Received: January 2, 2023. Revised: February 8, 2024.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Integrated Data, Task and Resource Management to Speed Up Processing Small Files in Hadoop Cluster

Shwetha K S[1]* **Chandramouli H[1]**

[1]*Department of Computer Science and Engineering, East Point College of Engineering and Technology, Bengaluru, Karnataka, India*
* Corresponding author's Email: shwethaise.nhce@gmail.com

Abstract: Huge demand on business intelligence applications over large volume of enterprise data has resulted in rapid adoption of High performance data analytics. Hadoop based high performance computing environment are optimized for large files. Data centric execution with localization of computing proximal to data provided higher performance for large files. But for small files, the performance reduced and overhead increased due to the way Hadoop handles files. Resource allocation and scheduling policies of Hadoop has to be improvised to handle small files. This work proposes an integrated data, task, and resource management technique to speed up processing of small files in Hadoop based high performance computing environment. As part of data management, the data placement is made dynamic to access frequency and inherent data semantics. As part of task management, tasks were grouped based on fine grained data semantics correlation they process. As part of resource management, data blocks or storage nodes are replicated to improve the access latency with minimal cost overheads. With this integrated management, the proposed solution is able to increase the speedup of handling small files by at least 8.7% compared to most recent works on individual management of data, task and resources.

Keywords: Hadoop, Small file, Speed up, Integrated management.

1. Introduction

Business environment has become more competitive, and data centric orientation is the way forward for enterprises to thrive in this environment. Realizing it many businesses are using cutting edge data analytics over the large volumes of data to mine valuable knowledge for fine tuning their business decisions. High performance computing (HPC) along with machine learning and artificial intelligence are the enabler technologies for this data centric business transformation. Hadoop is the open source big data processing component of HPC. It is designed for optimal execution of data intensive applications. Hadoop's distributed file system (HDFS) deal with the partioning, distribution and access of massive file shares with sequential data access patterns, running on clusters of nodes. HDFS organizes the files as blocks and assign a map task to each block. By collocating map tasks near to data blocks, Hadoop reduces the latency in data access and speeds up the execution of tasks. With emergence of IoT big data integration, the files need to be mined were of small size typically less than 2KB. Hadoop performance dropped due to working with these small files as the overhead in meta-data management and map spawning overhead were higher [1-3]. Over the years, many resource allocation, data management and scheduling algorithms have been proposed to solve this problem and improve the Hadoop performance for small files. The solution lacked adaptation to data semantics, tasks nature and service level agreements of the customers. Also works addressing small file problem to Hadoop clusters in the background of high performance data analytics environment are very meager.

This work addresses the problem of working with small files in Hadoop based HPC environment and proposes an integrated strategy involving joint management of data, task, and resources. The

objective is to speed the execution of data analytics on small files. The approach involves categorization of data based on data semantics and placement of them in storage nodes based on the categorization. The tasks are grouped based on data access nature and localized proximal to data blocks in such way to optimize the data access latency and resource consumption. The data blocks are replicated to maximize the speed with minimal resource consumption costs. Following are the contributions of this work.

(i) A novel on fly data semantics extraction and data semantics correlation technique to aid in data co-location decision. The data co-location based on the proposed data semantics reduced the data access latency and helped to achieve speed up of processing small files.
(ii) A task grouping algorithm based on task nature and fine grained data access semantics is proposed to group relevant tasks and collocate their execution. This is done to minimize the data transfer overheads and achieve speedup.
(iii) A data replication technique to speed up the execution of tasks with multi objective optimization of task speedup and resource costs in Hadoop clusters. The optimization problem is solved using elephant herding optimization (EHO) algorithm.

The rest of the paper is organized as follows. Section II provides the survey of existing algorithms in category of data, task and resource management and details the open issues. Section III details the proposed integrated data, task, and resource management technique. Section IV provides the results of the proposed technique and its comparison to existing works. Section V provides the concluding remarks and presents the scope for future work.

2. Survey

The survey is in three categories of data, task and resource management techniques.

A. Data management

Chen et al [4] proposed a data locality management technique to speed up tasks. The Datanode's storage capability is set based on its execution capability. Blocks are given importance to be stored in the fastest Data node. By this way the performance of the map tasks was improved. But the scheme did not consider application and data characteristics in data block allocation and scheduling of tasks. Xie et al [5] proposed a data management technique for optimal placement of data in heterogeneous Hadoop clusters. A new metric called computing ratio was proposed to profile the nodes and data is placed in proportion to computing ratio. Authors also redistributed the data based on node utilization. The data placement strategy did not consider the data semantics and tasks correlation. Also the redistribution scheme proposed in this work has higher communication overhead. Marquez et al [6] proposed a genetic programming based data placement technique. The data placement was optimized based on reduction of data write time using genetic algorithm. Resource allocation was optimized based on maximum utilization of physical machine. But the approach did not consider data semantics and task relation for data placement and allocation. Jeyaraj et al [7] proposed a fine grained data locality aware scheduling algorithm for reducing the makespan of map reduce jobs in Hadoop clusters. This scheduler attempts to minimize the amount of intermediate data in the shuffle phase by using multi level per node combiner. This scheme reduced the intermediate data by executing group of map jobs belonging to same application together. But the scheme did not consider data proximity for grouping map jobs. Amin et al [8] proposed a data provisioning strategy to localize Hadoop clusters proximal to data in high performance cloud computing environment. The strategy minimizes the data access delay in the environment by placing VM's in Hadoop cluster close to the storage node. By this way, the file transfer time and hence the MapReduce job completion time is reduced. The provisioning scheme does not consider collocating inter related data and tasks together to achieve maximal performance. Li et al [9] proposed two different data locality optimization for geo distributed clouds. In the first optimization scheme, tasks are assigned according to node locality, and access data of non-node-locality tasks are migrated in advance by using the idle network bandwidth. In the second optimization scheme, hot files are predicted and synchronized among the nodes. The tasks were not grouped based on similarity on data access and thus the data transfer cost is higher in this approach.

