International Journal of Early Childhood Special Education (INT-JECSE)
DOI:10.9756/INTJECSE/V14I6.338 ISSN: 1308-5581 Vol 14, Issue 06 2022

ANALYSIS ON BIG DATA BASED MAPREDUCE WORKLOADS FOR IMPROVING JOB ORDERING AND SLOT CONFIGURATIONS

[1]J Sravanthi, [2]Dr.Meghna Dubey
Research Scholar Mewar University,Chittorgarh,Rajasthan
Research Supervisor Mewar University,Chittorgarh,Rajasthan

ABSTRACT
Mapreduce is a simultaneous operational model for huge information refinement in groups and datacenters. The work of a Mapreduce consists of a group of tasks that contains more number of matching jobs and reducing the jobs. The matching jobs and reducing jobs can be executed in mapping a position and reducing the positions, the general mapping jobs are processed earlier for reducing jobs, various task processing the requests and mapreduce configuration positions of a Mapreduce has various achievement and variety of computer usage based on the case load. Two types of precise rules that is utilized in minimization of the make span and the entire finishing period of a logged off Mapreduce case load. Initial algorithm concentrates on the task organizing improvement for a Mapreduce case load for the given mapreduce position being set up. In difference, the second algorithm expects the procedure that appears for optimized mapreduce position configuration in a Mapreduce case load. We carry out the modeling observations on Amazon EC2, facebook and it shows that planned precise rules yields the outcome up to 20% - 75% improvised than the present optimized Hadoop, Almost it guides to remarkable simplifications during the operative period.

1. Introduction

MapReduce is the parallel programming paradigm which is widely used in distributed programming frameworks like Hadoop to process big data. Big data is the data with characteristics such as volume, variety and velocity. This data is available in different formats such as structured, semi-structured and unstructured. It does mean that big data has variety of formats. It is voluminous due to exponential growth of data produced by enterprises, wireless sensor networks, satellites and World Wide Web (WWW). Velocity refers to the fact that big data also includes the data which is streamed live from time to time. Thus there is complexity of big data and this complexity demands a distributed environment associated with cloud computing for processing big data. Hadoop is one of the best examples for distributed programming framework which supports MapReduce programming paradigm. It is the new model of programming where Map and Reduce phases are involved. In Map phase data is processed by thousands of nodes presented in commodity clusters while the reduce phase is to get the final output. The result of Map is a set of intermediate key/value pairs while the result of Reduce is the final output which is also in the form of key/value pairs.

Many researchers contributed towards improving performance of MapReduce programming. However, there is little research on the performance of Map Reduce workloads. There are two key performance metrics that are used to measure the performance of Map Reduce workloads. They are makespan and total completion time (TCT). The former is the metric to know the time taken when first job is started to the last job is finished from a batch of MapReduce jobs. The latter refers the time taken to execute all jobs including the queuing time and waiting time of individual jobs in the workload. Recently Tang et al. [11] studied the role of slot configuration and job ordering for MapReduce workloads. Their study revealed that map slots and reduce slots are pre-configured and the utilization is not optimal. They proposed two kinds of algorithms that improved makespan and total completion time. They focused on the optimization of job ordering and also slot configurations. They used Amazon EC2 for empirical study.

In their study it is found that the job ordering and slot configurations can be improved further and considered in this research as an optimization problem. The aim of this research is to have further investigation into MapReduce workloads and provide alternative algorithms for job ordering and slot configurations for improving performance of workloads in terms of maketime and total completion time.

2. LITERATURE SURVEY

This section provides review of literature on MapReduce workloads of big data and its related aspects with respect to job scheduling and execution. Mashayekhy et al. [1] studied scheduling of MapReduce jobs with energy efficiency. They used Hadoop for empirical study in the presence of jobs with Service Level Agreements (SLAs). They proposed two algorithms for energy aware scheduling. They studied energy consumption and execution time for different workloads in Hadoop environment. They used benchmark workloads like TeraSort, PageRank and so on for testing the efficiency of the proposed algorithms. They found that their algorithms

could consume 40% less energy when compared with other algorithms. It shows that managing MapReduce workload can be optimized to improve energy efficiency. Dimopoulos et al. [2] studied the big data frameworks and their behaviour in restricted private clouds. They found interference among popular big data frameworks. They evaluated frameworks like Storm, Spark and Hadoop in the presence of multi-tenance workloads. They found that in a constrained environment, different frameworks behave differently. However, they show certain common issues such as deadlock of resources, performance variability, and failed fair sharing. They proposed an architecture known as Mesos architecture that supports resource scheduling and handling of jobs in restricted environments.

