IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

A COMPREHENSIVE ANALYSIS OF VERSATILE DYNAMIC DATA PLACEMENT POLICY IN HETEROGENEOUS CLUSTER ENVIRONMENTS

Yogesh Kumar Namdeo[1]

Dr. Aarti Kumar2
(Professor)
Dept. of Computer Science & Engineering
Rabindranath Tagore University, Bhopal

Abstract:

Hadoop is a widely used system for managing enormous volumes of data. Hadoop cluster performance is significantly impacted by the effectiveness of job rescheduling algorithms. However, Hadoop's current scheduling methods are unable to properly account for variations in node efficiency across heterogeneous clusters. Issues like uneven work distribution and underutilization of resources may develop as a result. We suggest a novel strategy termed the *Versatile Dynamic Data Placement Algorithm (VDDP)*, created especially for heterogeneous Hadoop clusters, to tackle these difficulties.

Our method makes use of the cluster heartbeat mechanism to gather crucial data, such as node CPUs and memories. This data permits a thorough evaluation of each node's actual load capacity. The VDDP algorithm improves the overall efficiency and resource utilisation of Hadoop clusters in the presence of heterogeneity by dynamically adjusting and balancing the distribution of information across the cluster based on the condition of each node.

Our approach meets the needs of heterogeneous clusters where nodes have different processing capacities, in contrast to the current Hadoop implementation, which assumes homogenous computing nodes. Our proposed system effectively handles diverse data types in Hadoop clusters, thereby improving efficiency and performance in large-scale data collection and analysis. Furthermore, we introduce a method to calculate the remaining rate of hardware resources on each node, prioritizing tasks on nodes with higher resource availability based on the task type. Additionally, we incorporate data compression techniques for Input-Output intensive tasks to minimize disk function and expedite task execution.

Our VDDP approach outperforms existing scheduling algorithms, according to experimental data, especially in heterogeneous Hadoop clusters, where it considerably improves scheduling efficiency and task execution speed. The VDDP algorithm showcases task-specific enhancements in execution time, delivering superior performance for various task types.

Keywords: Job scheduling, heterogeneous cluster, Hadoop, and big data.

1. INTRODUCTION

Many industries and scientific communities generate vast volumes of data, commonly known as "big data" [1]. However, when employing large clusters composed of heterogeneous nodes, the performance of MapReduce implementations can deteriorate. Furthermore, the absence of data locality consideration in geographically dispersed environments poses another drawback, as Hadoop assumes all data to be stored locally.

This article concentrates on enhancing two fundamental aspects of the traditional MapReduce architecture: data homogeneity and data locality. Neglecting these factors can significantly impact MapReduce performance. In our approach, particularly for critical and resource-intensive applications, achieving a balanced distribution of tasks based on the allocated resources for each node can considerably improve the performance of the Hadoop platform, especially during MapReduce operations.

A heterogeneous cluster consists of servers with different CPU, memory, and disc resource performances on each node [2]. Hadoop frequently requires the installation of more nodes in order to improve the cluster's storage and computational capacity. The setup of a new node, however, is different from the configuration of the original nodes. A good job scheduling method is therefore essential to improving the overall performance of a heterogeneous Hadoop cluster [3].

The Versatile Dynamic Data Placement algorithm (VDDP) is the answer we suggest. We identify the best scheduling strategy to increase resource utilisation and task execution speed by assessing the real

computing performance of each node in the heterogeneous cluster environment. The following are the primary contributions of our strategy:

- Using the Versatile Dynamic Data Placement algorithm (VDDP) to find the best mapping between tasks and resources while taking into account the real load capacity of each heterogeneous node and task execution time as the goal function.

- Increasing the rationale of the scheduling algorithm by giving priority to assigning CPU-intensive jobs to nodes with high CPU idle rates (by calculating Cluster Typical CPU Utilisation) and taking into account node performance and task monitoring.

- Taking into account the demands of various workloads on node performance, we introduce Expected Data Mean calculations to determine the additional workload necessary to align the present node's data more consistently with its data state, thereby improving resource utilization.

2. RELATED WORK AND MOTIVATION

Numerous research studies have explored dynamic data deployment techniques. For instance, Pius et al. [10] proposed a method that tracks and estimates Typical CPU and memory usage of DataNodes and the cluster. During a file read operation, the solution evaluates cluster and DataNode utilization, employing a threshold value. It selects DataNodes whose utilization differs from the cluster by a value lower than the threshold for block placement. OverutilizedDataNodes are discarded, and the process continues with other nodes. Ye et al. [4] introduced a block Versatile Dynamic Data Placement technique that dynamically selects DataNodes for block placement to enhance Hadoop's efficiency.

To improve Hadoop's efficiency, Ye et al. focused on load balancing by selecting optimal DataNodes based on remaining space utilization. Lin et al. [5] proposed a strategy where the NameNode selects DataNodes based on their load status to achieve a balanced distribution. They introduced a new node called BalanceNode, which facilitates load balancing by transferring load from heavily-loaded DataNodes to lightly-loaded ones. Lee et al. C.-W. Lee, K.-Y.Hsieh, S.-Y.Hsieh, and H.-C. Hsiao proposed a dynamic block placement policy that considers the computing capacity of DataNodes to enhance MapReduce performance [6]. They created a RatioTable in the NameNode based on computing capacity to determine the ratio of data blocks to be placed on each DataNode.

