International Journal of Engineering & Technology, 7 (4.19) (2018) 578-585

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

An Efficient Makespan Model for Hybrid Dual Parallel Computing Framework

Wahida Banu[1*], Nandini N[2]

[1]Research Scholar,VTU, Dr.AIT Research Centre Bangalore
[2]Associate Professor, VTU, Dr.AIT.,Bangalore
**Corresponding Author Email: wahidanisar@gmail.com*

Abstract

MapReduce (MR) is the most widely adopted and used computing platform for processing complex scientific and data intensive application. Hadoop MapReduce (HMR) is widely used MR framework across various organization due to its open source nature. Cloud service provider (CSP) such Azure HDInsight offers computing resources to its user and only pays for their use. MapReduce framework currently been used are not efficient due to sequential computing of Map and Reduce phase. As a result, incurs higher computing cost and exhibit underutilization of cloud resources. Minimizing cost of execution on such platform is most desired. To overcome research challenges, this work firstly present Hybrid Dual Parallel Computing (HDPC) framework. HDPC offers parallel computation of Map and Reduce phase. To further enhance resource utilization parallel execution of map and reduce operation is carried out considering multi-core environments available with virtual computing workers. Lastly, this work presented job makespan/execution model and working structure of HDPC framework. Experiment are conducted on Microsoft Azure HDInsight cloud platform considering stream and non-stream application to evaluate performance of HDPC framework over existing computing model. The outcome shows significant performance improvement in terms of execution time. Overall good correlation is seen among practical execution and theoretical execution outcome shows proposed HDPC framework is robust, scalable, cost efficient and support dynamic analysis on cloud computing environment.

Keywords: *Big data, Bioinformatics, Cloud computing, GPU, Hadoop, Linear regression, MapReduce, Multi-core, Parallel computing.*

1. Introduction

Cloud computing play a major role for attaining scalable computing for scientific and data intensive application. The cloud computing adopts distributed architecture which is capable of processing large amount of data collected by various organization such as social network, sensor network, bioinformatics etc. Performing scalable computing on these unstructured data is most desired across organizations. The exiting model such as Phoenix [1], Mars [2] and Dryad [3], and Spark [4] are not efficient in performing in real-time analysis on continuous/stream data. Google came up with parallel computing architecture namely MapReduce framework [5] for performing real-time analysis for scientific and data intensive applications. Among all Hadoop MapReduce (HMR) [6] is the most widely used and adopted [7] framework due its open source nature and ease of deployment, and scalability.
The HMR is composed of Map, Shuffle, Sort and Reduce phase. In Map phase it read all input data and divide it into chunks of small data and perform execution parallel across different virtual machine. Shuffle phase begins with completion of Map phase that collects the intermediate output from all the Map task. A sort operation is performed on the intermediate output of map phase. For simplicity sort and shuffle phases are cumulatively considered in the shuffle phase. Post completion Reduce phase is initialized. In this phase it reads the intermediate output and aggregate the user defined functional output and store it in Hadoop distributed file system (HDFS). Detail of Hadoop MapReduce execution can be obtained from [6]. The basic architecture of Hadoop MapReduce framework is shown in Fig. 1.

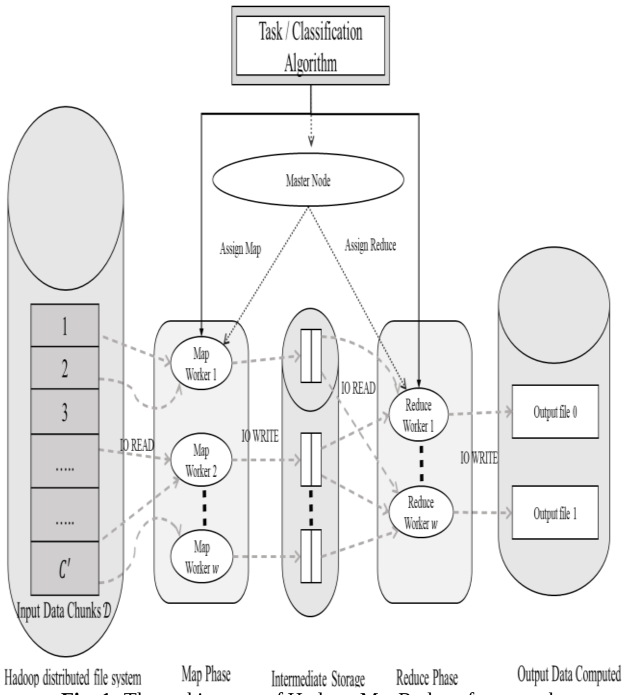

Fig. 1: The architecture of Hadoop MapReduce framework

HMR framework suffers from number of drawback such as it incurs buffer concurrency among jobs and heavy disk read seeks. As a result, incurs I/O overhead and increase execution time [8]. Further, HMR scheduler does not considers performance parameter such memory requirement and multi core environment for linear scalability effecting performance [9] and also considers homogenous map execution time considering homogenously distributed data, which is not true [10]. As a result, cloud resource are not utilized efficiently [11]. HMR adopts serial execution strategy adopted i.e., post completion of Map phase reduce is initialized. As a result, incurs higher cloud expense and effects performance [12]. HMR does not offer flexible pricing [13], scalability issues due to cluster based nature of HMR and are not efficient for streaming data analysis [11]. Recently number of optimization and makespan approaches has be presented to overcome the limitation of HMR framework.

In [8] addressed Hadoop memory management issues adopting global memory management technique. For attaining global memory management multi-thread execution engine is used. There model improves the memory utilization and balanced the performance of I/O and CPU. However, they did not considered the network I/O performance into consideration. Then, [9] presented a GPU based design to overcome linear scalability issue of Hadoop. They addressed the challenges existed in integrating HMR and GPU and how the MR job can be executed using CUDA based GPU. Further, [11] presented Cloud MapReduce (CMR) framework to address issue pertaining to sequential execution. The model offers parallel execution of Map and Reduce phase. However, no theoretical justification of model is given.