B. Task management

Hammoud et al [10] proposed a task scheduler to improve the map reduce tasks performance in Hadoop clusters. In addition to scheduling map based on data locality, reduce tasks are also scheduled based on data locality in this scheme. The early merge of map outputs is delayed and once after a sufficient time, the reduce task is scheduled on the node which is close to maximum size of map outputs. This improves the performance of reduce tasks by minimizing the data shuffling volume. But the scheme not does fully exploit the nature of data and

its semantics in its scheduling decision. Yao et al [11] improvised the Hadoop's YARN task scheduler to address the problems in fairness and efficiency while processing inter related tasks in Hadoop. The goal of the scheduler was to reduce the makespan of the batch of jobs by using the information on requested resources, resource capacities and dependency between tasks. Though authors considered maximizing the CPU and memory usage, they did not consider data nature and localization of tasks in their scheduling decision. Kousiouris et al [12] experimented with CPU percentage allocated to tasks and their impact on application performance in clusters. Authors found that collocating inter related tasks on same node improves the performance. Thus identifying the inter related based on data characteristics and scheduling them on same node increases the performance. But the authors did not propose any scheme for identifying inter related tasks. Wang et al [13] proposed a two stage scheduling algorithm to increase the throughput in Hadoop clusters. The algorithm has separate queue for each node and a common queue for all nodes. Nodes process task from its corresponding local queue when it is free, it process task from global queue using MaxWeight policy. Task processing did not consider data localization. Yang et al [14] proposed two techniques to optimize the internal overlap between map and reduce jobs to reduce the map reduce execution time in Hadoop clusters. Lazy start of reduce tasks and batch finish of map tasks are done for better alignment of map and reduce tasks. This alignment improved node throughput. But the proposed scheduling did not consider node localization in allocation of reduce tasks to node. Liu et al [15] proposed a Fair Sojourn Protocol in YARN scheduler to improve the responsiveness and ensure fairness in Hadoop clusters. It is a size based scheduler where job size is predicted and based on job size, resources are allocated to it. The efficiency of the scheduling depends on the accuracy of the job prediction. But prediction is based on moving average. But job size prediction did not consider data semantics. Zhang et al [16] proposed a task scheduling algorithm for heterogeneous Hadoop clusters. The tasks are allocated to node based on objective of minimizing the waiting time and transmission time for data. A metric based on predicted waiting time and transmission time if formulated. For batch of jobs, the metric is calculated, and the jobs are allocated to node with overall goal of minimizing the metric value. Task correlation and data correlation was not exploited to maximize the performance in this work. But the idea of evaluating the tasks for allocation using a metric is novel contribution in this work. Chen et al [17] proposed a data locality aware real time scheduling technique for Hadoop clusters. The tasks were categorized to data intensive and CPU intensive. Data intensive tasks are scheduled to data proximal nodes and CPU intensive tasks are scheduled to nodes with minimal load interference. Authors did not consider collocating interrelated tasks to same node to maximize data proximity. Wei et al [18] extended the default first in first out (FIFO) scheduler of Hadoop with data locality awareness and sharing. Tasks requesting same data are grouped as batches and processed in same node, so that data to be shared across tasks is maximized. The scheduling algorithm was designed only for homogenous clusters. The proposed tasks grouping mechanism works did not consider the granularity of data and semantics of the data for grouping the tasks.Gandomi et al [19] proposed a hybrid scheduling algorithm combining dynamic priority and localization. The algorithm aimed to increase the data locality rate and reduce completion time. With dynamic priority and proportional share assignment, tasks requesting same data are processed in batches. This reduces the makespan for related tasks. But the proposed scheme did not exploit the data collocation and grouping data based on semantics to maximize the performance gain. Choi et al [20] proposed a task scheduling algorithm to solve the performance problem due to input split consisting of multiple data blocks. The algorithm operates in two stages. In the first stage, tasks are classified to three types based on data proximity. In the second stage, classified tasks are assigned to nodes based on priority criteria. Task classification method proposed in this paper classifies task only based on data location in rack and neglects the data semantics. The performance of this method can be still improved by data grouping based on semantics. Convolbo et al [21] proposed a heuristic scheduling algorithm called GeoDis to optimize the makespan for data intensive jobs in geo distributed clouds. Authors formulated the task placement and data access as a linear programming problem and used heuristics linear problem solver to find optimal task placement schedule. But the solution is not scalable and computing complexity is higher. Xie et al [22] proposed a stochastic delay optimal algorithm called Pandas to reduce the makespan of data intensive tasks. It is task-level algorithm that specifies the priority among tasks of any data-processing phase by considering data locality considering data locality. Pandas predict the contents for data blocks and create replicas to avoid contention. By this way task processing time is reduced at cost of minor storage overhead. The contention blocks are predicted only

Table 1. Survey summary

Work	Solution for small file problem	Gap
Liu et al [18]	File content based merging	Constructing a global feature space for streaming data is difficult and thus this approach is not suitable for streaming data
Lyu et al [19]	optimized merging strategy to solve small file problem.	Only block size utilization was considered as the only criteria for merging without considering content characteristics and semantic relations
Wang et al [21]	probabilistic latent semantic analysis to determine the user access pattern and based on it small files are merged to a large file	scheme is not suitable for multi user environment as for each user, a merging order must be kept and this increases the storage overhead
He et al [22]	merging the small files based on balance of data blocks	Merging did not consider content characteristics and their semantic relation
Fu et al [23]	flat storage architecture collocating metadata and file in same object	the scheme is not suited for Hadoop as collocation causes higher access overhead for large files
Tao et al [24]	merged small files to large file and built a linear hash to small files to speed up access	File size was the only criteria considered for merging
Bok et al [25]	integrated file merging and caching to solve the small file problem	The merging was based only on size without considering the content characteristics and semantic similarity
Sharma et al [29]	Authors created Hadoop archive from small files but made access faster using two special hash functions.	Archiving was done without analysis of content characteristics and similarity
Li et al [30]	Improved genetic algorithm is used to schedule the resources to the tasks with goal of minimizing the task completion time.	Task collocation based on data sharing characteristics was not considered.

based on access frequency and performance can be still improved by using data semantic correlation to predict zero day blocks for replication. Li et al [23] proposed a performance aware scheduler (PAS) to schedule jobs in Hadoop clusters. The proposed solution automatically adjusts the scheduling policies to improve application performance and resource utilization. Multiple concurrent tasks are scheduled using different policies based on predicted job completion time. Greedy policy based adjustment of policy is done to maximize average job performance frequently. Task grouping did not consider data access similarity and data location. If these are considered, the makespan of tasks can be still reduced.

