|| Volume 3 || Issue 7 || July 2018 || ISSN (Online) 2456-0774
INTERNATIONAL JOURNAL OF ADVANCE SCIENTIFIC RESEARCH
AND ENGINEERING TRENDS

Modeling the Performance of Hadoop MapReduce

Sneha Shegar[1], Prof. Kore K. S.[2]
Student, ME Computer, SPCOE, Department Of Computer Engineering, Otur[1]
Assistant Professor, SPCOE, Department Of Computer Engineering, Otur[2]

Abstract— **In recent years the amount of data stored worldwide has exploded, increasing by a factor of nine in the last five years. However, the quantity of data is often far too large to store and analyse in traditional relational database systems, or the data are in unstructured forms unsuitable for structured schemas, or the hardware needed for conventional analysis is just too costly. An accurate performance model for MapReduce is increasingly important for analysing and optimizing MapReduce jobs. It is also a precondition to implement cost based scheduling strategies or to translate Hive like query jobs into sets of low cost MapReduce jobs. This paper presents a Hadoop job performance model that accurately estimates job completion time and further provisions the required amount of resources for a job to be completed within a deadline. Propose system introduces a Hadoop work execution demonstrate that precisely gauges work consummation instance and additional arrangements the required measure of assets for a vocation to be finished inside a due date. The propose scheme expands on authentic employment implementation report and utilizes Locally Weighted Linear Regression System to guess the implementation time of a trade. Besides, it utilizes Lagrange Multipliers system for asset provisioning to fulfil occupations with due date prerequisites. The propose scheme is at first assessed on an in-house Hadoop bunch and therefore assessed in the Amazon EC2 Cloud.**

Keywords: *Map Reduce; Locally Weighted Linear Regression System (LWLRS); Hadoop.*

I INTRODUCTION

Hadoop evolved as a distributed software platform for managing and transforming large quantities of data, and has grown to be one of the most popular tools to meet many of the above needs in a cost -effective manner . By abstracting away many of the high availability (HA) and distributed programming issues, Hadoop allows developers to focus on higher level algorithms. Hadoop is designed to run on a large cluster of commodity servers and to scale to hundreds or thousands of nodes. Each disk, server, network link, and even rack within the cluster is assumed to be unreliable. This assumption allows the use of the least expensive cluster components consistent with delivering sufficient performance, including the use of unprotected local storage (JBODs).As the amount of data generated by enterprises and organizations has exploded, conventional warehouse systems are unable to efficiently store and analyse the big data. MapReduce distributed and parallel programming model on clusters of commodity hardware, has emerged as the de facto standard for processing a large set of unstructured data. Since big data analytics requires distributed and parallel computing at scale, usually involving hundreds to thousands of machines, access to such facilities becomes a significant barrier to practising big data processing in small businesses. Second, coupling computation and storage leads to inefficient use of resources in virtualized environments. To exploit the parallelism of multicore processors, usually multiple Hadoop nodes are consolidated onto one physical machine. On the other hand, collocated virtual nodes often access the disk concurrently, causing interleaved IOs on the host machine. These almost random IO operations can significantly affect the performance of Hadoop jobs that are bottlenecked by disk accesses. There is existing work focusing on exploiting the additional layer of data locality, i.e., host-local, in virtual Hadoop clusters [1],[11]. Data stored on co-located Hadoop nodes is considered local on the same host. Thus, Hadoop scheduler is modified to launch "local" task even its data is on a different virtual node. However, this approach does not address the inefficient IO accesses.

A high accurate performance model for MapReduce is needed, in order to optimize and dynamic adjust job parameters (such as number of Reduce tasks, Buffer size, etc.), to shorten the job execute time, to implement more efficient scheduling strategies, and to compile Hive[8] like query jobs into sets of low cost MapReduce jobs.

Traditional Approaches:

- In existing system Hadoop MapReduce is its support of public cloud computing that enables the organizations to utilize cloud services in a pay-as-you-go manner [2]. This facility is beneficial to small and medium size organizations where the setup of a large scale and complex private cloud is not feasible due to financial constraints.

- Hence, executing Hadoop MapReduce applications in a cloud environment for big data analytics has become a realistic option for both the industrial practitioners and academic researchers. For example, Amazon has designed Elastic MapReduce (EMR) that enables users to run Hadoop applications across its Elastic Cloud Computing (EC2) nodes.

Drawbacks of Existing System:

1. The data volume of RDF closure is ordinarily larger than original RDF data.

2. The storage of RDF closure is thus not a small amount and the query on it takes nontrivial time.
3. The data volume increases and the ontology base is updated, these methods require the recomputation of the entire RDF closure every time when new data arrive.
4. Which takes much time (usually several hours or even days for large) and space (generally the ontology size is more than original data size).

