce phases. Our experimental results show that Hadoop-A significantly speeds up data movement in MapReduce and doubles the throughput of Hadoop. In addition, Hadoop-A significantly reduces disk accesses caused by intermediate data.*

Key Words: Big Data, Hadoop, HDFS, MapReduce, Hadoop acceleration, CPU Usage.

1.INTRODUCTION

In Hadoop using MapReduce emerged as a popular and easy-to-use programming model for cloud computing. It has been used by multiple organizations to process explosive amounts of data, perform difficult computation, and extract critical knowledge for organisation intelligence. Hadoop implements MapReduce framework with two categories of components: 1) JobTracker and 2) Task-Trackers. The JobTracker commands TaskTrackers (slaves) to process data in parallel through two main functions: map and reduce. In this process, the JobTracker is in charge of scheduling map tasks (MapTasks) and reduce tasks (ReduceTasks) to TaskTrackers. It also monitors their progress, collects runtime execution statistics, and handles possible faults and errors through task re-execution. Between the two phases, a ReduceTask needs to fetch a part of the intermediate output from all finished MapTasks. Globally, this leads to the shuffling of intermediate data (in segments) from all MapTasks to all ReduceTasks. For many data-intensive MapReduce programs, data shuffling can lead to a significant number of disk operations, contending for the limited I/O bandwidth. This presents a severe problem of disk I/O contention in MapReduce programs, which entails further research on efficient data shuffling and merging algorithms. It remains as a critical issue to examine the relationship of Hadoop MapReduce's three data processing phases, i.e., shuffle, merge, and reduce and their implication to the efficiency of Hadoop.

2.RELATED WORK

Hadoop is a popular open source implementation of the MapReduce programming model for cloud computing. MapReduce implementation enables a convenient and easy-to-use data processing framework.

In today's technical era handling the big data is major issue. The many Authors are working on the different techniques and algorithm for handling the big data. The author Paolo Costa, et al. built Camdoop technique. Social media like Facebook , Google, Microsoft etc. generates the huge amount of data which is responsible for high network traffic and difficult to support using traditional system. Thus, Camdoop technique is used to reduce network traffic and it also provides the high performance [1].

To simplify fault tolerance in map-reduce programming model to actualize the output of each map and reduce the task before it used. The authors Tyson Condie, et .al proposed modified map-reduce architecture data to be pipeline between operator where intermediate data is pipelined. There are three techniques introduced by author Online Aggregation, HOP(Hadoop Online Prototype),Pipeline. 1.Online Aggregation:-As soon as mapper produces the data which has been immediately processed by reducer. Thus task can be completed in efficient time.2.HOP(Hadoop Online Prototype):-It is used to support continuous queries that map-reduce job run continuously, accept the new data when it arrives and analyze it. 3.Pipeline:-It increases the chances of of parallelism, it improves utilization and reduces the response time [3].

Current map-reduce system requires dataset to be loaded into cluster before running queries because of that there is issue arise like high delay to start query processing. Thus authors Edward Mazur, Boduo Li introduced one pass analytic technique. This technique is used to reduce the delay in query processing [5]. Map-reduce based system is slower

than Parallel Database system in performing variety of tasks. Because there is performance gap between the Map-reduce and Parallel Database system. For achieving scalability and flexibility in Map-reduce system authors Dawei Jiang, et .al designed a system such that it is independent of storage system and it analyzes various kind of data like structured and unstructured. And it improves the performance of map-reduce system [6].

In traditional system to run single program in map-reduce framework the system administrator need to set number of running parameter. The author Shivnath Babu, make a case for technique to automate setting of tuning parameter of map-reduce programming which used to provide ad-hoc map-reduce program on large amount of data [7]. Parallel programming techniques like message passing, shared memory, it is difficult to write correct and scalable parallel code .The author Colby Ranger, et .al implemented phoenix technology it automatically manages thread creation, data partitioning, and fault tolerance across processor nodes. Phoenix technology used to increase the performance of multicore system and recover the errors [8].