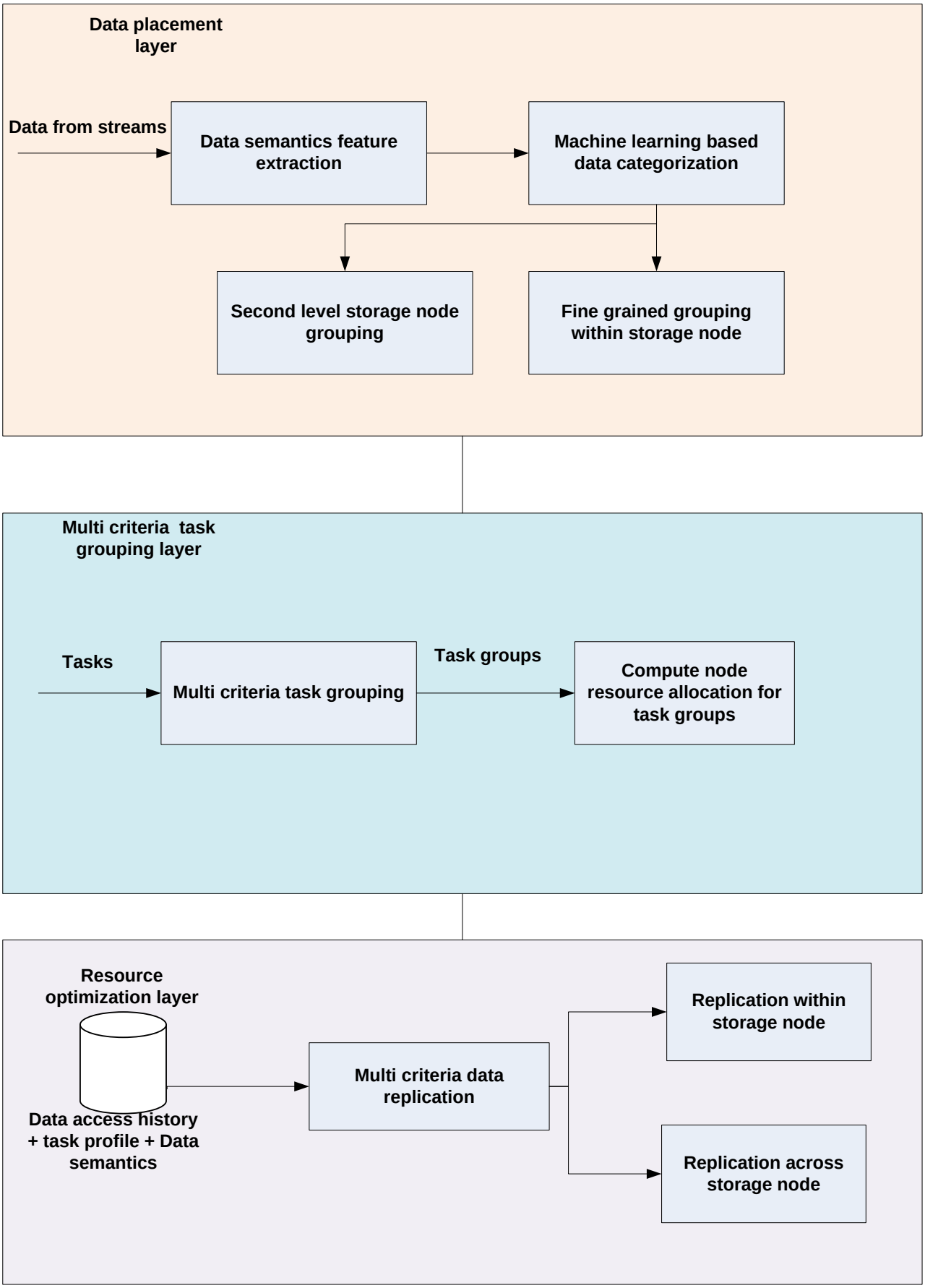

Figure. 1 Integrated management architecture

C. Resource management

Lim et al [24] modeled resource allocation of scheduling on Hadoop as a optimization problem and proposed a constraint programming based resource allocation algorithm for Hadoop map reduce jobs. Each job has a service level agreement (SLA) in terms of deadline time for execution. A batch of jobs is considered for resource allocation and they are scheduled in such way to minimize theSLA miss ratio. The proposed scheduling algorithm was data locality aware and it placed jobs proximal to data. In case of small files, the number of jobs shoots up and the computational complexity for resource allocation using constraint programming becomes NP hard in this approach. Lim et al [25] improvised the constraint programming based resource management in Hadoop clusters to achieve higher performance. The job execution times were estimated using a presheduling error handling technique. But the error estimation technique proposed in this work did not consider the localization and proximity of data. Bader et al [26] proposed a resource allocation algorithm called Tarema. The resource allocation algorithm was designed to process scientific workflows in Hadoop clusters effectively. The nodes are clustered based on profile similarity and task were grouped based on semantics. The task groups are allocated to node clusters. But the scheme did not consider the data semantics while grouping the tasks. Tao et al [27] proposed a dynamic Hadoop cluster on cloud infrastructure. The clusters were scaled based on load by varying the number of virtual machine. The VM were split to two categories of data storage and computing node. Computing nodes and storage node are placed proximal to minimize access delay. The tasks requesting same data are placed in computing node and data is placed in storage node. But the authors did not consider grouping data based on semantics and placing them on storage node. Due to this, the make span of Map reduce jobs increased in this work. Li et al [30] minimized execution time through optimal allocation of tasks to resources. But the work did not consider task collocation based on data sharing characteristics.

The summary of most important works in the survey is presented in Table 1. From the survey, it can be seen the existing works did not consider the joint influence of data, resource and task management in each other perceptive and thus performance gains were limited. Datasemantics and correlation of tasks based on semantics were not considered in scheduling andresource management decisions. To address this gap, this work integrates the data, task , and resource management with consideration for data semantics and data semantics based correlation in the task scheduling decisions.

