Deadline Constraint Aware Scheduler for Executing High Performance Computing Application on Hadoop MapReduce Framework

D C Vinutha, G T Raju

Abstract: ***MapReduce (MR) is a parallel computing programing framework used for executing scientific and data-intensive High Performance Computing (HPC) application in parallel nature using Hadoop platform. Certain jobs come with Service Level Agreement (SLA) prerequisite for their job computation. The state-of-art SLA aware MR scheduler methods do not consider the problems of dynamic makespan time and varying virtual computing machine performance. Hence, this paper presents Deadline Constraint Aware Scheduler (DCAS) for Hadoop MapReduce Framework. The DCAS can obtain the optimum results for the scheduling of deadline constrained problems using Hadoop job history logs. The DCAS makespan model is designed by considering heterogeneous Hadoop framework. This model assumes that some of the virtual computing machines cannot guarantee SLA prerequisite. Further, DCAS takes data locality into consideration for allocating resources to reduce the makespan time of data access. However, the available resources cannot guarantee SLA prerequisite of all jobs. Hence the proposed DCAS makes an attempt to guarantee the SLA prerequisite of all jobs. From extensive analysis it can be seen that no prior work has considered dynamic scientific and data-intensive computing on HMR framework. Further, no prior work has considered alignment considering long read genomic sequence alignment. DCAS model offers parallel execution of gene sequence alignment process under multi core environment. Thus, aid in improving resource utilization. Experimental analysis on the proposed approach has been carried out on gene sequence alignment using BWA-SW, CAP3 assembly, and text mining applications. Experimental results revealed that an average makespan performance (resources utilization) improvement of 43.95%, 42.33%, and 52.52% is achieved using DCAS when compared to HMR framework for Cap3, gene sequence alignment, and text mining applications.***

Keywords: ***Big data, Cloud Computing, Hadoop, High performance computing, MapReduce, SLA, Task scheduler QoS.***

I. INTRODUCTION

Hadoop framework offers scalable storage and computing resources on user demand by connecting large density of servers through network. Thus, scientific and data intensive high performance computing applications are developed using Hadoop framework. The scientific and data intensive applications generally requires huge amount of data, for example stock market analysis, text mining, business intelligence, bioinformatics (gene sequence analysis) and so on. For running these applications, Google came up with MapReduce (MR) Framework which is designed using parallel programing model [1]. Hadoop is a well-known MapReduce framework used across various organizations due to its open source nature. The architecture of Hadoop MapReduce (HMR) Framework is presented in Figure. 1. In Hadoop MapReduce, a job is divided into two stages, namely Map and Reduce stage. These jobs are segmented into chunks and are collaboratively executed across different virtual computing platform.

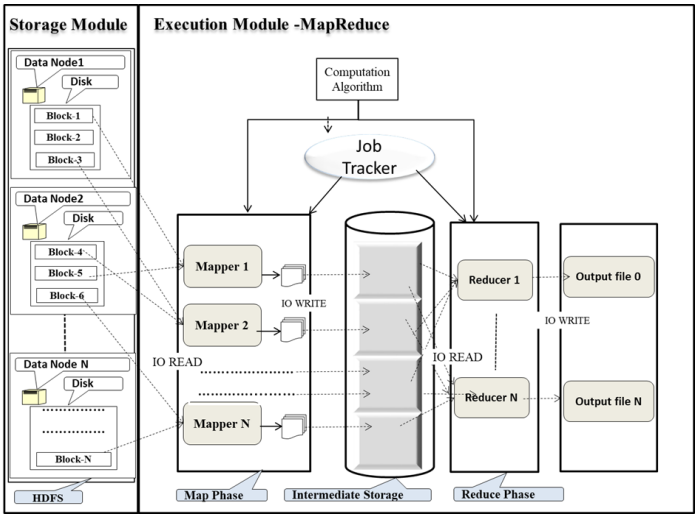


Fig 1. Architecture of Hadoop MapReduce Framework [19]

Number of scientific and data intensive jobs can be simultaneously executed using Hadoop Framework. HMR framework has number of schedulers [2] such as Fair, Capacity, and FIFO which can be used to allocate resources to several concurrent jobs. Further, number of optimization methods of scheduling metric has been presented to improve resource management in HMR [10, 11].Executing jobs with SLA prerequisite in HMR is a challenging and difficult task. Thus, designing an efficient makespan model is a key for estimating resources needed for meeting jobs' SLA prerequisite. More details of state-of-art makespan model along with the performance requirements are discussed in next section.

Revised Manuscript Received on June 05, 2019

D C VINUTHA, Research Scholar, Dept. of CSE, RNS Institute of Technology, Bengaluru, Associate Professor, Dept. of ISE Vidyavardhaka College of Engineering, Mysuru.Visvesvaraya Technological University, Belagavi, Karnataka.

G T RAJU, Professor, Dept. of CSE, RNS Institute of Technology, Bengaluru, Visvesvaraya Technological University, Belagavi, Karnataka.

- **Heterogeneity of resources/slot performance:** Generally, the resource of a MR computing model assumed to be homogeneous in nature. However, in actual scenario slots depicts a part of virtual computing machine [17]. In Hadoop environment, the heterogeneity of virtual computing machine is expected as it is connected to large set of virtual computing machines [18]. Further, it is practically difficult to expect similar performance across different virtual computing machines (i.e., resource availability of memory, CPU, and disks). Considering the heterogeneous computing cluster environment, the slots available with each computing worker will possess different resources. Further, considering shorter makespan job with SLA prerequisite, if the MR tasks are assigned with less resource, the job might meet with SLA prerequisite.

