The International Journal of Interdisciplinary Organizational Studies
ISSN: 2324-7649 (Print), ISSN: 2324-7657 (Online)
Volume 19, Issue 1, January-June, 2024

OPTIMIZING DATA PLACEMENT AND CLASSIFICATION: A HYBRID APPROACH WITH GENETIC ALGORITHMS AND PARTICLE SWARM OPTIMIZATION

Dr. S. Annapoorani
Assistant Professor of Computer Science, Gobi Arts & Science College, Gobichettipalayam, Tamilnadu, India, Email: sannapoorani@gascgobi.ac.in

Corresponding Author:
Dr. T. P. Senthilkumar
Assistant Professor of Computer Science, Gobi Arts & Science College, Gobichettipalayam, Tamilnadu, India, Email: tps@gascgobi.ac.in

Abstract: This paper proposes an Effective Data Emplacement and Classification (EDEC) approach for large-scale streaming data applications in heterogeneous cloud environments. It introduces a clustering model utilizing map and reduce functions to efficiently provision and classify data across resources. The approach leverages distributed computing to handle large volumes of unstructured data, offering a cost-effective solution. To address load imbalance, the model incorporates heterogeneity-aware scheduling mechanisms. The methodology involves MapReduce cluster constraints, dynamic imbalance data processing, task clustering, and adaptive task tuning. Genetic algorithms and particle swarm optimization aid in task scheduling and regrouping for improved performance. Additionally, bipartite graph modeling facilitates efficient resource allocation. The proposed approach is evaluated using Hadoop and demonstrates enhanced scalability and efficiency in processing large datasets.

Keywords: Data Emplacement, Classification, MapReduce, Heterogeneous Cloud Environment, Task Scheduling, Genetic Algorithms, and Particle Swarm Optimization.

1. Introduction

In this paper, an Effective Data Emplacement and Classification (EDEC) approach for large scale streaming data application will be designed in the heterogeneous cloud environment. It acts as efficient data clustering model to provide scalability to the data provisioning in the resource under shortest duration using map and reduce function. The multi-node cluster index has been set to the resource characteristics. It also has proven to be an effective platform to process a large set of unstructured data.

The approach employs the distributed computing at scale usually involving hundreds to thousands of machines to facilitate the large scale data processing. It offers a more cost-effective model to implement data processing and data placement using multi-node cluster index.

The variation in refining capabilities on MapReduce nodes results in load imbalance which requires separate mechanism to make task scheduling and load balancing as heterogeneity aware model for various resource characteristics. MapReduce model uses different constraints and configurations based on the resources to balance the data against increasing the efficiency of the

resource clusters in the cloud environment on the optimal value of the fitness function of the resource schedulers.

2. Proposed Methodology

2.1. MapReduce Cluster Constraints

The data processing model consists of a service manager, several service manager candidates, several task execution managers, a cluster coordinator, and metadata storage to define data level constraints. The figure 4.1 represents the architecture of distributed data processing system. The constraints employed on the various components of the data processing model is as follows

Service Manager

Service manager manages entire cluster and schedules task instances of user-defined distributed stream processing service to distributed nodes for parallel execution. Task execution managers manage task executors on each node, and the task executors run each assigned task instances as separate threads in the same process space. Task instances are cloned from user-defined task and they run on distributed nodes in parallel by sharing and partitioning the same input data stream (Abhishek Verma *et al,* 2012).

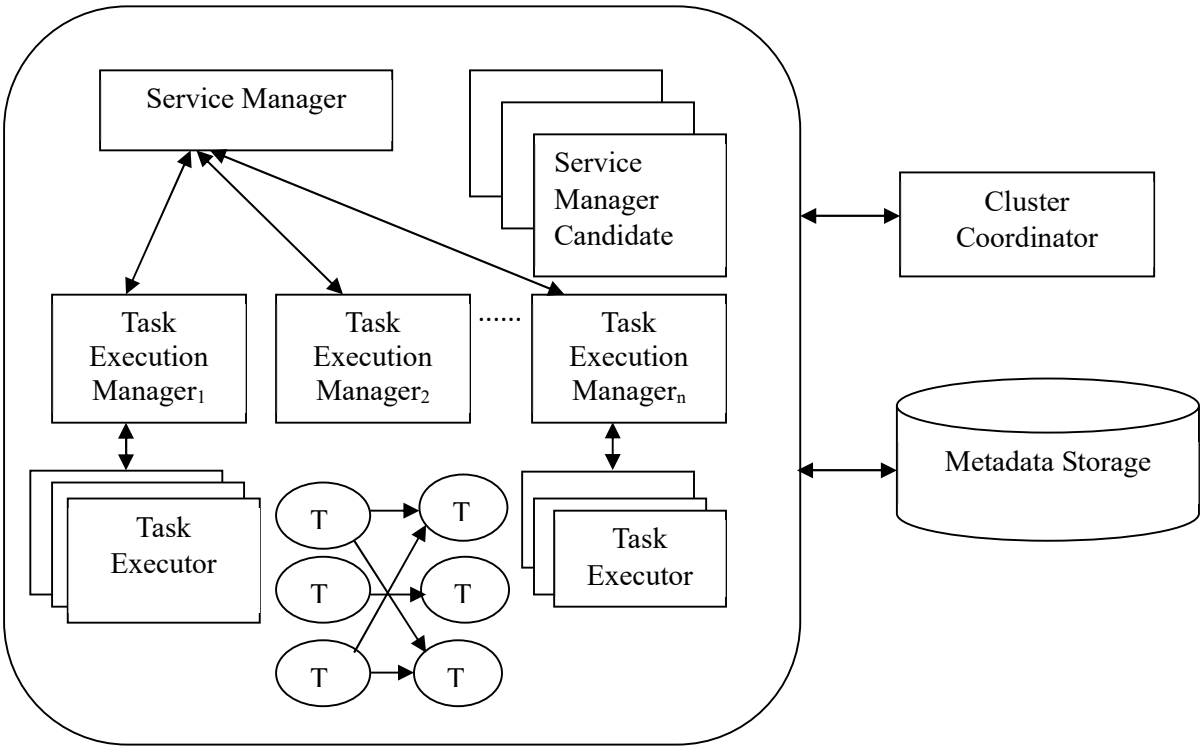

Figure 1: Distributed Data Processing System

Cluster Coordinator

Cluster coordinator is used for master selection in case of master node's failure, for shared storage and coordination of communication between several cluster components for data level evolution in the streams. Metadata storage is for the management of all the data related to the

cluster, service, user and QoS (Quality of Service) preferences. In the heterogeneous cloud cluster, there are five switches to connect the four computing nodes.