In the VDDP approach, the ApplicationMaster component assigns tasks to DataNodes. In a heterogeneous environment, when faster nodes complete processing their local data, the ApplicationMaster assigns the

remaining tasks to these nodes (Anjos et al. [13]; Pandey and Saini [14]). However, the newly assigned tasks to faster nodes do not have ownership of the required data blocks for processing. Therefore, data blocks from slower nodes must be transferred to the faster nodes to complete the processing. The faster nodes wait for task execution until the complete data block is transferred from the slower node.

This data transmission from slower nodes to faster nodes extends the execution time of MapReduce jobs in the Map phase of a heterogeneous environment. Excessive data transmission negatively impacts the overall performance of HadoopMapReduce. Reducing inter-node data transmission in the Map phase effectively decreases the idle waiting time of faster nodes in a heterogeneous environment.

3. SCENARIOS

During the monitoring phase, the NameNode actively monitors the Data State of each node and compares it with the values in the Modalities of Data Distribution Table. This process continues until new workloads are calculated for nodes that align better with their Data State. Following that, these calculated workloads are spread in succeeding rounds and stored in the Cluster-History table for each node.

Based on the procedure for each job sent to the cluster, the following part generally describes three scenarios:

a) ***New Job Type and No Information:*** In this case, a new job type is sent to the cluster without any information regarding the job type and its input Data Mean.
b) ***Existing Job Type with New Data Mean:*** In this case, the job type is not brand-new, but the input Data Mean is.
c) ***Existing Job Type and Data Mean:*** In this case, the cluster's job type and its input Data Mean are both old.

3.1 Scenario (Statements 1 through 16) of the VDDP:

- The NameNode, which is in charge of controlling the cluster, performs a number of checks and procedures whenever The Hadoop Distributed File System (HDFS) is used to store the data once a new job is submitted to the cluster.

- **Checking the RatioTable**: The NameNode first looks through the RatioTable to see if it has a record of the type of previously completed work. The job is a new kind if there isn't a record of the job type in the RatioTable.

a) ***Handling a new job type:*** If the job type is new and lacks information in the NameNode, several steps are taken:
   - **Updating the RatioTable:** The NameNode adds a record of the new job type to the RatioTable for future reference.
   - **Updating the Cluster-History Table:** The NameNode creates records in the Cluster-History Table for the new job type, including information about the job type and its data mean. This table assists in tracking and managing the distribution of input data blocks.
   - **Creating the Modalities of Data Distribution Table:** The NameNode generates the Modalities of Data Distribution Table specifically for the new job type. This table contains information on how the input data blocks should be distributed across the cluster.

b) **Distributing input data blocks:** Utilizing the information from the RatioTable and the Modalities of Data Distribution Table, the NameNode distributes the input data blocks across the cluster, ensuring efficient resource utilization.

c) **Monitoring phase:** Once the input data blocks are distributed, the monitoring phase commences. The NameNode monitors the job's progress, keeps track of node utilization within the cluster, and records relevant information in the Cluster-History Table for future reference.

3.2 Scenario (Statements 18 to29) of the VDDP -

- **Checking the RatioTable**: The NameNode looks through the RatioTable to see if the job type that was submitted has a record. If a record is discovered, it means that this kind of job has already been done.
- **Existing job type:** The presence of a record in the RatioTable signifies that the job type has been performed before. This implies that corresponding information is available in the Cluster-History Table, including the Modalities of Data Distribution Table.
- **Confirming input volume:** To confirm the input volume of the submitted work, the NameNode looks at the Cluster-History Table.
- **New job type:** The Cluster-History Table does not have a defined distribution pattern for the input data if the input volume of the submitted task cannot be identified in the RatioTable.
- **Data distribution among nodes:** In these situations, where there is no predetermined distribution pattern, the freshly written data is distributed across the cluster's nodes according to their computational power, as shown by the RatioTable.
- **Monitoring and workload calculation:** Once the input data blocks are assigned, the NameNode continuously monitors and compares the Data State of each node with the values in the Modalities of

Data Distribution Table. Utilizing load formulas present in the Modalities of Data Distribution Table, the NameNode calculates a workload that aligns better with each node's load situation.

- **Updating Cluster-History Table:** The calculated workload is then recorded in the Cluster-History Table for each node. This information serves as a basis for workload distribution to the nodes in future job submissions with the same job type and input data.

In summary, if a job type has been previously executed, the NameNode utilizes existing information in the Cluster-History Table for distributing the input data blocks. In the case of a new job type, data allocation is based on node capacity, and the NameNode dynamically adjusts workload distribution considering the load situation of each node. This information is subsequently recorded in the Cluster-History Table for future job submissions.

