Cost-aware optimal resource provisioning Map-Reduce scheduler for hadoop framework

Archana Bhaskar[1], Rajeev Ranjan[2]
[1]Department of Computer Science and Engineering, REVA University, Bangalore, India
[2]Department of Computer Science and Applications, REVA University, Bangalore, India

Article Info

Article history:

Received Dec 20, 2022
Revised Oct 28, 2023
Accepted Nov 30, 2023

Keywords:

Cost
Cost effective resource
Provisioning MapReduce
Hadoop
Hadoop MapReduce
MapReduce

ABSTRACT

Distributed data processing model has been one of the primary components in the case of data-intensive applications; furthermore, due to advancements in technologies, there has been a huge volume of data generation of diverse nature. Hadoop map reduce framework is responsible for adopting the ease of deployment mechanism in an open-source framework. The existing Hadoop MapReduce framework possesses high makespan time and high Input/Output overhead and it mainly affects the cost of a model. Thus, this research work presents an optimized cost aware resource provisioning MapReduce model also known as the cost-effective resource provisioning MapReduce (CRP-MR) model. CRP-MR model introduces the two integrated approaches to minimize the cost; at first, this model presents the optimal resource optimization and optimal Input/Output optimization cleansing in the Hadoop MapReduce (HMR) scheduler. CRP-MR is evaluated considering the bioinformatics dataset and CRP-MR performs better than the existing model.

Corresponding Author:

Archana Bhaskar
Department of Computer Science and Engineering, Assistant Professor, REVA University
Bangalore, India
Email: bhaskararchana3@gmail.com

1. INTRODUCTION

Big Data is generated by application industries, requiring a scalable storage and computation strategy. It's essential to make efficient use of limited resources - money, power, space, and people - to resolve an application of interest [1] for tackling large data and computational problems. In turn, this demands a grasp of the nature of the data and the analytic techniques and their application. Cloud computing and heterogeneous computational environments are relatively new ideas that solve a number of the aforementioned problems about the scalability and efficiency of computing for data-intensive and scientific applications.

High-performance computing (HPC) is a subfield of computer science that focuses on the development of high-performance machines and the software that runs on them. The creation of parallel processing techniques [2] or software, known as parallel computing, is a key aspect of this field. We get slowdown in uniprocessor performance because of diminishing returns in exploiting instruction level parallelism. The increase in servers and server performance, results in increase of scientific and data-intensive applications. The realization that increasing performance on the desktop is less important and a better understanding of how to use multiprocessors effectively, especially in scalar applications. Through the advancement of virtualization technology, HPC computing has become more accessible and economical. Virtualization software enables computers to function like a genuine physical computer, but with the ability to specify parameters such as the number of processors, memory and disc capacity, and operating system, with flexibility [3]. The use of heterogeneous multiple core (one-CPU) computers is a system that complements cloud-based computing. These computers have integrated specialized accelerators that boost peak arithmetic throughput by 10- to 100-fold and can turn individual computers, the workhorses of both desktop and cluster computers, into mini-supercomputers. The bulk of the Hadoop MapReduce paradigm's steps which are Setup, Map, shuffle, sort, and reduce are represented in Figure 1. A cluster of computer nodes and a master node make up the Hadoop architecture. Additional distribution of map and reduce jobs among Hadoop processes. The setup step involves conceptually partitioning the input data of a job to be processed into homogeneous volumes known as chunks for the map worker nodes. The input data is commonly housed on the Hadoop distributed file system (HDFS). Each MapReduce job is broken down into a sequence of map worker-performed tasks in Hadoop. Key/value pairs are entered into the map phase (k1, v1), and an output of intermediate key/value pairs (k2, v2) is produced. The map phase, which gathers all intermediate key/value pairs, ends with the start of the Shuffle phase. The intermediate key/value pair is sorted in the map stage. For ease of assessment, the sort and shuffle stages are evaluated simultaneously. The intermediate data is sorted in the reduction step using a user-defined function. The result of the reduction stage is kept in HDFS. Figure 1 shows the typical Hadoop MapReduce model.