II RELATED WORK

Due to the multisource, massive, heterogeneous, and dynamic characteristics of application data involved in a distributed environment, one of the most important characteristics of Big Data is to carry out computing on the petabyte (PB), even the exabyte (EB)-level data with a complex computing process. Therefore, utilizing a parallel computing infrastructure, its corresponding programming language support, and software models to efficiently analyze and mine the distributed data are the critical goals for Big Data processing to change from "quantity" to "quality."Currently, Big Data processing mainly depends on parallel programming models like MapReduce, as well as providing a cloud computing platform of Big Data services for the public. MapReduce is a batch-oriented parallel computing model [3]. There is still a certain gap in performance with relational databases. Improving the performance of MapReduce and enhancing the real-time nature of large-scale data processing have received a significant amount of attention, with MapReduce parallel programming being applied to many machine learning and data mining algorithms. Data mining algorithms usually need to scan through the training data for obtaining the statistics to solve or optimize model parameters. It calls for intensive computing to access the large-scale data frequently.

To improve the efficiency of algorithms, proposed a general-purpose parallel programming method, which is applicable to a large number of machine learning algorithms based on the simple MapReduce programming model on multicore processors. Ten classical data mining algorithms are realized in the framework, including locally weighted linear regression, k-Means, logistic regression, naive Bayes, linear support vector machines, the independent variable analysis, Gaussian discriminant analysis, expectation maximization, and back-propagation neural networks [4]. With the analysis of these classical machine learning algorithms, we argue that the computational operations in the algorithm learning process could be transformed into a summation operation on a number of training data sets. Summation operations could be performed on different subsets independently and achieve penalization executed easily on the MapReduce programming platform. Therefore, a large-scale data set could be divided into several subsets and assigned to multiple Mapper nodes. Then, various summation operations could be performed on the Mapper nodes to collect intermediate results. Finally, learning algorithms are executed in parallel through merging summation on Reduce nodes. R. Liammel et al. [8] proposed a MapReduce-based application programming interface Phoenix, which supports parallel programming in the environment of multicore and multiprocessor systems, and realized three data mining algorithms including k-Means, principal component analysis, and linear regression.

Herodotos [5] conducted a study of the integration of R (open source statistical analysis software) and Hadoop. The in-depth integration pushes data computation to parallel processing, which enables powerful deep analysis capabilities for Hadoop. Z. Zhang [6] achieved the integration of Weka (an open-source machine learning and data mining software tool) and MapReduce. Standard Weka tools can only run on a single machine, with a limitation of 1-GB memory. After algorithm parallelization, Weka breaks through the limitations and improves performance by taking the advantage of parallel computing to handle more than 100-GB data on MapReduce clusters. J. Dean et al. [7] proposed Hadoop-ML, on which developers can easily build task-parallel or data-parallel machine learning and data mining algorithms on program blocks under the language runtime environment.

B. Hew et al. [9] improved the MapReduce's implementation mechanism in Hadoop, evaluated the algorithms' performance of single-pass learning, iterative learning, and query-based learning in the MapReduce framework, studied data sharing between computing nodes involved in parallel learning algorithms, distributed data storage, and then showed that the MapReduce mechanisms suitable for large-scale datamining by testing series of standard data mining tasks on medium-size clusters. H. Wang [10] proposed a distributed collaborative aggregation (DisCo) framework using practical distributed data preprocessing and collaborative aggregation techniques. The implementation on Hadoop in an open source MapReduce project showed that DisCo has perfect scalability and can process and analyze massive data sets (with hundreds of GB). To improve the weak scalability of traditional analysis software and poor analysis capabilities of Hadoop systems.

III PROBLEM STATEMENT

Numerous associations are constantly gathering monstrous measures of datasets from different sources, for example, the World Wide Web, sensor networks and social networks. The capacity to perform adaptable and auspicious examination on these unstructured datasets is a high need task for some endeavours. It has gotten to be troublesome for traditional network storage and database frameworks to handle

these continuously developing datasets. Proposed system present a Hadoop work execution demonstrate that precisely gauges work consummation time and further arrangements the required measure of assets for a vocation to be finished inside a due date.

IV PROPOSED WORK

In Proposed System we present improved HP model for Hadoop job execution estimation and resource provisioning. The improved HP work mathematically models all the three core phases of a Hadoop job.

In contrast, the HP work does not mathematically model the non-overlapping shuffle phase in the first wave. The improved HP model employs Locally Weighted Linear Regression (LWLR) technique to estimate the execution time of a Hadoop job with the varied number of reduce tasks.

In contrast, the HP model employs a simple linear regress technique for job execution estimation which restricts to a constant number of reduce tasks. Based on the job execution estimation, the improved HP model employs Langrage Multiplier technique to provision the amount of resources for Hadoop job to complete within a given deadline.