4. EXPERIMENTAL PROBABILITY

TABLE 1 INDIVIDUAL NODE SPECIFICATION				
Machine	**Operating System**	**Memory (GB)**	**Number of Cores**	**Disk(GB)**
Master	Windows 11	16	4	930
Slave1	Ubuntu Linux20.4	4	1	19.8
Slave2	Ubuntu Linux20.4	6	2	19.8
Slave3	Ubuntu Linux20.4	8	4	583.4

Table 1: INDIVIDUAL NODE SPECIFICATION

The *Versatile Dynamic Data Placement Algorithm (VDDP)* and the Hadoop framework were tested against the Suggested algorithm in a TestBed. Using the WordCount application in a diverse Hadoop cluster, the proposed algorithm's performance was assessed. A MapReduce-based task called WordCount is used to count the words in an input file.The following devices made up the experimental set-up shown in Table 1,

Each Slave machine's specifications were as follows:

Slave1	Intel Core i5-4210U	1.70GHz	**1 CPU**	**4 GB**	**19 GB**
Slave2	Intel Core i5-4210U	1.70GHz	**2 CPUs**	**6 GB**	**19 GB**
Slave3	Intel Core i7-4790	3.60GHz	**4 CPUs**	**8 GB**	**538 GB**

Virtual Box 4.1.14 was used to set up the Slave1 and Slave2 computing nodes in a heterogeneous environment. The configuration of the nodes included different CPU and memory capacity. Utilising the WordCount application in a heterogeneous Hadoop cluster, The performance was assessed and contrasted using the TestBed of the proposed method against the DDP algorithm and the Hadoop framework. The WordCount programme is made to count the number of times a word appears in an input file.

Table 2 presents the ratios associated with the WordCount job as recorded in the RatioTable. To illustrate the distribution of input data blocks, Table 3 is created based on the ratios in the RatioTable, assuming the input data block size is 500 MB. According to Table 3, the allocation of data blocks is as follows:

TABLE 2 INDIVIDUAL NODE PARAMETER			
Job Type	**Slave1**	**Slave1 2**	**Slave1 3**
Word Count	**1**	**2**	**4**

Table 2: INDIVIDUAL NODE PARAMETER

Job Type	**Word Count**		
Input Data size	x	**Parametric Each Node workload**	**500**
Slave 1	$x = \frac{1}{1+2+4}$	y	100
Slave 2	$x = \frac{2}{1+2+4}$	$2y$	150
Slave 3	$x = \frac{4}{1+2+4}$	$4y$	250

Table 3: RATIO TABLE EXAMPLE

The amount of cores that are available on each node in the proposed algorithm determines how many jobs can run on that node at any one time. In light of each node's specifications:Because *Slave1 only has one core*, it only does one task every round.Because *Slave2 has two cores*, it can do two jobs concurrently throughout each round.Because *Slave3 has four cores*, it can do four jobs concurrently throughout each round.

Experiment 1 :

This Experiment focuses on comparing the DDP algorithm and the VDDP algorithm in the presence of an overloaded state within the cluster. The cluster's three Slaves' typical execution times are measured, considering different workloads during the normal load state. For the purposes of this experiment, the workloads assigned to Slave 1, Slave 2, and Slave 3 are 60 MB, 110 MB, and 210 MB, respectively.

1. Normal Load State :

WORKLOAD DISTRIBUTION AFTER REDISTRIBUTION

Slave	Workload (MB)	Execution Time (s)
Slave1	60MB	35s
Slave2	110 MB	85s
Slave3	210 MB	55s

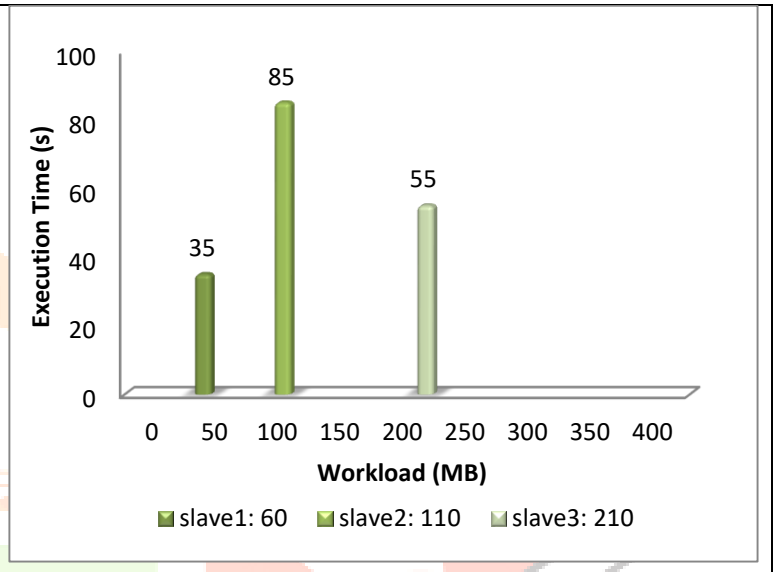


Chart: 1 : Execution time of each Slave in Normal load state

2. Overload State :

A. Round 1: Execution time for DDP in an overloaded state for each Slave

- Slave2 is overburdened; it takes it 240 seconds longer than usual to complete a task.

- Data blocks are distributed using the computing capacity ratios in the DDP and VDDP algorithms.

- Workload distribution and execution times are not specified.

