International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-1, November 2019

Enhhanced and Efficient Memory Model For Hadoop Mapreduce

Archana Bhaskar, Rajeev Ranjan

Abstract*—Usage of high-performance computing (HPC) infrastructure adopting cloud-computing environment offers an efficient solution for executing data intensive application. MapReduce (MR) is the favored high performance parallel computing framework used in BigData study, scientific, and data intensive applications. Hadoop is one of the significantly used MR based parallel computing framework by various organization as it is freely available open source framework from Apache foundation. The existing Hadoop MapReduce (HMR) based makespan model incurs memory and I/O overhead. Thus, affecting makespan performance. For overcoming research issues and challenges, this manuscript presented an efficient parallel HMR (PHMR) makespan model. The PHMR includes a parallel execution scheme in virtual computing worker to reduce makespan times using cloud computing framework. The PHMR model provides efficient memory management design within the virtual computing workers to minimize memory allocation and transmission overheads. For evaluating performance of PHMR of over existing model experiment are conducted on public cloud environment using Azure HDInsight cloud platform. Different application such as bioinformatics, tex mining, stream, and non-stream application is considered. The overall result obtained shows superior performance is attained by PHMR over existing model in term of makespan time reduction and correlation among practical and theoretical makespan values.

***Keywords*— Big data, Data intensive application, Caching, Cloud computing, high-performance computing, HMR, Scientific application.**

I. INTRODUCTION

Cloud computing plays a major role for attaining scalable computing for scientific and data intensive applications. The cloud computing adopts distributed architecture which is capable of processing large amount of information collected by various organization. For example genomic sequence technology, social network, sensor network etc. Performing scalable computing on this large, raw and dynamic unstructured information is most preferred across organizations. The exiting model such as Phoenix [1], Mars [2] and Dryad [3], and Spark [4] are not efficient in real-time analysis on stream information. Google came up with parallel computing architecture namely MapReduce framework [5]

for performing real-time examination of data intensive and scientific application.

Revised Manuscript Received on November 30, 2019.
***** Correspondence Author
Archana Bhaskar*, Research Scholar , Assistant Professor, Department of Computer Science and Engineering, REVA University, Bangalore, India. Email-archanabaskar007@gmail.com,
Dr. Rajeev Ranjan, Associate Professor, Department of Computer Science, Indian Institute of Information Technology-Allahabad, REVA University, Bangalore, India. Emailrajeevranjan@reva.edu.in

Among all Hadoop MapReduce [6] is the most widely used and adopted [7] framework due its open source nature and ease of deployment, and scalability.

The HMR framework is explicitly composed of Map and Reduce phase. Along with it also composed of Setup, Shuffle and Sort phase. The architecture of HMR is shown in figure 1. The HMR cluster is composed of set of computing worker (virtual machine) which is connected to master virtual computing worker. The user submit the jobs to HMR cluster. Then the master worker splits the jobs into tasks and assign it to Map and Reduce worker. In Setup phase, the input data from cloud storage location that needs to be processed by Map workers are logically segmented into identical blocks.

In Map phase, the master worker distribute the task to the Map worker. The Map worker takes input as key and value pairs as (k_1, v_1) and generate list of (k_2, v_2) intermediary key and value pairs as its outcome [16], [19]. Post completion of Map phase, shuffle phase is initialized. Shuffle phase collect the intermediary key and value pairs across Map worker. Then sort function is executed on collected intermediary data of Map phase. For easiness, shuffle and sort processed are merged together and named as Shuffle phase. Lastly, based on user defined operation Reduce phase are initialized. The outcome of Reduce phase is written on to cloud storage location.

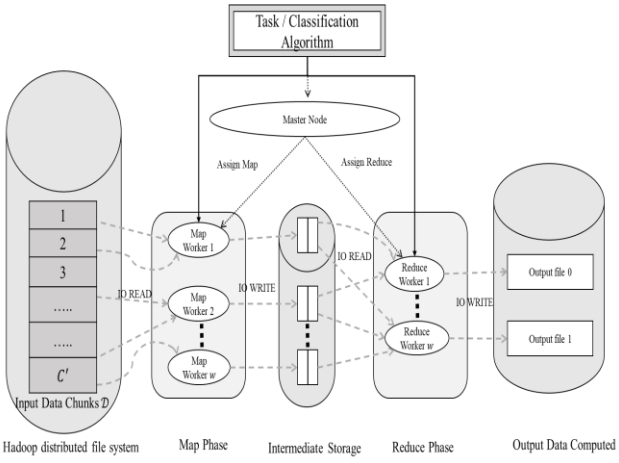

Fig. 1. Hadoop Map-Reduce (HMR) Model [21], [31]