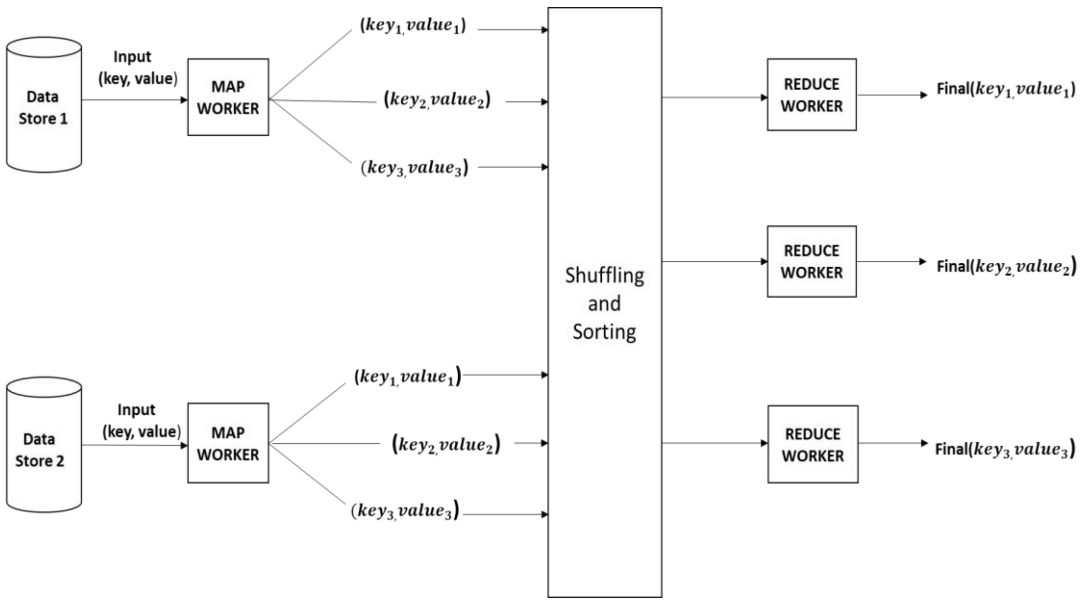

Figure 1. Hadoop MapReduce model

In recent years, multiple academics have proposed different efficient Hadoop models [4]–[9] to improve the performance of Hadoop applications [4]–[9]. Using the Starfish model, which collects an active Hadoop task profile at an adequate resolution, it is possible to offer complete data for job prediction and optimization. Elasticiser is offered for virtual machine (VM)-based resource allocation, which is regarded as superior to Starfish. However, establishing a thorough profile of a current Hadoop job might result in significant overhead and, thus, an over-optimistic time estimate. The shuffle stage is overlapping and non-overlapping phases are employed in [6]–[8], and a traditional linear regression approach is applied for task prediction. For a variety of time-sensitive tasks, this tool also helps with resource estimation. Cloud resource provisioning (CRESP) [9] successfully predicts task completion, which facilitates resource distribution under MapReduce slots. However, the impact of fewer jobs is disregarded in CRESP application models. A similar number of jobs were lost in both [10], [11].

The typical My structured query language (MySQL) technique proved unable to manage the immense volume of data. Google established the MapReduce framework, a distributed computer processing and programming architecture built in the Java programming language, to manage huge data. In the MapReduce algorithm, the two most critical jobs are Map and Reduce. Map converts a fixed quantity putting data into a new set of data in which single components are broken into tuples (key/value pairs). The output of the map job then becomes the input of the reduce task, which concatenates the data tuples into a small collection of tuples. The input data is managed by the map or mapper; the Hadoop file system (HDFS) stores the data as files or folders. Each line of the input files is passed on to the mapper function. The mapper does the job on the data and produces several little data slivers. Reducing the amount: The shuffle stage and the reduce stage make up the reduction stage. The task of the reducer is to alter the data generated by the mapper; the output of the mapper acts as the reduction's input. A collection of the freshly created output is stored in Hadoop distributed file system (HDFS); furthermore, the importance of HDFS suggests that there is a requirement for a Cost-effective MapReduce model. High-performance computing is becoming more and more crucial for the advancement of technology and the economy today. This type of computing also serves as a power measurement indication for a country's economy. Consequently, it is crucial and worthwhile to enhance High-performance computing effectiveness and accessibility, to make it more flexible and perform parallel computation, Hadoop Map Reduce (HMR) framework was introduced which performs the parallel execution of task assigned. Thus motivated by the Big data application of HMR, this research work proposes a CRP-MR framework for minimizing cost. A further contribution is given here.

– Cost effective resource provisioning MapReduce (CRP-MR) model is introduced in the Hadoop framework, which aims to minimize the cost through memory and Input/Output optimization.
– CRP-MR integrates the two mechanisms i.e. memory resource optimization and I/o optimization at the Hadoop scheduler level; also data cleansing is carried out which is ignored by several research works.
– CRP-MR is evaluated considering the Bioinformatics dataset of various lengths considering a read operation, write operation, and computing operation. Moreover, considering each service with a different sequence length CRP-MR observes marginal improvisation over the existing model.

This research work is organized as follows: the first section starts with the problem of data handling in the rising data era; further background of HMR is discussed with its typical operation and limitation. Thereafter, the first section ends with motivation and contribution of research work. The second section presents the literature work of various HMR framework.

2. RELATED WORK

A self-tuning model known as APlug [10], for parallel computing framework for substantial applications of various independent tasks. Another feature of APlug states that the ideal combination of smaller tasks and the amount of CPU resource used in terms of meeting client expectations for quality of service (QoS) requirements. APlug is also responsible for collecting a set of computations Makespan to build the model, then used by a distinct system that figures the performance parameter requirement. The mpiBLAST [11] is an open-source parallelization tool of basic local alignment search tool (BLAST), which achieves rapid speed by a section of the BLAST database for each hub via a computational model by seeking a type of segment in the database. Segmentation of the database leads each hub to seek a smaller segment portion of the database, to wipe out the circle of I/O, which leads to effortlessly enhancing the execution of the BLAST model.

A Deoxyribonucleic acid (DNA) structure [12] consists of a large number of genomic sequences assembled; however, these genomic sequences are a crucial part of the computational science environment. Consequently, the software-based genomic sequence takes many hours to complete. However, the hardware-based model accelerates the alignment procedure by using field programmable gate array (FPGA) Eosinophilic granulomatosis with polyangiitis (). The model associated with this is designed to minimize the genomic sequence assembly for makespan execution. The Hadoop MapReduce framework [13] has certain limitations. The memory configured before the allocation of Hadoop tasks leads to problems associated with buffer concurrency by the tasks associated with it and heavy disk read seeks. The makespan time enhances once there occurs any issues associated with resulting in high input/output. The performance is affected when the capability of the multi-core environment does not take into account the parameters such as memory requirements scheduled on tasks associated with Hadoop cloud environments [4].