Table 2. Notations used in equations

Notation	Detail
$x_{c_{i,j}}$	Current position of elephant in the clan
x_{center,c_i}	Center position of all elephants in the clan
$x_{worst,c_{i,j}} =$	Position of the elephant with worst fitness within the clan
RC	Overall resource cost
SL	Service level adherence
ERE	Overall distance between related task entities
f	Fitness function for task management
PC	Popularity counter
α	Popularity increment step (value is set as 1 in this work)
β	Popularity decrement step (value is set as 1 in this work)
NT	Number of tasks

3. Proposed methodology

The architecture of the integrated management technique is given in Fig. 1. It is layered architecture with management of data, task and resource in three layers of data placement; multi criteria task grouping and resource optimization. In the data placement layer, the incoming small files are analyzed and categorized based semantics. The small files are then organized to data blocks and moved to storage nodes. The metadata hash is then built to fetch the files in lowest latency and moved to Named node. Machine learning is applied for semantic analysis and categorization. In the multi criteria task group layer, the tasks are grouped based on the correlation between the semantics of data accessed by them. This is done to increase the volume of sharable information between the tasks and reduce the latency in data movement across computing nodes. The computing and storage resources are allocated based on task grouping. In the resource optimization layer, the data blocks or the storage nodes are replicated to increase the speed up with a minimal addition to resource cost. The details of each of the layers are presented in below subsections.

The notations used in equations below are summarized in Table 2.

A. Data placement layer

For categorization of incoming text, the data semantics must be learnt. This work relies on named entity recognition model to learn data semantics. This work adopts DBpedia Spotlight [28] for named entity recognition. It is a widely used open source named

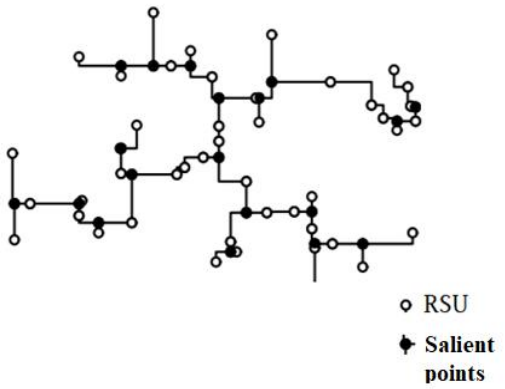

Figure. 2 Steiner minimal tree

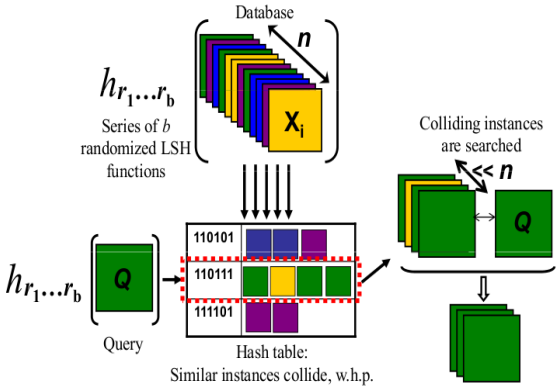

Figure. 3 LSH lookup

entity recognition system to label named references in text as entities in the DBpedia knowledge base. But the problem in use of DBpedia is that it provides vast number of named entities. Creating a data block for each named entity and keeping the small files containing the named entity in corresponding data blocks increases the storage volume and cost. To solve this, this work proposes named entity (NE) selection algorithm using the concept of Steiner minimal tree graph theory. Taking a batch of incoming text documents as input, the NE's are first extracted for each document. A graph is constructed with each NE as vertex. The weights between the NE is calculated using a combination of co-occurrence frequency and Word2vec similarity between the NE. Co-occurrence frequency is the number of documents in which the NE appear together. From the graph Steiner minimal tree is constructed as shown in Fig. 2. The Steiner minimal tree identifies the optimal vertex points called Salient points representing close proximal vertexes. Once the Salient points are identified, the set of vertices near to the Salient points are grouped as related entities (RE). Since finding the Salient point is non polynomial complexity problem, this work using graph iterated 1-steiner heuristics to find the optimal Steiner points. Once the RE is constructed, a data block is created for each RE. For each text document association score to RE is calculated based on the number of NE shared between the text document and the RE. The text is placed in data block corresponding to the RE having highest association score the text document. For each RE an index file is created and the metadata of file categorized to RE is kept in its corresponding index file. By this way, index file is generated for each RE. Locality sensitive hashing(LSH) is used for indexing the metadata. LSH is a method for approximate neighbor search in high dimensional space. It maps the high dimensional data to lower dimensional representation using random hash function such a way that points closer in higher dimensional space maps to same low dimensional space with higher probability. LSH hashes the item repeatedly several times, so that similar items are more likely hashed to same bucket than dissimilar items as shown in Fig. 3. Thus to find the items in a database, which is similar to a query, LSH maps to most relevant bucket and number of buckets are also less. Due to this lookup becomes faster in LSH compared to hashing based lookup. The idea of LSH is to construct hash functions g: $R^d \rightarrow U$ such that for any two points p, q

$$\text{if } ||p - q|| \leq r, \text{then } \Pr[g(p) = g(q)] \text{ is high}$$

$$\text{if } ||p - q|| > cr, then \Pr[g(p) = g(q)] \text{ is small}$$

This is achieved with family H of functions

$$g(p) = < h_1(p), h_2(p), \ldots . h_k(p) > \quad (1)$$

For all data point $p \epsilon P$, p is hashed to buckets $g_1(p), g_2(p), \ldots g_b(p)$

For an input query q, the points are retrieved from the buckets $g_1(q), g_2(q)$,… until all points from b buckets are retrieved. The effectiveness of LSH comes from use of multiple hash functions instead of single hash. Multiple hash reduces the number of buckets needed for mapping the items in the database The metadata mapping to a context is indexed using LSH to generate the buckets. The buckets corresponding to the context are written to a index file. When user/application queries for the small file, they can query the file by name or keywords. When the user queries by file name, parallel search is launched on each of index file using LSH. The bucket matching the filename is returned. From the bucket linear search is done with filename as key to return the metadata. When the query is done with keyword, linear search is done over the context file to find the matching context at first step and search is launched

the index file corresponding to that context using LSH. The bucket matching the filename is returned. From the bucket linear search is done with filename as key to return the metadata.

The RE can be grouped based on the cosine similarity between the NE contained in them. The related RE are placed in the same storage node. Through this approach, the data placement layer ensures higher co-occurrence between the semantic similar data in the storage nodes.

B. Task grouping layer

In most of existing works, tasks operating on same data in terms of storage node commonality were grouped. This increased the sharable intermediate data volume for inter related tasks thereby increasing speed up and system utilization. But the problem in method is that, without organizing the data based on their correlations and without operating at fine grained level of data organized, the sharable intermediate volume cannot be increased. Also the speed up cannot be increased. Another problem is grouping tasks and allocating to computing nodes without considering their service level requirements increases the SLA miss rates. This work proposes a multi criteria task grouping with consideration for both data locality and task service level requirements. The tasks are processed in batches. In a batch, the tasks requesting for content are first grouped based on RE to which the content belongs. The RE to which the content belongs can be found from the index file. The initial group of tasks found based on RE must be further optimized based on two goals of (i) minimization of overall cost for execution (ii) maximization of SLA adherence and (iii) minimization of overall distance of RE in each group. Solution to this optimization problem is found using modified elephant herding optimization (MEHO) algorithm in this work.