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 [14], [15]. However, these model incurs makespan overhead. Along with accuracy of makespan model are not efficient as they don't considers both overlapping and non-overlapping section during shuffling process. Thus, designing HMR makespan model is challenging. Since it composed of various stages such as Map, Shuffle and Reduce stage, where each stage involves memory and I/O operation. Besides, the initial wave of shuffling process is executed in parallel with Map stage (this stage is depicted as overlapping stage). Post completion of Map stage rest of the waves of shuffling phase are processed (this stage is depicted as non-overlapping stage).

Recently, for enhancing the performance of Hadoop application [16], various efficient Hadoop models are presented by numerous researchers [17], [18], [19], [20], [21], [22], [23], [24] and [25]. A comprehensive information for job prediction and optimization can be presented using Starfish [17] model which gathers an active Hadoop task profile at a satisfactory granularity. Elasticiser [18] is presented for resource allocation based on VMs which is considered above Starfish. However, gathering an active Hadoop task profile with comprehensive information can lead to large overhead and hence high over-predicted task run-time. In [19], [20], and [21] utilizes both overlapping and non-overlapping stages of shuffling process and for task prediction a conventional linear regression method is used. This application also helps to predict the required number of resources for various tasks with deadline constraints. CRESP [22] predicts task execution efficiently and helps to allocate resources based on MapReduce slots. However, in CRESP application models, the effect of number of reduced jobs are discarded. In [21] and [22], the number of reduced jobs are constant, incurs memory and I/O overhead and failed to provide data locality awareness. To address issue and challenges [24] introduced centralized caching mechanism. However, showed distributed caching [25], [26] and shared caching [27] aid in system performance. In [28] presented a model to provide lockless FIFO queue that connects HMR and external applications. However, such model are designed considering homogenous storage architectures and they fail under heterogeneous architecture [28]. Since they require dynamic I/O and memory optimization model.

From above analysis it is quite evident of the drawbacks of hadoop and Therefore, minimizing makespan time and utilizing resource efficiently with minimal costs is most desired of cloud based computing model. To attain this objective a Parallel Hadoop MapReduce makespan model is presented in this manuscript. The significance of PHMR makespan are described as follows. The PHMR present better in-memory caching mechanism. Further, PHMR utilize the resource more efficiently (i.e., minimize the existence of unutilized resources (i.e. slots)). The parallel execution strategy of PHMR framework is comparable with model presented in [29], [30]. The virtual computing node i.e., both Map and Reduce virtual computing node are processed using number of slots being allocated. Proper usage of resources (i.e. allocated slots) will aid in reducing makespan time. The local memory management technique used in this work allows efficient parallel execution with reduction of memory overheads. Similar to HMR framework, our PHMR makespan framework takes the information in the form of blocks. The Map virtual computing node in MR model carryout operations on these information blocks. In the PHMR framework, the information blocks is again segmented to parallel blocks of data to perform parallel execution under multi-core environment in the virtual computing workers. The execution under multicore environment with minimal I/O and memory overhead aid in utilizing resource more efficiently (i.e. reducing of unutilized slots) and minimize makespan.

The manuscript is articulated as follows. The proposed Parallel Hadoop MapReduce Makespan Model for utilizing unutilized slot more efficiently is presented is presented in section II. In section III, experiment study is presented. In last section, the conclusion along with future research direction is described.