Execution time of each Slave for DDP in overload state (Round (1))

Slave	Workload (MB)	Execution Time (s)
Slave1	60MB	35s
Slave2	110MB	240s
Slave3	210MB	55s

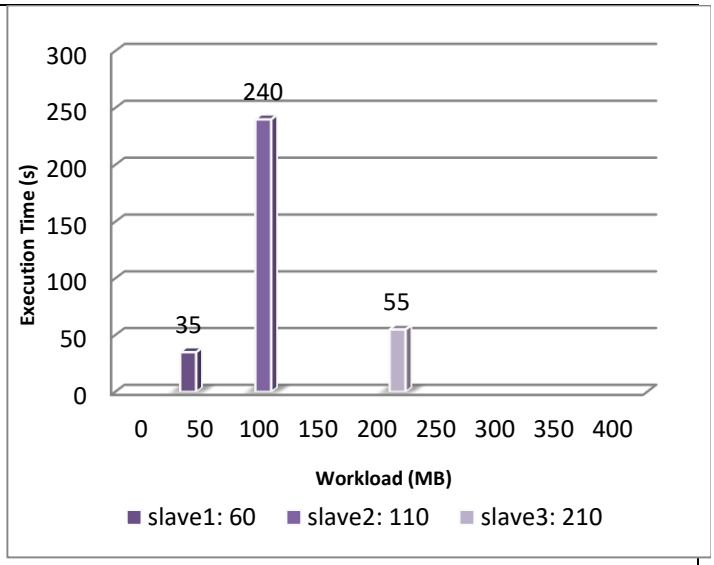


Chart: 2 : Execution time for each Slave for DDP when it is overloaded (Round 1)

Round 1: Execution time of each Slavefor VDDPin overload state

Execution time for each Slave for VDDP when it is overloaded (Round (1))

Slave	Workload (MB)	Execution Time (s)
Slave1	60MB	35s
Slave2	110MB	240s
Slave3	210MB	55s

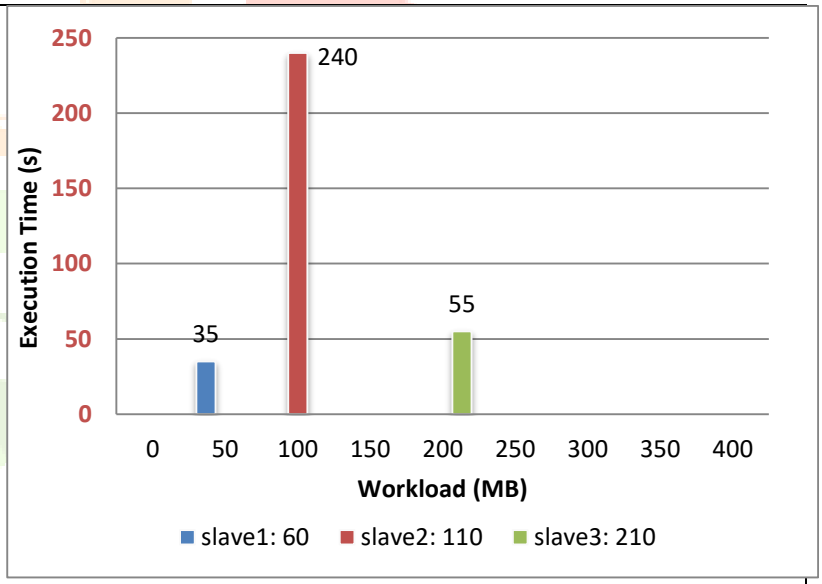


Chart: 3Execution time for each Slave for DDP when it is overloaded (Round 1)

B. Round 2: Each Slave's execution time for DDP in an overload state:

- Where the algorithm distributes data blocks in accordance with computational power.
- The Cluster-History table's values are used by the DDP algorithm to determine how data blocks are distributed.

Round 2: Each Slave's execution time for VDDP in the overload condition

- The NameNode assigns data blocks based on these values, which are derived using the Modalities of Data Distribution Table formulas, in Round 2. Due to the fact that Slave2 is overcrowded, 15% of Slave2's task must be added to Slave3's burden, which is underloaded.
- After shifting 15% of Slave2's workload to Slave3, determine the new workloads for Slave2 and Slave3.

Execution time for each Slave for DDP when it is overloaded (Round (2))

Slave	Workload (MB)	Execution Time (s)
Slave1	60MB	37s
Slave2	110MB	244s
Slave3	210MB	55s