Once the map tasks are completed, the reduced tasks are initiated in a Hadoop MapReduce model. However [5] states that the Hadoop model utilizes homogenously distributed data for homogenous map execution which is false. The performance is affected when a huge number of resources are utilized which results in a huge burden for various organizations and users in the cloud environment (storage, computation, and communication) [7]. The size of the HPC algorithm [14] increases the difficulties associated consisting of programmability, heterogeneity, energy proficiency, and fault tolerance that have emerged out of the distributed computing mechanism. To accommodate each one of them without compromising on the execution, customary techniques utilize the resources, programming models, and job schedules needed to be examined once more. A developing field in big data [15] that results in building short-term and long-term genomic sequences for the issue associated with cutting-edge genomic sequences concerning scientific research. However, the current philosophies developed by various research scientists for information sizes and straightforward entry for speeding up the current methodology to meet the prerequisites for huge information scales and their complex nature with it. In the Hadoop framework, jobs are executed based on the number of available CPU cores and the amount of memory allocated for the preset configuration. This leads to a memory bottleneck because of buffer concurrency and heavy disk seek, which causes i/o wait for occupancy and further increases the makespan time. The Hadoop memory management issues employ a global memory management strategy. A model for resource allocation based on prioritizing and revocation using a rule-based heuristic method.

In [16] Management of jobs, creation of jobs, and execution of jobs incur a computational cost, reducing the performance of a just-in-time compiler for brief tasks in the Java virtual machine (JVM). A direct consequence of information placement (region) in a computer cluster, task and job scheduling, and resource allocation in HMR. In the HMR [17], [18] parallel computing environment, the distribution of assets and resources remains a challenge. They proposed H2Hadoop (H2H), an enhanced version of the HMR architecture that reduces the cost of calculations associated with Big Data inspection. In addition, their solution solves the resource allocation issues in HMR. H2H [19], [20] provides a superior technique for "content information," such as detecting genomic sequence and motif relationships between reference genomes. In addition, H2H provides an efficient approach for cloud computing (CC) platforms. The H2H framework affects name node’s (NN's) ability to assign tasks to Task Trackers (i.e., Data Nodes) within the H2H computing cluster. By exploiting the attributes of the NN, H2H can logically guide and assign jobs to the DNs that hold the required information without communicating the task activity to the full computing cluster. In contrast to HMR, H2H reduces the number of reading functions and CPU time. Hadoop processes the Map and Reduce stages [21], [22] sequentially, its scalability is inefficient due to its cluster-based computing mechanism, it does not enable the processing of stream data, and it lacks variable pricing. To circumvent sequential execution, they devised a cloud-based parallel MapReduce paradigm in which Amazon EC2 instances (virtual machine (worker)) execute tasks. For parallel processing of stream and batch data, they devised a pipelining technique that offers adjustable pricing by utilizing Amazon Cloud spot instances [23]. Compared to the Hadoop MapReduce paradigm, the parallel processing of the HMR approach increases throughput and processing speed by 30% for larger datasets. In the field of bioinformatics, handling an ever-increasing amount of biological data necessitates high-performance processing. In this field, cloud-computing technologies are rapidly employed as handy, high-performance system. On a cloud system, we develop and evaluate prototypes of high-performance bioinformatics. In this thesis, a memory-optimized Hadoop MapReduce framework is provided.

The model for Hadoop Map Reduce framework (MOHMR) framework employs thread-based execution and dynamic memory management. Such an approach aids in effective resource use (i.e......, reduction of unutilized computing node resources). In addition, this thesis considers parallel execution leveraging the existing multi-core environment on computer nodes. A makespan model to describe the memory and I/O efficient parallel HMR framework's functionality. As may be observed, several methods are offered for performing bioinformatics applications. Due to iterative computations that frequently, include distinct, successive repetitions of computing, this might incur costs and lengthen computation time. HMR system is utilized by the state-of-the-art long genomic sequence alignment model for the CC platform. Memory optimization [24] for the Hadoop MapReduce (HMR) architecture, dubbed MOHMR, enables real-time data processing and optimal system resource use. The MOHMR schedule is intended to implement global memory management, resolve garbage collection (GC) difficulties in the virtual machine (VM), and decrease Disk I/O seek. Experiments are undertaken on the Microsoft Azure HDInsight cloud platform taking into account several applications, including text mining and bioinformatics applications, to evaluate the performance of MOHMR over the previous paradigm. The result demonstrates considerable performance enhancements in terms of calculation time and memory use.

MOHMR represents an improvement in MR by introducing a thread-based execution strategy for managing and effectively using memory resources. The MOHMR architecture, [25] MOHMR operates within a virtual machine (VM), which is one of the primary architectural design changes between the MOHMR and HMR, although the remainder of the upper layer architecture of HMR is kept. Compared to the state-of-the-art Hadoop-based parallel computing model, the proposed MOHMR framework model decreases execution time and makes better use of memory resources when performing steam and non-steam applications.

3. PROPOSED METHODOLOGY

MapReduce is a paradigm in programming, which enables huge scalability across several servers in the Hadoop cluster, it performs two distinctive tasks and Hadoop performs the same thus known as the Hadoop MapReduce framework, also popularly known as HMR. In HMR, various tasks are distinctly executed on various nodes; the memory scheduler is responsible for the tasks to be carried out for the allocation and deallocation of resources. To obtain high performance, a large-scale data center has to deal with various issues discussed earlier; these issues directly affect the model cost. However, previous research indicated memory resource utilization and Input/output utilization; this research develops a CRP-MR framework, which aims at reducing the cost. CRP-MR introduces optimal various workers will tend to have different information related to memory resource capacity. Memory resource is utilized here efficiently to reduce the makespan.