The exiting HMR based model does not offers job with deadline requirement on HDInsight cloud. Computing task deadline is a challenging task. Therefore, makespan modelling is considered to be a key performance component to compute amount of resource required to meet task deadline. In HMR, the first wave of shuffle phased is initialized in parallel fashion with Map phase (i.e. overlapping phase) and rest of the waves of the Shuffle phase are processed post completion of Map phase (i.e. non-overlapping phase). In [14] and [15] presented a makespan model for HMR to utilize cloud resource efficiently. However, they incurs computing overhead due to inaccurate estimation. Since they did not considered overlapping and non-overlapping phases of the Shuffle stage. Then, [16] presented a job prediction and optimization model namely Starfish. The model collected information of active Hadoop task profile at a satisfactory granularity. In [17], presented a model namely Elasticiser. The model enhanced the approach presented in [16] by adding resource allocation based on VMs. However, it incurs large overhead in collecting Hadoop task profile. As a result, attain high over-predicted task run-time. Further, [18], [19], and [20] used both overlapping and non-overlapping phases of shuffle stage and for task prediction a conventional linear regression method is utilized. CRESP [21] predicts task execution efficiently and aid in allocating resources based on MR slots. However, in CRESP models, the effect of number of reduced jobs are discarded. In [20] and [21], the number of reduced jobs are constant. As a result, incurs higher I/O disk seek. Thus, affecting resource utilization.

From above analysis it is quite evident Hadoop suffers from number of drawback in utilizing resource efficiently and accurate modelling of job makespan/estimation model. Therefore, minimizing execution time and utilizing resource efficiently with minimal costs is most desired of cloud based computing model. To attain this objective a Hybrid Parallel Dual Computing (HDPC) framework is presented in this manuscript. The HDPC framework adopts parallel execution among Map and Reduce phase similar to [11] i.e., reduce phase is initiated as soon as two or more Map task is completed. This usage of such strategy aid utilizing resource efficiently (i.e.., reduction of unutilized computing node resources). Further, this work consider parallel execution of Map and reduce operation using multi-core environment available with computing nodes. Lastly, presented a makespan model to describe function of the HDPC framework.

The Contribution of research work is as follows:

- This work present an accurate makespan model for HDPC aiding performance improvement.
- This work presented parallel execution design of Map and Reduce phase.
- Further, to enhance resource utilization parallel execution of map and reduce operation is carried out considering multi-core environments available with virtual computing workers
- Experiments are conducted considering diverse cloud configurations and varied application (stream and non-stream data) configuration.
- Experiment outcome shows good correlation among theoretical makespan model and experimental values.

The rest of the paper is organized as follows. In section II the proposed makespan modelling for Hybrid Dual Parallel Computing framework is presented. In penultimate section experimental study is carried out. The conclusion and future work is described in last section.

2. Makespan Modelling for Hybrid Dual Parallel Computing Framework

This work present an efficient makespan modelling for hybrid dual parallel computing (HDPC) framework. The HDPC adopts similar functionalities used in traditional MapReduce framework such as HMR. Therefore, it composed of Map, Shuffle, Sort and Reduce phase. For easiness, this work considers shuffle and reduce phase together in reduce phase. Map phase reads the input from the client and builds set of key value pairs as follows

$$M(k_1,v_1) \rightarrow l(k_2,v_2). \quad (1)$$

This constructed key k_1 and list of values are combined together in reduce phase. Reduce phase takes intermediate key k_2 and process and construct new set of values $l(v_3)$. The HDPC perform task/job computation on multiple virtual computing node that forms together as a computing cluster. Where one is master computing node and others are slave computing worker nodes. The master computing node distribute and assign task to different worker node and monitors the task among worker computing nodes. Further, the slave nodes updates its resource utilization information to the master node in a periodically manner. Master nodes schedule the task based on resource (worker) availability.

To reduce execution time for completing job and maximizing resource utilization of cloud platform, proposed HDPC framework adopts parallel execution strategy. That is, reduce phase is initialized in parallel when two or more map worker completes its tasks. This work adopts parallel execution of reduce phase using multi-core environment. Since the worker node considered to possess more than one computing cores. Further, this work present HDPC makespan model in subsection a.

The HDPC framework is combination of map and reduce job. Here firstly, the input data is segmented in to set of uniform block size of data, namely represented as chunks. These chunks are distributed among the virtual computing workers. Further, these chucks are further segmented to parallelize computation based on user defined map and reduce operation. User defined Mapper operation are applied to the input and intermediate output is built. These intermediate output is the input for the reduce job. Reduce phase is composed of shuffle and reduce stage. Output of map task is fed as input for shuffle stage, already completed map jobs is shuffled and then sort operation is performed. Post sorting, the sort data is fed into user defined reduce operation and output is generated and is written to cloud computing storage.

A map operation in terms of input/ output data dependencies and makespan or computing time can be expressed as a tuple as follows

$$\left(\vec{\mathbb{S}}_{\mathbb{M}}^{\mathbb{I}}, T_{\mathbb{M}}, \mathbb{M}_{min}, \mathbb{M}_{mean}, \mathbb{M}_{max}\right), \tag{2}$$

where $\vec{\mathbb{S}}_{\mathbb{M}}^{\mathbb{I}}$ is mean input data used by each Map computing node for processing, $T_{\mathbb{M}}$ is ratio among output and input (i.e., output of Map operation stored on cloud for reduce worker to process), $\mathbb{M}_{min}, \mathbb{M}_{mean}, \mathbb{M}_{max}$ depicts the maximum, mean and minimum makespan time of Map operation. Similarly, HDPC reduce operation is expressed as follows

$$(\mathbb{S}_{\mathbb{M}}, T_{\mathbb{R}}, \mathbb{R}_{min}, \mathbb{R}_{mean}, \mathbb{R}_{max}), \tag{3}$$

where $\mathbb{S}_{\mathbb{M}}$ is the output data of map operations placed in cloud storage (expressed as a ratio), $T_{\mathbb{R}}$ depicts the task assigned to HPDC or reduce operation (which is a ratio of reduce output to input), $\mathbb{R}_{min}, \mathbb{R}_{mean}, \mathbb{R}_{max}$ depicts the maximum, mean and minimum makespan time of Reduce operation. The Reduce stage is composed of shuffle and sort functions.
Minimizing makespan or computing time and cost of cloud usage is most desired. In this work we presented a makespan model to express functions of HDPC framework. Estimating accurate makespan time (job completion time) is very challenging, difficulty and complex. The makespan time estimation depends on number of factor such as network condition, data transfer rates, hardware parameters, virtual computing cluster node performance, cloud storage parameters and so on. In this work, HDPC only considers functional modification to enhance performance of traditional MapReduce frameworks. The makespan model presented in this work is based on model presented in [22].