- **Dynamic task SLA configuration:** In state-of-art SLA aware MR scheduler [12, 13], the jobs' SLA requirement is partitioned into set of task SLAs using a static method. However, to enhance resource utilization and minimize SLA violation in the proposed method, we consider dynamic SLA requirement configuration setup. This work split the jobs' SLA prerequisite as map tasks SLA perquisite and the reduce tasks SLA perquisite is computed using map and reduce makespan parameter. In existing scheduling model, when completing MR task, the actual makespan time of such task is returned to reduce or increase the actual SLA of awaiting MR tasks. However, the proposed DCAS method dynamically identifies the suitable resources for executing task that require longer execution time with SLA prerequisite. Thus, DCAS is adaptive in nature that uses the resources efficiently (i.e., without utilizing more resources for executing tasks that require longer makespan time with SLA prerequisite). Also, the jobs' makespan time can be optimized to avoid SLA violation.

- **Merging job SLA requirement and data locality awareness:** For attaining locality awareness for scheduling, it is important to consider MR task with its data residing on local worker. However, such assumption is made. There is high probability that long running tasks with SLA prerequisite are processed by high computing resources. Thus, there won't be enough resource available for executing shorter tasks that are executed later. The DCAS considers weak data locality awareness for MR task, if SLA prerequisite of that job has longer execution time. Thus resource utilization can be improved by allowing MR task with shorter makespan time by allocating additional resources to it.

The contributions of this work are as follows:

- Proposing Deadline Constraint aware scheduler for Hadoop MapReduce framework.
- Applying DCAS for parallel genomic sequence alignment utilizing system resources efficiently.
- Conducting Experimental Analysis on proposed DCAS for reducing makespan and compare the performance with existing HMR framework.

The rest of the paper is organized as follows. In section II, the literature survey is presented. The section III presents Deadline Constraint aware scheduler for Hadoop MapReduce framework. Experimental results are discussed in section IV. Section V concludes with scope for future direction.

II. LITERATURE SURVEY

This section presents a survey of various existing schedulers and makespan models for improving performance of HMR Framework. Makespan modelling of HMR is complex in nature. In HMR, the preliminary shuffle stage is started in parallel manner with map stage and rest of the shuffling stage is done after completion of map stage. Thus, different makespan is experienced in shuffling stage. In [3] to utilize cloud resource efficiently a makespan model is presented and [4] presented a job optimization and job forecasting method by collecting job makespan execution logs history/profiles. However, these models are not efficient as it does not consider both initial shuffle and regular shuffle of the Shuffle stage. Further, [5] enhanced the performance of [4] by allocating resources based on virtual computing machines. However, it induces I/O and storage overhead in collecting and storing job history logs.

In [6-[9] considered initial and regular shuffles makespan difference in Shuffle stage and linear regression methodology is used for task prediction. However, these approaches do not consider effects of reduce jobs being removed and the reduce jobs are constant in nature. Thus, induces I/O disk access overhead and affecting resource/slot utilization. Further, SLA requirement of jobs is not considered by these approaches.

The problems of designing SLA aware scheduling has been discussed in [12-16]. The existing HMR schedulers are not as efficient as they do not consider heterogeneity of resource/slot performance requirement and dynamic task SLA requirement.

Although various researchers have contributed towards addressing the issues related to schedulers, still they lack with respect to SLA requirement and data locality awareness. Hence a novice Deadline constraint aware scheduler is the need of the hour.

III. DEADLINE CONSTRAINT AWARE SCHEDULER

This work presents a Deadline Constraint Aware Scheduler for executing HPC application on Hadoop MapReduce framework. Firstly, we create a job history logs by executing different kinds of jobs. The accurate estimation of job makespan is presented [19]. Then using the job makespan history log, a Deadline Constraint Aware Scheduler (DCAS) method is presented for HMR framework.

A. Job execution history log:

MapReduce jobs are computed by distributing them across several virtual computing nodes. In Map phase, the jobs are segmented into set of map tasks and similarly in reduce phase, the jobs are segmented into set of reduce tasks. The segmented parts are logically split and are stored on HDFS. The map tasks execute user defined operation on each logical split data and store the output in HDFS. The intermediary output is then segmented to different reduce task and are written to local storage blocks of the virtual computing node processing the map task. The reduce phase is composed of stages such as shuffle stage, sort stage, and reduce stage. In shuffle stage, the reduce tasks obtain the intermediate output data from Map tasks. In sort stage, sorting operation is carried out on all the intermediate data of all map tasks. If intermediate output does not fit into available memory, an external merge sort is applied. Post completions of shuffle operation on intermediate outcome, lastly, the sorted files are merged. As a result, we combine shuffle and sort stages together. Lastly, the reduce stage, the sorted intermediary outcome is sent to user defined reduce operation. The outcome for reduce operation is then written to HDFS.

