International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Makespan Map Reduce Architecture for Efficient Memory Utilization

Archana Bhaskar, Rajeev Ranjan

Abstract: ***Makespan is referred to the total execution time taken to process the tasks or the jobs to the completion time.The High performance infrastructure in cloud computing provides extensive applications. These applications are preferred in Big Data. The existing Hadoop Map Reduce network incurs the input output memory overhead. The parallel Map Reduce network provides a parallel scheme to reduce makespan times in computing environments. The outcome provides improvement in the coefficient correlation and makespan time. The various challenges in computing dataset is handling large dataset efficiently and providing large amount of datasets with ease. The comprehensive method is to enhance data analyzation techniques. Massive large amount of data which are spread across the large number of machines needs to be parallelized.***

Index Terms: Big data, Bioinformatics, Caching, Cloud computing, Hadoop, MapReduce

I. INTRODUCTION

Cloud computing provides the various computing efforts for both scientific and large intensive applications. It provides distributed architecture to process large amount of data. The amount of data generated from various social network , sensors , bio informatics etc. The processing of the large unorganized data becomes important to process across the large organizations. The various existing models such as pyad and phoenix , mars , dryad and spark are not much effective in performing the real time analysis of streaming of data. Hadoop map reduce is most commonly and widely used and adapted framework due to the very open source nature of deploying and scalability. Moreover, in recent time, cloud computing environment has taken drastic growth due to large progress in internet usage. Therefore, in the early years of 21st century, Google has launched some new DBMS technologies such as GFS, Bigtable, and MapReduce to handle huge datasets [1-3]. Various institutions and industries have conveyed their interest in these techniques.

Manuscript published on 30 September 2019
* Correspondence Author
Archana Bhaskar*, Department of Computer Applications, Acharya Institute of Technology,Bangalore,Karnataka
Dr.Rajeev Raman, Department of Computer Applications, Reva University,Bangalore,Karnataka

Similarly, one more technique is launched by Apache Foundation which contains an open source distributed environment named as Hadoop. Since its launch, Hadoop has taken enormous growth which consist a MapReduce processing model to handle large amount of datasets.

Hadoop is a well-known technique which is highly established and widely utilized by various industries like Facebook, Yahoo and for academic research etc.
Moreover, Hadoop is a model which provides the distributed processing across various computer clusters to handle huge datasets.

Hadoop can handle different jobs like batch and interactive jobs for numerous concurrent subscribers. Hadoop consists of two key components such as HDFS and MapReduce which helps to handle large amount of data. HDFS (Hadoop Distributed File System) can be utilized for efficient data storage and its architecture is made in such a way that it can handle huge cluster of datasets whereas MapReduce is a technique which can handle structured and unstructured both type of data adaptively and this framework can tolerate different faults and provides high reliability. A Hadoop framework consists of different nodes such as Name Node, Data Node, and Client machine node.
The various phases of Hadoop map reduce are setup, map, reduce, shuffle and sort. The master and cluster node are distributed into map reduce tasks. The Map Reduce model takes input as key value pair and provides intermediate results. The value pair is sorted and the corresponding output is stored and written to Hadoop Distributed File System.

II. RELATED WORK

The various drawbacks are related to Hadoop Map Reduce. The Hadoop Map Reduce Scheduler does not consider memory efficiency and multi core environment. The efficiency of Cloud usage is minimized. Recently, for enhancing the performance of Hadoop application [16], various efficient Hadoop models are presented by numerous researchers [17], [18], [19], [20], [21], [22], [23], [24] and [25]. The streaming of Hadoop Map Reduce reduces the efficiency. The challenging task is to provide the makespan model with three core stages. I.e. Map, Reduce and Scheduler. CRESP [22] predicts task execution efficiently and helps to allocate resources based on MapReduce slots. However, in CRESP application models, the effects of number of reduced jobs are discarded.

In [21] and [22], the numbers of reduced jobs are constant, incur memory and I/O overhead and failed to provide data locality awareness. Map reduce models are designed homogeneously. The above analysis shows that utilizing resources are drawback and therefore we have to minimize the makespan time and resource utilization efficiently.

The major work of PHMR used to remove the unutilized null slots to increase the efficiency. PHMR is similar to HMR which uses the data which is divided in to chunks. The chunks are segmented further to process in order to improve the efficient in time production and reduce the empty slots.