II. PARLLEL HADOOP MAPREDUCE MAKESPAN MODEL

This work present a Parallel Hadoop MapReduce (PHMR) makespan model for utilizing unutilized slot more efficiently. Firstly, a system model is described. Then, sequential Hadoop MapReduce makespan model is described. Lastly, the proposed parallel Hadoop MapReduce makespan model is presented.

a) System model:

This work adopt cloud computing platform. Therefore, Parallel Hadoop MapReduce (PHMR) makespan model is composed of master computing node/worker, map and reduce computing node. The master computing node initialize q map and reduce virtual computing nodes. Each virtual computing nodes is composed of n computing cores. Let $U_{\mathcal{T}}$ depicts the processing time to initialize the virtual computing framework. Let $U_{\mathbb{M}}$ depicts the mean makespan time of the q map computing workers. Let $U_{\mathbb{R}}$ depicts the mean computing time required by q reduce computing workers to perform shuffle, sort and reduce computation. Let's assumes an execution operation g to be evaluated on heterogeneous computing environment using MR framework on cloud environment. The evaluation is done considering an application s is expressed using following equation

$$s = g(b) \tag{1}$$

Where b depicts the input argument.

The MR framework virtual computing (worker) nodes for performing computation of Map and Reduce task. There induce overhead or cost incurred in executing g on cloud computing framework. The overhead observed due to I/O operation are depicted by P^d and overhead observed due to computation is depicted by U.

The overhead are estimated using makespan incurred. Thus, the overall or cumulated makespan is depicted by U_S^P.

Let w depicts an I/O event occurring on cloud computing environment. The operation $Mem(y,n,w)$ depicts the operation to collect n bytes of information from blob container location of cloud to the local virtual computing storage blob location y. The makespan overhead observer for transferring data id depicted by $MkspnOvd(w)$. The operation to depict the process of storing the execution outcomes toward Hadoop distributed storage location on cloud computing platform in a blob storage container z is depicted as $DfsExe(z,n)$.Let x depicts the active virtual computing node at any given instance time u. P_J is the makespan for initializing x virtual computing nodes. Considering c bytes of information from the cloud computing platform location toward the virtual computing memory of the virtual computing worker is considered to posses p_u^c. Thus, $\mathrm{p_u^c(x)}$ depicts the I/O makespan perceived considering different x virtual computing workers.

b) *Makespan modelling of Hadoop MapReduce:*

This work assumes an operation g that will be evaluated using HMR framework. The Map virtual computing node and Reduce virtual computing node using cloud computing configuration are initialized. That is, number of virtual computing node x is initialized. The data e is divided into e' segments. Each segment is of $e'' * c$ bytes in size. The Map virtual computing nodes read the input array of information from remote blob container location and keeps the information in memory cache depicted using notation $\mathbb{C}$. The makespan of this process for different x is expressed as follows

$$P^d(e') = \left(P_J + (p_u^c * c * e')\right) \tag{2}$$

The application s is computed utilizing in-memory caching information and the resultant outcome of this operation information is stored in remote blob container location for future use (i.e., in process of Reduce virtual computing nodes. The execution of the dataset information e' by using Map virtual computing node is expressed using following equation

$$U(e') = (u_c * e'' * e') \tag{3}$$

Post completion of Map execution the outcome information (intermediary) are kept remote blob container location. Since different virtual computing have different makespan and performing synchronization are difficult. The Reduce virtual computing nodes are initialized post completion of Map task execution. The Reduce virtual computing node perform reduce operation on remotely stored intermediary information. The outcome of reduce jobs are obtained as array S. The makespan time induces by Reduce virtual computing node for performing reducing operation are expressed using following equation

$$\frac{e}{e''} - 1 \tag{4}$$

As HMR perform sequential execution, it can be seen the makespan at the start is significantly high due to segmenting and transmission of information across remote blob container $\left(i.e., P^D(e')\right)$ and then later for executing algorithm execution $\left(i.e., U(e')\right)$. Let e'_p depicts the session period through which the application computation and segmenting process considering transmission makespan are identical $\left(i.e., U(e') = P^D(e')\right)$. The makespan time period is measured as a set depicts as $\left[1 \ldots e'_p \ldots \widehat{e'}\right]$. The makespan at start that is., $e' < e'_p$ are expressed using following equation

$$U_S^P(e') = \left(\frac{e}{e''} + 1\right) P^D(e'), \forall e' < e'_p \tag{5}$$

The makespan time $U_S^P(e')$ can also be computed as follows

$$U_S^P(e') \cong \left(\frac{e.P_J}{e''}\right) + (e * p_u^c * c * e'), \forall e' < e'_p \tag{6}$$