2.1. Makespan Modelling Limit Computations

This section describes makespan modeling limit computation. Here makespan operation is evaluated as time needed to finish a job of input with respect to number of resource assigned to HDPC. Let consider a job $\mathbb{K}$ to be processed on the HDPC cloud computing framework considering input data $\mathbb{E}$. Let the cloud platform poses $\mathcal{O}+1$ number of workers or nodes. Each nodes is composed of $\mathcal{Q}$ cores that can be used for processing. One node act as a master worker node to process and remaining $\mathcal{Q}$ number of nodes to carryout Map and Reduce computation. Job $\mathbb{K}$ is considered to be distributed and processed utilizing y number of Map and Reduce tasks. The input data $\mathbb{E}$ is segmented into y chunks expressed as $\mathbb{E}'$. In traditional $\mathbb{E}'$ MR framework is computed as follows

$$\mathbb{E}' = (\mathbb{E}/\mathcal{O}) \tag{4}$$

HDPC also adopts similar methodology for evaluating $\mathbb{E}'$. The time taken to finish y task is depicted as $\mathcal{U}_1, \mathcal{U}_2, \ldots\ldots, \mathcal{U}_y$. In HDPC framework the chunks $\mathbb{E}'$ is further split to attain parallel computation which is expressed as follows

□″=□′□ (5)

In HDPC computation of y^{th} task is expressed as follows

$$\mathcal{U}_y = \mathcal{U}_{max} = \max\{u_1, \cdots u_{\mathcal{Q}}\} \tag{6}$$

where $u_{\mathcal{Q}}$ depicts processing of task on $\mathcal{Q}^{th}$ computing core with respect to data $\mathbb{E}''$.
The maximum time (μ) taken by y task to finish job $\mathbb{K}$ is expressed as follows

$$\mu = \max_{k}\{\mathcal{U}_k\} \tag{7}$$

Similarly, the mean time (φ) taken by y task to finish job $\mathbb{K}$ is expressed as follows

$$\varphi = \frac{\sum_{k=1}^{y} \mathcal{U}_k}{y} \tag{8}$$

Let consider an optimistic condition, that y task are distributed uniformly across $\mathbb{W}$ worker (minimum time needed to complete $(y * \varphi)$ job). The total time required to process these task and its lower limits time is expressed as follows

$$\mathbb{L}_{lim} = \frac{y * \varphi}{\mathbb{W}} \tag{9}$$

Similarly, we computed the upper limit considering pessimistic condition, i.e., the longest completion time of task $\overleftarrow{\mathcal{U}} \in (\mathcal{U}_1, \mathcal{U}_2, \ldots\ldots, \mathcal{U}_y)$ with execution time of μ is last completed task. Then, the time engaged before completion of last task $\overleftarrow{\mathcal{U}}$ is upper limited as follows

$$\frac{\left(\sum_{k=1}^{y} \mathcal{U}_k\right)}{\mathbb{W}} \leq \frac{(y-1) \times \varphi}{\mathbb{W}} \tag{10}$$

The likable job execution time varies due to non-deterministic and scheduling is attained by variance of lower and upper limit. This is critical performance parameter when time taken of longest task is small as compared to the total makespan i.e. $\mu \ll \left(y \times \frac{\varphi}{\mathbb{W}}\right)$. Therefore, the total execution for the longest task $\overleftarrow{\mathcal{U}}$ is upper limited as follows

$$\mathbb{U}_{lim} = \frac{(y-1) \times \varphi}{\mathbb{W}} + \mu \tag{11}$$

2.2. Makespan of Job on HDPC Framework

This section present job execution time of HDPC framework. Firstly, let consider a job $\mathbb{K}$ which is submitted by user to HDPC framework. The job $\mathbb{K}$ is segmented into $\mathcal{Y}_{\mathbb{M}}^{\mathbb{K}}$ number of Map task and number of Reduce task is depicted as $\mathcal{Y}_{\mathbb{R}}^{\mathbb{K}}$. Let $\mathbb{T}_{\mathbb{M}}^{\mathbb{K}}$ depicts the number of Map nodes assigned for the $\mathbb{K}^{th}$ job and $\mathbb{T}_{\mathbb{R}}^{\mathbb{K}}$ depicts the number of Reduce nodes assigned for the $\mathbb{K}^{th}$ job. To estimate the computing time of map task, upper and lower limits is evaluated. Using Eq. (7) and (8) maximum and mean computation time of Map tasks of $\mathbb{K}$ is evaluated. Using $\mathbb{M}_{max}$ and $\mathbb{M}_{mean}$ computed in Eq. (9) lower limits of Map stage is expressed as follows

$$\mathcal{U}_{\mathbb{M}}^{\mathbb{L}_{lim}} = \frac{\mathcal{Y}_{\mathbb{M}}^{\mathbb{K}} \times \mathbb{M}_{mean}}{\mathbb{T}_{\mathbb{M}}^{\mathbb{K}}} \tag{12}$$

Similarly, we compute the maximum makespan time of Map stage or upper limit in HDPC using eq. (11) is expressed as follows

$$\mathbb{U}_{\mathbb{M}}^{\mathbb{U}_{lim}} = \frac{(\mathcal{Y}_{\mathbb{M}}^{\mathbb{K}} - 1) \times \mathbb{M}_{mean}}{\mathbb{T}_{\mathbb{M}}^{\mathbb{K}} + \mathbb{M}_{max}} \tag{13}$$

Using eq. (12) and (13) this work evaluated the total execution time of Map stage in HDPC as follows

$$\vec{\mathbb{U}}_{\mathbb{M}} = \frac{\left(\mathcal{U}_{\mathbb{M}}^{\mathbb{L}_{lim}} + \mathbb{U}_{\mathbb{M}}^{\mathbb{U}_{lim}}\right)}{2} \tag{14}$$

The mean execution time of each map virtual computing node is estimated as follows

$$\vec{\mathbb{U}}_{\mathbb{M}_{mean}} = \frac{\vec{\mathbb{U}}_{\mathbb{M}}}{\mathbb{T}_{\mathbb{M}}^{\mathbb{K}}} \tag{15}$$