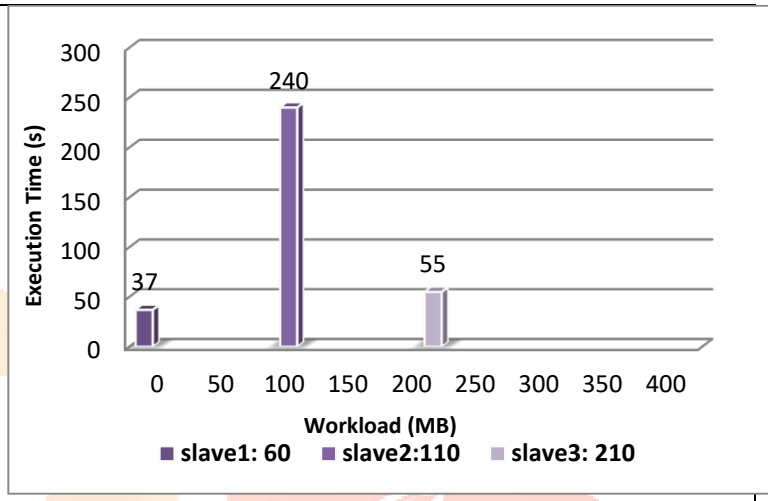
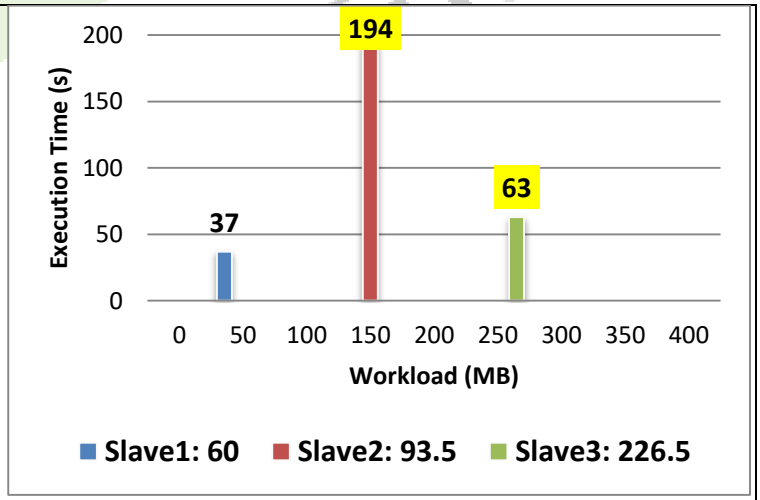


Chart: 4: Execution time for each Slave for DDP when it is overloaded (Round (2))

Execution time for each Slave for VDDP when it is overloaded (Round (2))

Slave	Workload (MB)	Execution Time (s)
Slave1	60MB	37s
Slave2	93.5MB	194s



Slave3	226.5MB	63s

Chart: 5: Execution time for each Slave for VDDP when it is overloaded (Round (2))

C. Round 3: Execution time for each Slave for DDP when it is overloaded (Round (3))

Execution time for each Slave for DDP when it is overloaded (Round (3))

Slave	Workload (MB)	Execution Time (s)
Slave1	60MB	35s
Slave2	110MB	240s
Slave3	210MB	55s

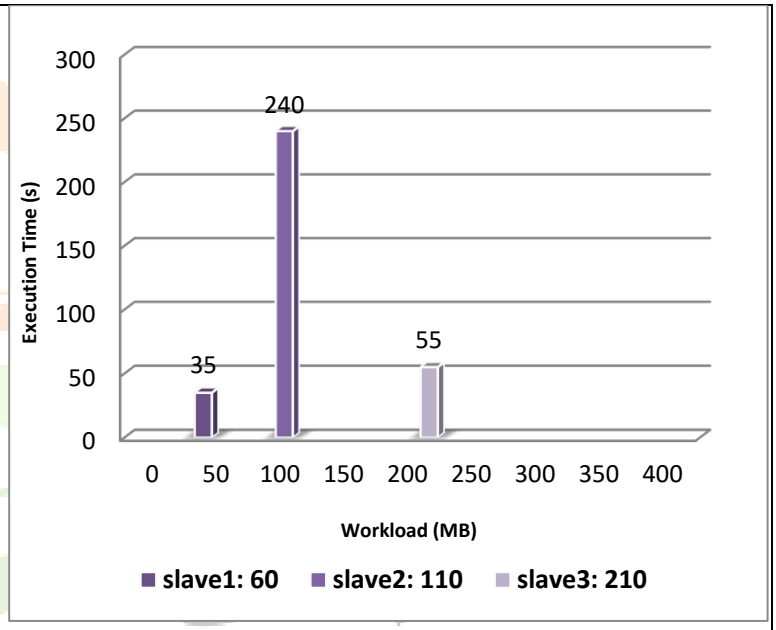


Chart: 6: Execution time for each Slave for DDP when it is overloaded (Round (3))

Round 3: Execution time for each Slave for VDDP when it is overloaded

To calculate the new workloads for Slave2 and Slave3 after redistributing 15% of Slave2's workload to Slave3, we need to determine the amount of workload to be redistributed and then update the values accordingly.

Execution time for each Slave for VDDP when it is overloaded (Round (3))

Slave	Workload (MB)	Execution Time (s)
Slave1	60MB	37s
Slave2	79.475 MB	144s
Slave3	240.525 MB	71s