The makespan time considering the time period when $\forall e' > e'_p$ is expressed as follows

$$U_S^P(e') = \left(x * P^D(e')\right) + (e) * u_c, \forall e' > e'_p \tag{7}$$

The makespan time $U_S^P(e')$ is approximated as follows

$$U_S^P(e') \cong (p_u^c * c * e'') + (e * u_c * p_u^c * e'), \forall e' < e'_p \tag{8}$$

c) *Parallel Hadoop MapReduce makespan model:*

The HMR model presented in above section adopts serial execution strategy. The virtual computing slots allocated in the HMR are at the start is dominated or used by the Map computing node. Post Map operation, Reduce computing node start utilizing the remaining slots. As a result, multiple slots remain unutilized. Thus, increases the makespan time for executing algorithm s. In the proposed Parallel HMR, both Map and Reduce slots allocated in such a manner to utilize slot more efficiently. The Map virtual computing node obtains the chunk of dataset information from the remote blob container location and keeps the dataset information in its virtual computing memory (in-memory caching). In the PHMR makespan the obtained dataset information e' is then segmented into parallel chunks e'' for executing under multicore environment. The makespan induced due to this process across the x workers is expressed as follows

$$P^d(e'', x) = \left(P_J + (p_u^c(x) * e'' * c)\right) \tag{9}$$

Each chunk of e' is then computed parallelly using multi-core environment available with virtual computing workers. Along with parallel execution the previously computed and redundant information is removed from the in-memory caching of virtual computing worker to reduce storage and makespan overheads. It can be seen that in-memory caching attainable/obtainable from the Map virtual computing nodes decreases with respect to increase in makespan. The usage of in-memory caching reduction is directly proportional to computational operation g used. The makespan is computed considering the time needed to finish the parallelized chuck e' used in executing application s by Map virtual computing node are expressed using following equation

$$U(e'') = (u_c * e'') \tag{10}$$

The Map virtual computing nodes keeps the processed output in the remote blob containers location.

The Reduce virtual computing node use these outcome information for executing task of reduce operation. The makespan time induced by Reduce virtual computing node are expressed using following equation

$$\left(\frac{e}{e''} - 1\right) \tag{11}$$

The overall or cumulated makespan induced is depicted using notation $U_S^P(e'')$, where the input dataset information vector is depicted using notation $B[y]$ and the algorithm outcome information vector is depicted using notation $S[y]$. Let $\left[1,2,\ldots e_p'' \ldots \widehat{e''}\right]$ depicts the discrete makespan time considering $e'' > e_p''$ is expressed as follows

$$U_S^P(e'') = \left(\frac{e}{e'' * x} + 1\right) P^D(e'', x), \forall e'' < e_p'' \tag{12}$$

At time period $e'' > e_p''$ the makespan time can be expressed as follows

$$U_S^P(e'') = \left(x * P^D(e'', x)\right) + \left(\frac{e}{x * u_c}\right), e'' > e_p'' \tag{13}$$

To utilize unoccupied slot of x virtual computing node more efficiently an optimization function is introduced as follows

$$P^D\left(e_p''(x), x\right) = U_S^P\left(e_p''(x)\right) \tag{14}$$

where e_p'' depicts the time period within array of $\left[1 \ldots \widehat{e''}\right]$ where $P^D\left(e_p'', x\right) = U(e'')$. The proposed parallelized model can attains better makespan performance using optimized slot scheduling mechanism of the virtual computing workers. Along with, efficient in-memory caching utilization mechanism aided in attaining better makespan performance which is experimentally shown below.

III. RESULT AND ANNALYSIS

Experimental study of PHMR performance attained over existing model [19] considering diverse application is presented in this section. HMR is the most preferred MR framework for executing data intensive and scientific application using cloud computing framework [31]. The Hadoop cluster is composed of 4 slave worker with 1 master worker. The Hadoop cluster is deployed on Microsoft Azure public cloud platform. This work used Hadoop 2.7, where each worker (both master and slave) is composed of 120 GB of cloud blobs/ container size, 7 Gigabytes RAM and 4 cores. Identical configuration is used for executing diverse application on both HMR and PHMR. For evaluating performance of PHMR over HMR diverse application from memory, CPU to I/O intensive application is considered. All three cases study are required to evaluate the robustness of any parallel computing framework. Thus, this work considered diverse application such as Gene sequence analysis, non-stream data and stream data analysis. Currently, the genetic data are extremely large, thus Gene sequence analysis are memory, CPU to I/O intensive. HMR model are good in handling CPU intensive. Thus it is important to see how it performs considering Memory and I/O intensive task. The PHMR model is designed to address both Memory and I/O and retaining CPU feature of HMR and it is important to see how it performs considering different genomic data (size) (i.e. considering both short and long read). More details of genomic data used are detailed in later section.