The Reduce stage is initiated post completion minimum two map task is completed by map worker (i.e., the reduce stage is started at $\left(\mathbb{U}_{\mathbb{M}}^{\mathbb{U}_{lim}} - \mathcal{U}_{\mathbb{M}}^{\mathbb{L}_{lim}}\right)$ time instance. The output of Map stage generates set of intermediate output which is later fed into user defined shuffle, sort and reduce operations. The maximum and mean computing time of Reduce stage considering $\mathbb{T}_{\mathbb{R}}^{\mathbb{K}}$ computing nodes is expressed using Eq. (7) and (8). The total execution time limit of Reduce stage that is the lower limit $\mathcal{U}_{\mathbb{R}}^{\mathbb{L}_{lim}}$ is evaluated as follows

$$\mathcal{U}_{\mathbb{R}}^{\mathbb{L}_{lim}} = \frac{\mathcal{Y}_{\mathbb{R}}^{\mathbb{K}} \cdot \mathbb{R}_{mean}}{\mathbb{T}_{\mathbb{R}}^{\mathbb{K}}} \tag{16}$$

and similarly, the upper limit $\mathcal{U}_{\mathbb{R}}^{\mathbb{U}_{lim}}$ is evaluated as follows

$$\mathcal{U}_{\mathbb{R}}^{\mathbb{U}_{lim}} = \frac{\left(\mathcal{Y}_{\mathbb{R}}^{\mathbb{K}} - 1\right) \cdot \mathbb{R}_{mean}}{\mathbb{T}_{\mathbb{R}}^{\mathbb{K}} + \mathbb{R}_{max}} \tag{17}$$

The total execution time of $\mathbb{K}^{th}$ job in HDPC platform is a sum of time engaged to compute Map tasks and time engaged to compute Reduce tasks. This work consider lower limits (i.e., best case scenario), the minimum execution time is expressed as follows

$$\mathcal{U}_{\mathbb{K}}^{\mathbb{L}_{lim}} = \mathcal{U}_{\mathbb{M}}^{\mathbb{L}_{lim}} + \mathcal{U}_{\mathbb{R}}^{\mathbb{L}_{lim}} - \left(\mathcal{U}_{\mathbb{M}}^{\mathbb{U}_{lim}} - \mathcal{U}_{\mathbb{M}}^{\mathbb{L}_{lim}}\right) \tag{18}$$

The Eq. (18) is simplified as follows

$$\mathcal{U}_{\mathbb{K}}^{\mathbb{L}_{lim}} = 2\mathcal{U}_{\mathbb{M}}^{\mathbb{L}_{lim}} + \mathcal{U}_{\mathbb{R}}^{\mathbb{L}_{lim}} - \mathcal{U}_{\mathbb{M}}^{\mathbb{U}_{lim}} \tag{19}$$

Further, this work considers worst case scenario (maximum execution time taken or upper limit) for computing which is expressed as follows

$$\mathcal{U}_{\mathbb{K}}^{\mathbb{U}_{lim}} = \mathcal{U}_{\mathbb{M}}^{\mathbb{U}_{lim}} + \mathcal{U}_{\mathbb{R}}^{\mathbb{U}_{lim}} - \left(\mathcal{U}_{\mathbb{M}}^{\mathbb{U}_{lim}} - \mathcal{U}_{\mathbb{M}}^{\mathbb{L}_{lim}}\right) \tag{20}$$

The Eq. (20) is simplified as follows

$$\mathcal{U}_{\mathbb{K}}^{\mathbb{U}_{lim}} = \mathcal{U}_{\mathbb{M}}^{\mathbb{L}_{lim}} + \mathcal{U}_{\mathbb{R}}^{\mathbb{U}_{lim}} \tag{21}$$

The total execution time of $\mathbb{K}$ on HDPC platform is expressed as follows

$$\vec{\mathcal{U}}_{\mathbb{K}} = \frac{\left(\mathcal{U}_{\mathbb{K}}^{\mathbb{U}_{lim}} + \mathcal{U}_{\mathbb{K}}^{\mathbb{L}_{lim}}\right)}{2} \tag{22}$$

Using Eq. (19) and (21), the total execution time is computed as follows

$$\vec{\mathcal{U}}_{\mathbb{K}} = \frac{\left(\left(\mathcal{U}_{\mathbb{M}}^{\mathbb{L}_{lim}} + \mathcal{U}_{\mathbb{R}}^{\mathbb{U}_{lim}}\right) + \left(2\mathcal{U}_{\mathbb{M}}^{\mathbb{L}_{lim}} + \mathcal{U}_{\mathbb{R}}^{\mathbb{L}_{lim}} - \mathcal{U}_{\mathbb{M}}^{\mathbb{U}_{lim}}\right)\right)}{2} = \frac{\left(3\mathcal{U}_{\mathbb{M}}^{\mathbb{L}_{lim}} + \mathcal{U}_{\mathbb{R}}^{\mathbb{L}_{lim}} + \mathcal{U}_{\mathbb{R}}^{\mathbb{U}_{lim}} - \mathcal{U}_{\mathbb{M}}^{\mathbb{U}_{lim}}\right)}{2} \tag{23}$$

2.3. Data Dependency Modelling on Total Execution Time on HDPC Framework

This section present data dependency modelling on total execution time on HDPC framework. This work adopt a similar approach presented in [21], [22], and [23] to model data dependency using linear regression. Therefore, the mean execution time of the $\mathcal{y}^{th}$ map computing node is expressed as follows

$$\mathbb{M}_{mean}^{\mathcal{y}} = \mathcal{W}_0^{\mathbb{M}} + \sum_{\mathcal{X}=1}^{\mathbb{T}_{\mathbb{M}}^{\mathbb{K}}} \left(\mathcal{W}_{\mathcal{X}}^{\mathbb{M}} \left(\frac{\mathbb{E}'}{\mathcal{Q}}\right)\right), \tag{24}$$

where $\mathcal{W}_{*}^{\mathbb{M}}$ depicts parameter that are dependent on map operation of user i.e., they are application dependent. Therefore, the mean execution time of the $\mathcal{y}^{th}$ reduce computing worker is expressed as follows

