IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 6, June 2020

Efficent Makespan Model for Hadoop MapReduce Framework for Provisioning Bioinformatic Applications

Vinutha D C[1], G T Raju[2]

Associate Professor, Department of ISE, Vidyavardhaka College of Engineering, Mysuru, Visvesvaraya Technological University, Karnataka[1]

Professor, Department of CSE, RNS Institute of Technology, Bengaluru, Visvesvaraya Technological University, Karnataka[2]

Abstract: Increasing demand to provision real-time smart application services has led many organization to collect data continuously. As a result has led to huge amount of size. MapReduce is the preferred framework to process massive data. Hadoop is a widely used MapReduce framework across 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. Minimizing execution time on such platform is most desired. 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. Experiment are conducted on Microsoft Azure HDInsight cloud platform considering 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.

Keywords: Big data, Cloud computing, Hadoop, MapReduce, Parallel computing, Schedular.

I. INTRODUCTION

Many organizations are continuously collecting massive amounts of datasets from various sources such as the World Wide Web, sensor networks and social networks. The ability to perform scalable and timely analytics on these unstructured datasets is a high priority task for many enterprises. It has become difficult for traditional network storage and database systems to process these continuously growing datasets. MapReduce [1], originally developed by Google, has become a major computing model in support of data intensive applications. It is a highly scalable, fault-tolerant and data parallel model that automatically distributes the data and parallelizes the computation across a cluster of computers [2]. Among its implementations such as Mars [3], Phoenix [4], Dryad [5] and Hadoop [6], Hadoop has received a wide uptake by the community due to its open source nature [7].

One feature of Hadoop MapReduce is its support of public cloud computing that enables the organizations to utilize cloud services in a pay-as-you-go manner. 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.

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.

The HDInsight Cloud makes it easier for users to set up and run Hadoop applications on a large-scale virtual cluster. To use the HDInsight Cloud, users have to configure the required amount of resources (virtual nodes) for their applications.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 9, Issue 6, June 2020

However, the HDInsight Cloud in its current form does not support Hadoop jobs with deadline requirements. It is purely the user's responsibility to estimate the amount of resources to complete their jobs which is a highly challenging task. Hence, Hadoop performance modeling has become a necessity in estimating the right amount of resources for user jobs with deadline requirements. It should be pointed out that modeling Hadoop performance is challenging because Hadoop jobs normally involve multiple processing phases including three core phases (i.e. map phase, shuffle phase and reduce phase). Moreover, the first wave of the shuffle phase is normally processed in parallel with the map phase (i.e. overlapping stage) and the other waves of the shuffle phase are processed after the map phase is completed (i.e. non-overlapping stage).

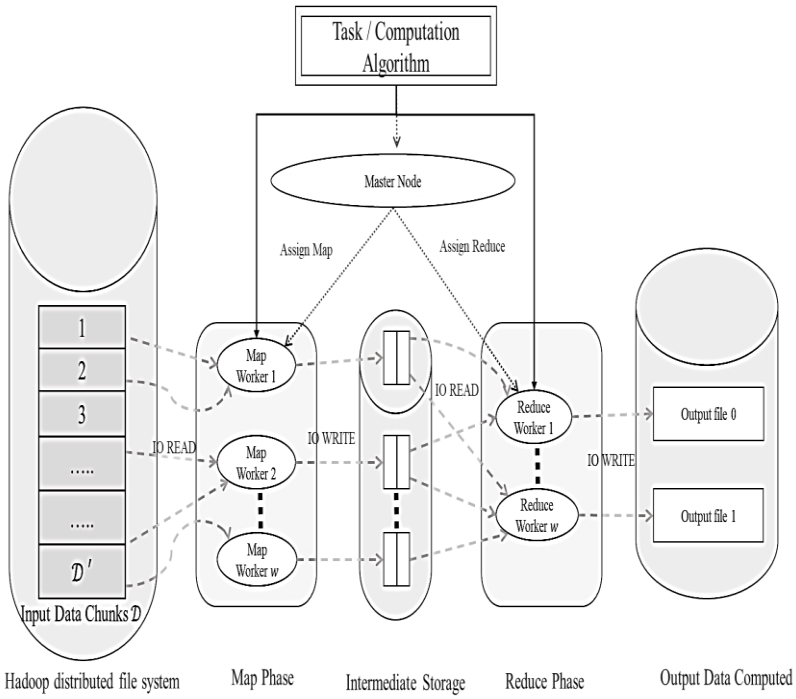

Fig. 1. Architecture of Hadoop MapReduce Framework

To effectively manage cloud resources, several Hadoop performance models have been proposed [8], and [9]. However, these models do not consider the overlapping and non-overlapping stages of the shuffle phase which leads to an inaccurate estimation of job execution.

