Indonesian Journal of Electrical Engineering and Computer Science
Vol. 12, No. 3, December 2018, pp. 1132~1142
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v12.i3.pp1132-1142

An Accurate and Efficient Scheduler for Hadoop MapReduce Framework

D C Vinutha[1], G.T.Raju[2]
[1]Department of information science and engineering, Vidyavardhaka college of engineering, mysuru, India
[1,2]Department of Computer science and engineering,R.N.S.I.T, Bengaluru, India

Article Info

Article history:

Received Dec 28, 2017
Revised Jan 9, 2018
Accepted Aug 21, 2018

Keywords:

Big data
Bioinformatics
Cloud computing
Hadoop
MapReduce
Parallel computing

ABSTRACT

MapReduce is the preferred computing framework used in large data analysis and processing applications. Hadoop is a widely used MapReduce framework across different community due to its open source nature. Cloud service provider such as Microsoft azure HDInsight offers resources to its customer and only pays for their use. However, the critical challenges of cloud service provider is to meet user task Service level agreement (SLA) requirement (task deadline). Currently, the onus is on client to compute the amount of resource required to run a job on cloud. This work present a novel makespan model for Hadoop MapReduce framework namely OHMR (Optimized Hadoop MapReduce) to process data in real-time and utilize system resource efficiently. The OHMR present accurate model to compute job makespan time and also present a model to provision the amount of cloud resource required to meet task deadline. The OHMR first build a profile for each job and computes makespan time of job using greedy approach. Furthermore, to provision amount of resource required to meet task deadline Lagrange Multipliers technique is applied. Experiment are conducted on Microsoft Azure HDInsight cloud platform considering different application such as text computing and bioinformatics application to evaluate performance of OHMR of over existing model shows significant performance improvement in terms of computation time. Experiment are conducted on Microsoft Azure HDInsight cloud. Overall good correlation is reported between practical makespan values and theoretical makespan values.

Corresponding Author:

D C Vinutha,
Department of Information Science & engineering,
Vidyavardhaka college of engineering, mysuru, India,
Email: vinudc@gmail.com

1. INTRODUCTION

The Many organizations such as industrial, government and education institution collects massive amount of data from various sources such as sensor network, social network, bioinformatics and World Wide Web etc. for various application uses. Performing scalable and analysis on these unstructured data is most desired across many organization. The state-of-art model finds difficulties in performing real-time analysis on continuous/stream data. For performing real-time analysis for data intensive applications, Google have come up with parallel programming model called MapReduce framework [1]. It is highly scalable, fault tolerant and parallelize execution in distributed nature across cluster of computing nodes. Hadoop MapReduce framework [2] has been widely adopted across various organization when compared with counter parts Phoenix [3], Mars [4] and Dryad [5] due to open source nature [6].

The Hadoop MapReduce model predominantly consist of following phases, Setup, Map, Shuffle, Sort and Reduce which is shown in Figure 1. The Hadoop frameworks consists of a master node and a cluster of computing nodes. Jobs submitted to Hadoop are further distributed into Map and Reduce tasks. In setup phase, input data of a job to be processed (residing generally on the Hadoop Distributed File Systems (HDFS)) is logically partitioned into homogenous volumes called chunks for the Map worker nodes. Hadoop divides each MapReduce job in to set of tasks were each chunk is processed by Map worker. Map phase takes input as key/value pair as (k_1, v_1) and generate list of (k_2, v_2) intermediate key/value pair as output. Shuffle phase begins with completion of Map phase that collects the intermediate key/value pair from all the Map task. A sort operation is performed on the intermediate key/value pair of map phase. For simplicity sort and shuffle phases are cumulatively considered in the shuffle phase. Reduce phase processes sorted intermediate data based on user defined function. Output of reduce phase is stored/written to HDFS.

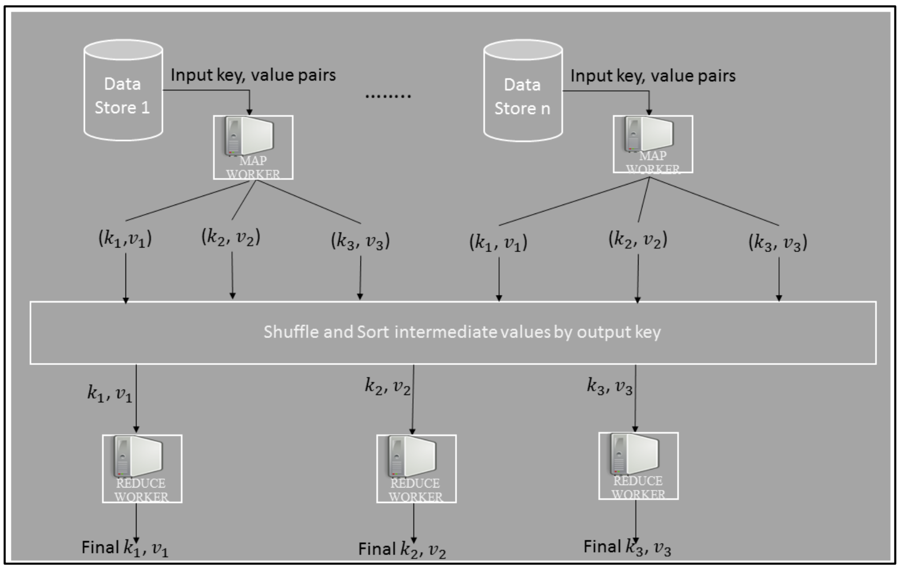

Figure 1. Hadoop MapReduce Computation Model

The Azure HDInsight Cloud aid in achieving scalable performance i.e. user can set up and run Hadoop application on a large-scale cluster. Azure HDInsight Cloud allow user to configure the amount of resource (virtual computing node) required to perform certain task. However, at present Hadoop job with deadline requirement is not supported in HDInsight cloud. The onus is on the cloud user/client to compute the amount of resource requirement to meet task deadline which is a challenging task. Therefore, Hadoop makespan modelling has become an important criteria in computing amount of resources required to meet task deadline. It should be noted that makespan modeling is a challenging task since Hadoop jobs involves multiple processing stage which composed of three core stage (i.e. Map, Shuffle and Reduce stage). Moreover, the first wave of shuffle stage is generally processed in parallel fashion with Map stage (i.e. overlapping phase) and rest of the waves of the Shuffle stage are processed post completion of Map stage (i.e. non-overlapping phase). To utilize the cloud resources efficiently, numerous makespan models for Hadoop is presented [7], and [8]. However, these approaches are not accurate and incurs high computing overhead/time. Since these approaches did not consider overlapping and non-overlapping phases of the Shuffle stage.