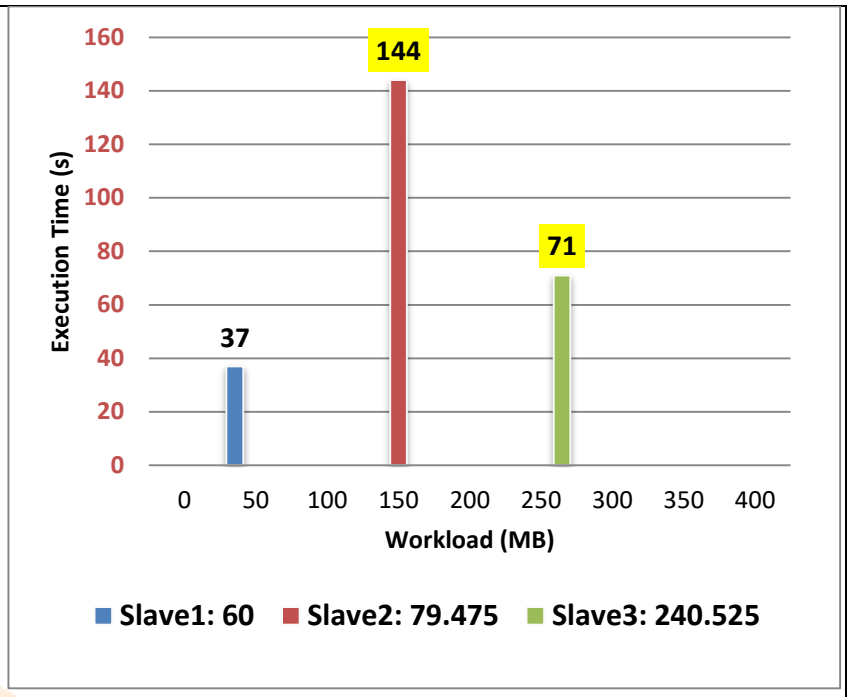


Chart: 7: Execution time for each Slave for VDDP when it is overloaded (Round (3))

D. Round 4: Execution time for each Slave for DDP when it is overloaded (Round (4))

Execution time for each Slave for DDP when it is overloaded (Round (4))

Slave	Workload (MB)	Execution Time (s)
Slave1	60MB	35s
Slave2	110MB	240s
Slave3	210MB	55s

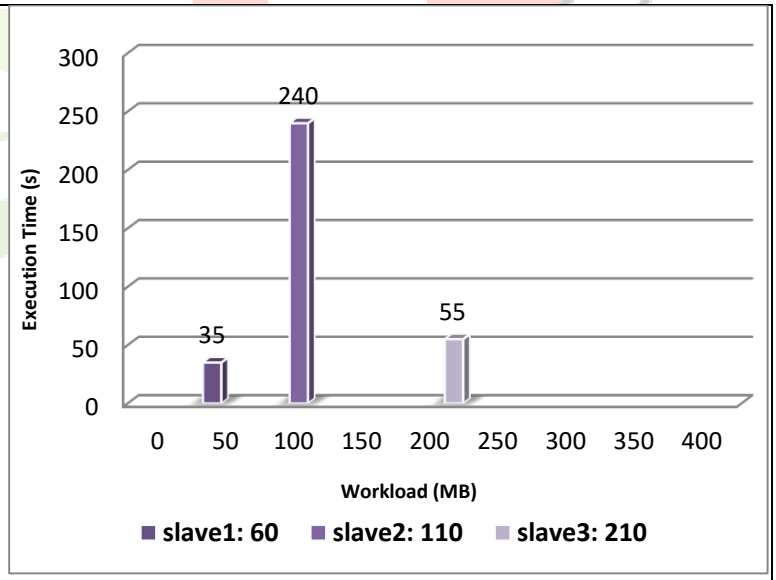


Chart: 8:Execution time for each Slave for DDP when it is overloaded (Round (4))

Round 4: Execution time for each Slave for VDDP when it is overloaded

To calculate the new workloads for slaves 2 and 3 after redistributing 15% of slave 2's workload to slave 3.

Execution time for each Slave for VDDP when it is overloaded (Round (4))

Slave	Workload (MB)	Execution Time (s)
Slave1	60MB	37s
Slave2	67.55375 MB	94s
Slave3	252.44625 MB	79s

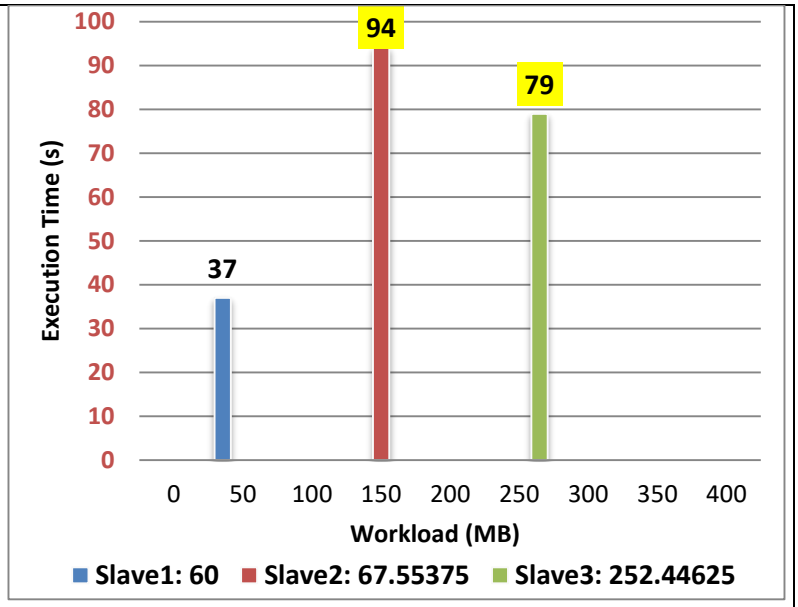


Chart: 9:Execution time for each Slave for VDDP when it is overloaded (Round (4))

