IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

ANALYSIS OF DYNAMIC DATA ALLOCATION & REDISTRIBUTION MODEL IN CLOUD ENVIRONMENT

[1]S Annapoorani, [2]Dr. B Srinivasan

[1]Assistant Professor, [2]Associate Professor

[1,2]Department of Computer Science,

sannapoorani@gascgobi.ac.in, 2 srinivasanb@gascgobi.ac.in

[1,2] Gobi Arts & Science College, Gobichettipalayam, India

Abstract: Dynamic Opportunistic Data Allocation and Redistribution model for Heterogeneous Hadoop Clusters for different level of constraints to represent its interaction on the data in adoption to the virtualisation. The importance of the proposed methodology is to minimize the completion length (i.e., makespan) of a set of MapReduce jobs. The current Hadoop only permits static slot configuration, i.e., fixed numbers of map slots and reduce slots all through the lifetime of a cluster. Dynamic configuration may lead to high system resource utilisations as well as less completion length. It is considered as effective schemes which use slot ratio between map and reduce tasks as a tunable knob for reducing the makespan of a given set leveraging the workload information of recently completed jobs, schemes dynamically allocates resources (or slots) to map and reduce tasks. The corresponding constraints have been placed to that particular node for effective data allocation to the specified streaming data. Further it eliminates the data duplication occurring in the data clusters. In addition incremental clustering eliminates the inconsistent instance considered as an outlier would be processed effectively.

Index Terms **– Data Allocation, MapReduce, Redistribution, data duplication**

I. INTRODUCTION

MapReduce is a paradigm for parallel big data processing through utilising open source implementation Apache Hadoop. On incorporation of cloud system, MapReduce cluster has to launch on the cloud[1]. It helps to set the optimal values for those parameters and affords to adjust a basic system parameter with the goal to improve the performance of a batch of MapReduce jobs. The primary goal of the new mechanism is to improve the completion time (i.e., the makespan) of a batch of MapReduce jobs while retain the simplicity in implementation and management. The proposed model is to automate the slot assignment ratio between map and reduce tasks in a cluster as a tunable knob for reducing the makespan of Map Reduce jobs. The Workload Monitor (WM), Slot Assigner (SA), Data Redistribution and Data Deduplication are the major components introduced in the model.

II. HADOOP FILE SYSTEM

Hadoop is primarily employed for job scheduling and resource management in the file system. Hadoop framework uses fixed numbers of map slots and reduces slots on each node throughout the lifetime of a cluster Hadoop which is configured with a large set of system parameters and provides the flexibility to customise the cluster on the file system. Hadoop cluster incorporates a single master node and multiple slave nodes. The master node runs the Job Tracker routine which is responsible for scheduling jobs and coordinating the execution of tasks of each job. Each slave node runs the Task Tracker daemon for hosting the execution of Map Reduce jobs. The figure 1 represents the Process Flow of the MapReduce Architecture. Important parameter controlling the Hadoop cluster for DODAR model is as follows,

JobTracker

It periodically collects the execution time information of recently finished tasks and estimates the present map and reduces workloads in the cluster. An effective dynamic data allocation approach should tune the slot assignments such that the execution times of map and reduce phases can be well balanced and the makespan of a given set can be reduced.

Slot Assigner

The primary function of this parameter is to estimate and decide the slot ratio between map and reduce tasks for each slave node.

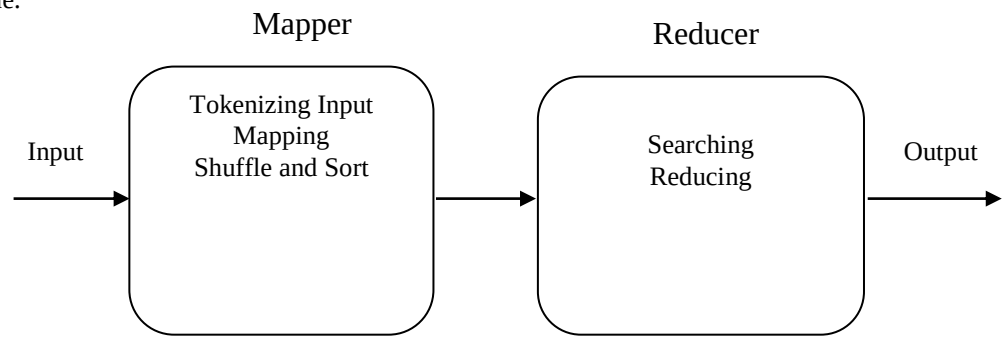

Figure 1: Process Flow of MapReduce Architecture

In order to diminish the makespan of a batch of jobs, more resources (or slots) should be allocated to map tasks if it have map intensive jobs.