Recently, a number of sophisticated Hadoop performance models are proposed [9-14]. Starfish [9] collects a running Hadoop job profile at a fine granularity with detailed information for job estimation and optimization. On the top of Starfish, Elasticiser [10] is proposed for resource provisioning in terms of virtual machines. However, collecting the detailed execution profile of a Hadoop job incurs a high overhead which leads to an overestimated job execution time. In [11], [12], and [13] considers both the overlapping and non-overlapping stages and uses simple linear regression for job estimation. This model also estimates the amount of resources for jobs with deadline requirements. CRESP [14] estimates job execution and supports resource provisioning in terms of map and reduce slots. However, both the HP model and CRESP ignore the impact of the number of reduce tasks on job performance. The HP model is restricted to a constant number of reduce tasks, whereas CRESP only considers a single wave of the reduce phase. In CRESP, the number of reduce tasks has to be equal to number of reduce slots. It is unrealistic to configure either the same number of reduce tasks or the single wave of the reduce phase for all the jobs. It can be argued that in practice, the number of reduce tasks varies depending on the size of the input dataset, the type of a Hadoop application (e.g. CPU intensive, or disk I/O intensive) and user requirements. Furthermore, for the reduce phase, using multiple waves generates better performance than using a single wave especially when Hadoop processes a large dataset on a small amount of resources. While a single wave reduces the task setup overhead, multiple waves improve the utilization of the disk I/O.

To address the research challenges this work present an accurate and efficient makespan model for Hadoop MapReduce framework namely OHMR (Optimized Hadoop MapReduce) to process data in real-time and utilize system resource efficiently. The OHMR present accurate model to compute job makespan time and also present a model to provision the amount of cloud resource required to meet task deadline. The OHMR first build a profile for each job and computes makespan time of job using greedy approach. Furthermore, to provision amount of resource required to meet task deadline Lagrange Multipliers technique is applied.

The Contribution of research work is as follows:

1) This work present an accurate makespan model for HMR aiding performance improvement.
2) Experiments considering diverse cloud configurations and varied application configuration.
3) Correlation between theoretical makespan model and experimental values.

The rest of the paper is organized as follows. Extensive research survey is carried out in Section 2. In Section 3 the proposed makespan modelling for Hadoop MapReduce framework is presented. In penultimate section experimental study is carried out. The conclusion and future work is described in last section.

2. RELATED WORK

In this section, a detailed literature is presented about the conventional state-of-art data analytic techniques. In [9], a locality based Hadoop cluster model is adopted which rely upon the distance between input information and processing nodes. This technique try to overcome from various issues of state-of-art techniques such as high overhead, required large storage capacity and expensive in real time. However, it also induces large delay and causes performance degradation.

In [10], a cloud based optimization framework is adopted to meet deadlines and accomplish data locality. They presented heuristic technique to provision task SLA requirement of cloud user. This technique presented an optimization technique to meet task dead line and minimize the number of nodes required for task processing. They solved single node failure and presented a tradeoff between minimizing deadline and locality constraint. Outcome shows reduction of storage and computation overhead. However they did not considered task deadline awre scheduling and performance evaluation considering compute intensive application.

In [11], a performance enhancement technique is introduced for Hadoop model based on metadata of interrelated tasks. This technique permits Name Nodes to find block which are preset in the cluster to store specific data. Their model attained superior performance than Hadoop framework. For performance evaluation they considered Bioinformatics application. Experiment outcome shows good performance in terms of I.O cost minimization and makespan time reduction. However, they did not considered performance evaluation considering different application and they considered performance evaluation for small genomic data size.

In [12], a Hadoop model is presented based on MapReduce performance modules to reduce delay and contention in the network and enhance performance of the system. And it also helps to decrease synchronization delay and schedule different tasks at a time. They also presented a theoretical evaluation of their makespan model. Attained good accuracy and performance evaluation is carried out for word count applications. However, they did not considered performance evaluation considering diverse application and evaluation on cloud platform.

In [13], an AffordHadoop application is adopted to reduce cost in finishing various tasks and to allocate data and schedule tasks and hence efficiency of system get enhanced. However, a NP-hard problem occurs while scheduling different tasks in state-of-art technique. To address NP-hardness, they adopted integer programming techniques and heuristic reduction and optimization to enable an optimal solution. Experiment are conducted considering Word count and Sort application attained good results in terms of cost minimization. However, theoretical accuracy performance evaluation is not presented.

In [14], a Hadoop model is proposed to predict tasks run-time and allocate some specified resources to accomplish tasks in an assigned time period. Hence, the deadline constraints are met. It uses multiple waves of a shuffle stage. Experiment are conducted considering word count and sort application. Theoretical accuracy performance evaluation of makespan model is presented shows good accuracy. However, it induces high overhead to finish tasks and data intensive and diverse application such as bioinformatics application is not considered for performance evaluation.

In [15], A Hadoop model is adopted to optimize Hadoop parameters with the help of programming based PSO. The PSO technique helps to find optimal parameters in Hadoop networks for a specified task. However, performance evaluation under cloud computing environment is not considered. In [16], a BigData computational model is adopted to reduce cost with the help of geo-distributed datacenters. This technique helps to decide the parameters to select the final data center. Here, a framework for efficient information movement and to provide resource allocation and to select a required data center to decrease cost of the system is described. However, task deadline requirement of task is not considered.