Recently, a number of sophisticated Hadoop performance models are proposed [10], [11], [12], [13], [14], and [15]. Starfish [10] collects a running Hadoop job profile at a fine granularity with detailed information for job estimation and optimization. On the top of Starfish, Elasticiser [11] 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 [12], [13], and [14] 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 [15] 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.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 9, Issue 6, June 2020

a) Research Contribution are as follows:

- This work presents an accurate makespan model for HMR aiding performance improvement.
- Experiments considering diverse cloud configurations and varied application configuration.
- Correlation between theoretical makespan model and experimental values

The rest of the paper is organized as follows. In section II the proposed makespan model is presented. In penultimate section experimental study is carried out. The conclusion and future work is described in last section.

II. MAKESPAN MODELLING FOR PROPOSED OPTIMIZED SCHEDULAR FOR HADOOP MAPREDUCE FRAMWORK

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

a) 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 considers 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 $\left(\sum_{n=1}^{q-1} \mathcal{W}_n\right)\Big/_{j} \leq (q-1)\cdot{}^{\beta}/_{j}$. Therefore, total execution time of all assignment is at least $(q-1)\cdot{}^{\beta}/_{j+\varphi}$. 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 $q\cdot{}^{\beta}/_{j}$. 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 $\varphi \ll {}^{q\cdot\beta}/_{j}$.

b) 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 $\mathcal{Q}_{\mathcal{H}}^{\mathcal{K}}$ map tasks and $\mathcal{Q}_{\mathcal{B}}^{\mathcal{K}}$ 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

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

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

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

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

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

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 9, Issue 6, June 2020

Finally, taking Eq. (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 Eq. (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 Eq. (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 Eq. (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 $\mathcal{W}_{\mathcal{K}}^{\rightarrow}$ can be written.

c) 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 $\mathcal{I}$. For achieving it, this work considers $\mathcal{W}$ 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 Eq. (11), where $\mathcal{X}_{\mathcal{K}}^{\downarrow}$, $\mathcal{Y}_{\mathcal{K}}^{\downarrow}$, and $\mathcal{Z}_{\mathcal{K}}^{\downarrow}$ are defined.

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

The Eq. (12) can be simplified as follows

$$\frac{x}{\hbar} \cdot \frac{y}{\mathcal{b}} = \mathcal{I} \tag{13}$$

where $\hbar$ and $\mathcal{b}$depicts the number of map and reduce slots allocated to job $\mathcal{K}$ respectively, and x, y and $\mathcal{I}$ depicts the corresponding expression from Eq. (12).

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

$$\frac{\partial \mathcal{L}}{\partial \hbar} = 1 - \varphi \frac{x}{h^2} = 0 \tag{14}$$

$$\frac{\partial \mathcal{L}}{\partial \mathcal{b}} = 1 - \varphi \frac{x}{\mathcal{b}^2} = 0 \tag{15}$$

$$\frac{\partial \mathcal{L}}{\partial \varphi} = \frac{x}{\hbar} + \frac{y}{\mathcal{b}} - \mathcal{I} = 0 \tag{16}$$

Solving Eq. (14), (15) and (16) simultaneously, we obtain

$$\hbar = \frac{\sqrt{x}\left(\sqrt{x} + \sqrt{y}\right)}{\mathcal{I}}, \quad \mathcal{b} = \frac{\sqrt{x}\left(\sqrt{x} + \sqrt{y}\right)}{\mathcal{I}} \tag{17}$$

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.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 6, June 2020

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

III. Simulation Result and Analysis

This section presents performance evaluation of proposed OHMR over state-of-art Hadoop MapReduce Framework [16]. 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 Bioinformatics application are considered such as Gene sequencing (BLAST) [18] and CAP3 sequence assembly [19]. The dataset for experiment analysis is obtained from NCBI [20]

a) Gene Sequencing performance :

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 [21]. For evaluation here the improved BLAST algorithm of [21] 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.

Experiments are conducted to evaluate OHMR and HMR performance for performing gene sequence alignment. 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 [21] which are tabulated in Table I. All six experiments are conducted using BLAST algorithm on HMR and OHMR frameworks. The total makespan time of both HMR and OHMR for all six experiments is noted and graph is plotted as shown in Fig. 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 Fig. 2. The outcomes shows significant performance in terms of reduces 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 are shown in Fig. 3. Minor variations are 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 I. Information of the Genome Sequences used as queries considering equal section lengths from the Homo sapiens chromosome 15 as reference

Experiment Id	Query genome	Query genome size	Experiment Id	Reference genome
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