Firstly, this work aims to construct a job execution summary log. The log files collected is composed of performance information specific to certain jobs that composed of amount of resources allocated or used by a specific job over time considering different stages of execution such as Map stages, Shuffle stages, Sort stages, and reduce stages. This information can be obtained by master node during job execution process for allocating resources. To efficiently analyze the makespan spent in each phase is described using following parameters given in equation 1.

$$\left(H_{\downarrow}, H_{\uparrow}, H_{\rightarrow}, G_H^{inp}, Sel_H\right) \tag{1}$$

Where $H_{\downarrow}$ depicts the minimum map stage makespan. $H_{\downarrow}$ aid as a function to initialize (compute) Shuffle stage since it is initialized post completion of map tasks. $H_{\uparrow}$ depicts the maximum makespan of map stage. It is utilized for considering worst case scenario of map stage execution. $H_{\rightarrow}$ is the mean makespan of map stage to encapsulate complete map wave. G_H^{inp} is the mean size of input data for each map task which is being utilized in our proposed work to compute the amount of map task to be produced for computing the data. Sel_H is ratio of the map input size with respect to the map output size. It is used to compute the size of intermediate output obtained by the map phase.

The shuffle and sort stage is initialized during post completion of first map task. Post completion of any reduce wave i.e. shuffles stage is a said to be finished when entire map phase is done and entire intermediate data obtained by map operation has been shuffled during reduce operations and sort operation is performed on them. For easiness, shuffle and sort is combined in Shuffle stage. Post completion of Shuffle stage, the reduce stage is initialized. Thus the job execution logs of shuffle and reduce stages are depicted by maximum and averages of their task makespan. An important things to be noted is, the computed shuffle stage makespan is composed of network latency (delay) incurred due to data transfer and is dependent on Hadoop cluster used. Further the Shuffle stage of first execution may be considerably different form shuffle stage that belongs to forthcoming reduce execution. This arises due to the Shuffle stage of first Reduce execution overlaps with overall map phase and is dependent on the amount of map execution and their makespan. Thus, we log to kind of estimates $\left(S_{\rightarrow}^{1}, S_{\uparrow}^{1}\right)$ for Shuffle stage of the first reduce execution and $\left(S_{\rightarrow}^{gen}, S_{\uparrow}^{gen}\right)$ for Shuffle stage of other execution depicted as general Shuffle operation. Since this work aim to compute performance invariant which are autonomous with respect to resource allocated to the jobs. Thus this work describes a Shuffle stage of the first Reduce execution includes only non-overlapping section of the preliminary shuffles in $\left(S_{\rightarrow}^{1}, S_{\uparrow}^{1}\right)$. Therefore, the job execution logs in Shuffle stage is described in equation 2.

$$\left(S_{\rightarrow}^{1}, S_{\uparrow}^{1}, S_{\rightarrow}^{gen}, S_{\uparrow}^{gen}\right). \tag{2}$$

Reduce phase is initialized post completion of Shuffle stage. The job execution log of the reduce stage is depicted by the following estimate given in equation 3.

$$\left(B_{\rightarrow}, B_{\uparrow}, Sel_B\right) \tag{3}$$

Where $B_{\rightarrow}$ depicts mean, $B_{\uparrow}$ depicts maximum and Sel_B depicts reduce selectivity (depicts ratio of reduce output with respect to input) of reduce tasks makespan.

B. Job makespan computation:

In our previous work [19], we have presented a makespan model to compute job makespan and amount of resource required to execute/complete a job. However, [19] did not consider Job SLA requirement. Thus, to provision Deadline Constraint aware job execution, this work presents a Deadline Constraint Aware Scheduler for executing HPC application on HMR framework. As discussed, reduce stage is initialized post completion map stage. Therefore, makespan time lower limit of map stage can be written as equation 4.

$$W_H^{\downarrow} = \frac{Q_H^K \cdot H_{\rightarrow}}{A_H^K} \tag{4}$$

In similar manner the upper limit of makespan time can be written as equation 5.

$$W_H^{\uparrow} = \frac{(Q_H^K - 1) \cdot H_{\rightarrow}}{A_H^K + H_{\uparrow}} \tag{5}$$

where Q_H^K is the set of map tasks, $H_{\rightarrow}$ is the mean makespan time of map task, $H_{\uparrow}$ is the maximum makespan time of map task, and A_H^K is the number of map slots assigned to job K.

For easiness, the sort stage is merged with shuffle stage. Therefore, the shuffle stage in the remaining reduce phase is estimated using equation 6.

$$W_S^{\downarrow} = \left(\frac{Q_B^K}{A_B^K} - 1\right) \cdot S_{\rightarrow}^{gen} \quad (6)$$

Where Q_B^K is the set of reduce tasks, A_B^K is the number of reduce slots assigned to job K and $S_{\rightarrow}^{gen}$ is the mean time taken for shuffle stage.

$$W_K^{\downarrow} = W_H^{\downarrow} + S_{\rightarrow}^{1} + W_S^{\downarrow} + Q_B^{\downarrow} \quad (7)$$

The job completion makespan (lower limit) time is given in equation 7 and it can be expressed as equation 8.

$$W_K^{\downarrow} = X_K^{\downarrow} \cdot \frac{Q_H^K}{S_H^K} + Y_K^{\downarrow} \cdot \frac{Q_B^K}{S_B^K} + Z_K^{\downarrow}, \quad (8)$$

Where $X_K^{\downarrow} = H_{\rightarrow}$, $Y_K^{\downarrow} = \left(S_{\rightarrow}^{gen} + B_{\rightarrow}\right)$, and $Z_K^{\downarrow} = S_{\rightarrow}^{1} - S_{\rightarrow}^{gen}$. The equations 8 represents a makespan time of job as a function/operation of map and reduce slots assigned to job *K* for performing its map and reduce tasks, that is, as a function of (Q_H^K, Q_B^K). In similar way $W_K^{\uparrow}$ and $W_K^{\rightarrow}$ can be written.

