International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 04 Issue: 06 | June -2017 www.irjet.net p-ISSN: 2395-0072

An Efficient Technique to Improve Resources Utilization for Hadoop MapReduce in Heterogeneous system

Ahmed Qasim Mohammed[1] , Rajesh Bharati[2]

[1]Department of Computer Engineering, Dr. D.Y. Patil Institute of Technology, Pune University, [2]Department of Computer Engineering, Dr. D.Y. Patil Institute of Technology, Pune University,

---

Abstract— Oughties witness releasing one of the most reputed platform for processing and storing Big Data which known by a strange name is Hadoop, mainly Hadoop consist of two main application MapReduce for processing data and Hadoop Distributed File System (HDFS) for storing data, in fact even all the good features of Hadoop there's still some struggles specially with high speed of data growth, therefor there is need to improve the performance of the main two components to increase Hadoop capability to treat data in efficient way. In this paper our main focus is improving resources utilization in MapReduce as a result of this there will be maximum usage of resources and minimizing time for processing data so we implemented different techniques on different level of MapReduce, our work started by adding a classifier level by using lightweight classification algorithm Support Vector Machine (SVM)to overcome heterogeneity issues that face Hadoop and generate problem to assign proper job to proper slave node, by this technique we decreased failed Tasks to approximately zero, second level will start scheduling into phase level by using PRISM fine grained algorithm here also we find using this algorithms increased resources utilization and decreased job running time by 10-30 % compared to existing schedulers, finally we add a dynamic slot configuration algorithm to give prepare slave node with proper required number of Map and Reduce slots. In overall using improving on different levels show improvement in performance of MapReduce and decreased running time by 30% depending on variation of Jobs and there requirements.

Keywords— Hadoop; MapReduce; Resources Utilization; Phase-level; Classification Algorithm; Prism algorithm; Heterogeneous System; Dynamic Slot Configuration;

1. INTRODUCTION

With the sunshine of third millennium the organization submerged with a lot of data and data types start changing day after day.

Due to uprising of new technologies (Social media, Smart phone, IOT, etc.) there was more and more data which known today as a BigData, according to some survey reported that 90% of data was generated in last five years also 4.1 Exabyte's it was in 2014 and this number increasing with incredible speed day by day.

For all this amount of data there was a need for an efficient platform to handle the data with cost effective, here Hadoop came to solve many of BigData problems with efficient way and less time, Fig.1 shows Hadoop's time line till it became stable in 2009.

But before down to brass tack of Hadoop, we need to know more about BigData[1, 10, 11], which is nothing only the problems that cannot be solved by traditional tools and there is characteristics that define BigData problems which publically known as 3 V's[2] of Big Data but actually they are more than three and the most high effected V's are Volume which explain the problem of enormous amount of data, Variety show the problem that there's many types of data, each required different tools to get served, Velocity show the problem of handling the data in real time, and last which is always neglected in literatures but it's the most important from our view it is Veracity where data getting process must have meaningful for particular problem, There's more V such as validity and volatility.