Extensive research survey carried out shows numerous approach is presented to minimize cost, time and amount of resource required to compute a task on Hadoop MapReduce framework. The survey shows need to develop a new makespan model that minimize amount of resource required to task deadline with good accuracy considering diverse application. In next section the proposed makespan model for Hadoop MapReduce framework is presented.

3. MAKESPAN MODELLING FOR PROPOSED OPTIMIZED SCHEDULAR FOR HADOOP MAPREDUCE FRAMEWORK

This work present an optimized scheduler for scheduling job to meet task deadline to meet QoS requirement of application on Hadoop MapReduce (HMR) framework. Firstly, this work present a mathematical model to compute completion time of MapReduce job. Secondly, the amount of resource required to meet task deadline of application is presented.

3.1. Makespan modelling/proposition

Firstly we evaluate the performance limits for a given makespan of a specified set of q tasks that is processed by j slots/servers. Let $\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3, \dots, \mathcal{W}_q$ be the time period of q tasks of a particular jobs. This work consider slot allocation to a task based on slot with Minimum Execution Time (MET) by adopting Greedy algorithm.

Let φ be the maximum time period of q task which is represented as:

$$\varphi = \max_n \{\mathcal{W}_n\} \tag{1}$$

and β be the average time period of q task which is represented as:

$$\beta = \left(\sum_{n=1}^{q} \mathcal{W}_n\right) \Big/ q. \tag{2}$$

The makespan of a task to meet MET is at least $q \cdot \frac{\beta}{j}$ and at most $(q - 1) \cdot \frac{\beta}{j+\varphi}$. We consider the worst case scenario for upper limit, that is, the longest task $\mathbb{W} \in \{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3, \dots, \mathcal{W}_q\}$ with time period φ is the last processed task. Considering this scenario, the time taken before commencement of last task $\mathbb{W}$ is scheduled is at least $\left(\sum_{n=1}^{q-1} \mathcal{W}_n\right) \Big/ j \leq (q - 1) \cdot \beta \big/ j$. Therefore, total execution time of all assignment is at least $(q - 1) \cdot \beta \big/ j + \varphi$. The lower limit is smaller, since the best case is when q task distributed equally among the j available slots. Therefore, the total execution time of is at least $q \cdot \beta \big/ j$. The total job completion time for scheduling lies between the lower and upper limit. These limit are mostly beneficial in case when the time period of longest task is small as compared to total execution time, i.e. when $\varphi \ll q \cdot \beta \big/ j$.

3.2. Computing job completion time

Let consider job $\mathcal{K}$ with known execution time that is obtained from previous execution. Let $\mathcal{K}$ be executed with new set of data that is segmented into $Q_{\mathcal{H}}^{\mathcal{K}}$ map tasks and $Q_{\mathcal{B}}^{\mathcal{K}}$ reduce tasks. Let $\mathcal{A}_{\mathcal{H}}^{\mathcal{K}}$ be the number of map slots assigned to job $\mathcal{K}$ and $\mathcal{A}_{\mathcal{B}}^{\mathcal{K}}$ be the number of reduce slots assigned to job $\mathcal{K}$. Let $\mathcal{H}_{\rightarrow}$ be the mean time period of map task of a particular job $\mathcal{K}$ and $\mathcal{H}_{\uparrow}$ be the maximum time period of map tasks of a particular job $\mathcal{K}$. Then, using makespan modelling (proposition) in section a, the lower limits $\mathcal{W}_{\mathcal{H}}^{\downarrow}$ and upper limits $\mathcal{W}_{\mathcal{H}}^{\uparrow}$ on time period of all map phase are computed as follows:

$$\mathcal{W}_{\mathcal{H}}^{\downarrow} = \frac{Q_{\mathcal{H}}^{\mathcal{K}} \cdot \mathcal{H}_{\rightarrow}}{\mathcal{A}_{\mathcal{H}}^{\mathcal{K}}} \tag{3}$$

$$\mathcal{W}_{\mathcal{H}}^{\uparrow} = \frac{\left(Q_{\mathcal{B}}^{\mathcal{K}} - 1\right) \cdot \mathcal{H}_{\rightarrow}}{\mathcal{A}_{\mathcal{H}}^{\mathcal{K}} + \mathcal{H}_{\uparrow}} \tag{4}$$

The reduce phase is composed of shuffle, sort and reduce stage. Similar to map phase, the makespan modelling (proposition) can be applied to estimate the lower limit $\left(\mathcal{W}_{\mathcal{B}}^{\downarrow}\right)$and upper limits $\left(\mathcal{W}_{\mathcal{B}}^{\uparrow}\right)$ of reduce stage completion time. Since, we possess measurement of mean and maximum tasks time periods in reduce stage, allocated reduce slots $\mathcal{A}_{\mathcal{B}}^{\mathcal{K}}$and the number of reduce task $\mathcal{Q}_{\mathcal{B}}^{\mathcal{K}}$.

The refinement lies in computing the time period of the shuffle stage. For easiness, the sort stage is merged with shuffle stage. Therefore, the shuffle stage in the remaining reduce phase is estimated as follows:

$$\mathcal{W}_{\mathcal{S}}^{\downarrow} = \left(\frac{\mathcal{Q}_{\mathcal{B}}^{\mathcal{K}}}{\mathcal{A}_{\mathcal{H}}^{\mathcal{K}}} - 1\right) \cdot \mathcal{S}_{\rightarrow}^{t} \tag{5}$$

$$\mathcal{W}_{\mathcal{S}}^{\uparrow} = \left(\frac{\mathcal{Q}_{\mathcal{B}}^{\mathcal{K}}}{\mathcal{A}_{\mathcal{H}}^{\mathcal{K}}} - 1\right) \cdot \mathcal{S}_{\rightarrow}^{t} + \mathcal{S}_{\uparrow}^{t} \tag{6}$$

