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 programming 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](image)

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.
Deadline Constraint Aware Scheduler for Executing High Performance Computing Application on Hadoop MapReduce Framework Paper Title Name

- **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.

\[
(H_{1}, H_{r}, H_{w}, G_{H}^{inp}, Sel_{H})
\]

Where \(H_{1}\) depicts the minimum map stage makespan, \(H_{r}\) aid as a function to initialize (compute) Shuffle stage since it is initialized post completion of map tasks, \(H_{w}\) depicts the maximum makespan of map stage. It is utilized for considering worst case scenario of map stage execution. \(H_{w}\) 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 \((s_{1}, s_{r}, s_{w})\) for Shuffle stage of the first reduce execution and \((s_{gen}, r_{gen})\) 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 \((s_{1}, s_{r}, s_{w})\). Therefore, the job execution logs in Shuffle stage is described in equation 2.

\[
(s_{1}, s_{r}, s_{w}, s_{gen}, r_{gen})
\]

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.

\[
(B_{w}, B_{r}, Sel_{R})
\]

Where \(B_{w}\) depicts mean, \(B_{r}\) depicts maximum and \(Sel_{R}\) 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.

\[
W_{H_{r}} = \frac{Q_{K}^{H} \cdot H_{r}}{A_{H_{r}}^{K}}
\]

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

\[
W_{H_{w}} = \frac{(Q_{K}^{H} - 1) \cdot H_{r}}{A_{H_{w}}^{K} + H_{r}}
\]

where \(Q_{K}^{H}\) is the set of map tasks, \(H_{r}\) is the mean makespan time of map task, \(H_{w}\) is the maximum makespan time of map task, and \(A_{H_{r}}^{K}\) 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.
Where $Q^R_K$ is the set of reduce tasks, $A^R_K$ is the number of reduce slots assigned to job K and $S^gen_{\Delta}$ is the mean time taken for shuffle stage.

\[ W^L_K = W^H_B + S^\gamma + W^S_{\Pi} + Q^R_K \] (7)

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

\[ W^L_K = X^K \cdot \frac{Q^H_K}{S^H_B} + Y^K \cdot \frac{Q^E_K}{S^B_\mu} + Z^K \] (8)

Where $X^K = H - \mu$, $Y^K = (S^gen_{\Delta} + B_{\Delta})$, and $Z^K = S^\gamma - S^gen_{\Delta}$. 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 $(Q^H_K, Q^E_K)$. In similar way $W^L_K$ and $W^R_K$ 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 $J$, thus we compute $J$ rather than $W^L_K$ in equation 9.

\[ J = X^K \cdot \frac{Q^H_K}{S^H_B} + Y^K \cdot \frac{Q^E_K}{S^B_\mu} + Z^K \] (9)

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

\[ \sum_{1 \leq i \leq Q} W^L_K(Q^H_B, Q^E_B) = J \] (10)

The solution for building Deadline Constraint Aware Scheduler for HMR for completing task K within SLA $J$ 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.
– 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.

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 obtains 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’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.

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 Logk 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.
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 // 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 ProcessingReduce_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 Framework

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 assembly 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](image)

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 considering parallelizing under multi-core environment.

![Fig 4. Makespan performance for executing CAP3 sequence assembling achieved by proposed DCAS over HMR scheduling model.](image)

### Table 1. Simulation Parameters Considered

<table>
<thead>
<tr>
<th>Experiment Id</th>
<th>GenBank accession number</th>
<th>Dataset</th>
<th>Genome size (base pairs)</th>
<th>Average genome size (base pairs)</th>
<th>Size of provided genomic sequence pairs (base pairs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AC004669</td>
<td>203</td>
<td>815</td>
<td>269</td>
<td>40,400</td>
</tr>
<tr>
<td>2</td>
<td>AF123462</td>
<td>526N18</td>
<td>1449</td>
<td>434</td>
<td>81081</td>
</tr>
<tr>
<td>3</td>
<td>AF111103</td>
<td>322F16</td>
<td>1933</td>
<td>454</td>
<td>26630</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Experiment ID</th>
<th>GenBank accession number</th>
<th>Makespan time in HMR</th>
<th>Makespan time in DCAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AC004669</td>
<td>877</td>
<td>494</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Reference genomic sequence</th>
<th>Genome size (base pairs)</th>
<th>Query genome sequence</th>
<th>Query Genome size (base pairs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saccharomyces cerevisiae S288c chromosome XII</td>
<td>1001933 base pair</td>
<td>Saccharomyces cerevisiae KillerVirusM1_1996_NC001782</td>
<td>1859 base pair</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Saccharomyces cerevisiae L_Bc_96_NC_001641.1</td>
<td>4478 base pair</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Saccharomyces cerevisiae S288c chromosome V_BK006939.2</td>
<td>576874 base pair</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Saccharomyces cerevisiae S288c chromosome XVI_BK006949.2</td>
<td>948066 base pair</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Number of base pair considered for gene sequence alignment</th>
<th>DCAS</th>
<th>HMR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.001M</td>
<td>71.65</td>
<td>36.9</td>
</tr>
<tr>
<td>2</td>
<td>0.005M</td>
<td>92.5</td>
<td>53.3</td>
</tr>
<tr>
<td>3</td>
<td>0.1M</td>
<td>175.7</td>
<td>102.6</td>
</tr>
<tr>
<td>4</td>
<td>0.5M</td>
<td>216.39</td>
<td>135.9</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Experiment ID</th>
<th>Review dataset used</th>
<th>Number of reviews</th>
<th>Number of product</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Health and personal cares</td>
<td>1,342,046</td>
<td>118,364</td>
</tr>
<tr>
<td>2</td>
<td>Clothing shoes and jewelry</td>
<td>2,587,014</td>
<td>676,522</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Review Size</th>
<th>Makespan time in HMR</th>
<th>Makespan time in DCAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,342,046</td>
<td>28.49</td>
<td>14.4</td>
</tr>
<tr>
<td>2,587,014</td>
<td>51.7</td>
<td>22.9</td>
</tr>
</tbody>
</table>

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 authors 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