$$\mathbb{R}_{mean}^{\mathcal{y}} = \mathcal{W}_0^{\mathbb{R}} + \sum_{\mathcal{W}=1}^{\mathbb{T}_{\mathbb{R}}^{\mathbb{K}}} \left(\mathcal{W}_{\mathcal{X}}^{\mathbb{R}} \left(\frac{\mathbb{E}'}{\mathcal{Q}}\right)\right), \tag{25}$$

where $\mathcal{W}_{*}^{\mathbb{R}}$ depicts parameter that are dependent on reduce of user operation and $\mathbb{E}'$ depicts intermediate output data collected from Map stage. For parallel execution realization and to utilize resource efficiently it is further segment similar to Map stage. On similar lines maximum and mean makespan of map and reduce workers is evaluated. Data dependent estimations of$\mathbb{M}_{mean}, \mathbb{M}_{\max}, \mathbb{R}_{\text{mean}}, \mathbb{R}_{\max}$ are utilized in (19), (21) and (23) to estimate total execution time of $\mathbb{K}^{th}$ job on HDPC platform considering data $\mathbb{E}$. More detail of proof and modeling of data dependency using linear regression can be obtained from [21] and [23]. The proposed makespan modeling for HDPC framework presented in this paper is accurate and efficient which is experimentally proven in next section.

3. Result and Annalysis

This section present performance evaluation of proposed HDPC framework over stat-of-art HMR framework [18]. This paper considers HMR framework for comparison. Since it is widely used MR platform for computing on cloud computing environments [24]. HDPC is developed using C#.net framework 4.0, C++ programming language and Node.js. HDPC is deployed on azure cloud computing platform for performance evaluation on larger data. The HDPC framework is deployed on Azure cloud computing environment. The HDPC cluster is composed of one master and four cloud computing node. Each computing node is deployed using A3 instance virtual computing instance which is composed 4 computing cores, 7 GB Ram and 120 GB of HDD storage space. Same cluster configuration for both HMR and OHMR is considered for experiment analysis. This work considers experiment analysis considering both stream and non-stream application. For stream analysis Word frequency statistic computation is considered and for non-stream analysis Hot-word detection computation is considered.

3.1. Non-stream Data Analysis Performance Evaluation of HDPC over HMR

This section carryout performance evaluation of non-stream application on HDPC and HMR framework. The word frequency statistic application is considered and is developed using C#

programming language. For performance evaluation Wikipedia dataset [25], [32] is considered. The Microblog dataset is composed of data greater than 30 GB. For experiment analysis this work split data into 4086 MB, 8192 MB, 16384 MB, and 32768 MB (i.e., around 16 GB of data is considered) is stored in Azure blob storage container. The word frequency test [21], [26] is carried out on stored data on both HMR and HDPC and result are noted as shown in Fig. 2. The outcomes shows an execution time reduction of 47.98%, 46.9%, 49.02%, and 54.16% is achieved by HDPC over HMR considering non-stream data size of 4086 MB, 8192 MB, 16384 MB, and 32768 MB respectively. An average execution time reduction of 49.51% is attained by proposed HDPC over exiting HMR considering varied non-stream data size.

Theoretical execution time of HDPC i.e., $\vec{u}_{\mathbb{K}}$ given in Eq. (23) is computed and is compared against the experimented value attained considering varied non-stream data size. Result attained is shown in Fig. 3. Slight variation is seen from practical and theoretical job execution computation. Overall good correlation is seen among theoretical and practical execution time. From experiment outcome it is clear execution of non-stream data analysis on proposed HDPC framework attain superior performance when compared with HMR framework. Accuracy and correctness of theoretical job execution model of HDPC framework presented is proved through correlation measures.

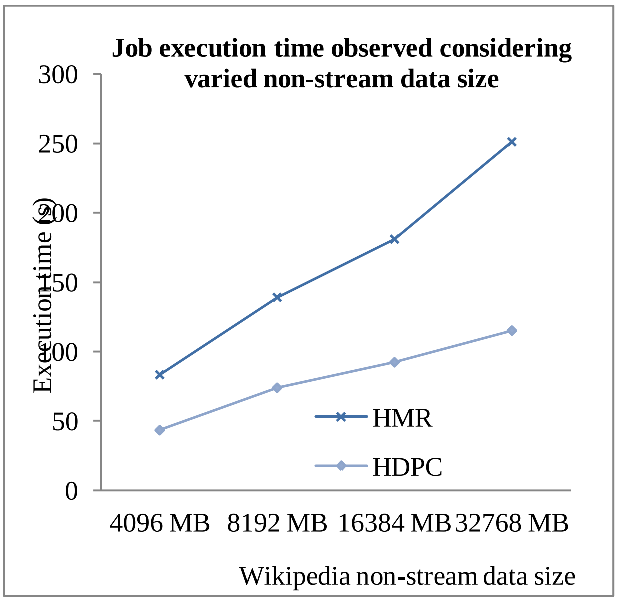

Fig. 2: Non-stream data analysis total execution time observed for varied data size conducted on HDPC and HMR frameworks

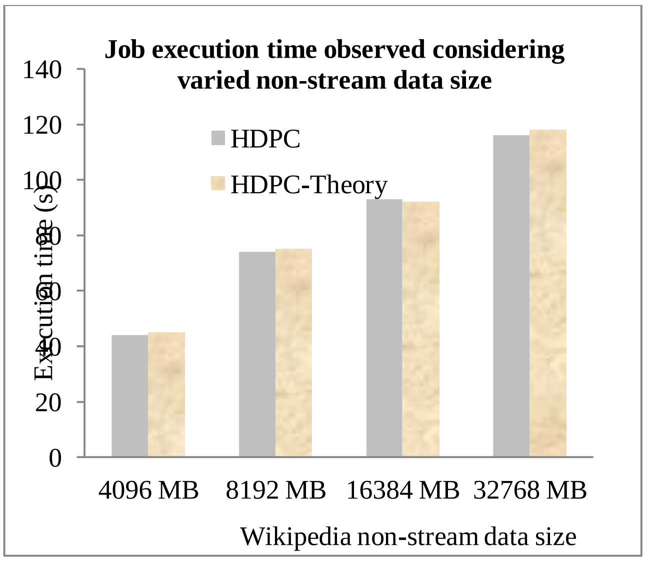

Fig. 3: Correlation between theoretical and practical execution time for varied non-stream data analysis on HDPC framework

3.2. Stream Data Analysis Performance Evaluation of HDPC over HMR

This section carryout performance evaluation of stream application on HDPC and HMR framework. The hot word detection application [26] is considered and is developed using C# programming language. For performance evaluation The “Movietweetings” dataset [27] is considered. For experiment analysis this work considers stream tweet size of 25K, 50K, 75K, and 100K and is stored in Azure blob storage container. The hot word detection test is carried out on stored data on both HMR and HDPC and result are noted as shown in Fig. 4. The outcomes shows an execution time reduction of 56.63%, 46.84%, 62.65%, and 56.51% is achieved by HDPC over HMR considering stream data size of 25K, 50K, 75K, and 100K respectively. An average execution time reduction of 55.66% is attained by proposed HDPC over exiting HMR considering varied stream data size.