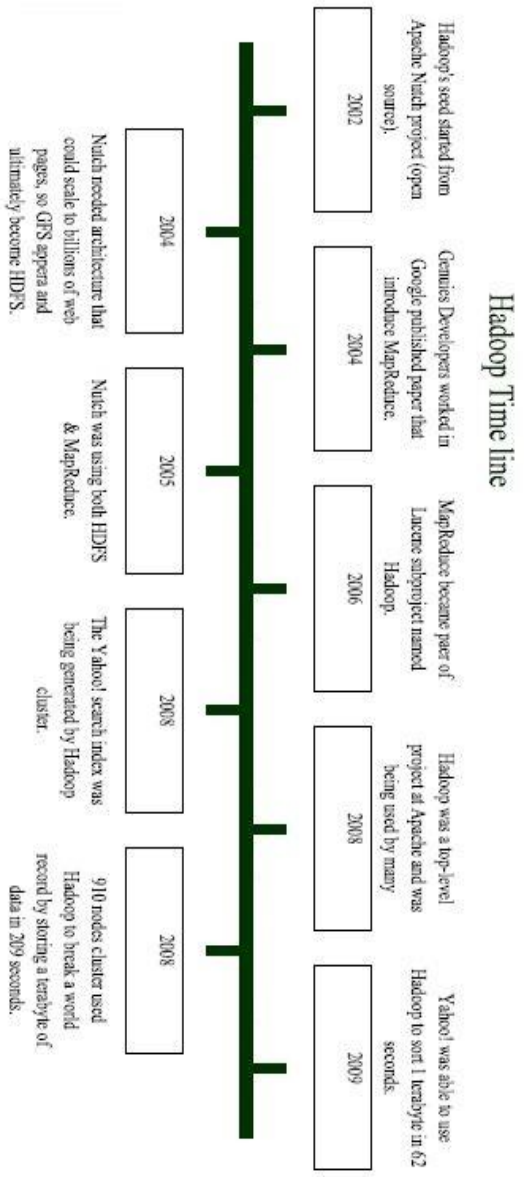

Fig -1: Hadoop's time line till it became stable in 2009

When one or more of V's appears so we have to know that we are going to face BigData problems.

Because of BigData problems Hadoop came to provide solutions for many of BigData problems by using a traditional information technology tools, the main two are MapReduce for processing data and HDFS[3,4] for storing data.

Fig.2 shows dataflow in Hadoop and the main two components of Hadoop, also it show the phases of MapReduce.

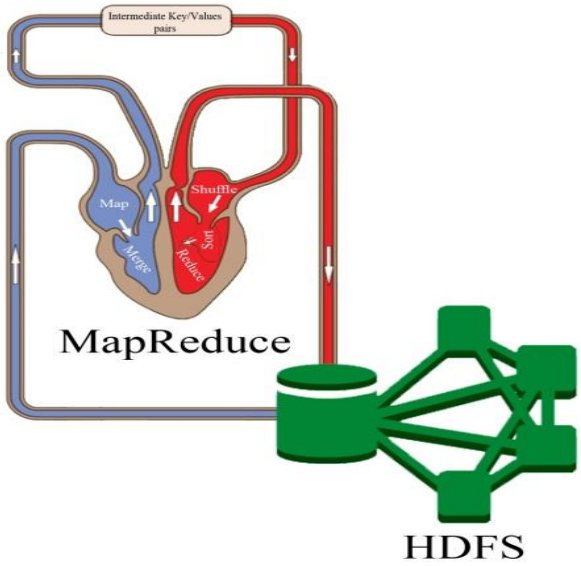

Fig -2: Dataflow in Hadoop

But Hadoop in fact not only two components Hadoop is a family that contain several applications most of them adopted by Apache and the number of these application regularly increasing last application is Apache Flink which is use to process data functionally in real time, Also there's Ambari and Hue used for administration, Habse[8], Hive and Sqoop[9] are applications dedicated to structured data, Flume and Sqoop used for moving data into Hadoop ecosystem, there's Pig which is used with query language, and many other applications for controlling workflow like Oozie[5] and other which is used for analysis such as Mahout and finally there's Zookeepers which is simple interface used to coordinate work of other application in systematic way.

We took look on Hadoop from different angels so in next sections we are going to focus on MapReduce as our paper work is improving the utilization of MapReduce by using different techniques on different level of it.

Our paper organized as follow. Section 2 will study MapReduce architectures in details, in Section 3 we will describe different literatures and provide more information about going research to improve scheduling of MR, while in Section 4 will discuss our implementation and result and finally conclusion and future work in section 5.

2. MapReduce Architecture

MapReduce it is one of the most powerful tool to process Big Data in parallel but it process rest data, originally MapReduce [6] designed to process data in distributed system by using simple technique of splitting data into

fragments and Map work to Map the input with key-value pairs while Reduce will combine similar keys together to produce one key- summation values, in practical MapReduce it's more complex than what we said and Fig.3 shown the flow of MapReduce in Hadoop with HDFS to process data in fast and efficient way.

MapReduce consist from Map and Reduce but inside this main two components there's five phases, each phase differ from the other with the resources consuming and requirements.