Task Execution Manager

Task execution manager is responsible for the task mapping to resource for execution based on the conditions. Task executor computes the each node characteristics and makes optimal decision. It switches between different computing nodes based on the deadline priorities. The optimal performance can be achieved on implementing it as separate service utility.

2.2. Design of the Proposed Methodology

The figure 4.2 represents the architecture of an Effective Data Emplacement and Classification Approach. The configurations on heterogeneous nodes inevitably lead to sub-optimal performance. Deriving the optimal configuration is a tedious and error prone process. A large number of performance-critical parameter can have complex interplays on data emplacement and data classification. However, it is difficult to construct models to connect parameter setting for optimal solution on the MapReduce performance towards data processing.

The different configurations are needed for different execution phases on the dynamic imbalance data. The data classification configurations should also be changed in response to the changes in the heterogeneous cluster resource performance.

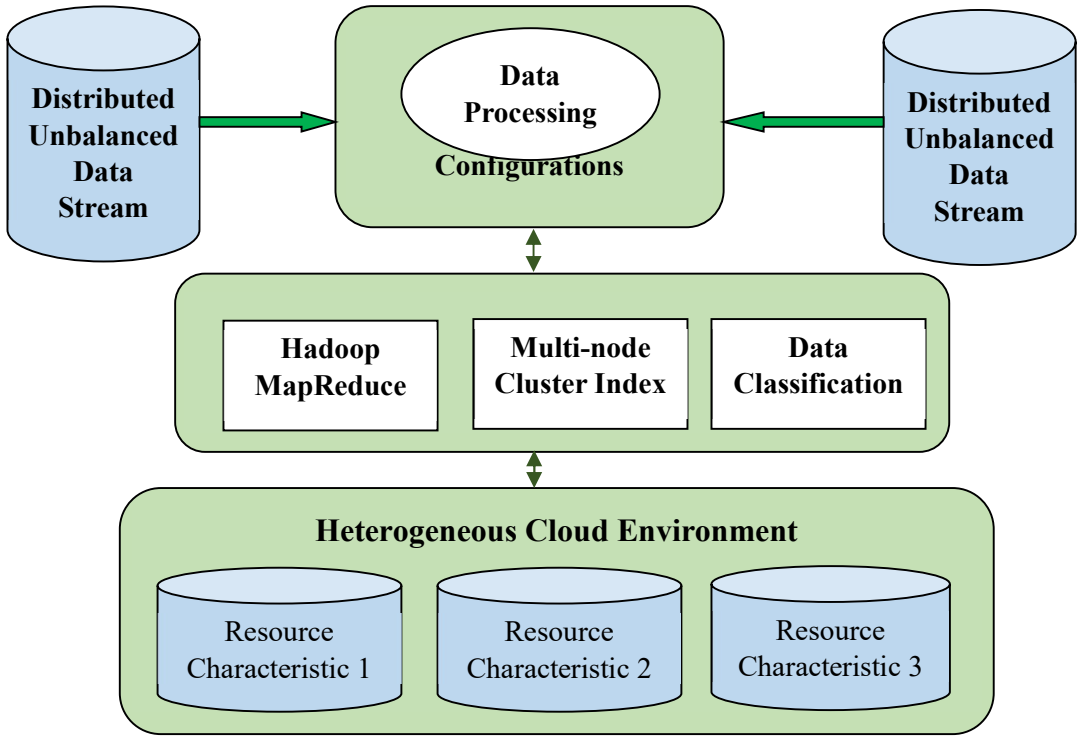

Figure 2: Architecture of an Effective Data Emplacement and Classification Approach

The Map Reduces model launches with different configurations and considers the ones that complete data processing in less execution time as good setting. To accelerate the execution speed and avoid trapping in local optimum, the model uses genetic functions crossover and mutation to

generate data classification configurations for streaming the data. Hadoop has been used to perform comprehensive evaluations with representative MapReduce architecture.

Distributed Unbalanced Scheduling framework

MapReduce is a distributed parallel programming model originally designed for processing a large volume of data in a heterogeneous cloud environment. Based on the default, Hadoop framework, a large number of parameters need to be set before a processing of the imbalance data to cluster in the available resources. These parameters control the behaviours of data classification on obtaining resource characteristics including their memory allocation, level of concurrency, I/O optimisation, and the usage of network bandwidth. In Hadoop, there are more than 190 configuration parameters, which determine the settings of the Hadoop cluster, describe a MapReduce job to the Hadoop framework, and optimise the data classification.

Cluster-level parameters specify the organisation of a Hadoop cluster and some long-term static settings. Changes to such parameters require rebooting the cluster to take effect. Data classification parameters which considered as job level parameters determine the overall execution settings, such as input format, number of map/reduce tasks, and failure handling. These parameters are relatively easier to tune and have uniform effect on all imbalance and exploring data even in a heterogeneous cloud environment. Task level parameters which related to the conceptual changes in data or theme of data, control the fine-grained task execution (data classification) on individual nodes and can possibly be changed independently and on-the-fly at runtime. Parameter tuning which is considered as data pre-processing at the task level opens up opportunities for improving performance in heterogeneous environments using multi-node cluster index structures.

Multi-node Cluster Index Structure

Multi-node cluster index structure set the configuration parameters to default values assuming a reasonably sized cluster and typical MapReduce jobs. Cluster indexing will predefine number of map and reduce slots for the task execution on the resource node based on the capabilities.

Each cluster has different processing capability and optimised task configurations can be quite different across sub clusters. The cluster index is represented in the Eq (1) & (2) as order to the resource providing maximum execution cost.

C (i) = Sort. Resource (percentage)..... Eq (1)

C (i) = Sort. Resource (task runtime)..... Eq (2)

Task configurations with high fitness values have been indexed to the resources. Cluster index structure for a task with inherently more job can be even harmful to normal tasks as allocating more memory to normal tasks. Cluster index will cost-effectively analyse large amounts of data without creating large infrastructures of their own. Virtual node will have capacity varying based on the collocated jobs. The two important cluster index structures are as follows

(i) Tasks belonging to the same job run with different configurations matching the capabilities of the hosting machines

(ii) The configurations of individual tasks dynamically change to search for the optimal settings

2.3. Imbalance Data Classification

Imbalance data is considered as task. The multiple tasks with different configurations concurrently reproduce new configurations based on the information of completed task on the cluster. The reproduction of generations is directed by a genetic algorithm which ensures that the good configurations in prior generations are preserved in new generations for the resource cluster based multi-node cluster index value. Each sub cluster has different processing capability; the optimised task configurations can be quite different across sub clusters.