Authors Jiuxin, et .al proposed design of MPI over infini-band which aquires the benefit of RDMA to large messages as well as small and control messages which improves the better scalability by combining RDMA operations like send or receive. RDMA based design used to reduce latency by 24% and increases the bandwidth by over 104% also reduce the overhead up-to 22% [9].In past decades author implemented some special purpose computation that process large amount of raw data like crawled documents, web request logs etc. for computing various kind of derived data but this computations are straightforward which cannot handle issues like how to parallelize the computation ,distribute the data and handle computations with simple computation on large amount of data. Jeffery Dean, et .al who propose a new approach which used for map reduce programming model where map function process key value pairs and generate intermediate key value pairs and reduce function merges all values associated with intermediate key [10].

Our characterization and analysis reveal a number of issues, including:

1)The serialization between Hadoop shuffle/merge and reduce phases.

2)Repetitive merges and disk access,

3)The lack of portability to different interconnects.

3.PROPOSED METHODOLOGY AND DISCUSSION

1. Network-levitated merge

Hadoop resorts to repetitive merges because of limited memory compared to the size of data. For each remotely completed MOF, each ReduceTask invokes an HTTP GET request to query the partition length, pull the entire data, and store locally in memory or on disk. This incurs many memory loads/stores and/or disk I/O operations.We design an algorithm that can merge all data partitions exactly once and, at the same time, stay levitated above local disks shows our network-levitated merge algorithm. The key idea is to leave data on remote disks until it is time to merge the intended data records.

2.CPU Usage

1. System A and System B calculate the cpu usage.
2. System A have dataset for processing.
3. System A request to system B for cpu usage.
4. After receiving tha cpu usage it compare its own cpu usage and System B cpu usage.
5. According to that cpu usage its create the chunk.
6. Upload data on HDFS of corresponding System.
Process that data.
7. After processing that data System A download the data from system B.

4. SYSTEM ARCHITECTURE

Two new user configurable plug-in components, MOFSupplier and Net- Merger, are introduced to leverage RDMA-capable interconnects and enable alternative data merge algorithms. Both MOFSupplier and NetMerger are threaded C implementations. A primary requirement of Hadoop-A is to maintain the same programming and control interfaces for users. To this end, we design the MOFSupplier and NetMerger plugins as native C programs that can be launched by TaskTrackers.

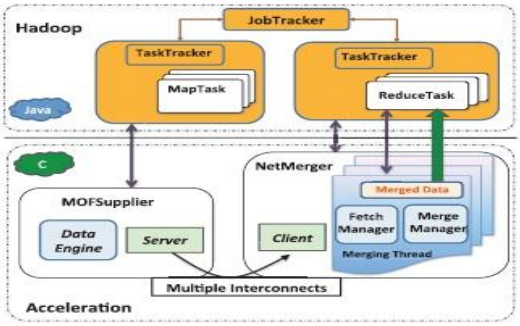

Fig-: Hadoop Architecture

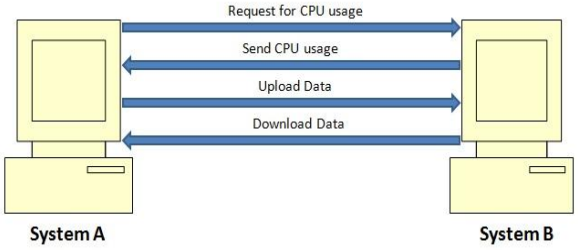

Fig-: CPU usage(Multi-node)

Task tracker accepts the tasks like Map, Reduce, Shuffle from job tracker. Job Tracker is used to keep the track of which Map-Reduce jobs are executing, schedules Maps, Reduces operation tasks to specific machine, monitors the success and failures of these tasks After finishing the tasks it notifies to jobtracker. Hadoop programs can run without any change when the Hadoop-A plug-in is activated.