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 9, Issue 6, June 2020

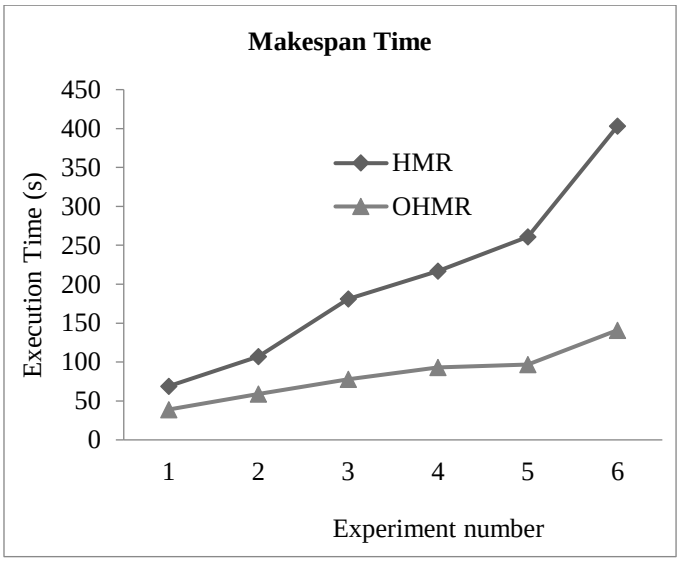

Fig. 2. BLAST sequence alignment total makespan time observed for experiments conducted on OHMR and HMR frameworks

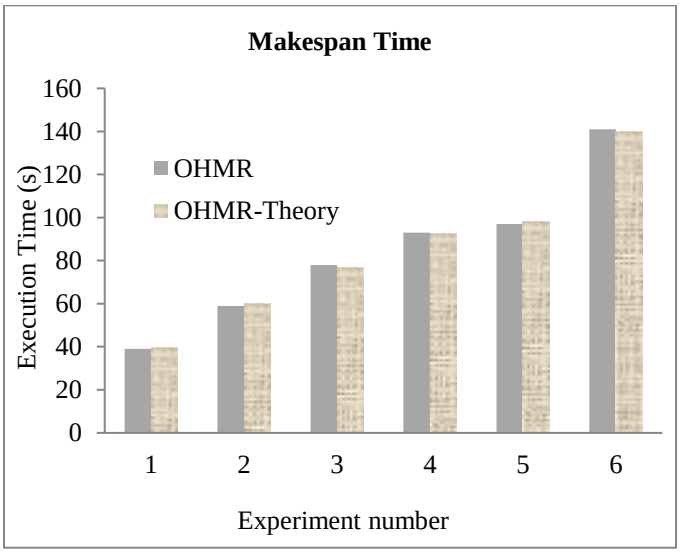

Fig. 3. Correlation between theoretical and practical makespan times for BLAST sequence alignment execution on OHMR framework

b) CAP3 assembly performance :

DNA sequence assembly tools are used in bioinformatics for gene discovery and understanding genomes of existing/new organisms. CAP3 is one such tool popular used to assemble DNA sequences. DNA assembly is achieved by performing merging and aligning operations on smaller sequence fragments to build complete genome sequences. CAP3 eliminates poor sections observed within DNA fragments, computes overlaps amongst DNA fragments, is capable of identifying false overlaps, eliminating false overlaps identified, accumulates fragments of multiple or one overlapping DNA segments to produce contigs and performs multiple sequence alignments to produce consensus sequences. CAP3 reads multiple gene sequences from an input FASTA file and generates output consensus sequences written to multiple files and also to standard outputs. Detailed explanation of the CAP3 gene sequence assembly is provided in [19].

The experiments conducted in CAP3 gene sequence assembly is directly adopted in Map phase of OHMR and Hadoop. In the reduce phase result aggregation is considered. To evaluate the performance of CAP3 execution on OHMR and Hadoop frameworks Homo sapiens chromosome 15 is considered as a reference. Genome sequences of various sizes are considered as queries and submitted to Azure cloud platform in the experiments. Query sequences for experiments are considered in accordance to [19]. CAP3 experiments conducted with query genomic sequences (BAC datasets) are summarized in Table II. All four experiments are conducted using CAP3 algorithm on Hadoop and OHMR frameworks. The total makespan time of both HMR and OHMR for all 4 experiments is noted and graph is plotted as shown in Fig. 4.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 6, June 2020

It must be noted that the initialization time of the VM cluster is not considered in 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 CAP3 sequence assembly experiments executed on HMR and OHMR frameworks is shown in Fig. 4. The outcomes shows significant performance in terms of reduces makespan times of OHMR over HMR. A makespan reduction of 30.4%, 24.12%, 16.69%, and 26.70% is obtained for six experiments by OHMR over HMR. An average makespan reduction of 24.47% 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 are shown in Fig. 5. Minor variations are 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 CAP3 sequence assembly 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 II. INFORMATION OF THE GENOME SEQUENCES USED IN CAP3 EXPERIMENTS