Theoretical execution time of HDPC i.e., $\vec{u}_{\mathbb{K}}$ given in Eq. (23) is computed and is compared against the experimented value attained considering varied stream data size. Result attained is shown in Fig. 5. Slight variation is seen from practical and theoretical job execution computation. Overall good correlation is seen among theoretical and practical execution time. From experiment outcome it is clear execution of non-stream data analysis on proposed HDPC framework attain superior performance when compared with HMR framework. Accuracy and correctness of theoretical job execution model of HDPC framework presented is proved through correlation measures.

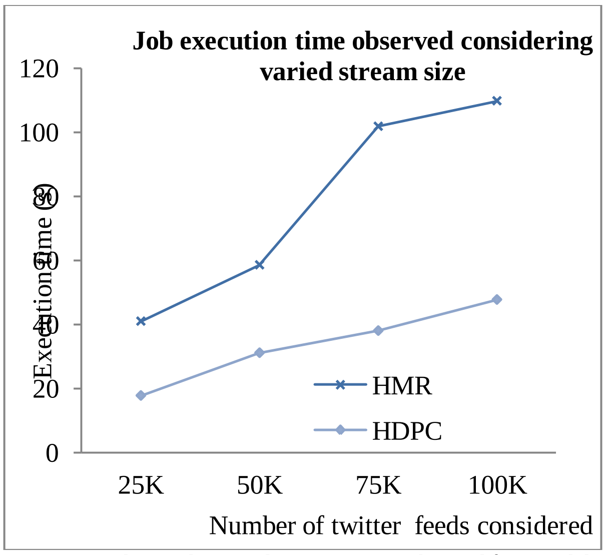

Fig. 4: Stream data analysis total execution time observed for varied data size conducted on HDPC and HMR frameworks

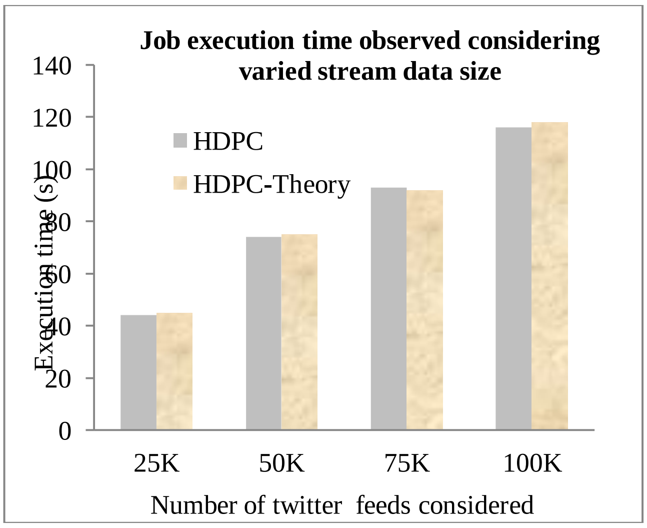

Fig. 5: Correlation between theoretical and practical execution time for varied stream data analysis on HDPC framework

3.3. Gene Sequence Analysis Performance Evaluation of HDPC over HMR

This section carryout performance evaluation of gene sequence analysis [18] on HDPC and HMR framework. Application related to Cancer research, Genetic Diseases identification, Reproductive Health and so on are dependent sequence alignment algorithms for analysis. This work used Homo sapiens chromosome (NC_000015.10) and bakers yeast genomic database as reference database obtained from [29] and the query sequences obtained from the influenza virus database [30]. The detail of query and reference genome used for experiment analysis is described in Table I. Gene sequence analysis is performed on query and reference genome used in Table I and the result are graphically plotted in Fig. 6.The experiment outcome shows an execution time reduction of 64.37%, 52.92%, 60.08%, 47.6%, and 44.83% is achieved by HDPC over HMR considering genomic data size of 4988 bp, 10207 bp, 15131 bp, 576874 bp, and 948066 bp respectively. An average execution time reduction of 53.96% is attained by proposed HDPC over exiting HMR considering varied genomic data size.

Theoretical execution time of HDPC i.e., $\vec{u}_{\mathbb{K}}$ given in Eq. (23) is computed and is compared against the experimented value attained considering varied genomic data size. Result attained is shown in Fig. 7. Slight variation is seen from practical and theoretical job execution computation. Overall good correlation is seen among theoretical and practical execution time. From experiment outcome it is clear execution of gene sequence (bioinformatics) analysis on proposed HDPC framework attain superior performance when compared with HMR framework. Accuracy and correctness of theoretical job execution model of HDPC framework presented is proved through correlation measures.

Table 1: Gene sequence considered for experiment analysis

Reference genome sequence	Sequence length	Query genome sequence	Sequence length
NC_000015.10	101991189 bp	NC 026141.2	4988 base pair
NC_000015.10	101991189 bp	NC 010955.1	10207 base pair
NC_000015.10	101991189 bp	NC 015123.1	15131 base pair
Saccharomyces cerevisiae S288c chromosome XII	1001933 bp	Saccharomyces cerevisiae S288c chromosome V_BK006939.2	576874 base pair
Saccharomyces cerevisiae S288c chromosome XII	1001933 bp	Saccharomyces cerevisiae S288c chromosome XVI_BK006949.2	948066 base pair