Scheduler

The map and reduce phases of jobs could be better pipelined under need based schedulers, and subsequently the makespan is reduced. On the other hand, a basic change in such slot configurations is not sufficient.

III. DESIGN OF THE PROPOSED METHODOLOGY

In this section, Dynamic Opportunistic Data Allocation and Redistribution model has been implemented in Hadoop framework. Hadoop needs to choose the job of the available slot (either a map slot or a reduce slot). In this model, dynamic configuration has been set to the slot configurations for each individual node to reduce the makespan of a batch of jobs. The notation used for slot configuration of Hadoop cluster for batch processing of job with map and reduce has been defined in the Table 1.

Table 1: Notation for Dynamic Data Allocation for Hadoop Clusters

Notation	Description
S_m , S_r	Number of Map cluster Slots and Number of Reduce cluster Slots
$n_m(i)$, $n_r(i)$	Number of Map and Reduce task of job i
$t_m(i)$, $t_r(i)$	Average Map Execution Time of job i and Average Reduce Execution Time of job i
J	Jobs

Hadoop cluster comprising of k nodes has received a group of n jobs for processing. In that set of job is represented by Eq (6.1),

$$J = \{j1, j2. . . jn\} \quad \text{..... Eq (1)}$$

Each job j1 is configured with $n_m(i)$ Map tasks and $n_r(i)$ Reduce tasks. The total slots number in the Hadoop cluster is equal to S, and let S_m , S_r be the number of Map slots and Reduces slots, respectively. In a Hadoop cluster, makespan of multiple jobs also depends on the job scheduling algorithm using machine learning principle which is coupled with solution towards allocating the Map and Reduce slots on each node.

IV. WORKLOAD MONITOR

The proposed model depends on prior knowledge of workload information. Workload can be derived from job profiles, training phase, or other empirical settings. In this module, we devise a new model that estimates the workload during the job execution without any prior knowledge. Let w^l_m and w^l_r represents the remaining workload of a map or reduce phase. In other words the summation of execution time of the unfinished map or reduce tasks. The model track the map/reduce workloads of running jobs, but not the jobs waiting in the queue. Basically, the workload is calculated as the multiplication of the number of remaining tasks and the average task execution time of a job. In particular, when a map or reduce task is done, the present workload information should be updated. The execution time of each completed task is already collected and answered to the JobTracker in current Hadoop frameworks. One pass algorithm can able to calculate the average of task execution times of the different jobs which incur very low overheads on both time and memory space.

Algorithm 1: Workload Monitor

Step 1: Validate If (map task of job j(i) is finished)
Then
Update the normal execution time of a map task $t_m(i)$
$\mathbf{w'_m (i) = t_m(i) * n'_m(i)}$

Step 2: Validate If (reduce task of job j(i) is finished)
Then
Update the normal execution time of a reduce task $t_r(i)$
$\mathbf{w'_r (i) = t_r(i) * n'_r(i)}$

The slots assignment is progressively changed, which affects the per task execution time in practice. Assigning more slots to one type of tasks may cause the conflict on a specific framework resource and lead to an increased execution time of each following task in the same type. The task assignment in Hadoop depends on the tasktracker and jobtracker. TaskTrackers report slots occupation situation to the Job Tracker.

V. DATA REDISTRIBUTION

The data redistribution model exploits the workload characteristics of the data on either replication or erasure codes to distribute data among multiple cloud storage providers. Replication-based scheme eliminates the redundant data for the large file or else use of the indexing structure to incorporate the changes. It distinguishes a large file from a small file is nontrivial as it sensitively depends on a file size threshold. The large files contribute to a disproportionally large storage capacity and thus the associated cost has been managed by inverted index. The degree of data replication for file system metadata and small files is resilient against the outages and failures. The figure 2 shows the data distribution among multiple cloud servers. During the service unavailable period, all the write/update operations are performed as usual. The large files are reconstructed using the erasure-code redundancy.