Along with we also considered large stream and non-stream application. This work considered Ecommerce (non-stream) data analysis. These application are generally CPU intensive with minimal I/O requirement. On the other side non-stream application we had considered Twitter dataset. These application involves both CPU and I/O intensive. Thus, it is important to see how both PHMR and HMR perform on these application. More detail of dataset used are discussed in later section.

a) Bioinformatics application (gene sequencing) performance evaluation of PHMR over HMR:

This section carryout performance evaluation of gene sequence analysis [19] on PHMR and HMR framework. Application related to Cancer research, Genetic Diseases identification, Reproductive Health etc. 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 [32] and the query sequences obtained from the influenza virus database [33]. 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. 2.The experiment outcome shows an execution time reduction of 64.39%, 52.98%, 47.76%, and 44.87% is achieved by PHMR over HMR considering genomic data size of 4988 base pairs, 10207 base pairs, 576874 base pairs, and 948066 base pairs, respectively. An average execution time reduction of 52.43% is attained by proposed PHMR over exiting HMR considering varied genomic data size.

Theoretical optimization of makespan time of PHMR i.e., U_S^P given in Eq. (14) is computed and is compared against the experimented value attained considering varied genomic data size. Result attained is shown in Fig. 3. Slight variation is seen from experimented and mathematical job makespan computation. Overall good correlation is seen among experimented and mathematical makespan time. From experiment outcome it is clear execution of gene sequence (bioinformatics) analysis on proposed PHMR framework attain superior performance when compared with HMR framework. Through correlation measure accuracy and correctness of mathematical job makespan model of PHMR is demonstrated.

TABLE I. 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
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

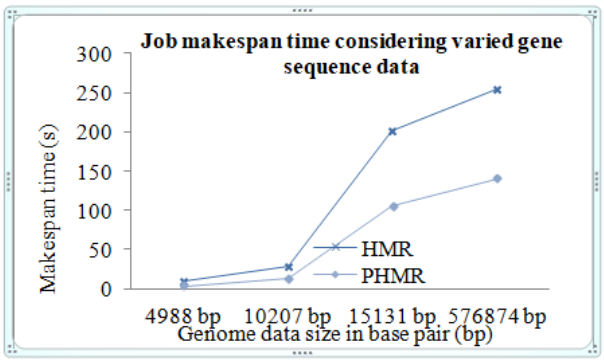

Fig. 2. Total makespan time for performing genomic sequence analysis on varied genome data size conducted on PHMR and HMR frameworks

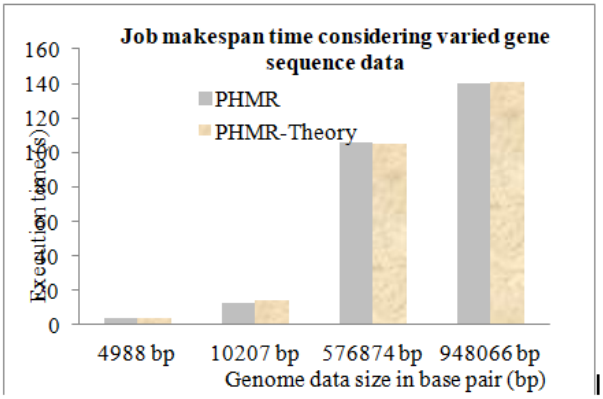

Fig. 3. Correlation between theoretical and practical execution time for varied genome data analysis on PHMR framework

b) E-commerce data analysis performance evaluation of PHMR 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 [22] is used. Amazon product data [34] is used for experiment analysis which composed of 142.8 million reviews from May 1996 to 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. 4. The experiment outcome shows an execution time reduction of 50.98%, 49.09%, and 55.64% is achieved by PHMR over HMR considering review data size of 3,268,695, 3,447,249, and 5,748,920 respectively. An average makespan time minimization of 51.9% is attained by proposed PHMR over exiting HMR considering varied review data size.