Finally, taking Equation (5) and (6) together, we can formulate the lower and upper limit of the overall job completion time of $\mathcal{K}$, which is shown as follows:

$$\mathcal{W}_{\mathcal{K}}^{\downarrow} = \mathcal{W}_{\mathcal{H}}^{\downarrow} + \mathcal{S}_{\rightarrow}^{1} + \mathcal{W}_{\mathcal{S}}^{\downarrow} + \mathcal{Q}_{\mathcal{B}}^{\downarrow} \tag{7}$$

$$\mathcal{W}_{\mathcal{K}}^{\uparrow} = \mathcal{W}_{\mathcal{H}}^{\uparrow} + \mathcal{S}_{\uparrow}^{1} + \mathcal{W}_{\mathcal{S}}^{\downarrow} + \mathcal{Q}_{\mathcal{B}}^{\uparrow} \tag{8}$$

where $\mathcal{W}_{\mathcal{S}}^{\downarrow}$ depicts the optimistic prediction of job $\mathcal{K}$ completion time and $\mathcal{W}_{\mathcal{S}}^{\uparrow}$ depicts the pessimistic prediction of job $\mathcal{K}$ completion time. In section c, we compare whether the prediction that is based on mean value between lower limit and upper limits tends to be closer to measured time period. Therefore, we state:

$$\mathcal{W}_{\mathcal{K}}^{\uparrow} = \frac{\left(\mathcal{W}_{\mathcal{H}}^{\uparrow} + \mathcal{W}_{\mathcal{K}}^{\downarrow}\right)}{2} \tag{9}$$

The Equation (7) can be re-written for $\mathcal{W}_{\mathcal{H}}^{\downarrow}$by replacing parts with Eq. (3) and (5), and similar equation for sort and reduce stages as follows:

$$\mathcal{W}_{\mathcal{K}}^{\downarrow} = \frac{\mathcal{Q}_{\mathcal{H}}^{\mathcal{K}} \cdot \mathcal{H}_{\rightarrow}}{\mathcal{S}_{\mathcal{H}}^{\mathcal{K}}} + \frac{\mathcal{Q}_{\mathcal{B}}^{\mathcal{K}} \cdot (\mathcal{S}_{\rightarrow}^{t} + \mathcal{B}_{\rightarrow})}{\mathcal{S}_{\mathcal{B}}^{\mathcal{K}}} + \mathcal{S}_{\rightarrow}^{1} - \mathcal{S}_{\rightarrow}^{t} \tag{10}$$

The Equation (8) can be simplified to compute the makespan time is as follows:

$$\mathcal{W}_{\mathcal{K}}^{\downarrow} = \mathcal{X}_{\mathcal{K}}^{\downarrow} \cdot \frac{\mathcal{Q}_{\mathcal{H}}^{\mathcal{K}}}{\mathcal{S}_{\mathcal{H}}^{\mathcal{K}}} + \mathcal{Y}_{\mathcal{K}}^{\downarrow} \cdot \frac{\mathcal{Q}_{\mathcal{B}}^{\mathcal{K}}}{\mathcal{S}_{\mathcal{B}}^{\mathcal{K}}} + \mathcal{Z}_{\mathcal{K}}^{\downarrow}, \tag{11}$$

where $\mathcal{X}_{\mathcal{K}}^{\downarrow} = \mathcal{H}_{\rightarrow}$, $\mathcal{Y}_{\mathcal{K}}^{\downarrow} = (\mathcal{S}_{\rightarrow}^{t} + \mathcal{B}_{\rightarrow})$, and $\mathcal{Z}_{\mathcal{K}}^{\downarrow} = \mathcal{S}_{\rightarrow}^{1} - \mathcal{S}_{\rightarrow}^{t}$. The Equation (11), represent a makespan time of job as a function/operation of map and reduce slots assigned to job $\mathcal{K}$ for performing its map and reduce tasks, that is, as a function of $\left(\mathcal{Q}_{\mathcal{H}}^{\mathcal{K}}, \mathcal{Q}_{\mathcal{B}}^{\mathcal{K}}\right)$. In similar way$\mathcal{W}_{\mathcal{K}}^{\uparrow}$ and $\mathcal{W}_{\mathcal{K}}^{\rightarrow}$ is written as follows:

$$\mathcal{W}_{\mathcal{K}}^{\downarrow} = \mathcal{X}_{\mathcal{K}}^{\rightarrow} \cdot \frac{\mathcal{Q}_{\mathcal{H}}^{\mathcal{K}}}{\mathcal{S}_{\mathcal{H}}^{\mathcal{K}}} + \mathcal{Y}_{\mathcal{K}}^{\rightarrow} \cdot \frac{\mathcal{Q}_{\mathcal{B}}^{\mathcal{K}}}{\mathcal{S}_{\mathcal{B}}^{\mathcal{K}}} + \mathcal{Z}_{\mathcal{K}}^{\rightarrow}, \tag{12}$$

$$\mathcal{W}_{\mathcal{K}}^{\downarrow} = \mathcal{X}_{\mathcal{K}}^{\uparrow} \cdot \frac{\mathcal{Q}_{\mathcal{H}}^{\mathcal{K}}}{\mathcal{S}_{\mathcal{H}}^{\mathcal{K}}} + \mathcal{Y}_{\mathcal{K}}^{\uparrow} \cdot \frac{\mathcal{Q}_{\mathcal{B}}^{\mathcal{K}}}{\mathcal{S}_{\mathcal{B}}^{\mathcal{K}}} + \mathcal{Z}_{\mathcal{K}}^{\uparrow}, \tag{13}$$

3.3. Resource requirement estimation to meet task deadline

Here we evaluate the minimum number of map and reduce slots required to meet task deadline. To assure guaranties of task deadline of a Job $\mathcal{K}$ in time $\mathcal{W}$ we need to compute what is the minimum number of MapReduce slots needed to be allocated to meet task deadline $\mathcal{W}$ with input data size $\mathcal{J}$. For achieving it the following questionnaires needs to be considered.

$\mathcal{W}$ is considered as a lower limit of the job makespan time. Generally, this aid in reducing amount of resources allocated for job to meet task deadline $\mathcal{W}$. This setting might not be idealistic in real environment.