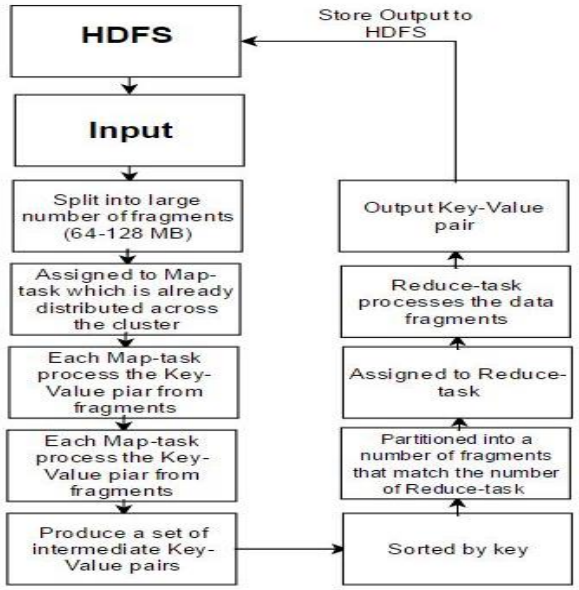

Fig -3: MapReduce Data Flow

Map consist two phases Map-phase and Merge-Phase both of them required CPU and memory resources to process the fragments and produce key-value pair, while Map-phase initiate key and assign value according to the job requirements, Merge-phase will gather related keys together to prepare for Reduce-task, actually sometimes the output of Map-task will be much bigger than the input for that there's a simple phase similar to Reduce but it work locally, this phase called combiner.
After Map-task done processing data there's first phase of Reduce-task which consume I/O [7] bandwidth to transfer key-value pairs to second

Phase of Reduce-task which is known as Sort-Phase and its work just to prepare the data in manner way that save time for Reduce-phase when it will start combining key-value pairs to produce one key many values and store them into HDFS.
The mechanisms that happened inside MapReduce inspired the author of PRISM algorithm to propose an algorithm that able to switch between task-level and phase-level and schedule task into phase-level with requirements monitoring.
Also we have to mention that JobTracker/TaskTracker architectures could only run on MapReduce.
In the end we have to mention the high scalability of MR model.

3. Related Work

These are the following papers that are related to my proposed as follows:-

Literature no.	Publication author and year	Advantages	Disadvantages
1	Dynamic Job Ordering and Slot Configuration for Map-Reduce Workloads, Tang S. , Lee B.S., He B., IEEE Trans- action (2016)	• Motivation of using Job ordering. • Dynamic slot Configuration.	• Using default Hadoop's scheduler which already have lot of draw back with consuming time.
2	PRISM: Fine-Grained Resource Aware Scheduling for MapReduce. Zhang Q., Zhani M., Yang Y., Boutaba R. and Wong, IEEE, (2015)	• Using new algorithm which work into Phase-level. • Increasing resource utilization.	• They didn't work on merging profiling job requirements with the new algorithm which result to need to switch back to Task-level. • Pause time effect on progress of job. • Using static slot configuration.
3	Hybrid Job-Driven Scheduling for Virtual MapReduce Clusters. Ming-Chang Lee, Jia-Chun Lin, IEEE (2015)	• Improve data locality for both map and reduce tasks. • Provide the good job performance.	• Overload memory of the Hadoop master server.
4	RFHOC: A Random-Forest Approach to Auto-Tuning Hadoop's Configuration. Zhendong Bei, Zhibin Yu, IEEE (2015)	• Performance increases. • Increase data size.	• Hadoop application needs several execution rounds.
5	Transformation-Based Monetary Cost Optimization for Workflows in the Cloud. Amelia Chi Zhou, Bingeing He,	• Effective transformation by using the transformation based technique.	• They consider only the single service provider. • They used only the few transformation sets. • High cost for optimization.
6	Resource-aware adaptive scheduling for MapReduce clusters. J. Polo, C. Castillo, D. Carrera, Y. Becerra, I. Whalley, M. Steinder, J. Torres, and E. Ayguade, (2011).	• An adaptive resource-aware scheduler that uses job profiling information to dynamically adjust the number of slots for scheduling.	• Required more work to increase performance on scheduling jobs.

Fig -4: Literatures

4. Discussion and Result

In this paper we did our improvements through three levels as shown in Fig.5 System Architecture.
In the first level we used SVM (support vector machine) which is an adaptive linear algorithm, does not overload our system.
In this level SVM fetch the job from jobs pool and classify it according to job requirements and node features

into non-executable which will back to first pool to be classified one more time with the node that have highest resources if it get executable label it will continue into node local pool otherwise administrator will interrupt to solve the problem which is really rare and it will be solved by upgrade the cluster with sufficient resources to serve such job smoothly in

Future in this level we overcome the problem with heterogynous system .