Elephant herding optimization (EHO) is a population based algorithm inspired by the herding behavior of elephants. In EHO, the entire population of the elephants is split to certain number of clans. An adult female elephant called matriarch leads the clan. It is found as the best positioned agent in each iteration. After each iteration, worst agent (male elephant) leaves the clan to live alone. The core of EHO is in two operations of: clan operation and separation operation.

The position of the elephant or search agent (j) in clan (ci) is updated in relation to clan leader or matriarch as

$$x_{new,c_{i,j}} = x_{c_{i,j}} + \alpha\left(x_{best,c_i} - x_{c_{i,j}}\right) \times r_1 \qquad (2)$$

Where $x_{c_{i,j}}$ is the current position, $x_{new,c_{i,j}}$ is new position x_{best,c_i} is the position of best agent in clan ci and r_1 is the random number in range of 0 to 1. The position of clan leader is taken as center of positions of all agents in the clan. It is calculated as

$$x_{center,c_i} = \frac{1}{n}\sum_{j=1}^{n} x_{c_{i,,j}} \qquad (3)$$

Where n is the number of agents in the clan.

Male elephants leave the clan to avoid in-breeding within the clan. This leaving behavior is represented as separator operator given as

$$x_{worst,c_{i,j}} = x_{min} + (x_{max} - x_{min} + 1) \times r_2 \quad (4)$$

Where x_{min}, x_{max} represents upper and lower bound of individual position and r_2 is random number with value from 0 to 1.

At each iteration of herd selection, best m elephants with higher fitness are selected to be explored for next iteration. By this way better performing agents are preserved. The steps in the algorithm are given below

Algorithm: EHO

Input: population size(N), number of clans(C), upper and lower bound of positions (x_{min}, x_{max})

Output: Best solution

1. Initialize population with random positions
2. Calculate fitness for all agents in population and sort based on fitness value
3. Save the best m agents
4. Divide the initial population into C clans
5. while max generation is reached
6. for ci=1:c
7. for j=1: number of agents in clan
8. update$x_{c_{i,j}}$ as in eq. 2
9. generatex_{center,c_i} as in eq. 3
10. end
11. end
12. for ci=1:c
13. replace agent with worst fitness $x_{worst,c_{i,j}}$
14. end
15. end
16. return best solution

In the EHO algorithm, the position of elephant is updated based on its current position and its clan leader position. Clan leader position is updated as center of new position of each elephant in the clan. This process of position update can get into local minimum problem. This work proposes a modified

EHO algorithm to avoid the local minima problem. The modified EHO allow elephants to also learn its position from other clan leaders in its neighborhood range. The center of all leaders within a configured neighborhood value is calculated as

$$M_l = \frac{1}{m}\sum_{i=1}^{m} x_{center,c_m} \quad (5)$$

Where m is the number of clans within the neighborhood. The position of the agents are now updated as in Eq. (11) replacing the Eq. (7).

$$x_{new,c_{i,j}} = x_{c_{i,j}} + \alpha\left(x_{best,c_i} - x_{c_{i,j}}\right) \times r_1 + (1 - \alpha)\left(\left(x_{best,c_i} - M_l\right) \times r_2 \quad (6)$$

By this way, local minima problem in original EHO is avoided in the MEHO algorithm.

The fitness function for MEHO for the problem of finding optimal group of tasks is formulated as

$$f = \frac{1}{RC} + \frac{1}{SL} + \frac{1}{ERE} \quad (7)$$

The overall resource cost (RC) is calculated as the aggregation of cost of all the computing nodes used for execution of tasks. The computation nodes to be used for execution are calculated for each group as the total MIPS (million instructions per second) divided by the maximal MIPS capacity of the computing node.

$$RC = \sum_{k=1}^{NG} \frac{\sum_{i=1}^{NT} MIPS_i}{\max MIPS} \quad (8)$$

Where NG is the number of groups and NT is the number of tasks in the group. The service level adherence (SL) for a group is calculated in terms of

$$SL = \begin{cases} 1, \sum_{k=1}^{NG} T \times \max MIPS \geq \sum_{i=1}^{NT} MIPS_i \\ 0, otherwise \end{cases} \quad (9)$$

Where T is threshold for maximum MIPS capacity that can be used by tasks in the computing node. It is fixed as 0.8 in this work.

Overall distance for RE (ERE) in the group is calculated as the sum of distance of RE between each task in group divided by the number of tasks. It is calculated as

$$ERE = \sum_{k=1}^{NG} \frac{\sum_{i=1}^{NT}\sum_{j=1}^{NT-1} |dis(R_i) - dis(R_j)|}{NT} \quad (10)$$

Starting with initial group of tasks, MEHO is invoked to optimize the fitness function Eq. (7). The result of MEHO is the optimized group of tasks. Each group of task is scheduled to computing node

Table 3. Node configuration

Parameter	Value
CPU	2 core with 2.13 GHZ
RAM	8 GB
Disk	500 GB
OS	Ubuntu
Hadoop version	2.9.1
Number of replicas	3
HDFS block size	512 MB

C. Resource optimization layer

In most existing works, data blocks are replicated for ease of faster access by computing nodes spread across multiple sites, but replicating storage node as whole increases the storage cost and also has poor resource utilization when few parts of storage node alone are accessed. This work proposes a replication strategy with less storage cost overhead by exploiting the fine grained access to data blocks in the storage node. A popularity counter (PC) is maintained for each RE in the Namenode. This counter is either incremented or decremented based on access of data blocks corresponding to the RE. At Namenode, for every time duration, when a data block corresponding to RE in accessed, PC is incremented by up step and when not accessed the PC value is decremented by down step.

$$PC = \begin{cases} PC + \alpha, & \text{when RE is accessed} \\ PC - \beta, & \text{when RE is not accessed} \end{cases}$$

The number of replicas of the data block is made proportional to the PC value of RE to which the data block belongs. By this way, the data blocks with higher popularity based on it RE context alone are replicated, thus speeding the applications accessing those popular contents. Since replication is based on RE, even the data blocks which don't have higher access are still replicated expecting a higher access pre-emptively.