Tang et al. [3] explored slot-based MapReduce programming paradigm and found that Hadoop MRv1 suffers from performance issues as it was not optimized. They identified three key aspects of resource allocation to alleviate issues with performance of such slot based MR. The first aspect is that due to pre-configuration, the map and reduce slots are underutilized. The second aspect is that the concept of speculative execution improves performance but at the cost of efficiency of commodity clusters. Delay scheduling improved performance but it is done at the cost of fairness. To overcome these three issues, they proposed alternative techniques such as Dynamic Hadoop Slot Configuration, speculative execution performance balancing, and slot pre-scheduling respectively. The framework they proposed is known as DynamicMR which could improve fairness for both single and multiple jobs with significant performance improvement over YARN.

Ren et al. [4] focused on workload analysis in the Hadoop environment. They used trace collected from real Hadoop environment and performed workload analysis. They also implemented a workload generator for experiments. It is known as Ankus which was built for expediting performance evaluation. It was used to have synthetic MapReduce workloads in the e-Commerce environment. They also proposed a job scheduling algorithm known as Fair4S which is meant for solving problems with small jobs. This is used to overcome the issues with Hadoop with respect to handling small jobs. Compared to Hadoop's fair scheduler, the Fair4S imporves performance by factor of 7. Understanding workloads of big data is very important. Pan et al. [5] studied inputs and outputs related big data. They characterized I/O of big data workloads in data centres. The rationale behind their study was the reason that I/O performance is very important for the success of any system. In fact it dominates overall performance of a system. Their study focused on disk read write dynamics, bandwidth requirements and MapReduce efficiency in terms of I/O. They used other parameters like memory usage, compression, and disk utilization.

Loghin et al. [6] studied the performance of big data processing on small nodes. Their study revealed that big data processing needs environments based on the requirements. There is no one size fit for all needs. They used small memory sizes, I/O and bandwidths and tested the performance of MapReduce programming. They found that Xeon nodes are necessary for I/O intensive workloads while database query related workloads can work on smaller nodes as well. Therefore, their conclusion is that based on the workloads, it is essential to have corresponding computing environment. Sometimes, in this context, it is not necessary to use data center services and sometimes, it cannot be avoided. Tian et al. [7] studied MapReduce workloads and the ways and means to reduce makespan of multiple MapReduce jobs. Minimizing makespan is considered is very important optimization problem in distributed programming environments. It is true with Hadoop and other MapReduce implementations. Hadoop is the open source implementation of MapReduce. They proposed a scheduling model known as HScheduler which takes care of optimal scheduling of multiple MapReduce jobs in order to reduce makespacn and improve overall performance of the system. They made experiments to reduce makespan in different phases of MapReduce programming in distributed environments.

Feller et al. [8] considered a Hadoop case study to know the energy efficiency and performance of big data applications. They evaluated Hadoop performance both in the traditional models and computing services. The experiments are made by separating data and computing services to have more useful insights. They observed many interesting facts in the empirical study. When virtual machines (VM) are coexisted on a server, the performance is degraded. Performance of real clusters is always better that of virtual ones. Separation of services cause performance degradation based on ratio between data and compute. Application performance is mapped with energy consumption which is again application specific. They compared both traditional model and alternative model. In the alternative model data and compute are separated for performance improvement. They found performance improvements in alternative deployments.

3. METHODOLOGY

This section provides methodology used to complete the research. MapReduce programming paradigm has two important phases. They are Map and Reduce. In these two phases, the slot configurations are pre-defined. This approach wasted slots and reduced performance of MapReduue with different workloads. MapReduce workloads which are benchmarks are used for experiments. The traces are used to have a test bed with simulations. Different workloads are considered for experiments. In each experiment specific number of Map slots and specific number of Reduce slots are used. The results are observed without any job ordering and slot configurations. Then these results are saved for comparing with optimized results. Afterwards, the proposed framework with optimized job ordering and slot configurations is used for experiments with same workloads.

The performance difference is observed in terms of two evaluation metrics aforementioned. They are makespan and total completion time.