Theoretical optimization of makespan time of PHMR i.e., U_S^P given in Eq. (14) is computed and is compared against the experimented value attained considering varied amazon review data size. Result attained is shown in Fig. 5. Slight variation is seen from experimented and mathematical job makespan computation. Overall good correlation is seen among experimented and mathematical makespan time. From experiment outcome it is clear execution of Wordcount (text computing/mining) analysis on proposed PHMR framework attain superior performance when compared with HMR framework. Through correlation measure accuracy and correctness of mathematical job makespan model of PHMR is demonstrated.

TABLE II. E-COMMERCE DATA CONSIDERED FOR EXPERIMENT ANALYSIS

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

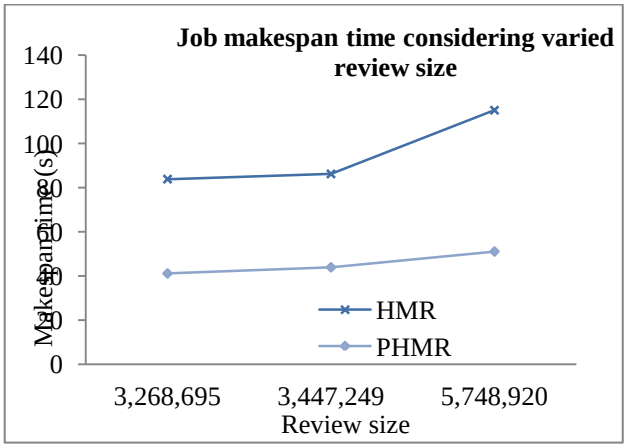

Fig. 4. Total makespan time for performing E-commerce analysis on varied datasets and review size conducted on PHMR and HMR frameworks

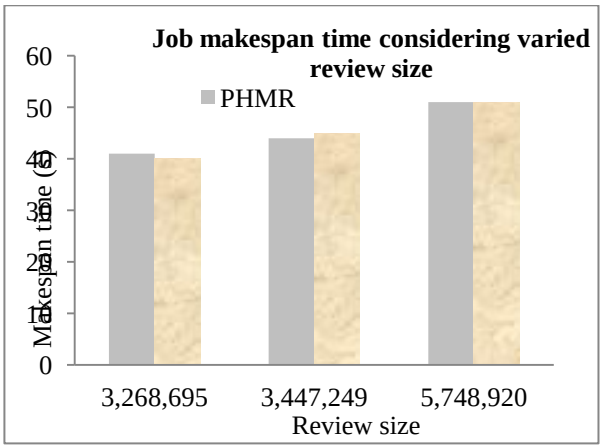

Fig. 5. Correlation between theoretical and practical execution time on varied datasets and review size conducted on PHMR framework

c) Stream data analysis performance evaluation of PHMR over HMR:

This section carryout performance evaluation of stream application on PHMR and HMR framework. The hot word detection application [35] is considered. The application and is developed using C# programming language. For performance evaluation The "Movietweetings" dataset [36] is considered. For experimental evaluation this work considers stream tweet size of 25K, 50K, 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 PHMR and result are noted as shown in Fig. 6. The outcomes shows an execution time reduction of 56.63%, 46.84%, and 56.51% is achieved by PHMR over HMR considering stream data size of 25K, 50K, and 100K respectively. An average makespan time minimization of 53.33% is attained by proposed PHMR over exiting HMR considering varied stream data size.

Theoretical optimization of makespan time of PHMR i.e., U_S^P given in Eq. (14) is computed and is compared against the experimented value attained considering varied tweet data size. Result attained is shown in Fig. 7. Slight variation is seen from experimented and mathematical job makespan computation. Overall good correlation is seen among experimented and mathematical makespan time. From experiment outcome it is clear execution of stream data analysis on proposed PHMR framework attain superior performance when compared with HMR framework. Through correlation measure accuracy and correctness of mathematical job makespan model of PHMR is demonstrated.