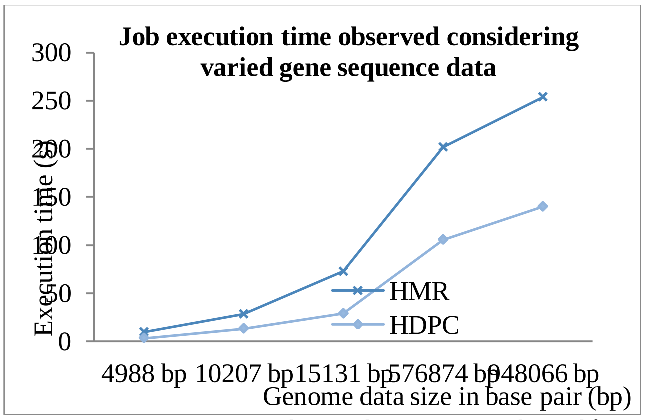

Fig. 6: Genomic sequence analysis total execution time observed for varied genome data size conducted on HDPC and HMR frameworks

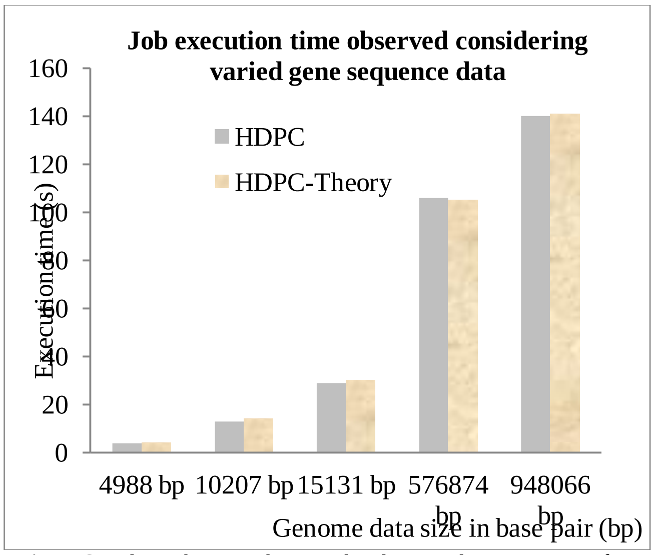

Fig. 7: Correlation between theoretical and practical execution time for varied genome data analysis on HDPC framework

3.4. E-commerce Data Analysis Performance Evaluation of HDPC over HMR

This section carryout performance evaluation of E-commerce data analysis on HDPC and HMR framework. For experiment analysis Wordcount (Text computation/mining) application [21] is used. Amazon product data [31] is used for experiment analysis which composed of 142.8 million reviews spanning May 1996 - July 2014. However, this work considers the experiment case shown in Table II. Wordcount analysis is performed on Amazon review dataset and result is graphically shown in Fig. 8.

The experiment outcome shows an execution time reduction of 47.4%, 50.98%, 49.09%, and 55.64% is achieved by HDPC over HMR considering review data size of 2,982,326, 3,268,695, 3,447,249, and 5,748,920 respectively. An average execution time reduction of 50.77% is attained by proposed HDPC over exiting HMR considering varied review data size.

Theoretical execution time of HDPC i.e., $\vec{u}_{\mathbb{K}}$ given in Eq. (23) is computed and is compared against the experimented value attained considering varied amazon review data size. Result attained is shown in Fig. 9. Slight variation is seen from practical and theoretical job execution computation. Overall good correlation is seen among theoretical and practical execution time. From experiment outcome it is clear execution of Wordcount (text computing/mining) analysis on proposed HDPC framework attain superior performance when compared with HMR framework. Accuracy and correctness of theoretical job execution model of HDPC framework presented is proved through correlation measures.

Table 2: E-commerce data considered for experiment analysis

Experiment ID	Dataset	Number of reviews	Number of product
1	Health and personal cares	2,982,326	263,032
2	Sports and outdoors	3,268,695	532,197
3	Cellphones and accessories	3,447,249	346,793
4	Clothing shoes and jewelry	5,748,920	1,503,384

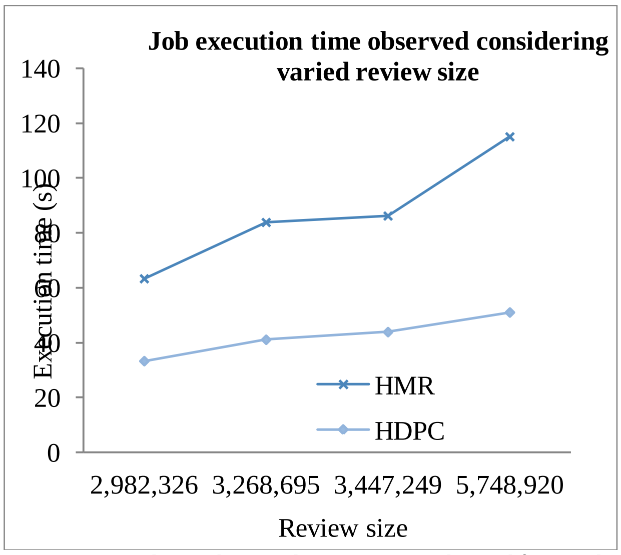

Fig. 8: Stream data analysis total execution time observed for varied genome data size conducted on HDPC and HMR frameworks

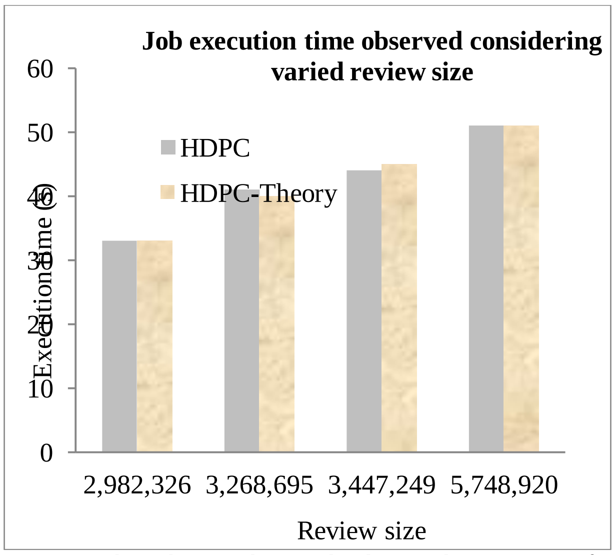

Fig. 9: Correlation between theoretical and practical execution time for varied genome data analysis on HDPC framework

3.5. Result Comparison of HDPC with state-of-art Technique

In this section the execution of the stream and non-stream applications namely word frequency statistics and hot word detection respectively is presented. The results presented here prove that the HDPC framework reduces the total job execution attained due to the proposed makespan model considering hybrid parallelization.