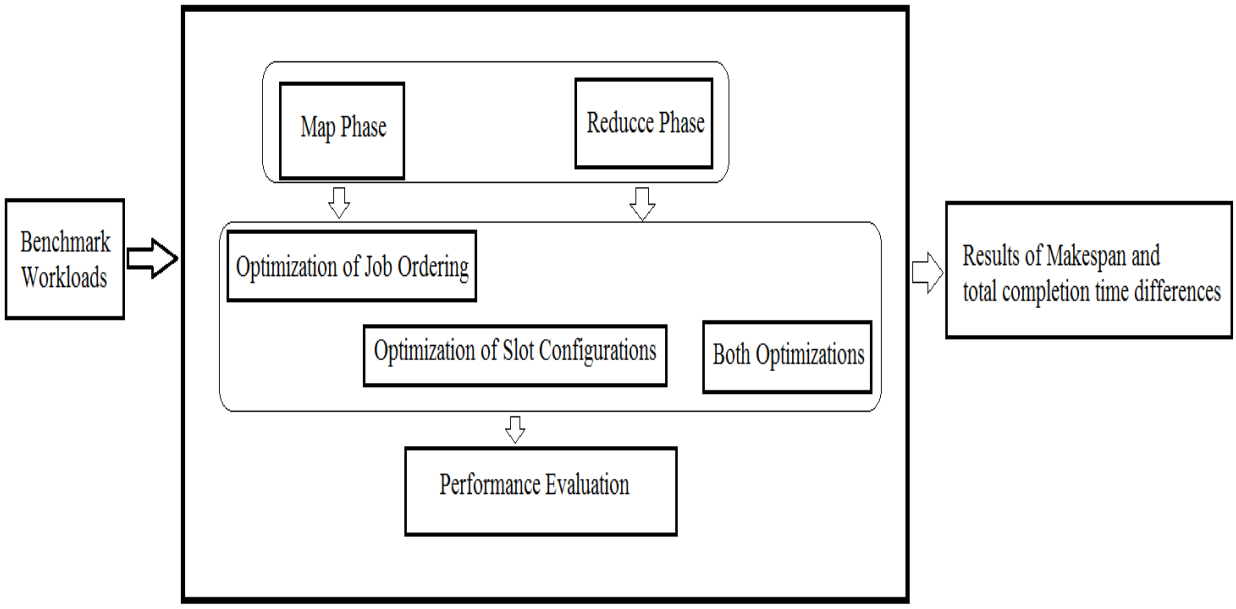

Figure 1: Overview of the methodology

As shown in Figure 1, it is evident that the Map and Reduce phases can have optimizations in terms of job ordering or slot configurations or both. The optimization improves performance of MapReduce applications. Then the performance is evaluated with makespan and total completion time. It is assumed that different slot configurations will have different performance results under specific job ordering. Even when the slot configurations are optimized, the job ordering dynamics differ in performance results. Under different job submission orders, the MapReduce performance is changed. However, the aim of the research is to optimize the performance of MapReduce by optimizing job ordering and slot configurations.

CONCLUSION It illustrates the task sequencing and map/reduce position set up difficulties for MapReduce presentation caseloads processing in a data repository, where the mean processing interval of map/reduce jobs for a Mapreduce job below FIFO planning in a Hadoop group. Two metrics are utilized Makespan and entire concluding interval. We first concentrate on the Makespan. We propose task sequencing optimization of precise rules and positional set up optimization algorithm. We investigate that the entire concluding interval can be bad content to optimal Makespan.

References

1. Lena Mashayekhy, Mahyar Movahed Nejad, Daniel Grosu, Quan Zhang, Weisong Shi. (2013). Energy-aware Scheduling of MapReduce Jobs for Big Data Applications. *IEEE*. 20 p1-14.
2. Stratos Dimopoulos, Chandra Krintz and Rich Wolski. (2016). Big Data Framework Interference In Restricted Private Cloud Settings. *IEEE*. p1-6.
3. Shanjiang Tang, Bu-Sung Lee and Bingsheng He. (2014). DynamicMR: A Dynamic Slot Allocation Optimization Framework for MapReduce Clusters. *IEEE*. p1-14.
4. Zujie Ren, Jian Wan, Me Weisong Shi, Xianghua Xu and Min Zhou. (2013). Workload Analysis, Implications, and Optimization on a Production Hadoop Cluster: A Case Study on Taobao. *IEEE*. p1-15.
5. Fengfeng Pan, Yinliang Yue, Jin Xiong and Daxiang Hao. (2014). I/O Characterization of Big Data Workloads in Data Centers. *IEEE*. p1-7.
6. Dumitrel Loghin, Bogdan Marius Tudor, Hao Zhang, Beng Chin Ooi and Yong Meng Teo. (2015). A Performance Study of Big Data on Small Nodes. *VLDB*. 8 (7), p1-12.
7. Wenhong Tian, Guozhong Li, Wutong Yang and Rajkumar Buyya. (2016). HScheduler: an optimal approach to minimize the makespan of multiple MapReduce jobs. *Springer*. p1-18.
8. Eugen Feller, Lavanya Ramakrishnan and Christine Morin. (2014). Performance and energy eciency of big data applications in cloud environments: A Hadoop case study. *Parallel and Distributed Computing*. p1-26.

9. Zhuoyao Zhang ,Ludmila Cherkasova and Boon Thau Loo. (2014). Exploiting cloud heterogeneity for optimized cost/performance MapReduce processing. *ACM*. p1-7.
10. Marcelo Veiga Neves, C´esar A.F. De Rose,Kostas Katrinis and Hubertus Franke. (2014). Pythia: Faster Big Data in Motion through Predictive Software-Defined Network Optimization at Runtime. *IEEE*. p1-10.