$\mathcal{W}$ is considered as an upper limit of the job makespan time. This will lead to over allocation of resources and might lead to very smaller job completion time than $\mathcal{W}$ because worst case scenario are very rare phenomenon in production environment.

$\mathcal{W}$ is considered as a mean between lower and upper limits on the job makespan time. This strategy may aid in providing balanced resource allocation/utilization that is closer to job makespan time $\mathcal{W}$.

The assignment of map and reduce slots to job $\mathcal{K}$ for meeting task deadline $\mathcal{W}$ considering known job profile are evaluated using variation in Equation (11), where $\mathcal{X}_{\mathcal{K}}^{\downarrow}$, $\mathcal{Y}_{\mathcal{K}}^{\downarrow}$, and $\mathcal{Z}_{\mathcal{K}}^{\downarrow}$ are defined.

$$\mathcal{X}_{\mathcal{K}}^{\downarrow} \cdot \frac{\mathcal{Q}_{\mathcal{H}}^{\mathcal{K}}}{\mathcal{S}_{\mathcal{H}}^{\mathcal{K}}} + \frac{\mathcal{Q}_{\mathcal{H}}^{\mathcal{K}}}{\mathcal{S}_{\mathcal{H}}^{\mathcal{K}}} + \mathcal{Y}_{\mathcal{K}}^{\downarrow} = \mathcal{W} - \mathcal{Z}_{\mathcal{K}}^{\downarrow} \tag{14}$$

The Equation (14) can be simplified as follows:

$$\frac{\mathscr{x}}{\mathscr{h}} \cdot \frac{\mathscr{y}}{\mathscr{b}} = \mathcal{J} \tag{15}$$

where $\mathscr{h}$ and $\mathscr{b}$depicts the number of map and reduce slots allocated to job $\mathcal{K}$ respectively, and $\mathscr{x}$, $\mathscr{y}$ and $\mathcal{J}$ depicts the corresponding expression from Equation (14).

The objective of our model is to minimize the number of map and reduce slot for job $\mathcal{K}$. i.e., we minimize $\mathcal{F}(\mathscr{h}, \mathscr{b}) = \mathscr{h} + \mathscr{b}$ over $\frac{\mathscr{x}}{\mathscr{h}} \cdot \frac{\mathscr{y}}{\mathscr{b}} = \mathcal{J}$. We consider Lagrange multiplier and set $\mathcal{L} = \mathscr{h} + \mathscr{b} + \varphi \frac{\mathscr{x}}{h} + \varphi \frac{\mathscr{y}}{\mathscr{b}} - \mathcal{J}$. By differentiating $\mathcal{L}$ with respect to $\mathscr{h}$, $\mathscr{b}$ and φ and equating to zero, we obtain,

$$\frac{\partial \mathcal{L}}{\partial \mathscr{h}} = 1 - \varphi \frac{\mathscr{x}}{h^2} = 0 \tag{16}$$

$$\frac{\partial \mathcal{L}}{\partial \mathscr{b}} = 1 - \varphi \frac{\mathscr{x}}{\mathscr{b}^2} = 0 \tag{17}$$

$$\frac{\partial \mathcal{L}}{\partial \varphi} = \frac{\mathscr{x}}{\mathscr{h}} + \frac{\mathscr{y}}{\mathscr{b}} - \mathcal{J} = 0 \tag{18}$$

Solving Equation (16), (17) and (18) simultaneously, we obtain,

$$\mathscr{h} = \frac{\sqrt{\mathscr{x}}\left(\sqrt{\mathscr{x}} + \sqrt{\mathscr{y}}\right)}{\mathcal{J}}, \quad \mathscr{b} = \frac{\sqrt{\mathscr{x}}\left(\sqrt{\mathscr{x}} + \sqrt{\mathscr{y}}\right)}{\mathcal{J}} \tag{19}$$

Using these equation the optimal value of map and reduce slot are obtained such that the number of slots is minimized while meeting task deadline constraint. Here we round up the values obtained from these equation for approximation. Since these values have to be integral.

In next section the performance evaluation of proposed scheduler over state of art technique is shown.

4. RESULT AND ANALYSIS

This section present performance evaluation of proposed OHMR over state-of-art Hadoop MapReduce Framework [11]. Hadoop is the most widely used/adopted MapReduce platform for computing on cloud environments [17], hence it is considered for comparisons. Hadoop 2.0 i.e. version 2.7 is used and is deployed on azure cloud using HDInsight. The Hadoop cluster is composed of one master worker node and four worker/slave nodes. Each worker node is deployed on A3 virtual machine instances which composed of 4 virtual computing cores, 7 GB RAM and 120 GB of storage space. Uniform configuration is considered for both OHMR and HMR. For experiment analysis different application are considered such as Gene sequencing (Bioinformatics), Word frequency statistics computation and Hot-word detection.

4.1. Gene sequencing

Gene sequence alignment is a fundamental operation adopted to identify similarities that exist between a query protein sequence, DNA or RNA and a database of sequences maintained. Sequence alignment is computationally heavy and its computation complexity is relative to product of two sequences being currently analyzed. Massive volumes of sequences maintained in the database to be searched induces additional computation burden. BLAST is a widely adopted bioinformatics tool for sequence alignment which perform faster alignments, at expense of accuracy (possibly missing some potential hits) [18]. Drawbacks of BLAST and its improvements is discussed in [19]. For evaluation here the improved BLAST algorithm of [19] is adopted. To improve computation time a heuristic strategy is used compromising accuracy minimally. In the heuristic strategy initial match is found and is later extended to obtain the complete matching sequence.

Experiment are conducted to evaluate OHMR and HMR performance for performing gene sequence alignment. The dataset for experiment analysis is obtained from NCBI [20]. For performing alignment Drosophila database as a reference database and Query sequence of varied sizes of from Homo sapiens chromosomal sequences and genomic scaffolds is considered similar to [19] which are tabulated in Table 1. All six experiment are conducted using BLAST algorithm on HMR and OHMR frameworks. The total makespan time of both HMR and OHMR for all six experiment is noted and graph is plotted as shown in Figure 2. It must be noted that the initialization time of the VM cluster is not considered is computing makespan as it is uniform in both OHMR and HMR owing to similar cluster configurations.