Task-level parameters control the behaviour of task execution, which is critical to the Hadoop. Task-level parameters which have significant performance impact the resource mapping. The Task distribution to Map is given by Eq (3),

$$TD = \sum_{n=0}^{x} D^k K \quad \text{..... Eq (3)}$$

Where D^k is the document instance or data instance on the cluster

K is the cluster

$$TR = T + K \sum_{n=1}^{\infty} \left(\text{rand} \frac{T}{l} + \log \frac{T}{l} \right) \quad \text{..... Eq (4)}$$

Task on reduce function is computed randomly by using Eq (4.4) as well logarithmically at different task configurations. Scheduling on the heterogeneous cluster is to minimize job execution time, task completion time.

Algorithm 1: Imbalance Data Scheduling

Step 1: Compute the fitness for each job

Step 2: If (jobs== fitness value)

Determine slot availability & Select two configuration candidates as parents;

Else

Determine Crossover and Mutation operations;

Use the obtained new generation C_{new} to assign task

End if

Step 3: Iterate until the running job is completed

Genetic is applied to scheduling in order to minimize job execution time and task completion time. Further longer task completion time does not necessarily indicate a worse configuration as some tasks are inherently longer to complete.

Most cases task misconfigurations are related to task memory allocations and incur excessive data spill operations. Finally fitness function does not address the issue of data skew due to non-uniform record distributions in task inputs.

2.4. Task Clustering

Performance Based Sampling (PBS) approach uses genetic algorithm to search the optimal task-level configurations for heterogeneous cloud resources. MapReduce jobs composed of multiple clusters of map tasks. The execution of individual task T is denoted by its parameter set C. A set candidate C_i consisting of a number of selected parameters represents a task configuration set which is represented by the Eq (5),

$$C_i = [g_1, g_2, g_3 \ldots . g_n] \quad \text{..... Eq (5)}$$

An initial configuration of randomly generated candidates for the task assignment has been initiated for genetic process. Further it evolves individual task configuration to breed good solutions during each interval by using the genetic reproduction operations. Fitness of all completed tasks has been computed on the each resource.

Task Scheduling using PCA

The constraints for task scheduling using Principle Component Analysis are as follows:

- Available slot in the cluster selects two configuration candidates as task execution configuration
- The new generation configuration candidates by using the proposed genetic reproduction operations assign the task with the new generated configuration set to the available slot

The crossover operator determines how task are exchanged between the resources to create those offspring's.

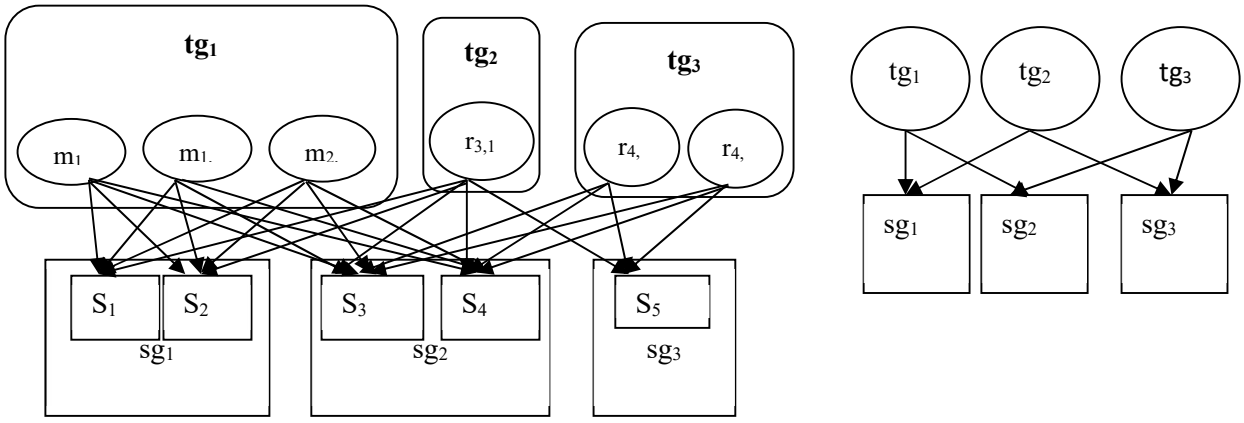

Figure 3: General Task Regrouping

The mutation allows for random alteration of task. While the selection and hybrid administrators tend to improve the quality of the task execution in the new generation and force convergence, transformation tends to bring in divergence. The figure 4.3 represents the general task regrouping.

Resource Selection for Task Placement

Towards ensuring the capacity of the queues for various jobs, capacity scheduler can also achieve fairness, however it requires a progressively control over the cluster.

The candidates with good fitness values to have a higher probability of being selected as resource cluster for data emplacement. The selection module ensures reproduction of more highly fit candidates compared to the number of less fit candidates. Roulette Wheel (RW) mechanism fitness $f(C_i)$ is the fitness of completed task performance in the candidate population, its probability of being selected by the Eq (6),

$$P_i = \frac{f(Ci)}{\sum f(Ci)} \quad \text{..... Eq (6)}$$

Where P is the number of tasks completed

Static Task Assignment

A crossover function is used to cut the sequence of elements from two chosen candidates and swap them to produce two new candidates. Crossover operation is employed for each individual resource. Relative fitness crossover is computed instead of absolute fitness crossover operation, because it moderates the selection pressure and controls the rate of convergence. Crossover operation is exercised on configuration candidates with a probability, known as crossover probability. The figure 4 depicts the Genetic Scheduling of the MapReduce Cluster.

.