The equation can also be utilized to establish suitable number of map and reduce slots for executing job K with given SLA $\mathcal{J}$, thus we compute $\mathcal{J}$ rather than $W_K^{\downarrow}$ in equation 9.

$$\mathcal{J} = X_K^{\downarrow} \cdot \frac{Q_H^K}{S_H^K} + Y_K^{\downarrow} \cdot \frac{Q_B^K}{S_B^K} + Z_K^{\downarrow} \quad (9)$$

This work aims to find the least number of slots or resources required to complete job K with SLA $\mathcal{J}$ using makespan method [19]. It is given in equation 10.

$$\sum_{1 \le i \le Q} W_K^{\downarrow}(Q_H^K, Q_B^K) = \mathcal{J} \quad (10)$$

The solution for building Deadline Constraint Aware Scheduler for HMR for completing task K within SLA $\mathcal{J}$ is described in below section.

C. Deadline Constraint Aware Scheduler for HMR framework:

The objective of this work is to present a Deadline Constraint Aware Scheduler (DCAS) for HMR framework. The DCAS enables jobs to be submitted with desired deadline makespan for completing it. Then, the DCAS will compute amount of resource required and assign required resources (i.e. map and reduce slot) to meet job deadline requirement. The architecture of the proposed Deadline Constraint Aware Scheduler for HMR framework is shown in Figure 2. It consists of Job execution logger, resource allocator, resource estimator, Job summary logger and DCAS modules. These modules are explained in the following sections.

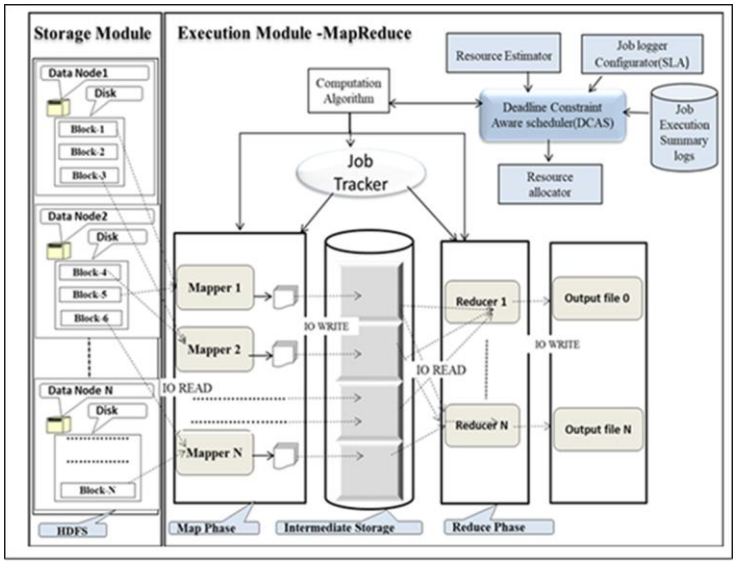


Fig 2. Deadline Constraint Aware Scheduler for Hadoop MapReduce Framework

- **Job execution logger:** The job execution logger logs the job execution summary information for completed and presently running jobs. For building job logger or profiles HMR counter is utilized which are transmitted by the virtual computing node to the master node along with heartbeat information. The execution log information is gathered from Hadoop MapReduce Distributed File System or with master worker post completion of execution of jobs. Then, the information is stored in Job execution logger.
- **Job summary logger database:** In our work the job logging information of past jobs are kept in HDFS location. The logs are recognized using job and user name which are described during job execution. Further, optionally the log information can also be store in SQL.
- **Resource estimator:** With access job logging information and its deadline information, resource estimators computes the minimum amount of map and reduce resources required to be given to the job for meeting SLA requirement. In our work it is uses makespan method to identify minimum resource required which are discussed in [19].
- **Resource allocator:** Using resource computed from resource allocator, the resource allocator allocates set of task to jobs in such a way that the job is all the time below the predefined constrained by tacking the runnable map and reduce tasks. In scenario there exist empty or free slots available, they can be assigned to other contending jobs. In general, there exist various kinds of jobs (i.e., a jobs with or without SLA). This work considers jobs with SLA requirement have highest selectivity for resources than job without SLA. However, once jobs with SLA are assigned their prerequisite for meeting the SLAs, the remaining resources can be given to the other job types.

– **Deadline Constraint Aware Scheduler (**DCAS**):** The Deadline Constraint Aware Scheduler is incorporated into HMR framework. The DCAS offers global decision making process in sorting of jobs and assigning resources to jobs. The DCAS wait for incoming request like computing worker heartbeat, job submissions etc. Post obtaining heartbeat information composed of set of unallocated resources. The DCAS processes a set of task to be allocated to it.

Algorithm 1: Deadline Constraint Aware Scheduler for HMR framework

Steps:

1. **Start**
2. When K is submitted by user to HMR framework. \\ After submitting job k to HMR
3. **Collect** Log_k from **database**. \\ collect the log information of job k from database
4. **Evaluate** minimum number of resource (Map slots h_k and Reduce slots b_k) utilizing makespan model presented in [19], considering upgraded function Equation 10.
5. When information is obtained from computing worker q.
6. Arrange Jobs with Minimum Execution Time (MET).
7. ∀ resource a in free Map/Reduce resources on computing worker q **do** // for all map and reduce slots in computing worker
8. ∀ job k in jobs **do** // for job k in a set of jobs
9. **if** a is map slot and $ProcessingMap_k < h_k$ **then**
10. **if** job k processes the unprocessed map task w with data on computing worker q **then**
11. **Process** Map task w with data present locally on computing worker q.
12. **Else if** k has unprocessed Map task w **then**
13. Process map task w on computing worker q.
14. **End if**
15. **End if**
16. **If** a is reduce slot and $CompletedMap_k > 0$ and $ProceesingReduce_k < b_k$ then
17. **If** job k has unprocessed reduce task w **then**
18. Process reduce task w on computing worker q.
19. **End if**
20. **End if**
21. **End** ∀
22. **End** ∀
23. ∀ task W_k completed slots by computing worker q **do**
24. **Reevaluate** (h_k, b_k) with respect to present time, present processing condition and SLA of job k. \\ reevaluate the current map and reduce slots with respect to processing and SLA of job k.
25. **End** ∀
26. **Stop.**