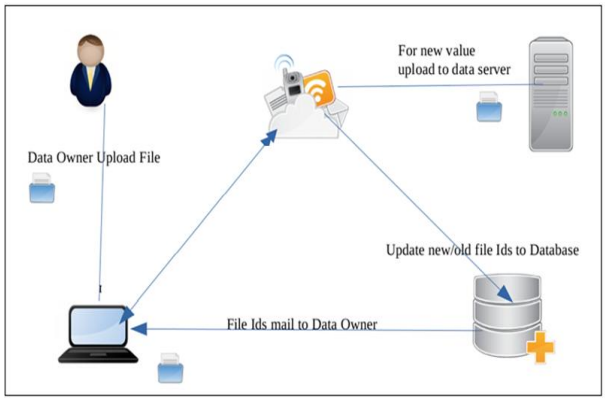

Figure 1. System Architecture

V PROPOSED ALGORITHM

Algorithm 1: Resource Allocation Algorithm:

Input: Job profile of *J*

- (N_M^J, N_R^J) ← Number of map and reduce tasks of *J*
- (S_M, S_R) ← Total number of map and reduce slots in the cluster *T* ←
- Deadline by which job must be completed

Output: *P* ← Set of

Steps:

- **Step 1:** Plausible resource allocations (S_M^J, S_R^J)
- **Step 2:** S^j_M←Min(N^j_M, S_M) to 1 do
- **Step 3:** for
- **Step 4:** Aj/ S^j_M + Bj/ S^j_M = T – C_j^{low} for S_R^J
- **Step 5:** Solve the equation
- **Step 6:** if $0 < S_R^J \leq S_R$ then
- **Step 7:** else

// Job cannot be completed within deadline T // with the allocated map slots

- **Step 8:** Break out of the loop end
- **Step 9:** end if
- **Step 10:** end for

VI CONCLUSION

Hadoop provides an open-source implementation of the MapReduce framework. But its design poses challenges to attain the best performance in the virtualized environment due to the coupled computation nodes and storage nodes. It results in inefficient I/O operations and VM scheduling in virtualized Hadoop clusters. The proposed model builds on historical job execution records and employs Locally Weighted Linear Regression technique to estimate the execution time of a job. The proposed model employed LWLR to estimates execution duration of a job that takes into account a varied number of reduce tasks.

Furthermore, it employs Lagrange Multipliers technique for resource provisioning to satisfy jobs with deadline requirements. The performance of the improved HP model was intensively evaluated on both an in-house Hadoop cluster and on the EC2 Cloud.

ACKNOWLEDGEMENT

I express my sincere thanks to my project guide Prof. K. S. Kore who always being with presence & constant, constructive criticism to made this paper. I would also like to thank all the staff of Computer Department for their valuable guidance, suggestion and support through the project work, who has given co-operation for the project with personal attention. At the last I thankful to my friends, colleagues for the inspirational help provided to me through a paper work.

REFERENCES

[1] Mukhtaj Khan, Yong Jin, Maozhen Li, Yang Xiang, Changjun Jiang, "Hadoop Performance Modelling for Job Estimation and Resource Provisioning," *IEEE Transaction on Parallel and Distributed Systems*, Volume: 27, Issue: 2, Feb. 1, 2016.

[2] Xuelian Lin, zide Meng, Chuan Xu, Meng Wang, "A Practical Performance Model for Hadoop MapReduce", In : *IEEE International Conference on Cluster Computing*, 2012.

[3] Keke chen, James Powers, Shumin Guo, Fengguang Tian, "CRESP: towards Optimal Resource Provisioning for MapReduce Computing in Public Clouds", In : *IEEE Transaction on Parallel and Distributed Systems*, 2013.

[4] Xiaolong Cui, Xuelian Lin, chunming Hu, "Modeling the Performance of MapReduce under Resource Contentions and Task Failures", In : *IEEE*, 2013.

[5] Herodotos Herodotou, Harold Lim, Gang Luo, Nedyalko Borisov, "Starfish: A Self-tuning System for Big Data Analytics", In : *CIDR*, January 2011.

[6] Z. Zhang, L. Cherkasova, and B. T. Loo, "Performance Modeling of MapReduce Jobs in Heterogeneous Cloud Environments," In : *Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing*, 2013, pp. 839.

[7] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun. ACM, vol. 51, no. 1, pp*. 107–113, 2008.

[8] R. Lämmel, "Google's MapReduce programming model — Revisited," *Sci. Comput.Program.vol. 70, no. 1, pp*. 1–30, 2008.

[9] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang, "Mars: a MapReduce framework on graphics processors," *in Proceedings of the 17th international conference on Parallel architectures and compilation techniques – PACT* '08, 2008, *p*. 260.

[10]. H. Yang, Z. Luan, W. Li, D. Qian, and G. Guan, "Statistics-based workload modeling for MapReduce," *in Proc. Parallel Distrib. Process. Symp. Workshops PhD Forum*, 2012, pp. 2043–2015.

[11]. Apache Hadoop. [Online]. Available: http://hadoop.apache.org/. [Accessed: 21Oct- 2013].