III. RESULTS AND DISCUSSIONS

Experimental study of PHMR performance attained over existing model [19] considering diverse application is presented in this section. HMR is the most preferred MR framework for executing data intensive and scientific application using cloud computing framework [31]. The Hadoop cluster is composed of 4 slave worker with 1 master worker. The Hadoop cluster is deployed on Microsoft Azure public cloud platform. This work used Hadoop 2.7, where each worker (both master and slave) is composed of 120 GB of cloud blobs/ container size, 7 Gigabytes RAM and 4 cores. Identical configuration is used for executing diverse application on both HMR and PHMR. For evaluating performance of PHMR over HMR diverse application from memory, CPU to I/O intensive application is considered. All three cases study is required to evaluate the robustness of any parallel computing framework. Thus, this work considered diverse application such as Gene sequence analysis, non-stream data and stream data analysis. Currently, the genetic data are extremely large, thus Gene sequence analysis are memory, CPU to I/O intensive. HMR model are good in handling CPU intensive. Thus it is important to see how it performs considering Memory and I/O intensive task. The PHMR model is designed to address both Memory and I/O and retaining CPU feature of HMR and it is important to see how it performs considering different genomic data (size) (i.e. considering both short and long read). More details of genomic data used are detailed in later section.

Along with we also considered large stream and non-stream application. This work considered Ecommerce (non-stream) data analysis. These applications are generally CPU intensive with minimal I/O requirement. On the other side non-stream application we had considered Twitter dataset. These applications involve both CPU and I/O intensive. Thus, it is important to see how both PHMR and HMR perform on these applications. More detail of dataset used is discussed in later section.

A. Bioinformatics application (gene sequencing) performance evaluation of PHMR over HMR

This section focuses on performance evaluation of gene sequence analysis [19] on PHMR and HMR framework. Application related to Cancer research, Genetic Diseases identification, Reproductive Health etc. are dependent sequence alignment algorithms for analysis. This work used Homo sapiens chromosome (NC_000015.10) and baker's yeast genomic database as reference database obtained from [32] and the query sequences obtained from the influenza virus database [33]. The detail of query and reference genome used for experiment analysis is described in Table I. Gene sequence analysis is performed on query and reference genome used in Table I and the result are graphically plotted in Fig. 2.The experiment outcome shows an execution time reduction of 64.39%, 52.98%, 47.76%, and 44.87% is achieved by PHMR over HMR considering genomic data size of 4988 base pairs, 10207 base pairs, 576874 base pairs, and 948066 base pairs, respectively. An average execution time reduction of 52.43% is attained by proposed PHMR over exiting HMR considering varied genomic data size.

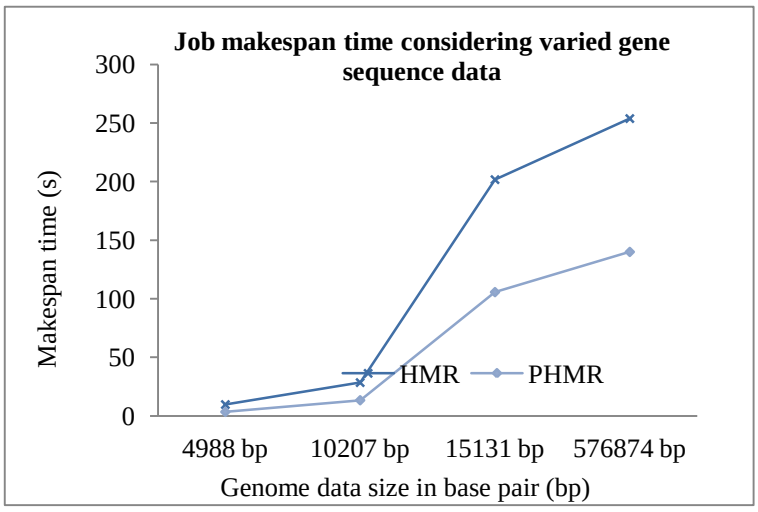

Figure. 1. Total makespan time for performing genomic sequence analysis on varied genome data size conducted on PHMR and HMR frameworks