Algorithm 1: Deadline Constraint Aware Scheduler for HMR Framemwork

The DCAS should compute which jobs the resources needs to be given first, how much resource must be allocated considering slot allocation done with minimum execution time for maximizing user utility parameter and compute minimum number of resource required to meet SLA using makespan computation [19]. The working process of DCAS is given in Algorithm 1. It has two segments. Firstly, when jobs are submitted to HMR, the scheduler obtain jobs log information from HDFS and evaluate the minimum amount of resource (both map and reduce) needed to finish executing the job satisfying SLA using makespan method [19]. Secondly, heartbeat information is sent by computing node to the master computing node in periodic manner reporting their currently processing task, node information, available free resources of both map and reduce resources. Then, the master computing obtains a set of tasks to be allocated to the computing node. The master computing node keep track of the amount completed and runnable map and reduce tasks of every job. For every jobs and available resources, if the amount of runnable map task is less than the amount of map resources, is allocated it to the forthcoming task initialized. As described in line 10 to 14, more priority or selectivity is given to data that are presented with local virtual computing worker. Lastly, if utmost one map task is completed, reduce task is initialized as needed. In few scenarios, the resource required obtainable for assigning is less than prerequisite for job K and then K is assigned only a part of prerequisite slots. As time passes on, the slot assignment is reevaluated during the job's computation and is optimized if required as described in line 23 to 25. This process aid DCAS in attaining better resource utilization if the job processing condition is behind the expected and target parameter. Whenever a virtual computing node gives information status of executed (finished) tasks, we reduce Q_H^K or Q_B^K in the DCAS and reevaluate the minimal amount of resources. The DCAS model attains better resource utilization (makespan reduction) for computing diverse HPC application on HMR.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

The performance evaluation of proposed DCAS over state-of-art HMR based scheduling model [6] is presented in this section. This work used HMR version 2.0 and above for evaluating both existing and proposed scheduling model considering Microsoft azure HDInsight platform [2]. This work considered HDInsight computing cluster size of 4 worker nodes with single master node. Each computing worker possess 4 core, 8 GB RAM with 120 GB of disk space. Experiments are conducted by considering different applications such as Cap3 sequence assemble gene sequence analysis and text mining.

A. Cap3 assembly performance evaluation

DNA sequencing assembly algorithm is utilized for identifying and knowledge representation of genomic of new or existing organism. CAP3 is one such algorithm

which is utilized to assemble genome sequence. The process to perform CAP3 sequence assembly is described in Figure 3. Gene assembling is performed by carrying out aligning and merging function on small set of gene sequence fragments to construct the whole DNA sequence. Cap3 removes poor sequences within genomic sequence fragments, evaluate overlaps among genomic fragments. Further, it establishes false overlaps, removing false overlaps established. Along with perform accumulation of fragments of single or multiple overlapping genomic segments to obtain contains and carryout multiple sequence alignment to obtain consensus genomic sequences written to set of logs and to standard outcomes. More detail of CAP3 assembly can be obtained from [20].

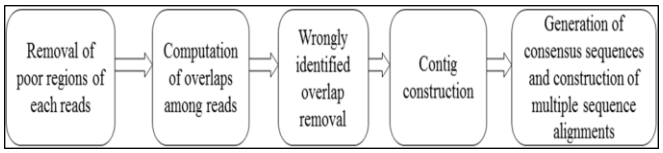


Fig 3. Process for performing CAP3 sequence assembling

For executing CAP3 application on HMR framework, CAP3 genomic sequence assembly is carried out in map stage of HMR considering base scheduler and proposed Dead Constraint Aware Scheduler (DCAS), in reduce stage aggregation of results are done. Homo sapiens chromosome 15 is considered as the reference genome. Query sequence (BAC datasets) used for analysis is considered with respect to [20] which are describe in Table 1. Experiment is conducted on all three cases for both DCAS and HMR and results are given in Table 2, which is graphically shown in Figure 4. From result it can be seen DCAS reduce makespan by 43.64%, 41.29%, and 46.93% over HMR scheduler considering all three cases, respectively. Average makespan time reduction of 43.95% is attained by DCAS over HMR. The overall result attained proves that DCAS model attain superior performance than state-of-art HMR model for executing CAP3 sequence assembly using Microsoft azure HDInsight cloud computing framework.