4. Results

The performance of the proposed solution is tested against experimental setup consisting of 6 nodes with 1 Namenode and 5 Data node. The configuration of each node is given in Table 3. Small text dataset of 100000,200000,300000 and 400000 files with file size from 1KB to 10 MB is used for experimentation. Three recent works in category of data management (Dynamic repository approach by Sharma et al [29]), task management (PAS scheme by Li et al [23]) and resource management (genetic

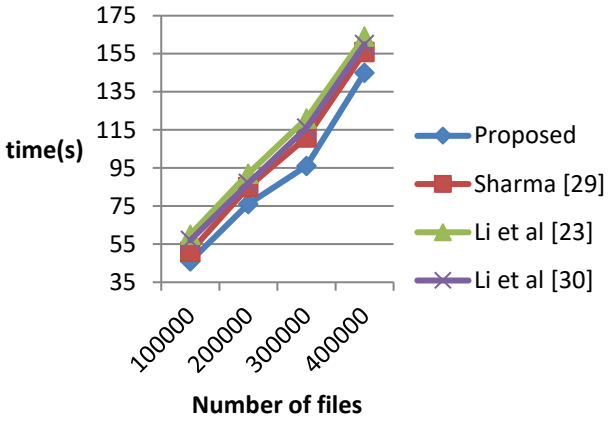

Figure. 4 Execution time for word count application

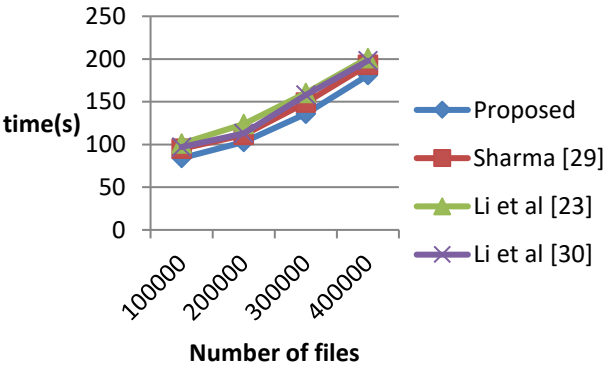

Figure. 5 Execution time for K mean clustering

algorithm based resource optimization scheme by Li et al [30]). Each of the above works attempted to speed up execution from three different perspectives of data, task and resource management and thus the proposed solution was compared with these works to prove the effectiveness of joint management of data, task and resources.

The application execution time was measured for two applications of word count and k-mean clustering for various sizes of small files and the result is given in Figs. 4 and 5. The average execution time in the proposed solution for word count application is 11% lower compared to Sharma et al [29], 20.4% lower compared to Li et al [23] and 15.7% lower compared to Li et al [30]. The average execution time in the proposed solution for K-means clustering application is 8.73 lower compared to Sharma et al [29], 16% lower compared to Li et al [23] and 12% lower compared to Li et al [30]. The application speedup has increased in the proposed solution due to combined three factors high relevant task grouping, fine grained data locality and popularity based data block replication at a fine grained level. All these factors increased the volume of sharable data and reduced data transportation costs. The access time is measured for different number of random files and

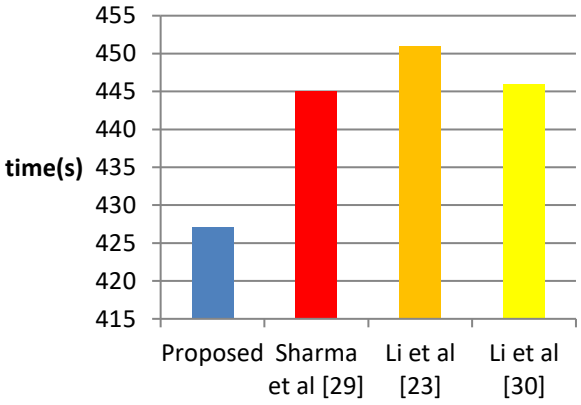

Figure. 6 Access time for 5 random files

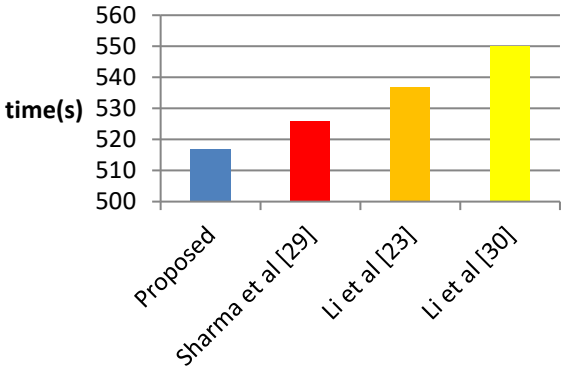

Figure. 7 Access time for 10 random files

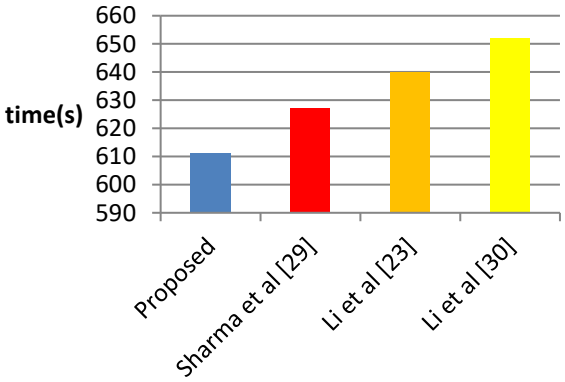

Figure. 8 Access time for 50 random files

the results are given in Figs 6-8. The average access time in proposed solution is atleast 2.7% lower compared to Sharma et al [29], 4.6 % lower compared to Li et al [23] and 5.9 % lower compared to Li et al [30]. The access time has reduced in the proposed solution due to LSH based hashing of metadata in the index file compared to sequential search used in existing works. Use of LSH hashing combined to organization of index file based on RE has facilitated faster access to data.