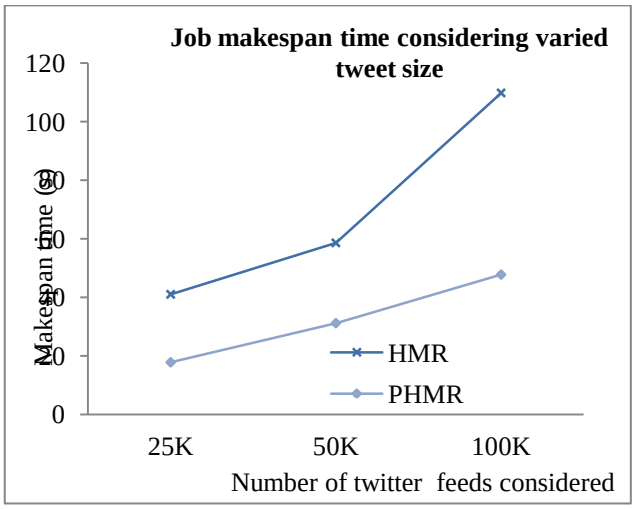

Fig. 6. Total makespan time for performing stream analysis on varied datasets (tweet) size conducted on PHMR and HMR frameworks

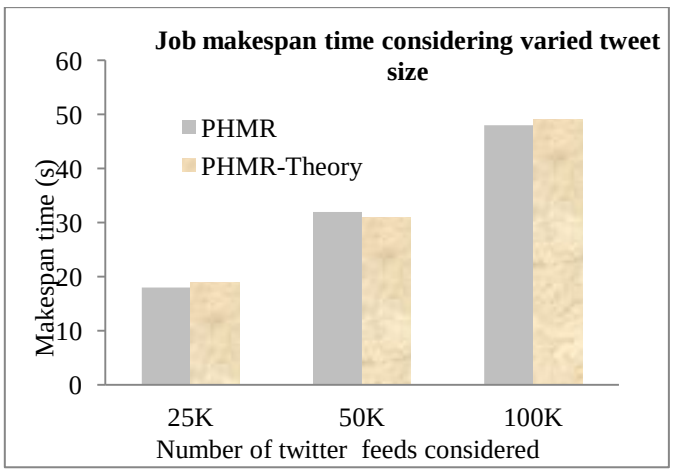

Fig. 7. Correlation between theoretical and practical execution time on varied datasets (tweet) size conducted on PHMR framework

d) Result and discussion:

This section discusses the result attained by PHMR over HMR and other Hadoop based MR model for executing diverse application such as genomic sequence analysis, non-stream application for analyzing ecommerce review using word frequency statistics and stream applications such as hot word detection is used. The results articulated here shows that the PHMR framework minimizes the total job makespan attained due to the proposed makespan model considering memory optimization and parallel execution strategy to reduce unutilized/null slots. An average job makespan time reduction of 52.43%, 51.9% and 53.33% is attained by proposed PHMR over exiting HMR [19] for bioinformatics application, e-commerce (non-stream) analysis, and text mining (stream) analysis, respectively. The comparative analysis over existing methods is tabulated in Table III shows the robust and scalable and effectiveness of PHMR over existing methods. Since, PHMR support execution of bioinformatics, text mining, stream, and non-stream application over cloud platforms. Along with PHMR makespan model utilizes cloud resource more efficiently. Correlation measure shows that our mathematical attain better accuracy than existing mathematical makespan model [20] and [22]. Usage of public cloud environment aid in providing scalable processing of huge amount of data (i.e., both stream and non-stream) on different Hadoop cluster size. The above described factor enabled PHMR to attain better performance than most standard Hadoop based parallel computing model.

TABLE III. . COMPARSON WITH EXISITNG PARALLEL COMPUTING MODEL

	[19]	[20]	[21]	[22]	[23]	PHMR
Type of application or algorithm used	Bioinformatics	Word count application	Tera sort, and Word count application	Sort and Word count	Sort and Word count application	Bioinformatics, stream and non-stream
MR computing environment considered	HMR	HMR	HMR	HMR	HMR	HMR
Cloud computing environment used	Yes	NO	Yes	Yes	No	Yes
Average makespan minimization over Hadoop (%)	40.01%	13.01%	34.08%	27.1%	43.67%	52.47%

IV. CONCLUSION

This work discussed the drawback of Hadoop MapReduce framework. Further, the significance of memory and I/O requirement for designing efficient HMR framework are discussed here.