An average job execution time reduction of 49.51%, 55.66%, 53.97%, and 50.7% is attained by proposed HDPC over exiting HMR [11] for non-stream application, stream application, bioinformatics application, and text computing/mining application respectively. The comparative analysis over state-of-art technique is tabulated in Table III shows the efficiency of HDPC over state-of-art technique in terms of robustness and scalability. Since, HDPC support execution of stream, non-stream, bioinformatics and text mining application over cloud platforms. Our HDPC job makespan/execution model aided in better cloud resource utilization. Theoretical comparison evaluation is considered and attained better result when compared with [19] and [21]. Adoption of cloud platform aid in providing scalable processing of huge amount of stream and non-stream data of various types on large computing clusters. All these feature attributed to the performance improvement of HDPC over state-of-art models.

Table 3: Comparson with state of art technique

	[11]	[18]	[19]	[20]	[21]	[28]	HDPC
MapReduce platform considered	Custom	Hadoop	Hadoop	Hadoop	Hadoop	Hadoop	Custom
Cloud support for MapReduce execution	Yes	Yes	No	Yes	Yes	No	Yes
Support both stream and non-stream application	Yes	No	No	No	No	No	Yes
Support execution of text mining application	Yes	No	Yes	Yes	Yes	Yes	Yes
Support execution of bioinformatics application	No	Yes	No	No	No	No	Yes
Makespan accuracy evaluation considered	No	No	Yes	No	Yes	No	Yes
Hybrid parallelization considered	No	No	No	No	No	No	Yes
Parallelization under multi-core environment	No	No	No	No	No	No	Yes
Parallelization of MapReduce phase	Yes	No	No	No	No	No	Yes
MapReduce Platform adopted	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Average percentage improvement over HMR framework	30.0%	40.28%	13.33%	34.83%	27.7%	43.91%	52.58%

4. Conclusion

This paper discussed about the significance of cloud computing framework. Further, disused the working process Hadoop MapReduce framework and highlighted the drawback and limitation of HMR. Further, to utilize resource efficiently and minimize execution time this manuscript introduced HDPC framework on cloud computing platform. HDPC offers parallel execution of Map and Reduce phase. To further enhance resource utilization parallel execution of map and reduce operation is carried out considering multi-core environments available with virtual computing workers. Lastly, this work presented job makespan/ execution model and working structure of HDPC framework. Experiment are conducted on Microsoft Azure HDInsight cloud platform considering stream and non-stream application to evaluate performance of HDPC framework over existing computing model. The outcome shows an average makespan/execution time performance improvement of 49.51%, 55.66%, 53.97%, and 50.7% considering non-stream application, stream application, bioinformatics application, and text computing/mining application, respectively. Overall good correlation is seen among practical execution and theoretical execution outcome shows proposed HDPC framework is robust, scalable, cost efficient and support dynamic analysis on cloud computing environment. The future work would consider performance evaluation of HDPC framework considering more diverse application applications and datasets.

References

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

[2] 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, p. 260, 2008.

[3] 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.

[4] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica,“Spark: Cluster Computing with Working Sets,” in Proceedings of the 2nd USENIX Conference on Hot topics in Cloud Computing, (Boston,MA), June 2010.

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

[6] “Apache Hadoop.” [Online]. Available: http://hadoop.apache.org/. [Accessed: 21-july-2018].

[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. Shi et al., "Mammoth: Gearing Hadoop Towards Memory-Intensive MapReduce Applications," in IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 8, pp. 2300-2315, Aug. 1 2015.

[9] J. 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.

[10] M. Zaharia, A. Konwinski, A.D. Joseph, R.H. Katz and I. Stoica, &ldquo,Improving Mapreduce Performance in Heterogeneous Environments,&rdquo, Proc. Eighth USENIX Conf. Operating Systems Design and Implementation (OSDI), pp. 29-42, 2008.

[11] D. Dahiphale et al., "An Advanced MapReduce: Cloud MapReduce, Enhancements and Applications," in IEEE Transactions on Network and Service Management, vol. 11, no. 1, pp. 101-115, March 2014.

[12] E. Deelman, G. Singh, M. Livny, B. Berriman and J. Good, "The cost of doing science on the cloud: The Montage example," 2008 SC - International Conference for High Performance Computing, Networking, Storage and Analysis, Austin, TX, pp. 1-12, 2008.

[13] N. Chohan, C. Castillo, M. Spreitzer, M. Steinder, A. Tantawi, and C. Krintz, “See spot run: using spot instances for mapreduce workflows,” in Proc. 2010 USENIX Conference on Hot Topics in Cloud Computing, ser. HotCloud’10. USENIX Association, pp. 7–7, 2010.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] Z. Zhang, L. Cherkasova and B. T. Loo, "Optimizing cost and performance trade-offs for MapReduce job processing in the cloud," 2014 IEEE Network Operations and Management Symposium (NOMS), Krakow, 2014, pp. 1-8.

[23] K. Chen, J. Powers, S. Guo and F. Tian, "CRESP: Towards Optimal Resource Provisioning for MapReduce Computing in Public Clouds," in IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 6, pp. 1403-1412, June 2014.

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

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

[26] Changjian Wang; Yuxing Peng; Mingxing Tang; Dongsheng Li; Shanshan Li; Pengfei You, "MapCheckReduce: An Improved MapReduce Computing Model for Imprecise Applications," Big Data (BigData Congress), 2014 IEEE International Congress on , vol., no., pp.366,373, June 27 2014-July 2 2014.

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

[28] 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.

[29] Saccharomyces genome database (SGD). (2015). [Online] Available: http://www.yeastgenome.org/

[30] Influenza Virus Resource. (2015). [Online] Available: http://www.ncbi.nlm.nih.gov/genomes/FLU/FLU.html.

[31] Amazon product dataset “http://jmcauley.ucsd.edu/data/amazon/”, last accessed sep 2, 2018.

[32] R. M. Esteves and C. Rong, "Using Mahout for Clustering Wikipedia's Latest Articles: A Comparison between K-means and Fuzzy C-means in the Cloud," 2011 IEEE Third International Conference on Cloud Computing Technology and Science, Athens, pp. 565-569, 2011.