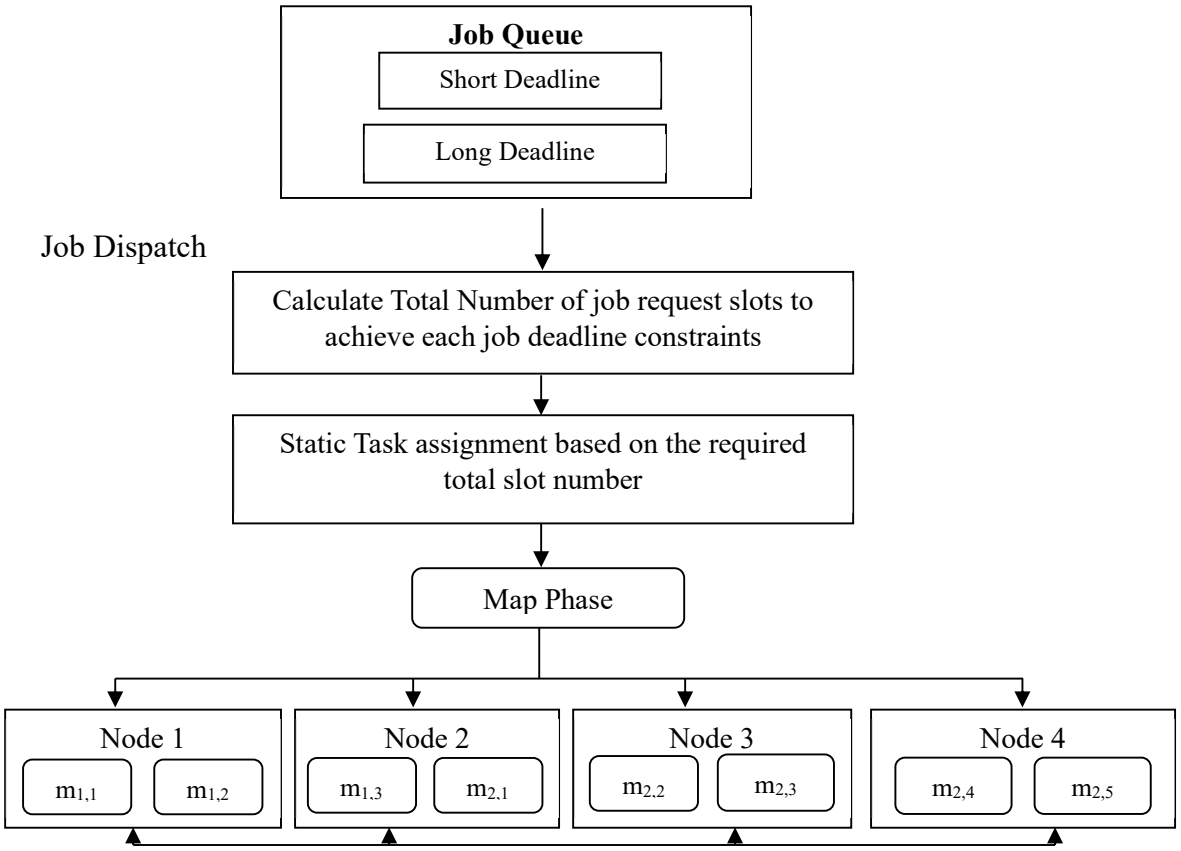

Figure 4: Genetic Scheduling for MapReduce Cluster

Dynamic Map Assignment

The mutation function aims to avoid trapping in the local optimum by randomly mutating an element with a given probability.

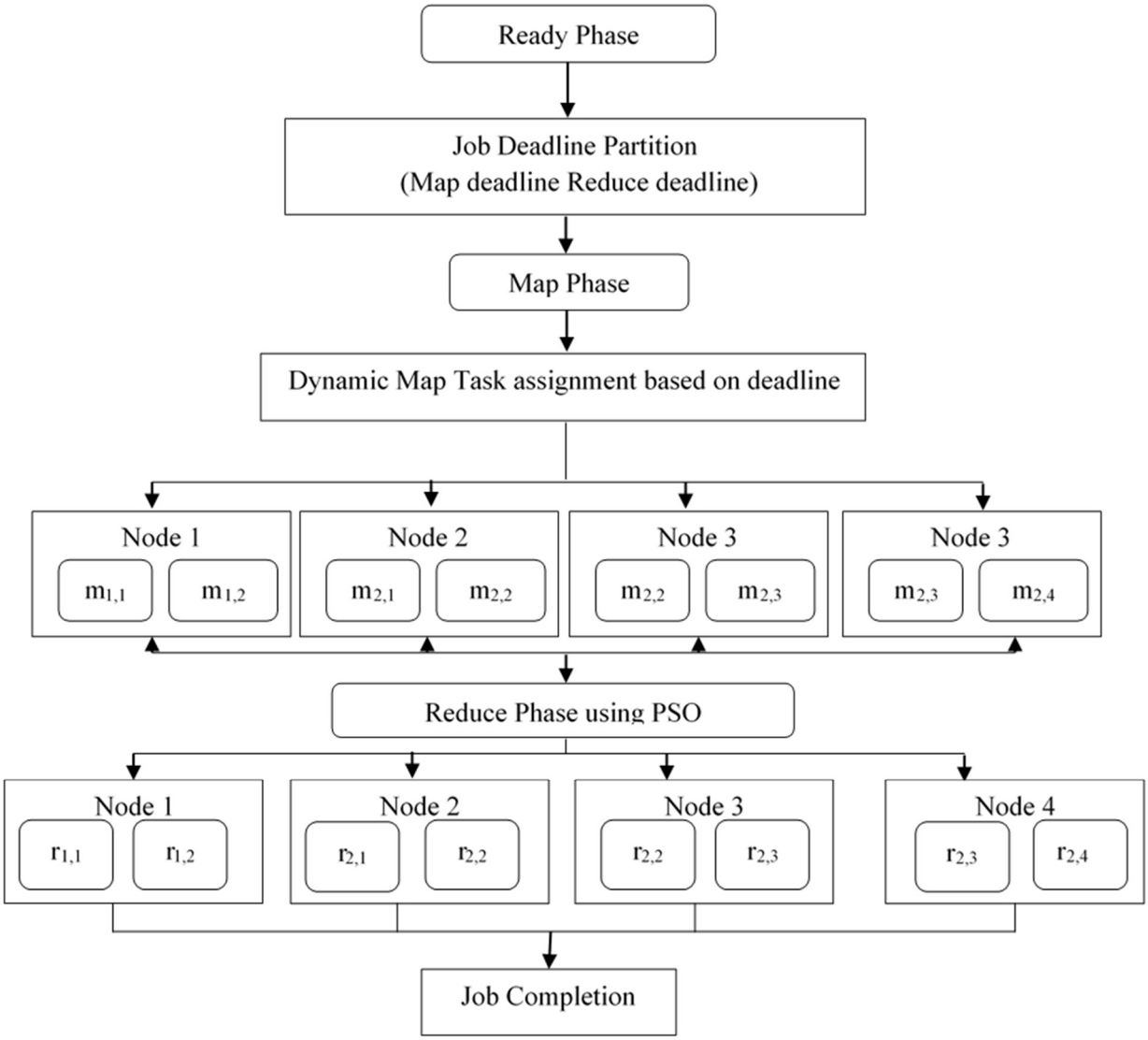

Figure 5: Task Regrouping based PSO

Instead of performing task by task mutation at each generation of the resource cluster, a random resource cluster is generated for each individual task. If R represents the population of the task is larger than the mutation probability, the particular individual undergoes the mutation process.