The normalized resource cost (NRC) is measured in terms of normalized percentage of overall time

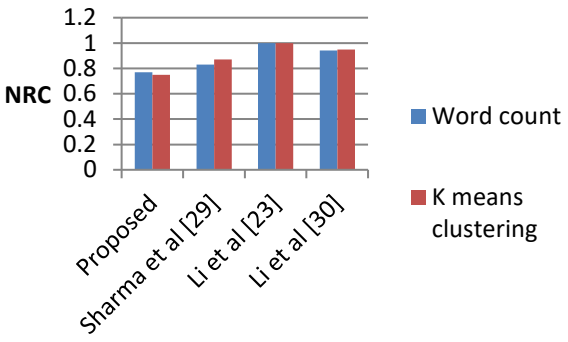

Figure. 9 Normalized resource cost

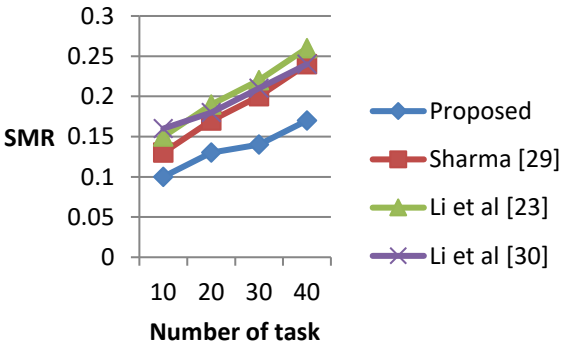

Figure. 10 SLA miss rate

resources were occupied for word count and K-means clustering application and the result is given in Fig. 9. The average resource cost in proposed solution is atleast 9% lower compared to existing works. The resource cost reduction is due to use of fine grained replication at level of RE in the proposed solution compared to replication of all data blocks or storage node as a whole in the existing works.

The SLA miss rate (SMR) is measured for various instances of tasks and the result is given in Fig. 10. The SLA miss rate in proposed solution is atleast 37% compared to existing works. The SLA has reduced in the proposed solution due to effective grouping of tasks based on fine grained data locality.

Discussion

The salient features of these existing works are given in Table 4. The proposed solution's application speedup increased by atleast 8.73% compared to Sharma et al, increased by atleast 16% compared to Li et al [23] and increased by atleast 12% compared to Li et al [30]. The reasons of the better performance of proposed solution were analyzed through ablation studies listed in Table 5. The studies were conducted with K mean clustering application by comparing the application execution time. The results for execution are presented in Table 6. In comparison to Sharma et al [29] (Data management), ablation 1 achieved

Table 4. Salient features of existing works

Sharma et al [29]	Authors created Hadoop archive from small files but made access faster using two special hash functions.
Li et al [23]	Dynamic policy update with the goal of improving job performance
Li et al [30]	Improved genetic algorithm is used to schedule the resources to the tasks with goal of minimizing the task completion time.

Table 5. Ablation postulates

Ablation 1	Only fine grained data locality + LSH based hashing (data management)
Ablation 2	Only fine grained task grouping (task management)
Ablation 3	Only fine grained data replication (resource management)
Ablation 4	Combined fine grained task grouping, fine grained data locality and fine grained data replication (Integrated management)

Table 6. Results of ablation studies

Data management	
Sharma et al [29]	137
Ablation 1	133
Ablation 4	126
Task management	
Li et al [23]	146.5
Ablation 2	140
Ablation 4	126
Resource management	
Li et al [30]	141.5
Ablation 3	139.7
Ablation 4	126

3.17% speedup. Thus the contribution of fine grained data locality+ LSH based hashing to application speed up in the proposed solution is 3.17%. But the overall integration (ablation 4) achieved 8.73% speed up. The LSH based hashing with data localization brings semantically similar data to same bucket and this in turn reduces the data transfer cost and increases application speed up. This proves that LSH based hashing with fine grained data localization management is very effective compared to Hadoop archive with hashing scheme adopted in Sharma et al [29].

In comparison to Li et al [23] (task management), ablation 2 achieved 5.51%. speedup. Thus the contribution of fine grained grouping in proposed solution is 5.51%. EHO based task grouping with

goal of task execution time minimization at same time exploiting task data sharing characteristics has increased the speedup compared to dynamic policy update strategy followed in Li et al [23]. The task grouping based on similarity is enhanced by using it as a important factor in fitness function to be optimized by modified EHO in the proposed task management scheme. This in combination with integrated management increased the speedup by another 3.2% to overall 8.73% speedup.

In comparison to Li et al [30] (resource management), ablation 3 achieved 1.4% speed up. The speedup in ablation 3 is less compared to ablation 1 and ablation 2. The fine grained resource replication performed only marginally higher compared to improved genetic algorithm based resource matching strategy proposed by Li et al [30]. But still the overall speed in integrated is higher due to contribution of ablation 1 and ablation 2.

From these observations, it is proved that integrated management strategy adopted in proposed solution has contributed significantly positive to the application speedup.

5. Conclusion

An integrated management technique considering data, task and resource is proposed for speed up of small files processing in Hadoop clusters. The proposed scheme grouped files at fine grained level based on semantics using natural language processing. The tasks are grouped based on multi criteria optimization using modified elephant herding algorithm. In addition replication of data blocks were done based on popularity at a fine grained level. Due to the joint integration of data, task and resource management, the proposed solution is able to achieve at least 8.7% speed up, reduce resource cost by 9% and reduce SLA miss rate by at least 37% compared to most recent works on individual management of data, task and resources.

Conflicts of Interest

Authors declare no conflict of interest.

Author Contributions

Shwetha is the primary author who conceptualized, implemented and documented this paper work. Chandramouli reviewed the work and contributed to improvising the paper work.

Acknowledgments

Shwetha was the primary author who conceptualized, implemented the concept, collected results and documented the paper. Chandramouli reviewed the work, suggested changes and verified the results.

References

[1] R. Aggarwal, J. Verma, and M. Siwach, “Small files’ problem in Hadoop: A systematic literature review”, *Journal of King Saud University - Computer and Information Sciences*, Vol. 34, No. 10, pp. 8658-8674, 2022.

[2] A. Mehmood, M. Usman, W. Mehmood and Y. Khaliq, “Performance efficiency in Hadoop for storing and accessing small files”, In: *Proc. of Seventh International Conference on Innovative Computing Technology (INTECH)*, pp. 211-21, 2017.

[3] B. Dong, Q. Zheng, F. Tian and K. Chao, “An optimized approach for storing and accessing small files on cloud storage”, *Journal of Network and Computer Applications*, Vol. 35 pp. 1847-1862, Elsevier, 2012.

[4] T. Chen, J. Hung, S. Hsieh and R. Buyya, “Heterogeneous Job Allocation Scheduler for Hadoop MapReduce Using Dynamic Grouping Integrated Neighboring Search”, *IEEE Transactions on Cloud Computing*, Vol. 8, No. 1, pp. 193-206, 2020.