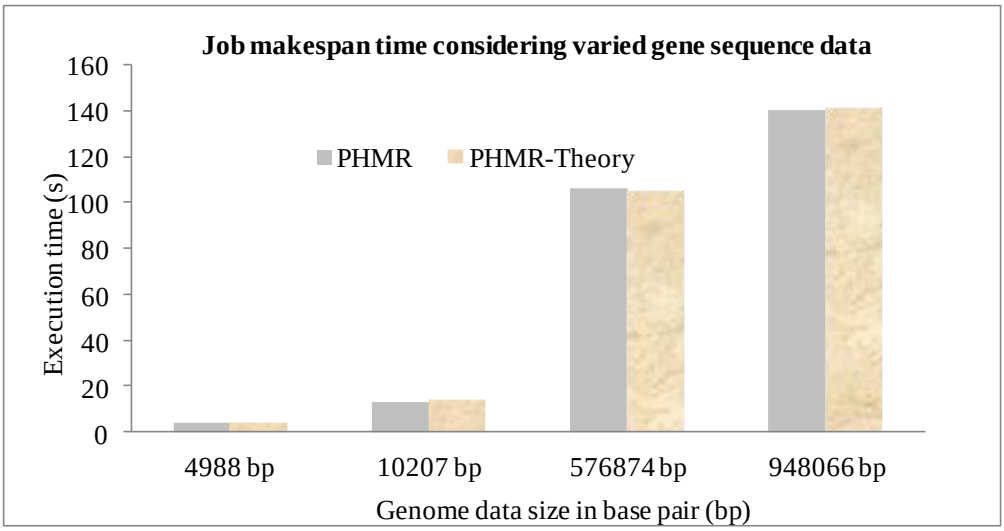

Fig. 2. Correlation between theoretical and practical execution time for varied genome data analysis on PHMR framework

B. E-commerce data analysis performance evaluation of PHMR over HMR

This section carries out performance evaluation of E-commerce data analysis on HDPC and HMR framework. For experiment analysis Word count (Text computation/mining) application [22] is used. Amazon product data [34] is used for experiment analysis which composed of 142.8 million reviews from May 1996 to July 2014. However, this work considers the experiment case shown in Table II. Word count analysis is performed on Amazon review dataset and result is graphically shown in Fig. 4. The experiment outcome shows an execution time reduction of 50.98%, 49.09%, and 55.64% is achieved by PHMR over HMR considering review data size of 3,268,695, 3,447,249, and 5,748,920 respectively. An average makespan time minimization of 51.9% is attained by proposed PHMR over exiting HMR considering varied review data size.This section discusses the result attained by PHMR over HMR and other Hadoop based MR model for executing diverse application such as genomic sequence analysis, non-stream application for analyzing ecommerce review using word frequency statistics and stream applications such as hot word detection is used. The results articulated here shows that the PHMR framework minimizes the total job makespan attained due to the proposed makespan model considering memory optimization and parallel execution strategy to reduce unutilized/null slots. An average job makespan time reduction of 52.43%, 51.9% and 53.33% is attained by proposed PHMR over exiting HMR [19] for bioinformatics application, e-commerce (non-stream) analysis, and text mining (stream) analysis, respectively. The comparative analysis over existing methods is tabulated in Table III shows the robust and scalable and effectiveness of PHMR over existing methods. Since, PHMR support execution of bioinformatics, text mining, stream, and non-stream application over cloud platforms. Along with PHMR makespan model utilizes cloud resource more efficiently. Correlation measure shows that our mathematical attain better accuracy than existing mathematical makespan model [20] and [22]. Usage of public cloud environment aid in providing scalable processing of huge amount of data (i.e., both stream and non-stream) on different Hadoop cluster size. The above described factor enabled PHMR to attain better performance than most standard Hadoop based parallel computing model.

TABLE I. . COMPARSON WITH EXISITNG PARALLEL COMPUTING MODEL

	[19]	[20]	[21]	[22]	[23]	PHMR
Type of application or algorithm used	Bioinformatics	Word count application	Tera sort, and Word count application	Sort and Word count	Sort and Word count application	Bioinformatics, stream and non-stream
MR computing environment considered	HMR	HMR	HMR	HMR	HMR	HMR
Cloud computing environment used	Yes	NO	Yes	Yes	No	Yes
Average makespan minimization over Hadoop (%)	40.01%	13.01%	34.08%	27.1%	43.67%	52.47%

IV. CONCLUSION

The main objective was to improve the memory efficiency by reducing the makespan time and utilizing the memory efficiency effectively by removing the unused null slots. This we can achieve by utilizing the PHMR model where the memory overhead is reduced and and this overhead aims at utilizing the unorganized slots. This improves the over makespan time over the existing HMR model.

Finally we can see a good correlation between the memory overhead and time efficiency. The future work will aim to implement the effect on diverse application and large volume of datasets.