Otherwise, the mutation operation involves replacing a randomly chosen parameter with a new value generated randomly in its search space. This process prevents premature convergence of the population and helps mutation sample the entire solution space using PSO. The figure 5 represents the Task Regrouping using PSO.

Adaptive Task Tuning

Roulette Wheel selection mechanism has a higher probability of selecting good candidates. However, this approach still results in too many task evaluations, which in turn reduces the speed of convergence in order to avoid those challenges, adaptive task tuning is performed as task

regrouping. The frequency of the re-grouping is determined by two factors: the dynamics of task inference and mutation in the scheduling and another one is due to the size of the virtual MapReduce cluster. The heuristic for task regrouping are as follows

Heuristics

- The re-grouping should be more frequent correspondingly so as to capture dynamic capacity changes of the virtual nodes which has not occur or less occur when the size of the virtual cluster is small
- Dynamic capacity change of the virtual nodes has been formulated by

$$\text{Change of task order } = p(m) + \frac{pi}{Ci} + \frac{p(Ci - (Ci-1))x^2}{Ci}$$

In addition to non-met heuristic scheduling algorithms, an alternative is the heuristic scheduling algorithms that can be used to provide better scheduling plans than do the rule-based scheduling algorithms for resource scheduling.

Algorithm 2: Dynamic Task Regrouping

Inputs: K: Number of groups (initialise k=2); M: Number of VMs in a virtual cluster.

Step 1: Randomly choose k VMs in the cluster as k groups

Step 2: Repeat if (VM ≠ Cluster group)
Then
Assign the VM to the Resource Cluster;
Update the mean of the group;
End if
until the minimum mean of the group is achieved;

Step 3: Continue if (number of cluster is fixed)
Then Compute: Task ordering| Task regrouping
End if

Task Regrouping

Parallel Hybrid Particle Swarm Optimisation

Parallel Hybrid Particle Swarm Optimisation (PHPSO) is a fine-grained resource-aware MapReduce scheduler that divides tasks into phases, where each phase has a constant resource usage profile. It performs scheduling at the phase level which exhibit the significance of phase-level scheduling by demonstrating the resource usage variability within the lifetime of a task utilising a wide-range of MapReduce jobs.

Algorithm 3: PHPSO Based Task Regrouping

Step 1: Assign No. of Particles – No .of Tasks

Step 2: Compute Search Space – No. of Resources

Step 3: Each task (particle) treated as point in N dimensional Space

Step 4: Each particle organises its best solution based on fitness function is treated as pbest

Step 5: Any particle organised in the particular space based on fitness function is treated a gbest

Step 6: Particle is iterated to attain the pbest locations and gbest locations by random weight assignment at each iteration – Position Modification

Step 7: Compute $V_i^{k+1} = wV_i^k + c_1\, rand_1(\ldots) \times (pbest_i - s_i^k) + c_2\, rand_2(\ldots) \times (gbest - s_i^k)$

Where

V_i^k : velocity (speed) of operator i at iteration k,
w : weighting capacity,
c_j : weighting factor,
rand : consistently distributed random number between 0 and 1,
s_i^k : current position of operator i at iteration k,
$pbest_i$: pbest of operator i,
gbest : gbest of the group

Step 8: w = wMax-[(wMax-wMin) x iter]/maxIter

Where

wMax : initial weight,
wMin : final weight,
maxIter : maximum iteration number,
iter : current iteration number

Step 9: $s_i^{k+1} = s_i^k + V_i^{k+}1$

A large inertia weight (w) facilitates an entire resource search while a small inertia weight facilitates a cluster search. By straightly decreasing the inertia weight from a relatively large value to a small value through the course of the PSO run provides the best PSO performance compared with fixed inertia weight settings.

Bipartite Remodeling

A weighted bipartite graph is to represent the scheduling (allocation) relationship between the task and resource on Service Level Agreement (SLA) and its strategies for efficient data emplacement and redistribution. The resource management is carried to manage resource slots which minimize the energy consumed when executing the application achieve optimal schedules.

Algorithm 4: Bipartite graph modelling

Step 1: Collect the running map and reduces tasks of such jobs for each task t of T
Step 2: Do calculate ft ← Find the feasible slots of t
Step 3: Collect ft in F
Step 4: Form a weight bipartite graph based on the T and F

Multiple jobs may submit their map or reduce tasks. These tasks will be allocated to appropriate slots without violating their respective map or reduce deadlines. The proposed adaptive deadline partition can re-calculate the new reduce deadline after the map phase. According to the new reduce deadline, the reduce tasks can take precedence to obtain the correspondingly feasible slots. After performing the node group technique, a reduced weighted bipartite graph is formed. Then, the heuristic algorithm slots the edges of the graph in increasing order of their corresponding weights.

Next, the algorithm runs in iteration. In each iteration, the edge with the lowest weight is selected. The corresponding task group and slot group on the selected edges are recorded. The

selected edge is then removed from the graph. Finally, the algorithm terminates when no edges exist in the graph.

Task Partitioning and Task Selection

The data is partitioned into groups or partitions using following strategies, such as the feature with inconsistent task value which has been considered as non actionable element to the node using push based method, which in composing task selection and task partitioning to reduce task characteristics. The figure 4.6 indicates the Task Reduction Process using Push Based Method.

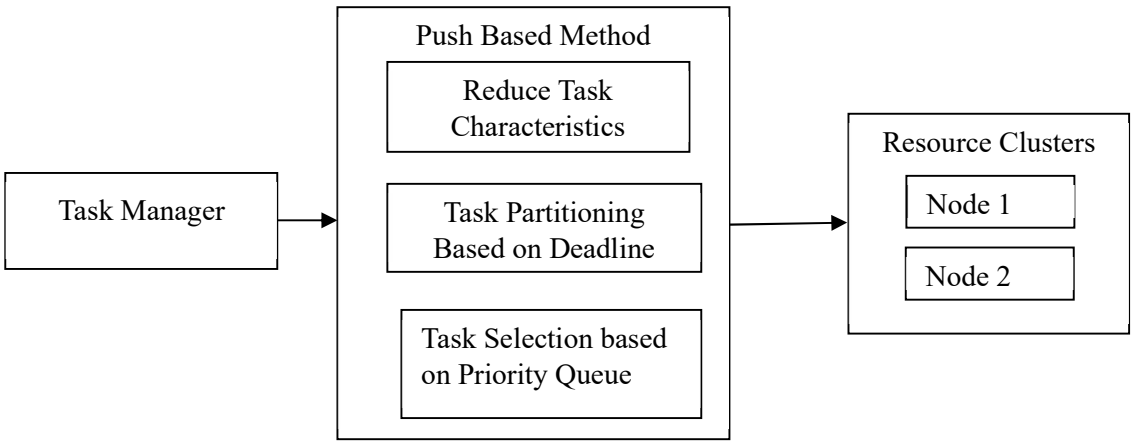

Figure 6: Task Reduction Process using Push Based Method

The reduced feature has to abide with the cases to aggregate the space of the task manager with retaining the important features. Application of the system performance can be improved on two dimensional task characteristics correlations. The correlation of the task function is calculated based on the similarity and distance function using correlation coefficient of the task parameter. The coefficient measures of the correlation between pairs of parameter value to remove one of two highly correlated data in the resource scheduler.