Experiment Number	Data set	GeneBank accession number	Number of reads (bp)	Average Length of reads (bp)	Length of provided sequences (bp)
1	203	AC004669	1812	598	89779
2	216	AC004638	2353	614	124645
3	322F16	AF111103	4297	1011	159179
4	526N18	AF123462	3221	965	180182

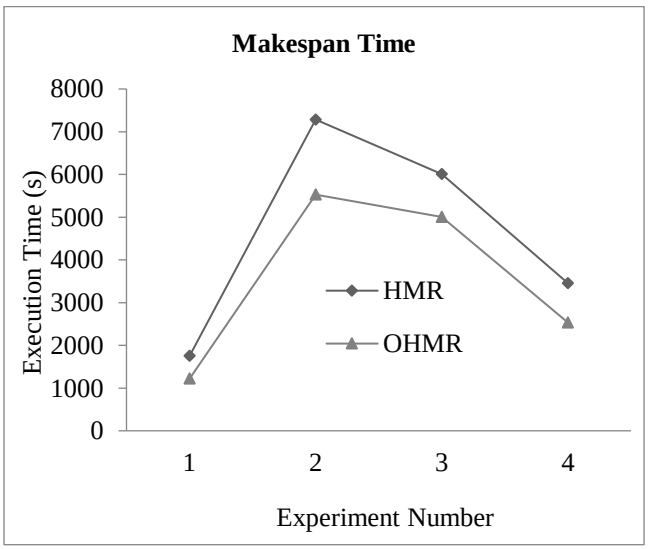

Fig. 4. CAP3 sequence assembly alignment total makespan time observed for experiments conducted on OHMR and HMR frameworks

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 6, June 2020

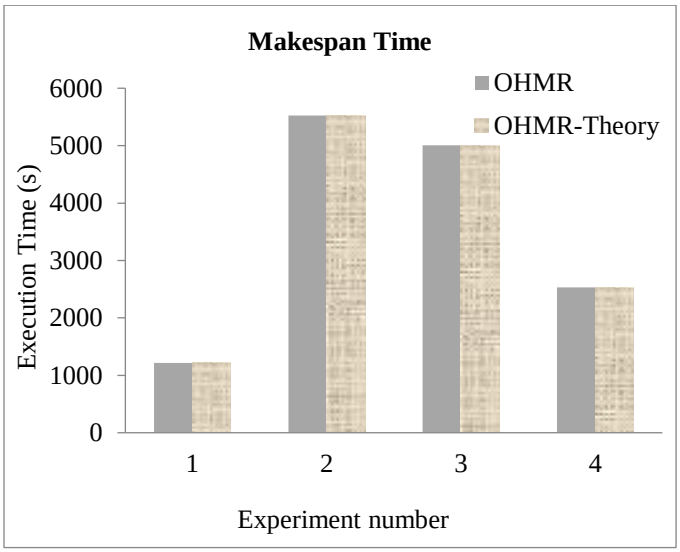

Fig. 5. Correlation between theoretical and practical makespan times for CAP3 sequence assembly execution on OHMR framework

IV. 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 is considered such as BLAST and CAP3. Performance of OHMR framework is compared with HMR framework in terms of makespan time. Average overall makespan times reduction of 54.03%, and 24.47% is achieved using OHMR framework when compared to HMR framework for BLAST and CAP3 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.

REFERENCES

[1] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008.
[2] R. Lämmel, “Google’s MapReduce programming model — Revisited,” Sci. Comput. Program., vol. 70, no. 1, pp. 1–30, 2008.
[3] 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.
[4] 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.
[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] “Apache Hadoop.” [Online]. Available: http://hadoop.apache.org/. [Accessed: 21-Oct-2013].
[7] 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.
[8] 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.
[9] 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.
[10] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu, “Starfish: A Self-tuning System for Big Data Analytics,” in In CIDR, pp. 261–272, 2011.
[11] H. Herodotou, F. Dong, and S. Babu, “No One (Cluster) Size Fits All: Automatic Cluster Sizing for Data-intensive Analytics,” in Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC ’11), pp. 1–14, 2011.
[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, March 21, 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] 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, Nov. 1 2017.
[15] K. Chen, J. Powers, S. Guo, and F. Tian, “CRESP: Towards Optimal Resource Provisioning for MapReduce Computing in Public Clouds,” IEEE Transcation Parallel Distrib. Syst., vol. 25, no. 6, pp. 1403 – 1412, 2014.

[16] 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.
[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] Xiaoqiu Huang and Anup Madanin, "CAP3: A DNA Sequence Assembly Program," in Genome Research, 9(9):868-77, 1999.
[20] National Center for Biotechnology Information. (2015). [Online]. Available : http://www.ncbi.nlm.nih.gov/
[21] 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.