3.1. The makespan model

This work is carried out on a cloud-computing platform, it consists of a master computing node based on the map and reduce computing node. The master node is initialized as the u for virtual computing workers. Each computing worker consists of x computing cores. Here let γ_h determines the processing time for initialization purposes. Then γ_l determines the mean Makespan time for s map computing workers. Let us assume that γ_h depict the mean computing time necessary for x computing workers to perform shuffling sorting as well as reduce the computation power. Let γ_r depict the mean computing time for x computing workers. The total computing time of the makespan model is denoted,

$$\mathrm{C}= \gamma_l + \gamma_h + \gamma_r \tag{1}$$

3.2. Map-Reduce makespan model optimization

Here the major work is focused on the optimization of the tasks associated with MapReduce. The optimization involved here deals with a set of low-level optimization mechanisms that enhance the speed of I/o, the indexes utilized to fingerprint for quick comparisons of the key, and block size tuning it. The MapReduce performance is enhanced via efficient I/O disk utilization. By choosing the specific number of Map Reducing tasks that are involved by partitioning intermediate data dynamically along the runtime. The methods are needless to reduce the cost involved in I/O in MapReduce by using MapReduce indexing structures. A scheduling technique proposed here is implemented along a prototype that adapts to a scenario that manages the workload performance for hardware awareness to optimize a variety of metrics, which involves response time, stretch, and makespan. The Hadoop mechanism is responsible for optimization, which is a self-tuning mechanism to adjust the Hadoop configuration used automatically for a MapReduce job for using Hadoop clusters maximized. Another cluster optimization approach is responsible for optimization and MapReduce task-level parameter optimization on the cloud level. To focus on the desired time for the allocation of Map-Reduce jobs to avoid redundant work thereby reducing or saving the processing time.

There exists another optimization technique for the pipeline. MapReduce pipeline is used to maximize the cluster utilization, by incorporating the MapReduce job clustering for slot configuration and task submission. These studies involve analysis of various techniques incorporated by the MapReduce frameworks, to enhance further improvisation.

The proposed Hadoop optimization methodology improves the system's performance. There are several forms of dynamic memory allocation buffers because memory is utilized at various phases of the MapReduce operation. The following equation is used to determine the storage capacity of different buffers,

$$\alpha_\theta = \mathrm{C}\uparrow - \beta_{\rho_\mathrm{C}} - \mathrm{A}_{\upsilon_\mathrm{C}} \tag{2}$$

here α_θ determines the cache list, $\mathrm{C}\uparrow$ is the maximum memory size determined used for data, β_{ρ_τ} is the average sum of the list, and $\mathrm{A}_{\upsilon_\mathrm{C}}$ is the total size of the Data pair list. $\mathrm{A}_{\upsilon_\mathrm{C}}$ is the total size of the memory utilized by the I/O scheduler for the I/O buffer. The size is given by the Map-tasks as shown in the (3),

$$\eta_\theta = \min\,(\gamma_{\rho_\mathrm{C}} + P_{\rho_\mathrm{C}}, \alpha_\theta \tag{3}$$

here γ_{ρ_C} is the sort capacity (buffer) size, and P_{ρ_C} is the size of the transmission ability. the proposed algorithm here suggests that the scheduler avoids frequent recycling of memory for transmission purposes and then buffer collection for controlling memory and accommodating the memory map tasks. When one Map job is initialized or completed, this task will shift away.

3.3. Memory resource optimization model

The Memory Resource optimization model is designed to improve the model efficiency for the Hadoop cache scheduler; the proposed model adopts the scheduler that avoids memory cycling. Map-Reduce task is initialized for completion, this is an indexed or un-indexed partition uniformly among the running reduce-jobs. During the entire execution process, the memory utilization will vary and the resource optimization dynamically. The optimization is based on the size of the previous update. No bottleneck for the memory is efficiently utilized, the optimized heap size is less than the updation, and the resource is an inadequate and specific area of the capacity/buffer to be processed into the disk conveniently.

3.4. Cleansing in the I/o optimization for the scheduler

A performance-enhancing I/O improvement for the Map-Reduce framework that minimizes the disc seeks to reach a concurrent computation. The Hadoop-Map Reduce paradigm is optimized to execute the Merge

Sort process. Parallel I/O incurs significant disc seek cost. The jobs are conducted independently and must interact with one another, reducing I/O performance. To improve I/O speed, sequential overlays for CPU execution and disc I/O are implemented. The proposed I/O scheduler is comprised of two components, ReadWorker (RW) and Clean Worker (CW), which independently account for reading and write operations. Both RW and CW necessitate the multi-buffering request capability. Each tool capacity includes a selection that RW and CW employ to reorganize the read/write operations. I/O operation in this suggested paradigm comprises both active and passive I/O. Active I/O receives input from the Hadoop distributed file system and writes the output to the Hadoop file system. In the flow chart given in Figure 2, a selection is established for each task-based framework by the scheduler to accommodate the request based on their selectivity based on class. High-priority requests are satisfied first as each request per block instance; however, requests with the same similarity are satisfied in the round-robin manner. Here an interleaved I/O performs a read request similar to that of the Clean Worker.