The total makespan of OHMR and HMR is dependent on task execution time of virtual computing/worker nodes during Map and Reduce phase. The total makespan observed in BLAST sequence alignment experiments executed on HMR and OHMR frameworks is shown in Figure 2. The outcomes shows significant performance in terms of reduce makespan times of OHMR over HMR. A makespan reduction of 43.44%, 44.85%, 56.9%, 57.17%, 62.83% and 65.01% is obtained for six experiment by OHMR over HMR. An average makespan reduction of 55.03% is achieved by OHMR over HMR across all experiments.

Theoretical makespan of OHMR i.e., $\mathcal{W}$ given by Equation (11) is computed and compared against the practical values observed in all the experiments. Results obtained is shown in Figure 3. Minor variations is observed between practical and theoretical makespan computations. Overall good correlation is reported between practical makespan values and theoretical makespan values. Based on the results presented it is evident that execution of BLAST sequence alignment algorithm on proposed OHMR yields superior results when compared to similar experiments conducted on existing HMR framework. Accuracy and correctness of theoretical makespan model of OHMR presented is proved through correlation measures.

Table 1. Simulation parameter considered

Experiment Id	Query genome	Query genome size	Reference genome	Reference genome size
1	NT_007914	14866257	Drosophila database	122,653,977
2	AC_000156	19317006	Drosophila database	122,653,977
3	NT_011512	33734175	Drosophila database	122,653,977
4	NT_033899	47073726	Drosophila database	122,653,977
5	NT_008413	43212167	Drosophila database	122,653,977
6	NT_022517	90712458	Drosophila database	122,653,977

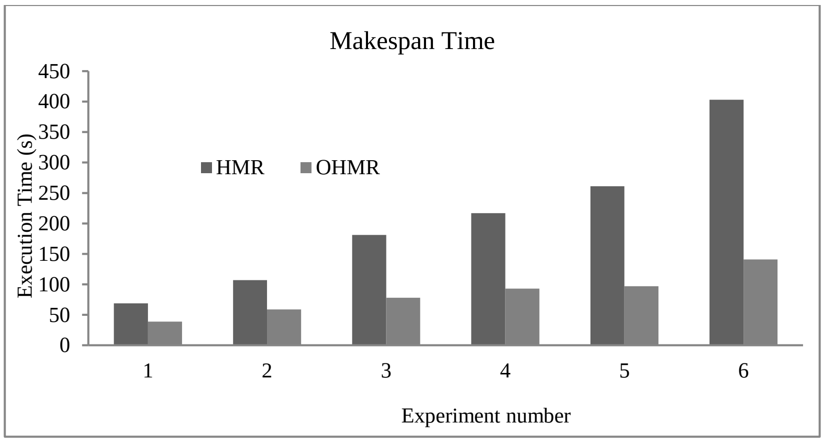

Figure 2. BLAST sequence alignment total makespan time observed for experiments conducted on OHMR and HMR frameworks

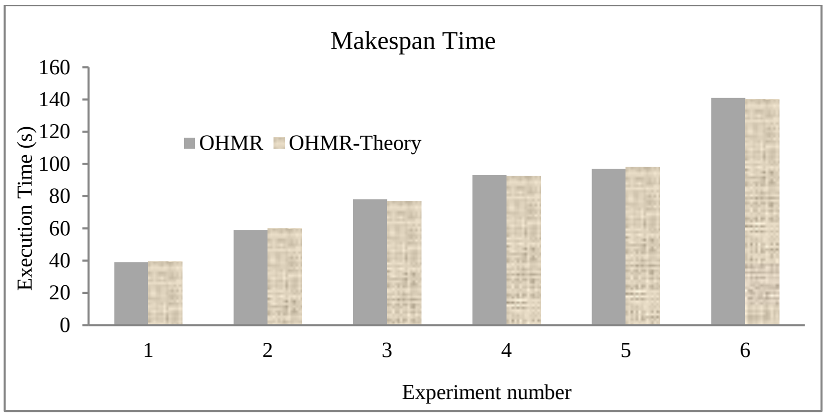

Figure 3. Correlation between theoretical and practical makespan times for BLAST sequence alignment execution on OHMR framework

4.2. Word frequency statistics computations

The word frequency statistic application is developed using Java programing language. The Wikipedia dataset [21] is considered for experiment analysis. The Wikipedia dataset is huge in size (i.e. >100 GB) and is split into2048 MB each and stored in Azure cloud container. For experimental analysis this work consider 16GB of data. The word frequency statistics applications were executed on the OHMR and HMR framework and the results obtained are noted. The outcomes shows significant performance in terms of reduce makespan times of OHMR over HMR. A makespan reduction of 43.7%, 44.34%, 45.69% and 51.57% is obtained for data size of 2048 MB, 4096 MB, 8192 MB and 16384 MB respectively by OHMR over HMR. An average makespan reduction of 46.39% is achieved by OHMR over HMR across all experiments.

Theoretical makespan of OHMR i.e., $\mathcal{W}$ given by Equation (11) is computed and compared against the practical values observed in all the experiments. Results obtained is shown in Figure 5. Minor variations is observed between practical and theoretical makespan computations. Overall good correlation is reported between practical makespan values and theoretical makespan values. Based on the results presented it is evident that execution of word frequency statistic application on proposed OHMR yields superior results when compared to similar experiments conducted on existing HMR framework. Accuracy and correctness of theoretical makespan model of OHMR presented is proved through correlation measures.