Table1. Simulation Parameters Considered

Experiment Id	GenBank accession number	Dataset	Genome size (base pairs)	Average genome size (base pairs)	Size of provided genomic sequences (base pairs)
1	AC004669	203	815	269	40,400
2	AF123462	526N18	1449	434	81081
3	AF111103	322F16	1933	454	26630

Table 2. Makespan time of CAP3 sequence assembling on both DCAS and HMR scheduling model

Experiment ID	GenBank accession number	Makespan time in HMR	Makespan time in DCAS
1	AC004669	877	494
2	AF123462	1778	944
3	AF111103	3141	1844

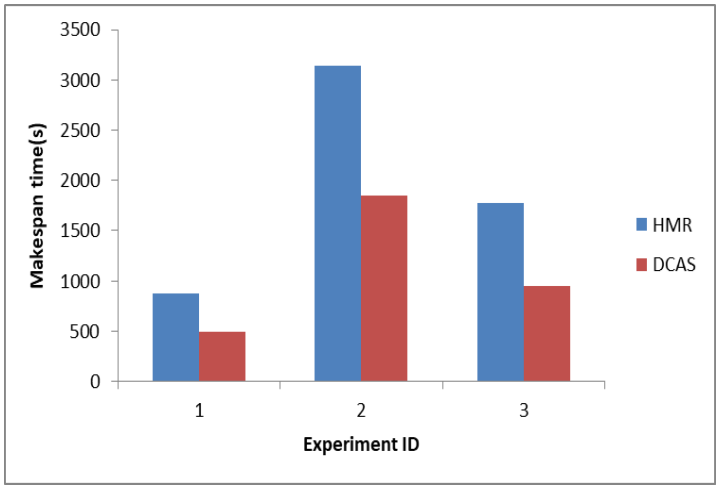


Fig 4. Makespan performance for executing CAP3 sequence assembling achieved by proposed DCAS over HMR scheduling model.

B. Short and Long read gene sequencing performance evaluation

This section presents performance evaluation for performing gene sequencing alignment considering both short (0.001 million base pair) and long genomic sequence (1 million base pair). For performing gene sequence alignment for both short and long read this work uses an approach Burrows Wheeler Aligner Smith Waterman (BWASW) [21] [22]. The alignment process of BWA is strongly dependent on smith waterman (SW) algorithm. Thus, parallelizing the computation process of smith waterman algorithm (sequential execution is shown in Fig. 5a) will aid in reducing makespan. In smith waterman algorithm, the computation process of obtaining similarity matrix score induce higher computing time as described in Fig. 5a. Thus, in this work we are considering parallelizing under multi-core environment available with computing nodes. Thus, overcomes the memory and cost overhead of using GPU based model [23], [24]. Experiment is conducted using baker yeast database [25] as described in Table 3. Parallel smith waterman algorithm as shown in Fig. 5b is incorporated into BWASW and experiment is conducted on HMR considering base scheduler [6] and proposed Deadline Constraint aware scheduler (DCAS). Hadoop 2.7 framework is considered which deployed on Microsoft azure HDInsight cluster. Experiments are conducted for case study presented in Table 3 and makespan time of both DCAS and HMR is given in Table 4, which are graphically shown in Fig. 6. From result it can be seen that DCAS reduces makespan by 48.38%, 42.3%, 41.57%, and 37.085% over HMR scheduler considering all four cases, respectively. Average makespan time

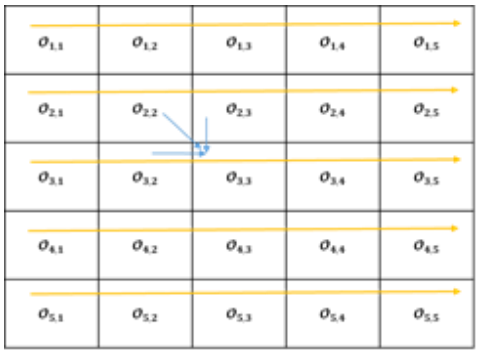
reduction of 42.33% is attained by DCAS over HMR. The overall result attained proves that DCAS model attain superior performance than state-of-art HMR model for executing smith waterman gene sequence algorithm using Microsoft azure HDInsight cloud computing framework.

Table 3. Genome sequence used for performance evaluation

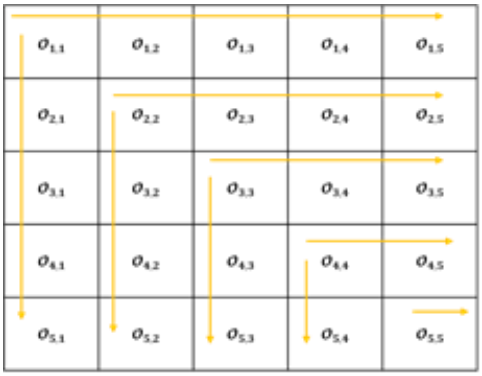
Reference genomic sequence	Genome size (base pairs)	Query genome sequence	Query Genome size (base pairs)
Saccharomyces cerevisiae S288c chromosome XII	1001933 base pair	Saccharomyces cerevisiae KillerVirusM1_1996_NC001782	1859 base pair
		Saccharomyces cerevisiae virus L-BC_ NC_001641.1	4478 base pair
		Saccharomyces cerevisiae S288c chromosome V_BK006939.2	576874 base pair
		Saccharomyces cerevisiae S288c chromosome XVI_BK006949.2	948066 base pair

Table 4. Makespan time for gene sequence alignment using smith waterman algorithm on both DCAS and HMR

Sl.No	Number of base pair considered for gene sequence alignment	DCAS	HMR
1	0.001M	71.65	36.9
2	0.005M	92.5	53.3
3	0.1M	175.7	102.6
4	0.5M	216.39	135.9



(a)



(b)

Fig 5. Gene sequence alignment using smith waterman algorithm (a) using sequential genomic sequence alignment method (b) using parallel genomic sequence alignment method.