The Merge Sort operation mainly focuses on the data stored in the memory, however, the Hadoop Map-Reduce model is also known as the external sort, since the sorting algorithm functions on the data embedded in the disks. The I/O and the CPU-bound operations are interleaved for implementations. Once the buffer is stored with lots of data, the CPU is responsible for blocking and waiting. For multi-capacity that is responsible for non-blocking. Figure 2 shows the flowchart depicting the I/O optimization of the scheduler.

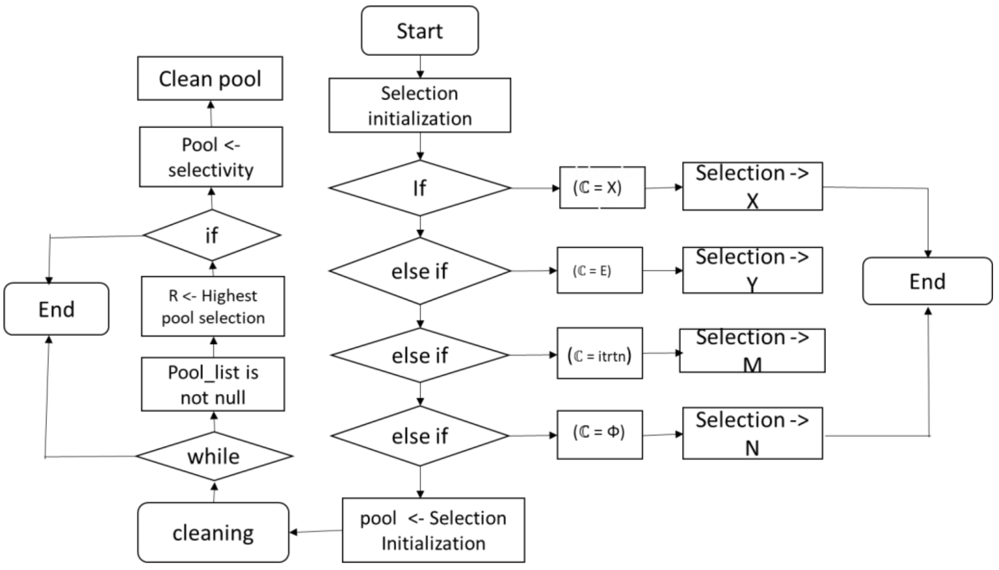

Figure 2. Flowchart depicting the I/O optimization of the scheduler

4. RESULTS AND ANALYSIS

To investigate scalability, both the proposed and current model system frameworks are installed on the Azure HDInsight cloud platform with many computational nodes in mind. Identical virtual computing employees are evaluated for both the proposed paradigm and the current approach. Consideration is given to a Microsoft Azure A3 virtual computing worker. Each A3 virtual computing node consists of 4 processor cores, 7GB RAM, and 120GB of local storage. For both the proposed and current Hadoop models, a cluster of four worker nodes from an A3 virtual computing instance is evaluated. Windows Server 2012 R2 is the operating system for the master nodes. The hourly cost of computation for each virtual computer node is \$0.296.

$$Blk\_size = \frac{D\_size}{Blk\_size} \tag{4}$$

where Blk_size is the default data block size and D size is the volume of data to be uploaded to HDFS (by default it is 64MB). Each block is given a compression ratio before being put into the HDFS to make it smaller. When a MapReduce job reads data from HDFS, the cost to read a single data block is $HDFSRCt$. The cost of reading all data from HDFS is known as $IOWCt,$ and it is calculated as (5). The price of writing a single data block to HDFS is known as $HDFSWCt$, The cost of writing any data, whether the output of a MapReduce process or raw data, is known as $IOWCt$ and is calculated as (6).

$$IORCt = NOBlk * HDFSRCt \quad (5)$$

$$IOWCt = NOBlk * HDFSWCt \quad (6)$$

The aforementioned computations clearly show that the total costs of reading and writing from HDFS depend on the number of blocks or the data size. Therefore, by reducing the amount of data, we may reduce the costs of these processes, which will enhance Hadoop's performance. Additionally, the quantity of blocks required for each Hadoop operation is inversely correlated with the cost. For instance, $CPUCompCost_R$ is used to calculate the CPU cost of reading.

$$CPUCompCost_R = NOBlk * InUncCPUCompCost + InpMapPair * MapCPUCompCost \quad (7)$$

$MapCPUCompCost$ is the cost of mapping one pair, and $InpMapPair$ is the total number of pairs involved in the mapping process.

4.1. Computation cost evaluation of the proposed model over the existing model in a public cloud environment

In Table 1, for computing read, write, and computation cost performance evaluation of the proposed model over a current model for conducting sequence alignment, this study considers the genomic sequence derived from the baker yeast database (Saccharomyces genome database (SGD), 2015) [26], [27] and shown in Table 1. Furthermore, Experiments are undertaken for each of the scenarios listed in Table 1, and the read, write, and computational costs to align genomic sequence are depicted in Figures 3, 4, and 5, respectively. As query sequence length rises, the cost of writing, reading, and calculating (analyzing) increases as well. Overall, the suggested model reduced reading, writing, and calculation costs by an average of 36.85%, 35.48%, and 32.97% in comparison to the old system in the Azure HDInsight public cloud environment. The total result achieved on the cloud platform indicates that the proposed model can efficiently align genomic sequences with low read, write, and compute costs, hence decreasing the overall cost of service delivery and optimizing resource utilization. Figure 4 presents the comparison of genomic sequence alignment reading cost on a public platform; to evaluate the CRP-MR model two distinctive data of 0.5 million and 1 million are considered. For 0.5 million data, the existing model costs 0.03971 whereas the proposed model costs 0.0271. Similarly, for 1 million data, the existing model cost is 0.1219 whereas the proposed model cost is 0.07071.