After four cycles, the cluster with 380 MB of input data volume is balanced, with an average cluster execution time of **70 seconds**; while, the average cluster execution time for the DDP algorithm is **110 seconds**.

$$\mathbf{MExTavg(i)} = \frac{\sum_{i=1}^{N} \mathbf{JobExeTime(i)}}{\boldsymbol{NumberofJob}}$$

Here, MExTavg(i) represents the average execution time of jobs, JobExeTime(i) represents the execution time of each individual job, N represents the total number of jobs, and Number of Job represents the count of jobs.Let's consider an example scenario with 3 jobs and their corresponding execution times:

DDP	VDDP
Job 1: 35 seconds,Job 2: 240 seconds Job 3: 55 seconds, In our example, the sum of job execution times is:35 + 240 + 55 = 330 seconds Since we have 3 jobs, Number of Job = 3. Plugging these values into the formula, we get: **MExTavg(i) = 330 seconds / 3 = 110 seconds**	Job 1: 37 seconds, Job 2: 94 seconds, Job 3: 79 seconds, In our example, the sum of job execution times is:37 + 94 + 79 = 210 seconds Since we have 3 jobs, Number of Job = 3. Plugging these values into the formula, we get: **MExTavg(i) = 210 seconds / 3 = 70 seconds**

Comparison of the total cluster's Hadoop, DDP, and VDDP algorithm execution times during each round of overload

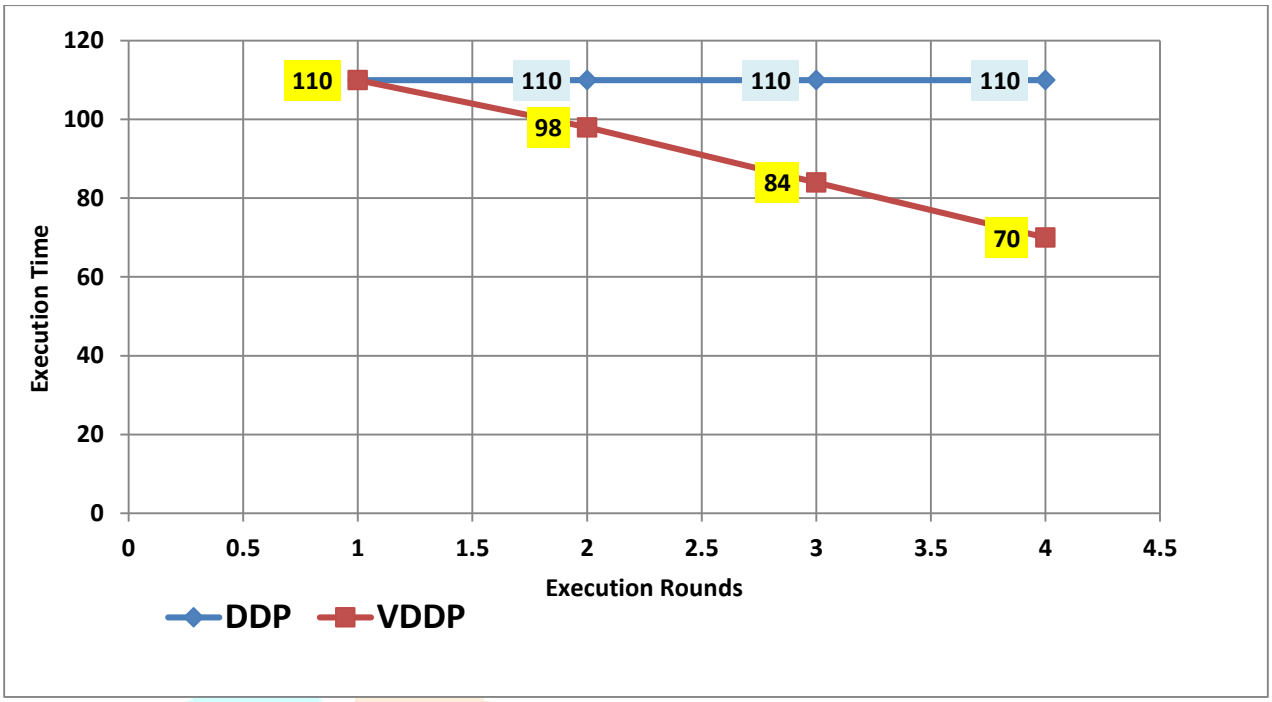


Chart: 10: Comparison of the total cluster's Hadoop, DDP, and VDDP algorithm execution times

These results demonstrate that the *VDDP* algorithm outperforms the *DDP* algorithm in terms of cluster balancing and average execution time reduction by redistributing depending on values recorded in the Cluster-History database, or in blocks of data.

5. CONCLUSION

This study introduces the VDDP (Versatile Dynamic Data Placement) algorithm as a means of boosting Hadoop's functionality in heterogeneous clusters. In order to manage load imbalances, the algorithm, which belongs to the class of resource-aware scheduling algorithms, seeks to reduce data flow between slow and fast nodes.