REFERENCES

1. X. Shi et al., "Mammoth: Gearing Hadoop Towards Memory-Intensive MapReduce Applications," in IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 8, pp. 2300-2315, Aug. 1 2015.
2. J. Zhu, J. Li, E. Hardesty, H. Jiang and K. C. Li, "GPU-in-Hadoop: Enabling MapReduce across distributed heterogeneous platforms," Computer and Information Science (ICIS), 2014 IEEE/ACIS 13th International Conference on, Taiyuan, pp. 321-326, 2014.
3. M. Zaharia, A. Konwinski, A.D. Joseph, R.H. Katz and I. Stoica, &ldquo,Improving Mapreduce Performance in Heterogeneous Environments,&rdquo, Proc. Eighth USENIX Conf. Operating Systems Design and Implementation (OSDI), pp. 29-42, 2008.
4. D. Dahiphale et al., "An Advanced MapReduce: Cloud MapReduce, Enhancements and Applications," in IEEE Transactions on Network and Service Management, vol. 11, no. 1, pp. 101-115, March 2014.
5. E. Deelman, G. Singh, M. Livny, B. Berriman and J. Good, "The cost of doing science on the cloud: The Montage example," 2008 SC - International Conference for High Performance Computing, Networking, Storage and Analysis, Austin, TX, pp. 1-12, 2008.
6. N. Chohan, C. Castillo, M. Spreitzer, M. Steinder, A. Tantawi, and C. Krintz, “See spot run: using spot instances for mapreduce workflows,” in Proc. 2010 USENIX Conference on Hot Topics in Cloud Computing, ser. HotCloud’10. USENIX Association, pp. 7–7, 2010.
7. X. Lin, Z. Meng, C. Xu, and M. Wang, “A Practical Performance Model for Hadoop MapReduce,” in Cluster Computing Workshops (CLUSTER WORKSHOPS), 2012 IEEE International Conference on, pp. 231–239, 2012.
8. X. Cui, X. Lin, C. Hu, R. Zhang, and C. Wang, “Modeling the Performance of MapReduce under Resource Contentions and Task Failures,” in Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on, vol. 1, pp. 158–163, 2013.
9. W. Xiao, W. Bao, X. Zhu and L. Liu, "Cost-Aware Big Data Processing Across Geo-Distributed Datacenters," in IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 11, pp. 3114-3127, 2017.
10. M. Khan, Y. Liu and M. Li, "Data locality in Hadoop cluster systems," 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Xiamen, pp. 720-724, 2014.
11. M. Xu, S. Alamro, T. Lan and S. Subramaniam, "CRED: Cloud Right-Sizing with Execution Deadlines and Data Locality," in IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 12, pp. 3389-3400, 2017.
12. H. Alshammari, J. Lee and H. Bajwa, "H2Hadoop: Improving Hadoop Performance using the Metadata of Related Jobs," in IEEE Transactions on Cloud Computing, vol. PP, no. 99, pp. 1-1, 2016.
13. Daria Glushkova, Petar Jovanovic, Alberto Abelló, “MapReduce Performance Models for Hadoop 2.x”, in Workshop Proceedings of the EDBT/ICDT 2017 Joint Conference, ISSN 1613-0073, 2017.
14. M. Ehsan, K. Chandrasekaran, Y. Chen and R. Sion, "Cost-Efficient Tasks and Data Co-Scheduling with AffordHadoop," in IEEE Transactions on Cloud Computing, vol. PP, no. 99, pp. 1-1, 2017.
15. M. Khan, Y. Jin, M. Li, Y. Xiang and C. Jiang, "Hadoop Performance Modeling for Job Estimation and Resource Provisioning," in IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 2, pp. 441-454, 2016.
16. Khan, M., Huang, Z., Li, M., Taylor, GA., – Optimizing Hadoop parameter settings with gene expression programming guided PSO. Concurrency Computation: Practice and Experience, DOI: 10.1002/cpe.3786, 2016.
17. Apache, Centralized Cache Management in HDFS. Update date 2014.
18. Y. Huang, Y. Yesha, M. Halem, Y. Yesha and S. Zhou, "YinMem: A distributed parallel indexed in-memory computation system for large scale data analytics," 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, pp. 214-222, 2016.
19. Zhang, J., et al., A Distributed Cache for Hadoop Distributed File System in Real-Time Cloud Services, in Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing. 2012, IEEE Computer Society. p. 12-21.