Table 1. Gene sequence considered for experiment analysis

Reference genome sequence	Sequence length	Query genome sequence	Sequence length
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

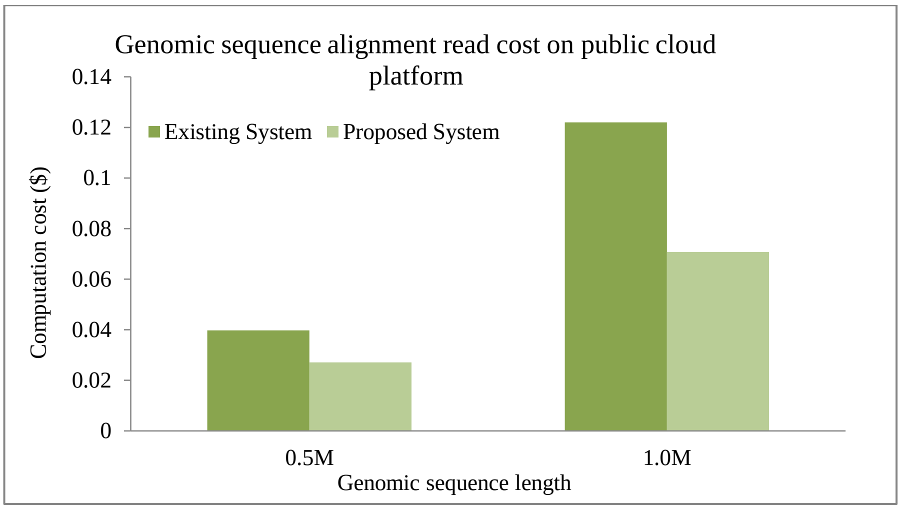

Figure 3. Genomic sequence alignment read cost considering multiple computing workers on a public cloud platform

Figure 4 presents the writing cost comparison of genomic sequence alignment considering the 0.5m and 1.0 million data. Moreover, In the vase of 0.5M data, the existing model requires the cost of 0.01456 and the proposed model requires 0.010175. Similarly, in the case of 1 million, the existing model requires a cost of 0.02240 and the proposed model requires a cost of 0.01325. Figure 5 presents the computation cost comparison of genomic sequence alignment considering the Genomic sequence length of 0.5 million and 1.0 million. In the case of 0.5M length, the computation cost of the existing model is 0.0189 and the computation cost of the proposed model is 0.0135. Similarly, for 1.0 million computations, the cost of the existing model is 0.028 and the computation cost of CRP-MR is 0.0167.

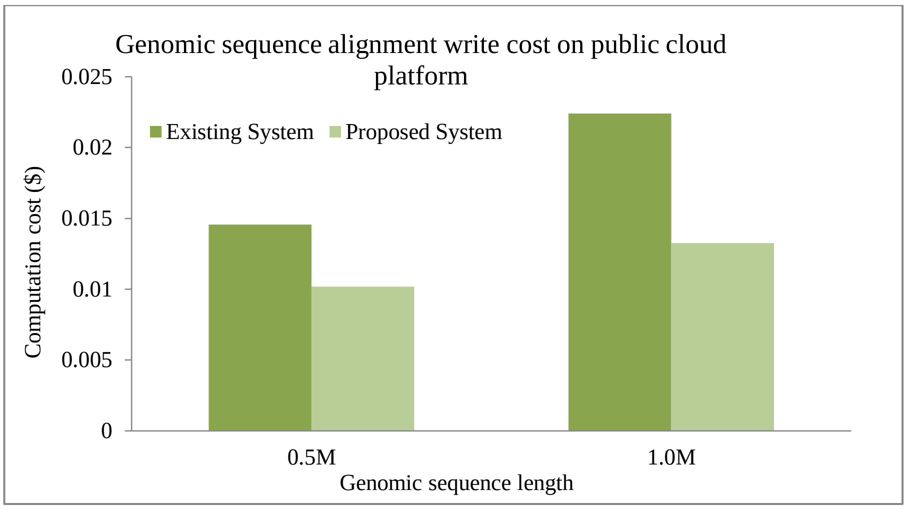

Figure 4 Genomic sequence alignment write cost considering multiple computing workers on a public cloud platform