This work aimed to minimize makespan time and utilize resource efficiently using cloud computing environment. For attaining such model a Parallel Hadoop MapReduce makespan model is presented. The PHMR makespan model is designed considering minimizing unutilized/null slot of virtual computing node (i.e., presenting an optimization model for local worker memory management model). Thus, our model allows efficient parallel execution with reduction of memory overheads. Further, the chunked data is again segmented to allow parallel execution within Map computing workers (virtual machines). The minimal memory usage (memory I/O overhead) and parallel execution strategy aid in utilizing resource (slots) efficiently. Experiment are conducted on using Microsoft Azure HDInsight public cloud platform. Different application such as bioinformatics, text mining, stream and non-stream application are used to validate the performance of PHMR framework over state-of-art parallel computing method. The outcome shows an average makespan time performance improvement of 52.43%, 51.9% and 53.33% is attained by proposed PHMR over exiting HMR for bioinformatics application, e-commerce (non-stream) analysis, and text mining (stream) analysis, respectively. An average of 52.47% improvement is attained by PHMR over existing HMR model considering diverse application. Overall good correlation is seen among practical execution and theoretical execution outcome shows proposed PHMR framework is robust, scalable, cost efficient and support dynamic analysis on cloud computing environment. Future research direction will considers developing novel gene sequencing algorithm that utilizes memory more efficiently. And also considers developing SLA driven scheduling for supporting application deadline requirement and diverse application.

The future work would consider performance evaluation of PHMR 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. 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.
17. 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.
18. 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.
19. 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.
20. 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.
21. 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.
22. 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.
23. 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.
24. Apache, Centralized Cache Management in HDFS. Update date 2014.
25. Y. Huang, Y. Yesha, M. Halem, Y. Yesha and S. Zhou, "YinMem: A distributed parallel indexed in-memory computation system for large scale data analytics," 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, pp. 214-222, 2016.
26. Zhang, J., et al., A Distributed Cache for Hadoop Distributed File System in Real-Time Cloud Services, in Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing. 2012, IEEE Computer Society. p. 12-21.
27. Longbin, L., et al. ShmStreaming: A Shared Memory Approach for Improving Hadoop Streaming Performance. in Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on. 2013.
28. J. Kim, H. Roh and S. Park, "Selective I/O Bypass and Load Balancing Method for Write-Through SSD Caching in Big Data Analytics," in IEEE Transactions on Computers, vol. 67, no. 4, pp. 589-595, April 1 2018.
29. L. G. Valiant. A bridging model for parallel computation. Communications of the ACM, 33(8):103–111, August 1990

30. G. Malewicz, M. Austern, A. Bik, J. Dehnert, J. Horn, I. Leiser, N. Czajkowski, “Pregel: a system for largescale graph processing,”. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pages 135–146, Indianapolis, Indiana, USA, 2010.
31. T. White, Hadoop: The Definitive Guide. O’Reilly Media, 2009.
32. Saccharomyces genome database (SGD). (2015). [Online] Available: http://www.yeastgenome.org/
33. Influenza Virus Resource. (2015). [Online] Available: http://www.ncbi.nlm.nih.gov/genomes/FLU/FLU.html.
34. Amazon product dataset “http://jmcauley.ucsd.edu/data/amazon/”, last accessed sep 2, 2018.
35. 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.
36. 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.

AUTHORS FROFILE

Ms. Archana Bhaskar, Assistant Professor is Currently pursuing Ph.D in Computer Science and Engineering from REVA University. She is Currently working in Acharya Institute of Technology. She has published around 5 papers in both national and international journals.. Her areas of interests includes Cloud , Big Data, Hadoop and Wireless Sensor Networks

Dr. Rajeev Ranjan, Associate Professor holds Ph.D. in Computer Science from Indian Institute of Information Technology-Allahabad. He joined Indian Institute of Information Technology-Allahabad (IIIT-Allahabad) as Junior Research Fellow (JRF), followed by Research Associate (RA). Currently, he is working in REVA University as an Associate Professor. He has published 11 papers in International and National journals and conferences of repute and has authored 02 book chapters. He has 10 years of experience in industry and academics. His area of interest include Wireless sensor networks- coverage & connectivity, Sensor deployment and localization, Wireless sensor statistical routing etc.