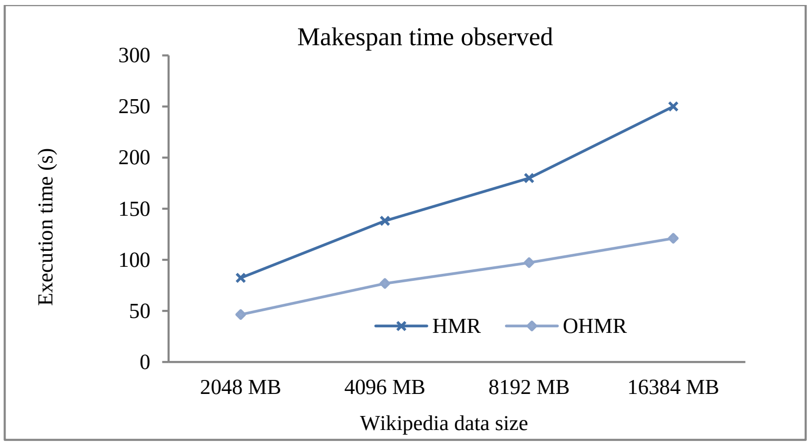

Figure 4. Word frequency statistic application total makespan time observed for experiment conducted on OHMR and HMR frameworks

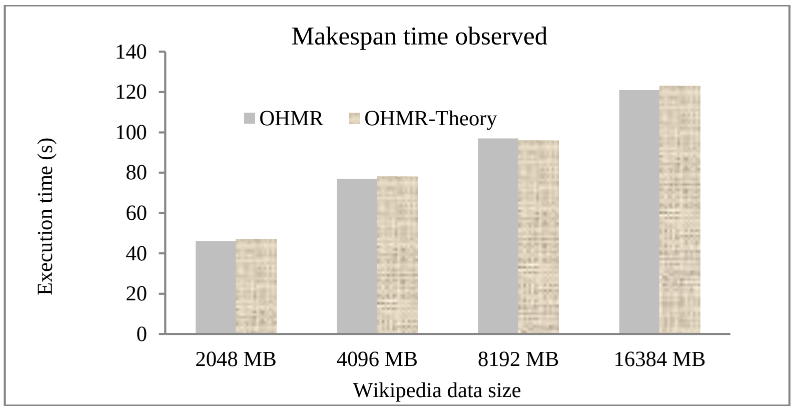

Figure 5. Correlation between theoretical and practical makespan times for word frequency statistic application execution on OHMR framework

4.3. Hot-word detection computations

The hot-word detection algorithm [22] is developed using Java programing language. The “Movietweetings” dataset [23] is considered for experiment analysis and stored in Azure cloud container. Tweets consisting of 20000, 40000, 60000 and 80000 movies is considered and is represented as 20K, 40K, 60K and 80K. The hot-word detection algorithm were executed on the OHMR and HMR framework and the results obtained are noted. The total makespan time of OHMR and existing model is noted and is shown in Figure 6. Experiment analyses shows as number of tweets increases the computation time of both OHMR and HMR increases. The outcomes shows significant performance in terms of reduce makespan times of OHMR over HMR. A makespan reduction of 54.19%, 45.13%, 60.68% and 54.69% is obtained for tweet size of 20K, 40K, 60K and 80K respectively by OHMR over HMR. An average makespan reduction of 53.67% is achieved by OHMR over HMR across all experiments.

Theoretical makespan of OHMR i.e., $\mathcal{W}$ given by Equation (11) is computed and compared against the practical values observed in all the experiments. Results obtained is shown in Figure 7. Minor variations is observed between practical and theoretical makespan computations. Overall good correlation is reported between practical makespan values and theoretical makespan values. Based on the results presented it is evident that execution of Hot-word detection on proposed OHMR yields superior results when compared to similar experiments conducted on existing HMR framework. Accuracy and correctness of theoretical makespan model of OHMR presented is proved through correlation measures.

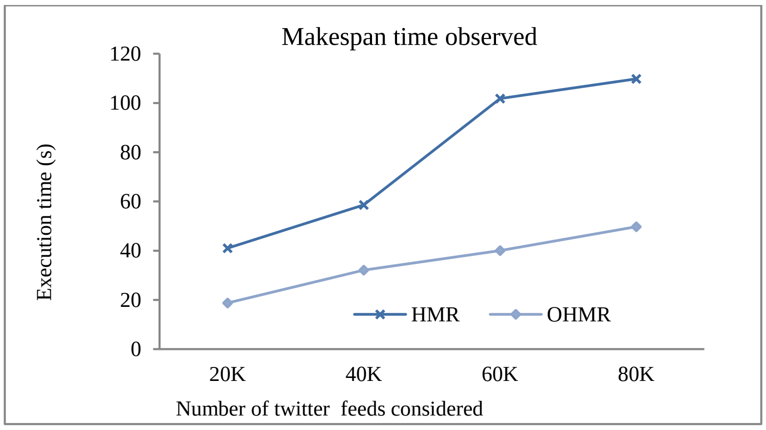

Figure 6. Hot-word detection total makespan time observed for experiment conducted on OHMR and HMR framework

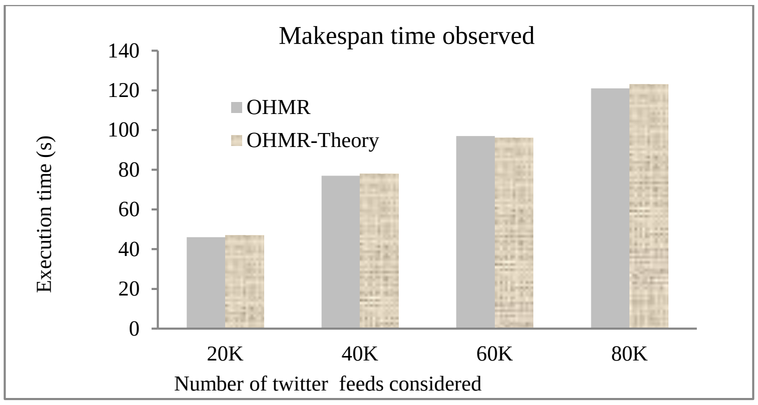

Figure 7. Correlation between theoretical and practical makespan times for BLAST sequence alignment execution on OHMR framework