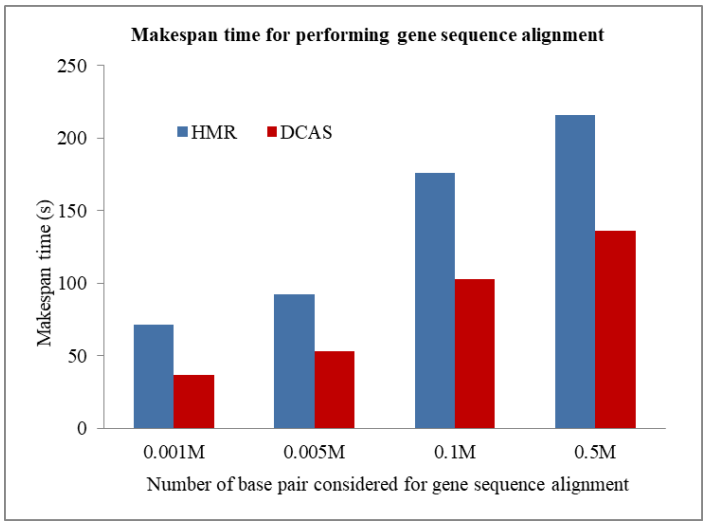


Fig 6. Makespan performance for executing smith waterman gene sequence alignment achieved by proposed DCAS over HMR scheduling model.

C. *E-commerce review analysis performance evaluation*

This section presents the performance evaluation for performing analysis on ecommerce review dataset using DCAS and HMR framework. Experiment similar to [9] such as text mining is used. The review dataset is obtained from [26] which is described in Table 5. Experiments are conducted on cases presented in Table 5 and experimental results obtained are given in Table 6, which are graphically shown in Figure. 7. From result it can be seen that DCAS reduces makespan time by 49.28%, and 55.76% over HMR scheduler considering both cases, respectively. Average makespan time reduction of 52.52% is attained by DCAS over HMR. The overall result attained proves that DCAS model attain superior performance than state-of-art HMR model for executing text mining using Microsoft azure HDInsight cloud computing framework.

Ta

Table 5. E-commerce review dataset used for experiment analysis

Experiment ID	Review dataset used	Number of reviews	Number of product
1	Health and personal cares	1,342,046	118,364
2	Clothing shoes and jewelry	2,587,014	676,522

Table 6. Makespan time for text mining on ecommerce data on DCAS over HMR scheduling model

Review Size	Makespan time in HMR	Makespan time in DCAS
1,342,046	28.49	14.4
2,587,014	51.7	22.9

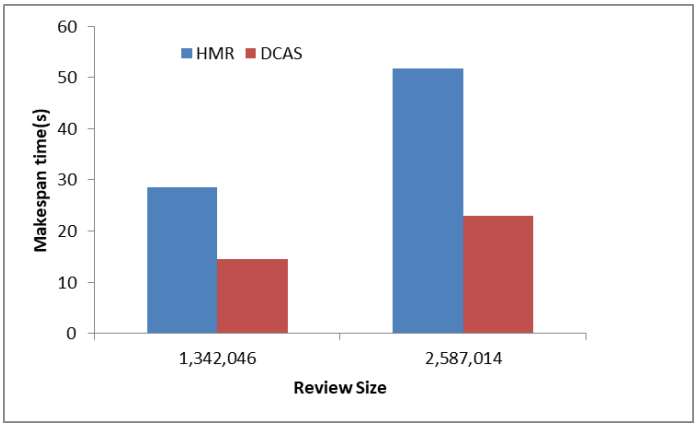


Fig. 7. Makespan performance for performing text mining on ecommerce data achieved by proposed DCAS over HMR scheduling model.

D. Results and discussions

This section presents comparative analysis over various state-of-art model [6-10]. The authors in [6] considered bioinformatics application on HMR framework considering cloud computing platform. The model attained an average makespan performance improvement of 40.28% over HMR model. However, the gene sequence alignment can align only shorter read gene sequence with low alignment accuracy. The authors in [7] considered word count application for performance evaluation over HMR. The model attained 13.33% makespan performance improvement over HMR and performance evaluation under cloud computing framework is not considered. The authors in [8] considered word count and Tera sort application for performance evaluation over HMR. The model attained 34.83% makespan performance improvement over HMR and performance evaluation under cloud computing framework is not considered. The author's in [9] considered word count and Sort application for performance evaluation over HMR. The model attained 27.7% makespan performance improvement over HMR and performance evaluation under cloud computing framework is not considered. The authors in [10], considered word count and Sort applications for performance evaluation over HMR. The model attained 43.91% makespan performance improvement over HMR and performance evaluation under cloud computing framework is not considered. From extensive analysis it can be seen that no prior work has considered dynamic scientific and data-intensive computing on HMR framework. Further, no prior work has considered alignment considering long read genomic sequence alignment. For providing computation on long read gene sequence the DCAS model offers parallel execution of gene sequence alignment process under multi core environment. Thus, aid in improving resource utilization. From result obtained it can be seen an average makespan performance (resources utilization) improvement of 43.95%, 42.33%, and 52.52% is achieved using DCAS when compared to HMR framework for Cap3, gene sequence alignment, and text mining applications. The overall result shows robustness, scalable nature of proposed DCAS method considering dynamic scientific and data intensive HPC algorithm and applications.