Executable jobs will be gathering into local pool specific for only this node and they will get schedule by PRISM algorithm into phase level to be process in parallel with another job processed in advance phase.

In fact it's not that easy to process this entire job in parallel due to phase dependences, for example Shuffle phase will always wait for Merge phase to complete transferring data from Map task to Reduce task.

But in overall these two level shows very pleasant result while we process Job contain different task of counting words, finding specific word, and count appearance of specific word.

Where there was no any task failed as showing in result and the time much faster compared to use existing fair schedulers in current system.

Finally while we classified task approved executable task requirements will directly sent to Task Tracker and saved in queue when this task get served will find already required Map slot and Reduce slot configured according to job need.

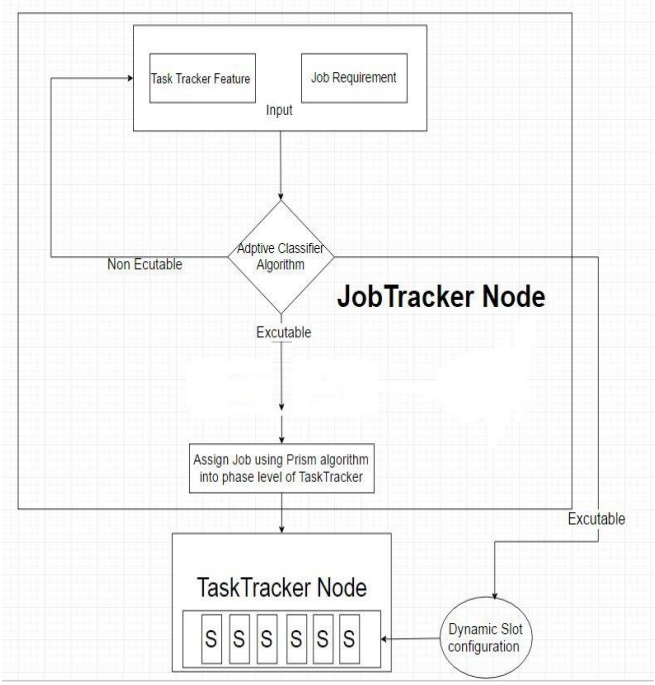

Fig -5: System Architecture

We have to mention that Map task can only processed in Map slot and same happened with Reduce task can only processed in Reduce slot after Map task complete processing.

Following figure show the dataflow in our system.

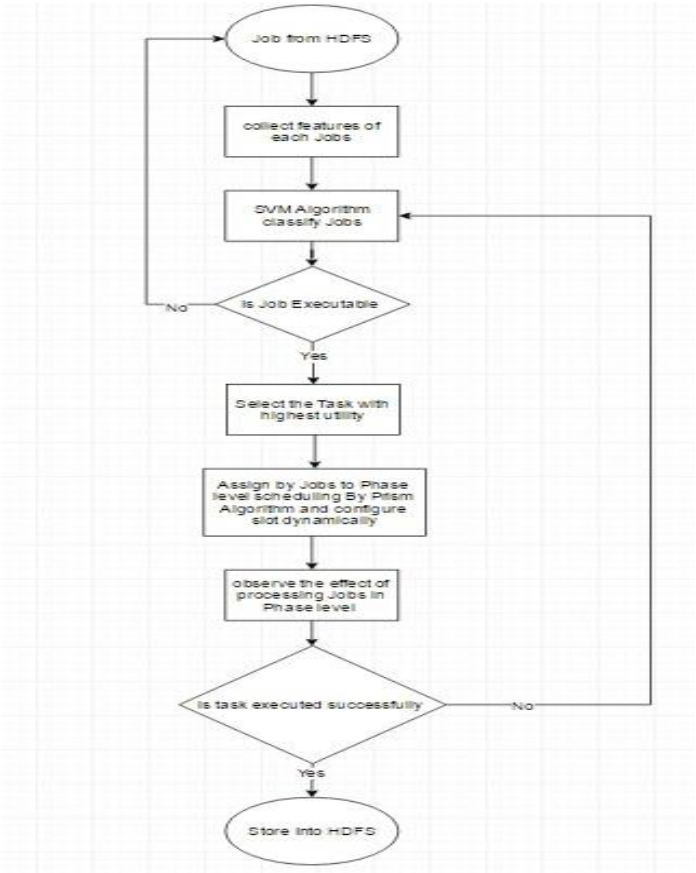

Fig -6: shows the dataflow in our system

For our experiments we configured a cluster of 3 nodes working on centos operating system with stable version of Hadoop on each node.