[5] J. Xie, S. Yin, X. Rua and X. Qin, “Improving MapReduce performance through data placement in heterogeneous Hadoop clusters”, In: *Proc. of IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pp. 1-9, 2010.

[6] J. Marquez, H. Mondragon, and D. Juan, “An Intelligent Approach to Resource Allocation on Heterogeneous Cloud Infrastructures”, *Applied Sciences*, Vol. 11, No. 21, pp. 9940, 2021.

[7] R. Jeyaraj, S. Ananthanarayana and A. Paul, “Fine-grained data-locality aware MapReduce job scheduler in a virtualized environment”, *J Ambient Intell Human Comput*, Vol. 11, pp.4261-4272 ,2020.

[8] F. Thaha, M. Amin, S. Kannan and M. Ahmad, “Data location aware scheduling for virtual Hadoop cluster deployment on private cloud computing environment,” In: *Proc. of 22nd Asia-Pacific Conference on Communications (APCC)*, pp. 103-109, 2016.

[9] C. Li, J. Zhang and M. Tao, “Data locality optimization based on data migration and hotspots prediction in geo-distributed cloud environment”, *Knowledge-Based Systems*, Vol. 165, 2018

[10] M. Hammoud and F. Sakr, “Locality-aware reduce task scheduling for MapReduce,” In:

Proc. of the IEEE Third International Conference on Cloud Computing Technology and Science, pp. 570-576, 2011.

[11] Y. Yao, H. Gao, J. Wang, B. Sheng and N. Mi, "New Scheduling Algorithms for Improving Performance and Resource Utilization in Hadoop YARN Clusters", *IEEE Transactions on Cloud Computing*, Vol. 9, No. 3, pp. 1158-1171, 2021.

[12] G. Kousiouris, T. Cucinotta and T. Varvarigou, "The effects of scheduling, workload type and consolidation scenarios on virtual machine performance and their prediction through optimized artificial neural networks", *Journal of Systems and Software*, Vol. 84, pp. 1270-1291, 2011.

[13] W. Wang, K. Zhu, L. Ying and J. Tan, "MapTask scheduling in mapreduce with data locality: throughput and heavy traffic optimality", *IEEE/ACM Transactions on Networking*, Vol. 24, pp. 190-203, 2016.

[14] A. Yang, J. Wang, Y. Mao and Y. Yao, "Optimizing Internal Overlaps by Self-Adjusting Resource Allocation in Multi-Stage Computing Systems", *IEEE Access*, pp. 88805-88819, 2021.

[15] Y. Liu, Y. Zeng and X. Piao, "High-Responsive Scheduling with MapReduce Performance Prediction on Hadoop YARN", In: *Proc. of IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 238-247,2016.

[16] X. Zhang, Y. Feng, S. Feng, J. Fan and Z. Ming, "An effective data locality aware task scheduling method for MapReduce framework in heterogeneous environments", In: *Proc. of International Conference on Cloud and Service Computing*, pp. 235-242, 2011.

[17] Y. Chen, W. Wei, F. Wei, and. J. Chen, "LaSA: A locality-aware scheduling algorithm for Hadoop-MapReduce resource assignment", In: *Proc. of International Conference on Collaboration Technologies and Systems*, pp. 342-346, 2013.

[18] H. Wei, H., Wu, Y. Lee, C. Hsu, "Shareability and Locality Aware Scheduling Algorithm in Hadoop for Mobile Cloud Computing", *J. Inf. Hiding Multim. Signal Process*, Vol. 6, pp.1215-1230, 2015.

[19] A. Gandomi, M. Reshadi and A. Movaghar, "HybSMRP: a hybrid scheduling algorithm in Hadoop MapReduce framework", *J Big Data*, Vol. 6, No.106, 2019.

[20] D. Choi, M. Jeon, N. Kim and D. Lee, "An Enhanced Data-Locality-Aware Task Scheduling Algorithm for Hadoop Applications", *IEEE Systems Journal*, Vol. 12, No. 4, pp. 3346-3357,2018.

[21] M. Convolbo, J. Chou, H. Hsu and C. Chung, "GEODIS: towards the optimization of data locality-aware job scheduling in geo-distributed data centers", *Computing*, Vol. 100, No. 12, pp. 21-46,2018.

[22] Q. Xie, M. Pundir, Y. Lu and Y., Abad, "Pandas: Robust Locality-Aware Scheduling with Stochastic Delay Optimality", *IEEE/ACM Transactions on Networking*, Vol.25,No.2,pp. 662-675,2017.

[23] Y. Li, T. Li and P. Shen, "PAS: Performance-Aware Job Scheduling for Big Data Processing Systems", *Security and Communication Networks*, Vol. 2022, pp. 14 pages, 2022.

[24] N. Lim, S. Majumdar and P. Ashwood-Smith, "MRCP-RM: A Technique for Resource Allocation and Scheduling of MapReduce Jobs with Deadlines", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 28, No. 5, pp. 1375-1389, 2017.

[25] N. Lim, S. Majumdar and P. Ashwood-Smith, "Techniques for Handling Error in User-Estimated Execution Times During Resource Management on Systems Processing MapReduce Jobs", In: *Proc. of IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pp. 788-793, 2017.

[26] Y. Chen, W. Wei, F. Wei and J. Chen, "LaSA: A locality-aware scheduling algorithm for Hadoop-MapReduce resource assignment", In: *Proc. of International Conference on Collaboration Technologies and Systems*, pp. 342-346, 2013.

[27] D. Tao, Z. Lin and B. Wang, "Load Feedback-Based Resource Scheduling and Dynamic Migration-Based Data Locality for Virtual Hadoop Clusters in OpenStack-Based Clouds", Tsinghua Science and Technology, Vol. 22, No. 2, pp. 149-159, 2017.

[28] https://www.dbpedia-spotlight.org/

[29] S. Sharma, A Afthanorhan, C. Barwar, S. Singh and H. Malik, "A Dynamic Repository Approach for Small File Management with Fast Access Time on Hadoop Cluster: Hash Based Extended Hadoop Archive", *IEEE Access*, Vol. 10, pp. 36856-36867, 2022.

[30] Y. Li and X. Hei, "Performance optimization of computing task scheduling based on the Hadoop big data platform", *Neural Comput&Applic*, 2022.