MapReduce Framework utilizing KNN Classification

The K Nearest Neighbour classification is modelled to compute the distance between the two tasks in the mapper's function which is considered using vector space model as it has ability to compute the similarity task for resource node. After computing the distance between the task and resources, sorting has to be made in ascending order to extract the results on the task.

Algorithm 5: Data processing using KNN classification

Input

Initialise the cluster centre ()

Form clusters () based on cluster centre

For all data points

Process

Take training data set <ds, f(ds)>

Take testing data set <ds, f(ds)>

Set N to sample values

Assign the learning rate LER

Assign N value for cross validation
Attribute values Normalize by standard deviation
Set feature weights for each cluster
Normalize the features
Set random Values vi For each attribute ATi
Divide the training examples into N sets
By cross validation Train the Values
For Each pair Ek in N, do
Assign Ek = Validation pair
For each sample xi in N such that xi does not belong to Ek do
Based on the Euclidean distance find the K nearest neighbors
Return the class values that represents the maximum of the k instances
If actual class(ac)Not Equal to predicted class(pc) then apply gradient descent
Err = no of partitions Predicted(P)
Class
For each vk
vk = vk + LER * Err * Ek
where Ek is the query attribute value
LER is the Least Error Rate

Output : Then form cluster based on the features weights.

KNN classification has been carried out with the different strategies used to finally compute and sort distances efficiently using MapReduce for reduced overhead of the scheduler. These different strategies can be divided into two categories, depending on the number of jobs they require for load balancing.

4. Result and Discussion

In this approach, data processing model has been applied with dataset of 50GB is used for conveniently calculating the data retrieval cost of a job. The experiments in heterogeneous cloud environments with parameter setting on Hadoop represent the performance variability associated with them. It is used to obtain the optimal solution for the deadline constrained task scheduling to the MapReduce framework. Table 1 provides the Performance Values of the Effective Data Emplacement and Classification approach.

Table 1: Performance Values of the Effective Data Emplacement and Classification approach

Technique	Job Transfer Time in ms	Job Elapsed Time in ms	VM Utilization in mb	Execution Time in ms
Sampling based Load Balancing (SLB) approach	25.32 ms	5.25 ms	12.56 mb	45.25 ms

Effective Data Emplacement and Classification (EDEC) approach	21.32 ms	1.56 ms	9.56 mb	28.98 ms

The proposed approach handles multiple jobs on adopting the FIFO and Fair schedulers to extend the performance. In addition, the model estimates the available slots for serving the job to meet job deadline. Further constraints utilise the slot heterogeneous performance to assign each task to the best suitable slot. Storage management in the cluster is to serve better on the task evolutions. Data placement constraint is used to determine the hidden features of the job which hide the unobserved features of the dataset which can be even eliminated as non valuable feature on estimating the hardware properties of the computing platform. It can be considered as resource sampling technique for the effective task redistribution. The figure 7 represents the Performance Evaluation of the EDEC approach against SLB approach in terms of Job Transfer Time.

The generated feature can be categorised as mixture model for the virtual machine for effective utilisation of the slot, in that each feature from the mixture can be selected as important structure for VM formation is represented as follows.

$$F[X_{t+1}] = x_{t+1} \mid X_t = x_t , X_{t-1} = x_{t-1} , \ldots ,$$

$$F[X_{t+1}] = X_1 = x_1 , X_0 = x_0$$

Where X_t is the feature at the collection of the mixture generated from the unobserved features. The fitness constraint has been derived for each job execution in the VM allotted.

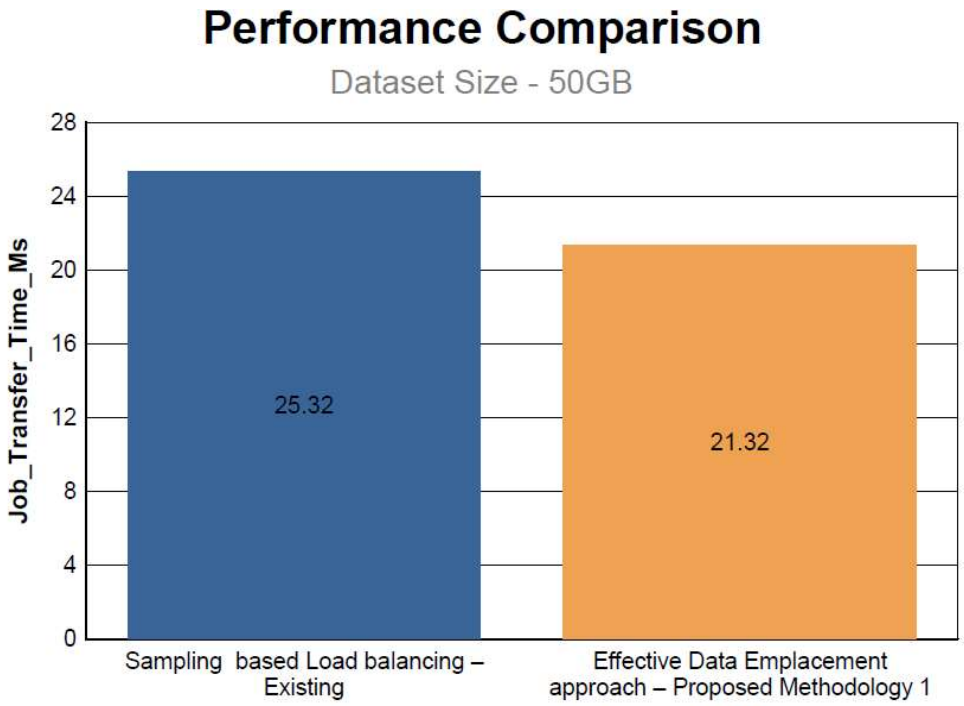

Figure 7: Performance Evaluation of the EDEC approach against SLB approach in terms of Job Transfer Time

Constraints:

Selection of the Data Instance = Objective Function (Fitness constraints)

For feature of the Feature vector

Feature = fitness of VM class 1

End for

For all feature based fitness

Probability of feature in VM class1 | VM class2

End for

The constraint is needed to calculate the number of slots required for each job on the particular VM class. The figure 8 represents the Performance Evaluation of the EDEC approach against SLB approach in terms of VM Utilization.