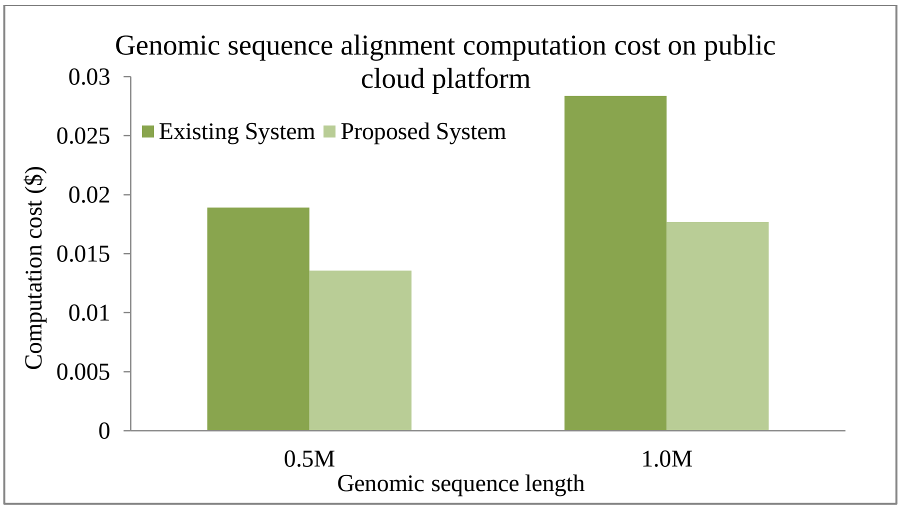

Figure 5 Genomic sequence alignment computation cost considering multiple computing workers on a public cloud platform

4.2. Comparative analysis

Table 2 shows the comparative analysis in terms of the percentage of improvisation over the existing model considering three operations Reading, writing, and Computation for sequence lengths of 0.5 and 10. Considering the reading operation, CRP-MR achieves improvisation of 31.67% and 42.00% respectively for both sequence lengths. Furthermore, considering the writing cost, CRP-MR achieves improvisation of 30.34% and 40.90% respectively for both sequence lengths. At last, considering computation operation, CRP-MR achieves improvisation of 40.90% and 37.80% for sequence length.

Table 2. Improvisation of CRP-MR over the existing model

Parameters	Existing model	Proposed model
Genomic Sequence Length	0.5	1.0
Reading cost	31.67%	42.00%
Writing cost	30.34%	40.90%
Computation cost	40.90%	37.80%

5. CONCLUSION

The HMR system has a number of flaws, including buffer concurrency between operations and intensive disc read searches. As a result, incurs I/O overhead and increases execution time. Further, the HMR scheduler does not consider performance aspects such as memory requirement and multi-core environment for linear scaling affecting performance and considers homogenous map execution time assuming homogenously dispersed data that makes it costly. This research introduces the CRP-MR model for a cost-effective model with two integrated approaches i.e. memory resource utilization and minimizing input/output overhead.CRP-MR is evaluated considering genomic sequence derived from the baker yeast database (Saccharomyces genome database (SGD) CRP-MR achieves improvisation of 31.67% and 42.00% respectively for both sequence lengths. Furthermore, considering the writing cost, CRP-MR achieves improvisation of 30.34% and 40.90% respectively for both sequence lengths. At last, considering computation operation, CRP-MR achieves improvisation of 40.90% and 37.80% for sequence length. Future work includes designing a forecasting model for accurately predicting incoming tasks for utilizing resources more efficiently and meeting SLA (Service Level Agreement) for supporting diverse application deadline requirements.

REFERENCE

[1] M. Khan, Y. Jin, M. Li, Y. Xiang, and C. Jiang, "Hadoop performance modeling for job estimation and resource provisioning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 441–454, 2016, doi: 10.1109/TPDS.2015.2405552.

[2] Z. Zhang, L. Cherkasova, and B. T. Loo, "Optimizing cost and performance trade-offs for MapReduce job processing in the cloud," *IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World*, 2014, doi: 10.1109/NOMS.2014.6838231.

[3] X. Cui, X. Lin, C. Hu, R. Zhang, and C. Wang, "Modeling the performance of MapReduce under resource contentions and task failures," in *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom*, 2013, vol. 1, pp. 158–163, doi: 10.1109/CloudCom.2013.28.

[4] J. Zhu, J. Li, E. Hardesty, H. Jiang, and K. C. Li, "GPU-in-Hadoop: Enabling MapReduce across distributed heterogeneous platforms," in *2014 IEEE/ACIS 13th International Conference on Computer and Information Science, ICIS 2014 - Proceedings*, Jun. 2014, pp. 321–326, doi: 10.1109/ICIS.2014.6912154.

[5] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, "Improving MapReduce performance in heterogeneous environments," *Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2008*, pp. 29–42, 2019.

[6] D. Dahiphale *et al.*, "An advanced MapReduce: Cloud MapReduce, enhancements and applications," *IEEE Transactions on Network and Service Management*, vol. 11, no. 1, pp. 101–115, Mar. 2014, doi: 10.1109/TNSM.2014.031714.130407.

[7] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: The montage example," Nov. 2008, doi: 10.1109/SC.2008.5217932.

[8] N. Chohan, C. Castillo, M. Spreitzer, M. Steinder, A. Tantawi, and C. Krintz, "See spot run: Using spot instances for MapReduce workflows," *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2010*, p. 7, 2010.

[9] X. Lin, Z. Meng, C. Xu, and M. Wang, "A practical performance model for hadoop mapreduce," in *Proceedings - 2012 IEEE International Conference on Cluster Computing Workshops, Cluster Workshops 2012*, 2012, pp. 231–239, doi: 10.1109/ClusterW.2012.24.

[10] M. Sahli, E. Mansour, T. Alturkestani, and P. Kalnis, "Automatic tuning of bag-of-tasks applications," in *Proceedings - International Conference on Data Engineering*, Apr. 2015, vol. 2015-May, pp. 843–854, doi: 10.1109/ICDE.2015.7113338.

[11] S. H. Sreedhara, V. Kumar, and S. Salma, "Efficient big data clustering using Adhoc Fuzzy C means and auto-encoder CNN," in *Lecture Notes in Networks and Systems*, vol. 563, Springer Nature Singapore, 2023, pp. 353–368.