In this section the execution of the imprecise and bioinformatics applications namely word frequency statistics, hot word detection, and gene sequencing (BLAST) is presented. The results presented here prove that the OHMR model reduces the makespan observed due to the optimized makespan model incorporated in to HMR. An average reduction of 53.67% for word frequency statistics and 46.39% for the hot word detection is reported and 53.67% for the gene sequencing (BLAST) considering the OHMR model when compared to the existing HMR model [11]. The cumulative analysis over state-of-art technique in Table II shows the efficiency of OHMR over state-of-art technique in terms of robustness and scalability. Since, OHMR support execution of different application such as Bioinformatics and text mining over cloud platforms. Our OHMR makespan model aided in better cloud resource utilization. Theoretical comparison evaluation is considered and attained better result when compared with [12] and [14]. Adoption cloud platform aid in proving scalability of processing of large amount of data of various types on large computing clusters. All these feature attributed to the performance improvement of OHMR over state-of-art models.

Table 2. Comparison with state of art technique

	[11]	**[12]**	**[13]**	**[14]**	**[15]**	**OHMR**
MapReduce platform considered	Hadoop	Hadoop	Hadoop	Hadoop	Hadoop	Hadoop
Cloud adopted	Yes	NO	Yes	Yes	No	Yes
Application considered	Bioinformatics	Word count	Word count and Tera sort	Word count and Sort	Word count and Sort	Bioinformatics and text mining
Makespan accuracy evaluation considered	No	Yes	No	Yes	No	Yes
Average percentage improvement over HMR framework	40.28%	13.33%	34.83%	27.7%	43.91%	51.16%

5. CONCLUSION

The significance of cloud computing platforms is discussed. Commonly adopted Hadoop map reduce framework working with its drawbacks is presented. To lower makespan times and enable effective utilization of cloud resources this paper proposes an OHMR framework. The main contribution of this work is presenting an accurate and efficient makespan model for Hadoop MapReduce framework. The amount resource required to meet task deadline is done based makespan model presented here. To evaluate the performance of proposed OHMR framework computationally heavy bioinformatics application and imprecise application such as word frequency statistics and hot word detection is considered. Performance of OHMR framework is compared with HMR framework in terms of makespan time. Average overall makespan times reduction of 55.03%, 46.39, and 53.67% is achieved using OHMR framework when compared to HMR framework for BLAST, word frequency statistics, and hot word detection applications. Experiments presented prove robustness of OHMR framework, its capability to handle diverse applications on public and private cloud platforms. Results presented through experiments conducted prove superior performance of OHMR against Hadoop framework. Good matching is reported between the theoretical makespan of OHMR presented and experimental values observed.

The future work would consider performance evaluation considering different application and also would further consider optimization of MapReduce scheduler for further reduction of computation time. We also consider presenting accurate and fast gene sequencing and novel bioinformatics applications. Then, evaluate the performance of OHMR considering different performance parameters.

REFERENCES

[1] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” ACM Commun., vol. 51, no. 1, pp. 107–113, Jan. 2008.

[2] “Apache Hadoop.” [Online]. Available: http://hadoop.apache.org/. [Accessed: 21-Oct-2017].

[3] K. Taura, T. Endo, K. Kaneda, and A. Yonezawa, “Phoenix: a parallel programming model for accommodating dynamically joining/leaving resources,” in SIGPLAN Not., 2003, vol. 38, no. 10, pp. 216–229.

[4] ZuKuan Wei1, Bo Hong, JaeHong Kim, “A New Memory MapReduce Framework for Higher Access to Resources”, Indonesian Journal of Electrical Engineering and Computer Science Vol. 4, No. 3, December 2016, pp. 629 ~ 636.

[5] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, “Dryad: distributed data-parallel programs from sequential building blocks,” ACM SIGOPS Oper. Syst. Rev., vol. 41, no. 3, pp. 59–72, Mar. 2007.

[6] U. Kang, C. E. Tsourakakis, and C. Faloutsos, “PEGASUS: Mining Peta-scale Graphs,” Knowl. Inf. Syst., vol. 27, no. 2, pp. 303–325, May 2011.

[7] Ning Chen, Chai Xiangyang, “Investigation of Distributed Search Engine Based on Hadoop”, TELKOMNIKA, Indonesian Journal of Electrical Engineering Vol. 12, No. 9, September 2014, pp. 6954 ~ 6957.

[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] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] Satish Londhe, Smita Mahajan, “Effective and Efficient Way of Reduce Dependency on Dataset with the Help of Mapreduce on Big Data”, TELKOMNIKAIndonesian Journal of Electrical Engineering Vol. 15, No. 1, July 2015, pp. 171 ~ 176.

[17] T. White, Hadoop: The Definitive Guide. O’Reilly Media, 2009.

[18] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. Journal of molecular biology, 215(3):403–410, 1990.

[19] K. Mahadik, S. Chaterji, B. Zhou, M. Kulkarni and S. Bagchi, "Orion: Scaling Genomic Sequence Matching with Fine-Grained Parallelization," SC14: International Conference for High Performance Computing, Networking, Storage and Analysis, New Orleans, LA, 2014, pp. 449-460.

[20] National Center for Biotechnology Information. (2015). [Online]. Available : http://www.ncbi.nlm.nih.gov/

[21] Kajdanowicz, T.; Indyk, W.; Kazienko, P.; Kukul, J., "Comparison of the Efficiency of MapReduce and Bulk Synchronous Parallel Approaches to Large Network Processing," Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on , vol., no., pp.218,225, 10-10 Dec. 2012.

[22] S. Dooms, T. De Pessemier, and L. Martens, “Movietweetings: a movie rating dataset collected from twitter,” in Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys, vol. 13, 2013.

[23] G. Zhai, L. Tian, Y. Zhou, Q. Sun and J. Shi, "A computing resource adjustment mechanism for communication protocol processing in centralized radio access networks," in China Communications, vol. 13, no. 12, pp. 79-89, December 2016.