Figure 8: Performance Evaluation of the EDEC approach against SLB approach in terms of VM Utilization

The VM classification is non parametric method for generation of the VM classes for deadline constrained task towards execution in the effective resources. The feature selection is generated using linear discriminate analysis and fisherman discriminate model for available resource pool for the task execution. The resource feature generated is stored in the feature vector to make acts as decision data for VM class generation and towards processing the task.

Markov Property has been enabled to generate the set of different instance for the task feature selected from the unobserved instance for execution in the specified VM. The unobserved feature can be categorised as important feature based on the some correlation. It follows the both forward and backward procedure.

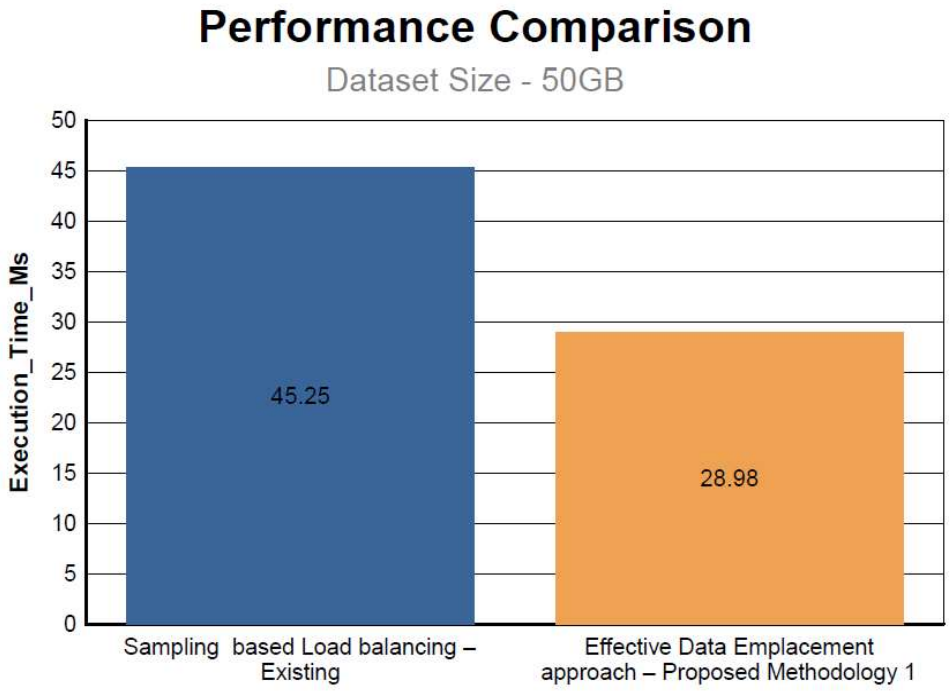

Figure 9: Performance Evaluation of the EDEC approach against SLB approach in terms of Execution Time

The VM feature transition probabilities from familiar to unobserved VM may be carried out using markov property analysis against slot computation time. The figure 9 represents the Performance Evaluation of the EDEC approach against SLB approach in terms of Execution Time.

Identifying the K value for Effective job Transfer

K value is determined through the number of the occurrence of the feature for the specified task on the available resource feature space, which can be termed as distinct features. It is capable of processing complex jobs. K value can act as class membership for VM slot and it can accept only task with specified deadlines. The resource feature space contains the all available resource instance and classifies new instance for the task execution based on the bandwidth availability measure. The feature vector (selection vector) is classified by assigning the label which is the most frequent among the *k* training samples nearest to that task initialisation point.

Criteria to select a Neighbour feature

Proposed redistribution approach is able to handle a large number of resource features and dynamic time evolving task. However, making a prediction based on the relevant feature for the job characteristics on thousands of feature instance is complex. Therefore, when the number of VM reaches a certain amount, a selection of the best neighbours has to be made on the selected neighbour of feature for the job. Two techniques, correlation-thresholding and best-n-neighbour, can be used to determine which neighbour to select in the feature vector for particular job.

The first technique selects only those neighbours correlation which is greater than a given threshold. The second technique selects the best *n* neighbours with the highest correlation in the feature similarity.

Most commonly used distance metric is for continuous feature is Euclidean distance and for discrete feature is hamming distance for task execution. The irrelevant feature still exists in the classifier for task execution. It produces the result with strong consistency.

Probabilistic methods are the most fundamental among all data classification methods for task scheduling. Probabilistic classification algorithms use statistical inference to find the best class of resources for the task execution. In addition to simply assigning the best class like other classification algorithms, probabilistic classification algorithms will yield a relating posterior probability of the test occurrence being an individual from each of the possible classes of resources.

The posterior probability is defined as the probability after observing the specific characteristics of the test instance of task. On the other hand, the prior probability is simply the fraction of training records belonging to each particular class on redistribution cycle with no knowledge of the test instance. After obtaining the posterior probabilities, decision theory is used to determine class membership for each new instance on task computing.

Basically, there are two ways in which one can estimate the posterior probabilities which effective value for job elapsed time on the set of resources. The figure 10 represents the Performance Evaluation of the EDEC approach against SLB approach in terms of Job Elapsed Time.