[12] A. Rosenthal, P. Mork, M. H. Li, J. Stanford, D. Koester, and P. Reynolds, "Cloud computing: A new business paradigm for biomedical information sharing," *Journal of Biomedical Informatics*, vol. 43, no. 2, pp. 342–353, Apr. 2010, doi: 10.1016/j.jbi.2009.08.014.

[13] X. Shi *et al.*, "Mammoth: Gearing Hadoop towards memory-intensive MapReduce applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 8, pp. 2300–2315, Aug. 2015, doi: 10.1109/TPDS.2014.2345068.

[14] M. Shang, Y. Zhou, and H. Fujita, "Energy-saving operation synergy for multiple metro-trains using Map-Reduce parallel optimization," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 1319–1332, Feb. 2022, doi: 10.1109/TVT.2021.3133858.

[15] C. Wang, X. Li, P. Chen, A. Wang, X. Zhou, and H. Yu, "Heterogeneous cloud framework for big data genome sequencing," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 12, no. 1, pp. 166–178, Jan. 2015, doi: 10.1109/TCBB.2014.2351800.

[16] M. Khan, Z. Huang, M. Li, G. A. Taylor, and M. Khan, "Optimizing hadoop parameter settings with gene expression programming guided PSO," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 3, Feb. 2017, doi: 10.1002/cpe.3786.

[17] Y. Wu, W. Guo, J. Ren, X. Zhao, and W. Zheng, "NO2: Speeding up parallel processing of massive compute-intensive tasks," *IEEE Transactions on Computers*, vol. 63, no. 10, pp. 2487–2499, Oct. 2014, doi: 10.1109/TC.2013.132.

[18] M. Sun, X. Zhou, F. Yang, K. Lu, and D. Dai, "Bwasw-cloud: Efficient sequence alignment algorithm for two big data with

MapReduce," in *5th International Conference on the Applications of Digital Information and Web Technologies, ICADIWT 2014*, Feb. 2014, pp. 213–218, doi: 10.1109/ICADIWT.2014.6814662.
[19] D. Borthakur, "The hadoop distributed file system: Architecture and design," *Hadoop Project Website*, pp. 1–14, 2007, [Online]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://web.mit.edu/mriap/hadoop/hadoop-0.13.1/docs/hdfs_design.pdf
[20] F. M. Awaysheh, M. N. Aladwan, M. Alazab, S. Alawadi, J. C. Cabaleiro, and T. F. Pena, "Security by design for big data frameworks over cloud computing," *IEEE Transactions on Engineering Management*, vol. 69, no. 6, pp. 3676–3693, Dec. 2022, doi: 10.1109/TEM.2020.3045661.
[21] X. Li, F. Chen, R. Ruiz, and J. Zhu, "MapReduce task scheduling in heterogeneous geo-distributed data centers," *IEEE Transactions on Services Computing*, vol. 15, no. 6, pp. 3317–3329, Nov. 2022, doi: 10.1109/TSC.2021.3092563.
[22] J. Wang, M. Qiu, B. Guo, and Z. Zong, "Phase-reconfigurable shuffle optimization for Hadoop MapReduce," *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 418–431, 2020, doi: 10.1109/TCC.2015.2459707.
[23] M. C. Schatz, B. Langmead, and S. L. Salzberg, "Cloud computing and the DNA data race," *Nature Biotechnology*, vol. 28, no. 7, pp. 691–693, Jul. 2010, doi: 10.1038/nbt0710-691.
[24] L. Person, "World hadoop market-opportunities and forecasts, 2014-2021", Press Release. 4 November 2016.
[25] S. Canzar and S. L. Salzberg, "Short read mapping: An algorithmic tour," *Proceedings of the IEEE*, vol. 105, no. 3, pp. 436–458, Mar. 2017, doi: 10.1109/JPROC.2015.2455551.
[26] H. Alshammari, J. Lee, and H. Bajwa, "H2hadoop: Improving hadoop performance using the metadata of related jobs," *IEEE Transactions on Cloud Computing*, vol. 6, no. 4, pp. 1031–1040, 2018, doi: 10.1109/TCC.2016.2535261.
[27] S. G. Database, "Saccharomyces Genome Database, http://yeastgenome.org," pp. 3–4, 2015, [Online]. Available: http://www.yeastgenome.org/.

BIOGRAPHIES OF AUTHORS

Ms. Archana Bhaskar assistant professor is currently pursuing Ph.D. in computer science and engineering from REVA University. She is currently working in REVA university. She has published around 5 papers in both national and international journals. Her areas of interests include cloud, big data, hadoop and wireless sensor networks. She can be contacted on this email: bhaskararchana3@gmail.com.

Dr. Rajeev Ranjan after completing a Ph.D. in wireless sensor network at Indian Institute of Information Technology, Allahabad (IIIT -A), he has joined as Associate Professor in the School of Computer Science and Applications at REVA University. He has published 41 Journal papers and 8 conference papers of repute. He is an author of 3 book chapters and filled 9 patents. He is the reviewer of many international journals like Journal of Ambient Intelligence and Humanized Computing (Springer), IEEE sensor Journal, IET journal of Sonar & Navigation etc. His area of work includes-: Wireless sensor networks-coverage & connectivity, Sensor deployment and localization, Wireless sensor statistical routing etc. He can be contacted on this email: rajeev.ranjan@reva.edu.in.