V. CONCLUSION

In this paper, an extensive survey on various scheduling model of Hadoop MapReduce framework has been presented. From survey it is observed that number of makespan optimization models have been presented to improve resource utilization in HMR. However, very limited work is carried out considering the provisioning of SLA or meet task deadline of a job in HMR framework. The work presented in this paper such as Deadline Constraint Aware Scheduler allocates resources to jobs using job history makespan logs of various application computations. Further, to utilize resources more efficiently, parallelization of smith waterman alignment has been explored. Experiments were conducted to evaluate the performance of DCAS over base scheduler on HMR framework considering gene sequence alignment using BWA-SW, CAP3 assembly, and text mining applications. Results revealed that an average makespan performance (resources utilization) improvement of 43.95%, 42.33%, and 52.52% is achieved using DCAS when compared to HMR framework for Cap3, gene sequence alignment, and text mining applications. Also, it can be seen that the DCAS model is robust and scalable for dynamic applications. Future work would consider minimizing makespan and computing cost through task prioritization.

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. T. White, Hadoop: The Definitive Guide, 3rd ed. Inc. O'Reilly Media, 2012.
3. 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.
4. 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.

5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. W. Xiao, W. Bao, X. Zhu and L. Liu, "Cost-Aware Big Data Processing Across Geo-Distributed Datacenters," in IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 11, pp. 3114-3127, 2017.
12. X. Dong, Y. Wang, and H. Liao, "Scheduling Mixed Real-Time and Non-real-Time Applications in MapReduce Environment," in Proc. IEEE ICPADS, Dec. 2011, pp. 9–16.
13. E. Hwang and K. H. Kim, "Minimizing Cost of Virtual Machines for Deadline-Constrained MapReduce Applications in the Cloud," in Proc. ACM/IEEE GRID, Sep. 2012, pp. 130–138.
14. J. Polo, Y. Becerra, D. Carrera, M. Steinder, I. Whalley, J. Torres, and E. Ayguad´e, "Deadline-Based MapReduceWorkload Management," IEEE Trans. Netw. Service Manag., vol. 10, no. 2, pp. 231–244, 2013.
15. B. Cho, M. Rahman, T. Chajed, I. Gupta, C. Abad, N. Roberts, and P. Lin, "Natjam: Design and Evaluation of Eviction Policies for Supporting Priorities and Deadlines in Mapreduce Clusters," in Proc. ACM SoCC, 2013, pp. 1–17.
16. M. Mattess, R. N. Calheiros, and R. Buyya, "Scaling MapReduce Applications Across Hybrid Clouds to Meet Soft Deadlines," in Proc. IEEE AINA, Mar. 2013, pp. 629–636.
17. Y. Li, H. Zhang, and K. H. Kim, "A Power-Aware Scheduling of MapReduce Applications in the Cloud," in Proc. IEEE DASC, Dec. 2011, pp. 613–620.
18. M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, "Improving MapReduce Performance in Heterogeneous Environments," in Proc. OSDI, 2008, pp. 29–42.
19. D C Vinutha, G.T. Raju, "An Accurate and Efficient Scheduler for Hadoop MapReduce Framework," Indonesian Journal of Electrical Engineering and Computer Science, Vol. 12, No. 3, pp. 1132~1142, 2018.
20. Xiaoqiu Huang and Anup Madanin, "CAP3: A DNA Sequence Assembly Program," in Genome Research, 9(9):868-77, 1999.
21. S. Canzar and S. L. Salzberg, "Short Read Mapping: An Algorithmic Tour," in Proceedings of the IEEE, vol. 105, no. 3, pp. 436-458, March 2017.
22. M. Sun, X. Zhou, F. Yang, K. Lu and D. Dai, "Bwasw-Cloud: Efficient sequence alignment algorithm for two big data with MapReduce," The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014), Bangalore, pp. 213-218, 2014.
23. Zhu, J. Li, E. Hardesty, H. Jiang and K. C. Li, "GPU-in-Hadoop: Enabling MapReduce across distributed heterogeneous platforms," Computer and Information Science (ICIS), 2014 IEEE/ACIS 13th International Conference on, Taiyuan, pp. 321-326, 2014.
24. Al Kawam, S. Khatri and A. Datta, "A Survey of Software and Hardware Approaches to Performing Read Alignment in Next Generation Sequencing," in IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. PP, no. 99, pp. 1-1, 2016.
25. Saccharomyces genome database (SGD). (2015). [Online] Available: http://www.yeastgenome.org/.
26. Amazon product dataset "http://jmcauley.ucsd.edu/data/amazon/", last accessed Feb 2, 2019.

AUTHORS PROFILE

D C Vinutha has received M.Tech., Degree from Visvesvaraya Technological University, Belagavi, Karnataka. Currently working as an Associate Professor in the Department of Information Sciemce and Engineering Vidyavardhaka college of Engineering,Mysuru, Research scholar RNS Institute of technology, Bengaluru. She has 19 years of experience in teaching. Her area of research interests includes Big Data, Data Mining. She is IETE, IEI, ISTE life member. She has published 6 papers in leading reputed International Conferences/Journals/National conferences.

Dr. G T Raju, has received M.E. Degree from Bangalore University, in 1995 and Ph.D. from Visvesvaraya Technological University, Belagavi, Karnataka in 2008. Currently working as Vice Principal, Professor and Head, in the Department of Computer Science & Engineering, RNS Institute of Technology, Bengaluru. He has 24 years of experience in teaching and research. His area of research interests include Web Mining, KDD, Image Processing, Pattern Recognition. He has published more than 90 papers in leading reputed International Journals/ Conference proceedings. He has authored five Technical books. He has completed two funded research projects. Ten Research Scholars have been awarded Ph.D. Degree under his supervision.