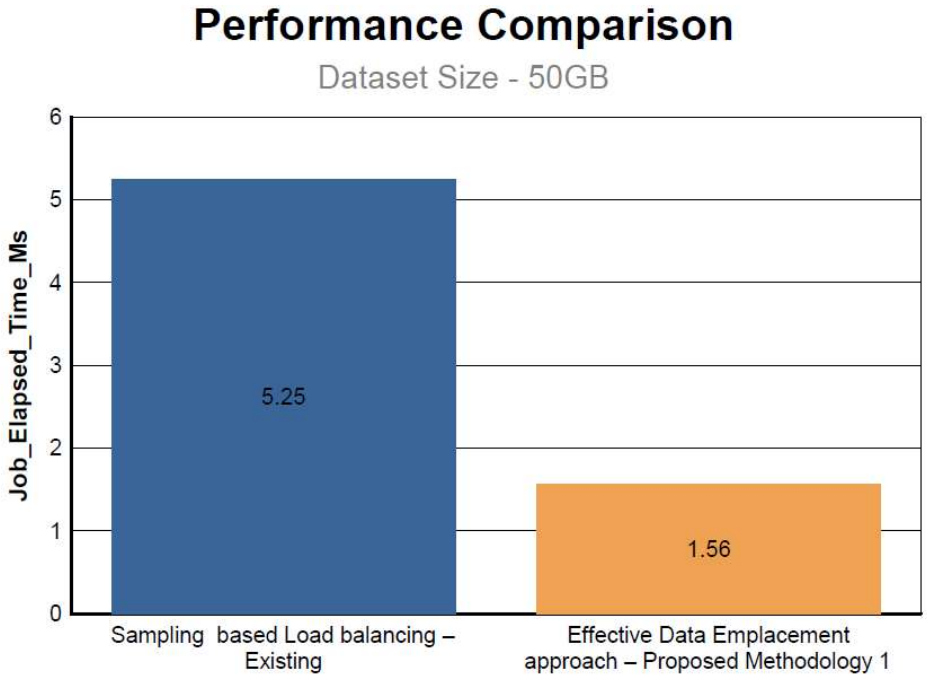

Figure 10: Performance Evaluation of the EDEC approach against SLB approach in terms of Job Elapsed Time

In the first case, the posterior probability of a particular class is estimated by determining the class-conditional probability and the prior class separately and then applying Bayes' theorem to find the parameters.

5. Conclusion

In this paper, a detailed architecture based on the data emplacement and redistribution has been carried out by implementation on heterogeneous Hadoop clusters. The proposed architecture is improving the performance on processing the large set of imbalance data through dynamic task partitioning and task regrouping on inclusions of scheduling and classification model. Those models have reduced the system complexity with high scalability on the cluster tasks. Clusters have been generated with least data instances.

Reference

1. Abhishek Verma, L Cherkasova, and R H. Campbell, *"ARIA: Automatic resource inference and allocation for mapreduce environments"*, in Proc. of the ACM International Conference on Autonomic Computing, pp. 235–244, 2011.
2. Abhishek Verma, L Cherkasova, and R H. Campbell, *"Two sides of a coin: Optimizing the schedule of mapreduce jobs to minimize their makespan and improve cluster performance"*, in International Symposium on MASCOTS Aug 2012.
3. Ahmed Amokrane, M.F. Zhani, R. Langar, R. Boutaba and G. Pujolle, *"Greenhead: Virtual Data Center Embedding Across Distributed Infrastructures"*, IEEE Transactions on Cloud Computing, Vol. 1, No. 1, pp. 36– 49, 2013.
4. Ali Ghodsi, M Zaharia, S Shenker and I Stoica, *"Choosy: Max-min fair sharing for datacenter jobs with constraints",* in Proc. of the ACM European Conference on Computer Systems, pp. 365–378, 2013.
5. Anastasios Arvanitis and Georgia Koutrika, *"Towards Preference-aware Relational Databases"*, in International Conference on Data Engineering (ICDE), IEEE Explore, pp. 426–437, 2012.
6. Andres Ferragut, F Kozynski and F Paganini, *"Dynamics of content propagation in BitTorrent-like P2P file exchange systems"*, in IEEE Conference on Decision and Control and European Control Conference, pp. 3116–3121, 2011.
7. Angus Roberts, R Gaizauskas, M Hepple et al, "The CLEF Corpus: Semantic Annotation of Clinical Text", In Proc. of Building and evaluating resources for biomedical text mining: workshop at LREC, pp. 625- 629, 2008.
8. Ashish Thusoo, J Sarma et al, *"HIVE: A Warehousing Solution over a Map-Reduce Framework"*, in Proc. of VLDB Endowment, Vol. 2, No. 2, pp. 1626-1629, 2009.
9. Avinash Lakshman and P Malik, *"Cassandra: structured Storage System on a P2P Network"*, in Proc. of the ACM Symposium on Parallelism in Algorithms and Architectures, 2009.

10. Avishan Sharafi and Ali Rezaee, *"Adaptive Dynamic Data Placement Algorithm for Hadoop in Heterogeneous Environments"*, Journal of Advances in Computer Engineering and Technology, Vol. 2, No. 4, pp. 17-30, 2016.
11. Azza Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Rasin and A. Silberschatz, *"HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads"*, in Proc. of the VLDB Endowment, Vol. 2, No. 1, pp. 922-933, 2009.
12. Balaji Palanisamy, A Singh and L Liu, *"Cost-Effective Resource Provisioning for MapReduce in a Cloud"*, IEEE Transactions on Parallel and Distributed Systems, Vol. 26, No. 5, pp. 1265-1279, 2014.
13. Bandar Alhaqbani and Colin Fidge, *"Privacy-preserving electronic health record linkage using pseudonym identifiers",* in 10th International Conference on e-health Networking, Applications and Services, pp. 108-117, 2008.
14. Bei Guan, Jingzheng Wu, Yongji Wang and Samee U. Khan, *"CIVSched: A Communication-Aware Inter-VM Scheduling Technique for Decreased Network Latency between Co-Located VMs"*, IEEE Transactions on Cloud Computing, Vol. 2, No. 3, pp. 320–332, 2014.
15. Bernhard Riedl, Veronika Grascher, Stefan Fenz, Thomas Neubauer, *"Pseudonymization for improving the Privacy in E-Health Applications"*, in Hawaii International Conference on System Sciences, IEEE Explore, pp: 1530-1605, 2008.
16. Bhaskar Vishnu Vardhan.Ch and Pallav Kumar Baruah, *"Improving the Performance of Heterogeneous Hadoop Cluster",* in Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC), IEEE Explore, 2016.
17. Bikash Sharma, R Prabhakar, S-H Lim et al, *"MROrchestrator: A Fine-Grained Resource Orchestration Framework for MapReduce Clusters*", in Proc. of IEEE International Conference on Cloud Computing, 2012.
18. Bikash Sharma, T Wood and C R. Das, *"HybridMR: A Hierarchical MapReduce Scheduler for Hybrid Data Centers"*, in Proc. of IEEE International Conference on Distributed Computing Systems, 2013.
19. Bo Mao, H Jiang, S Wu and L Tian, *"Leveraging Data Deduplication to Improve the Performance of Primary Storage Systems in the Cloud"*, IEEE Transactions on Computers, Vol. 65, No. 6, pp. 1775-1788, 2016.
20. Bo Zhang and F Zhang, *"An efficient public key encryption with conjunctive-subset keywords search"*, Journal of Network and Computer Applications, Vol. 34, No. 1, pp. 262–267, 2011.