5.Experimental results

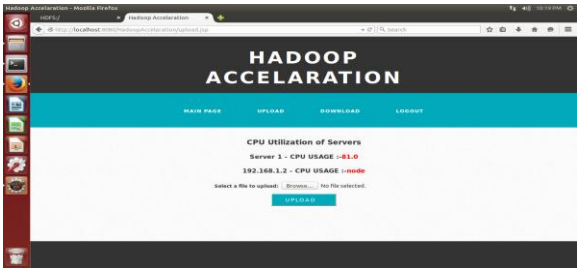

Fig-:CPU usage

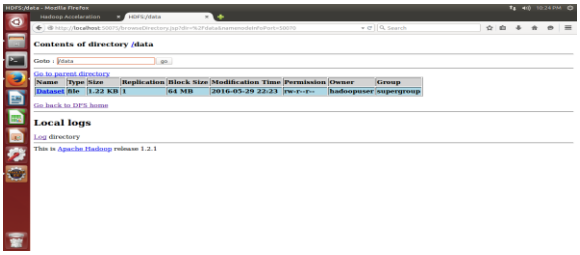

Fig-:Content of Directory.

CONCLUSIONS

In proposed system, we first calculate the cpu usage of the system. Depend on that cpu usage we distribute the dataset. The distributed dataset loaded on the corresponding HDFS system. So system is ready to process dataset. After processing dataset we can download that. So we distribute dataset depend on cpu usage of system, so performance of system is increases.

REFERENCES

[1] [1] P. Costa, A. Donnelly, A. Rowstron, and G. O'Shea, "Camdoop: Exploiting in-Network Aggregation for Big Data Applications," Proc. Ninth USENIX Conf. Networked Systems Design and Implementation (NSDI '12), p. 3, 2012.

[2] [2] X. Que, Y. Wang, C. Xu, and W. Yu, "Hierarchical Merge for Scalable MapReduce," Proc. Workshop Management of Big Data Systems (MBDS '12), pp. 1-6, 2012.

[3] [3] T. Condie, N. Conway, P. Alvaro, J.M. Hellerstein, K. Elmeleegy,and R. Sears, "MapReduce Online," Proc. Seventh USENIX Symp.Networked Systems Design and Implementation (NSDI), pp. 312-328,Apr. 2010.

[4] [4] Vaibhav Dhore and Sonali R. Jagtap, A Survey on Hierarchical Merge for Hadoop-A, International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064 Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

[5] [5] B. Li, E. Mazur, Y. Diao, A. McGregor, and P. Shenoy, "A Platform for Scalable One-Pass Analytics Using MapReduce," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '11), pp. 985- 996, 2011.

[6] [6] D. Jiang, B.C. Ooi, L. Shi, and S. Wu, "The Performance of MapReduce: An In-Depth Study," Proc. VLDB Endowment, vol. 3,no. 1, pp. 472-483, 2010.

[7] [7] S. Babu, "Towards Automatic Optimization of MapReduce Programs," Proc. First ACM Symp. Cloud Computing (SoCC '10), pp. 137-142, 2010.

[8] [8] C. Ranger, R. Raghuraman, A. Penmetsa, G.R. Bradski, and C. Kozyrakis, "Evaluating MapReduce for Multi-Core and Multiprocessor Systems," Proc. IEEE 13th Int'l Symp. High Performance Computer Architecture (HPCA '07), pp. 13-24, 2007.

[9] [9] J. Liu, J. Wu, and D.K. Panda, "High Performance RDMA-Based MPI Implementation over InfiniBand," Int'l J. Parallel Programming, vol. 32, pp. 167-198, 2004.

[10] [10] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Proc. Sixth Symp. Operating System Design and Implementation (OSDI '04), pp. 137-150, Dec. 2004.

[11] [11] http://www.tutorialspoint.com/hadoop/index.htm

[12] [12] Apache Hadoop Project, http://hadoop.apache.org/, 2013.