Each node has different resources to create a heterogynous system by configuring them on virtual machine.

- Master node (JobTracker)

Hard Disk	50 GB
Memory	6 GB
Processors	6 core

- Slave node1 (TaskTracker)

Hard Disk	50 GB
Memory	5 GB
Processors	3 core

- Slave node2 (TaskTracker)

Hard Disk	50 GB
Memory	4 GB
Processors	2 core

And we run a word count job contain different task (counting words, finding specific word, and count appearance of specific word).
Result shown in Fig.7 where there was no any failed Task and blue line shows more Jobs submitted and completed processing with in less time.

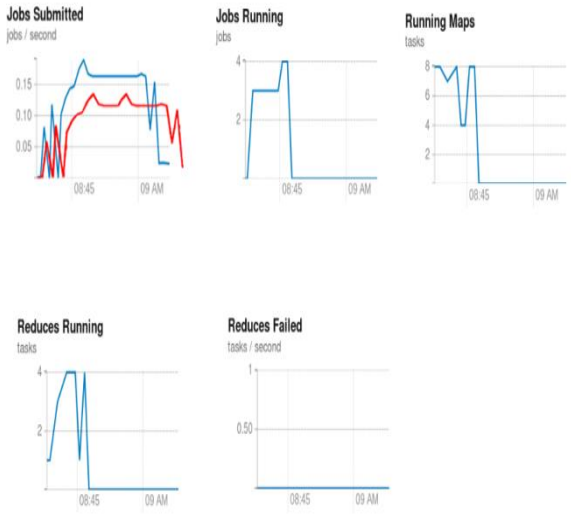

Fig -7: Results

5. Conclusion and Future Work

In this paper we are satisfied with result that we get from implementing different improvement in the architecture of MapReduce, by using adaptive algorithms on different level that can learn and improve they work with time and we overcome problem of heterogeneous system by using SVM algorithm.

But we still believe there's more to do in to improve MapReduce by understanding Job requirements and profile them in proper way to satisfy phase scheduling requirements.

For Future work we are going to go more deep with studying new coming application that work with Hadoop to process data in real time.

Also we are going to deploy Nash Equilibrium rule into Hadoop's slave node to make them smarter to take decision of choosing task by them self.

References

[1] T. White, ''How the MapReduce works,'' in Hadoop: The Definitive Guide, 3rd ed. Tokyo, Japan: O'Reilly Inc., 2012.

[2] I. Lahmer and N. Zhang, "MapReduce: MR model abstraction for future security study," in Proc. 7th Int. Conf. Secur. Inf. Netw., 2014, pp. 392–398.

[3] C. Lam, "Introducing Hadoop, and managing Hadoop," in Hadoop in Action. Greenwich, U.K.: Manning Publications Co, 2010.

[4] P. Zikopoulos, C. Eaton, D. Deroos, T. Deutsch, and G. Lapis, Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. New

York, NY, USA: McGraw-Hill, 2012.

[5] J. Dean and S. Ghemawat, ''MapReduce: Simplified data processing on large clusters,'' Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008.

[6] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica. Dominant resource fairness: fair allocation of multiple resource types. In USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2011.

[7] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica. Dominant resource fairness: fair allocation of multiple resource types. In USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2011.

[8] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. Cetin, and S. Babu. Starfish: A self-tuning system for big data analytics. In Conference on Innovative Data Systems Research (CIDR11), 2011.

[9] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, and K. Talwar. Quincy: fair scheduling for distributed computing clusters. In ACM SIGOPS Symposium on Operating Systems Principles (SOSP), pages 261–276, 2009.

[10] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang. Multi-resource allocation: Flexible tradeoffs in a unifying framework. In IEEE International Conference on Computer Communications (INFOCOM), pages 1206–1214, 2012.

[11] J. Polo, C. Castillo, D. Carrera, Y. Becerra, I. Whalley, M. Steinder, J. Torres, and E. Ayguade. Resource-Aware Adaptive ´ Scheduling for MapReduce Clusters. ACM/IFIP/USENIX Middleware, pages 187–207, 2011