The VDDP technique uses a data placement scheme to accomplish this goal by distributing data fragments among numerous heterogeneous nodes according to their computational capabilities and workloads. The approach enhances data locality and lowers extra overhead by determining each node's appropriate workload using load parameters and allocating data blocks accordingly.

The VDDP technique has been shown to be effective in enhancing Hadoop performance in heterogeneous clusters. By improving data placement, minimizing data transportation, and minimizing load imbalances, it offers considerable advantages to both DataNodes and NameNodes. These developments help to improve data processing's overall efficacy and efficiency in heterogeneous computing systems.

REFERENCES

[1] M. Hilbert, "Big data for development: A review of promises and challenges," *Development Policy Review*, vol. 34, no. 1, pp. 135–174, 2016.

[2] Pandey V, Saini P. How heterogeneity affects the design of Hadoopmapreduce schedulers: a state-of-the-art survey and challenges. *Big Data*. 2018;6(2):72-95.

[3] Gaur M, Minocha B, Muttoo SK. A study of factors affecting MapReduce scheduling. In: Aggarwal V, Bhatnagar V, Mishra D, eds. *Big Data Analytics*. Singapore: Springer; 2018:275-281.

[4] X. Ye, M. Huang, D. Zhu, and P. Xu, "A novel blocks placement strategy for Hadoop," 2012 IEEE/ACIS 11th International Conference on Computer and Information Science, pp. 3–7, May 2012.

[5] C.-Y. Lin and Y.-C.Lin," A load-balancing algorithm for hadoop distributed file system," 2015 18th International Conference on Network-Based Information Systems, pp. 173–179, Sep. 2015.

[6] C.-W. Lee, K.-Y.Hsieh, S.-Y.Hsieh, and H.-C. Hsiao," A dynamic data placement strategy for hadoop in heterogeneous environments," Big Data Research, vol. 1, pp. 14–22, Aug. 2014.

[7] Xie J, Yin S, Ruan X, Ding Z, Tian Y, Majors J, Manzanares A, Qin X (2010) Improving mapreduce performance through data placement in heterogeneous hadoop clusters. In: 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW). IEEE. pp 1–9. https ://doi.org/10.1109/IPDPS W.2010.54708 80)

[8] Wh LIN, Zm LEI, Jun L, Jie Y, Fang L, Gang H, Qin W (2013) Map Reduce optimization algorithm based on machine learning in heterogeneous cloud environment. The Journal of China Universities of Posts and Telecommunications 20(6):77–121. https ://doi.org/10.1016/S1005 -8885(13)60112 -0

[9] Kwon Y, Balazinska M, Howe B, Rolia J (2012) Skewtune: mitigating skew in mapreduce applications. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. ACM. pp 25–36. https://doi.org/10.1145/22138 36.22138 40 Lee CW, Hsieh KY, Hsieh SY, Hsiao HC (2014) A dynamic data placement strategy for hadoop in heterogeneous environments. Big Data Research 1:14–22. https://doi.org/10.1016/j.bdr.2014.07.002

[10] Krish K, Anwar A, Butt AR (2014) hats: A heterogeneity-aware tiered storage for hadoop. In: 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE. pp 502–511. https ://doi.org/10.1109/CCGri d.2014.51

[11] Kumaresan V, Baskaran R, Dhavachelvan P (2018) Aegeus++: an energy-aware online partition skew mitigation algorithm for mapreduce in cloud. Cluster Computing 21(2):1243–1260. https://doi.org/10.1007/s1058 6-017-1044-8

[12] Xiong R, Luo J, Dong F (2014) Sldp: A novel data placement strategy for large-scale heterogeneous hadoop cluster. In: 2014 Second International Conference on Advanced Cloud and Big Data. IEEE. pp 9–17. https://doi.org/10.1109/CBD.2014.57

[13] Anjos JC, Carrera I, Kolberg W, Tibola AL, Arantes LB, Geyer CR (2015) Mra++: Scheduling and data placement on mapreduce for heterogeneous environments. Future Generation Computer Systems 42:22–35. https ://doi.org/10.1016/j.futur e.2014.09.001

[14] Pandey V, Saini P (2018) How heterogeneity affects the design of hadoopmapreduce schedulers: A state-of-the-art survey and challenges. Big Data 6(2):72–95. https ://doi.org/10.1089/ big.2018.0013

[15] Eltabakh MY, Tian Y, Özcan F, Gemulla R, Krettek A, McPherson J (2011) Cohadoop: flexible data placement and its exploitation in hadoop. Proceedings of the VLDB Endowment 4(9):575–585. https ://doi.org/10.14778 /20029 38.20029 43

[16] Huang S, Huang J, Dai J, Xie T, Huang B (2010) The hibench benchmark suite: Characterization of the mapreduce-based data analysis. In: 2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010). IEEE. pp 41–51. https ://doi. org/10.1109/ICDEW .2010.54527 47

[17] Lee CW, Hsieh KY, Hsieh SY, Hsiao HC (2014) A dynamic data placement strategy for hadoop in heterogeneous environments. Big Data Research 1:14–22. https ://doi.org/10.1016/j.bdr.2014.07.002