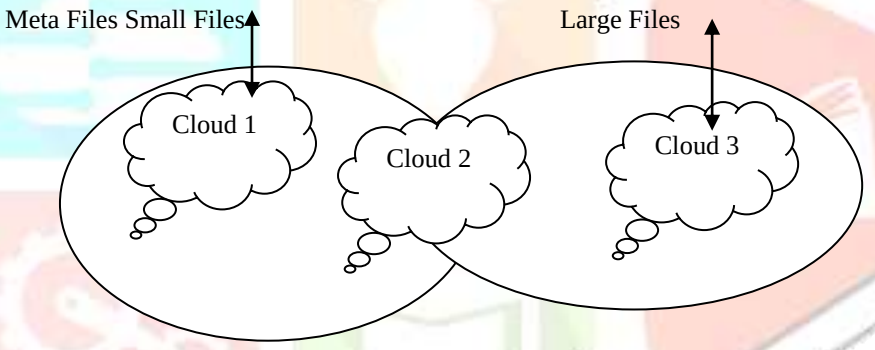

Figure 2: Data Distribution among Multiple Cloud Servers

VI. RESULTS & DISCUSSIONS

In this section, it is analysed the performance of Dynamic Opportunistic data Allocation and Redistribution for Heterogeneous Hadoop Clusters on the specified dataset using the various metrics has been detailed. The primary importance of methodology is to minimize the completion length (i.e., makespan) of a set of Map Reduce jobs. It is achieved using the classifier.

The most well-known named as Bayes classifier, which is known as a generative model has been utilised to reduce the execution time in the scheduler. Another probabilistic approach is to directly model the posterior probability, by learning a discriminative function that maps an input resource feature vector directly onto a class label on the task execution. Table 2 provides the Performance Values of the Dynamic Opportunistic Data Allocation and Redistribution model.

Table 2: Performance Values of the Dynamic Opportunistic Data Allocation and Redistribution model

Technique	Job Transfer Time in ms	Job Elapsed Time in ms	VM Utilization in mb	Execution Time in ms
Adaptive Job Scheduling (AJS) of Parallel Jobs	22.56 ms	5.24 ms	10.25 mb	41.15 ms
Dynamic Opportunistic Data Allocation and Redistribution (DODAR) model	20.32 ms	0.85 ms	7.86 mb	26.12 ms

MapReduce cluster to set the optimal values for those parameters and affords to adjust a basic system parameter with the goal to improve the performance of a batch of MapReduce job. The decision tree provides the effective computation to mitigate the pitfalls in the scheduling.

Decision Trees

Decision trees create a hierarchical partitioning of the data, which relates the different partitions at the leaf level to the different classes. The hierarchical partitioning at each level is made with the utilisation of a split criterion. The split rule may either use a condition (or predicate) on a single property, or it might contain a condition on multiple traits. The figure 3 provides the Performance Evaluation of the DODAR approach against AJS approach in terms of Job Transfer Time.

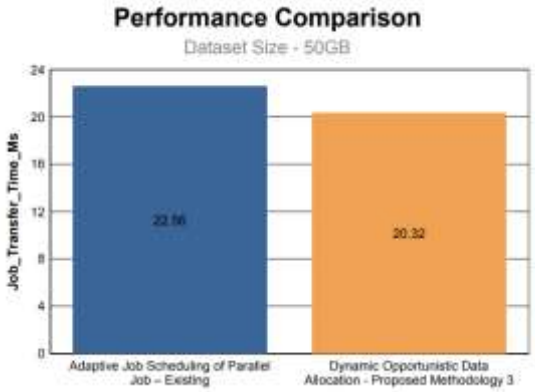

Figure 3: Performance Evaluation of the DODAR model against AJS approach in terms of Job Transfer Time

The former is referred to as a univariate split, whereas the latter is referred to as a multivariate split. The overall approach is to try to recursively split the data so as to maximize the discrimination among the different classes over different resources nodes predicted in the VM. The discrimination among the different classes is maximized, when the level of skew among the different classes in a given node is maximized to achieve best efficiency. The dynamic data allocation approach should tune the slot assignments such that the execution times of map and reduce phases can be well balanced and the makespan of a given set can be reduced according on allocated period. The figure 4 provides the Performance Evaluation of the DODAR approach against AJS approach in terms of Execution Time.

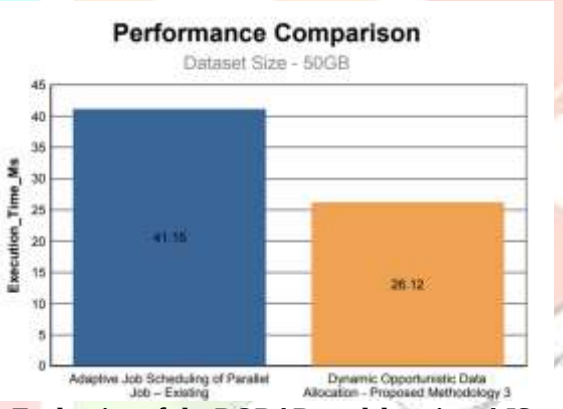

Figure 4: Performance Evaluation of the DODAR model against AJS approach in terms of Execution Time

In order to reduce the makespan of a batch of jobs, more resources have been assigned to the map. In instance-based learning, the first phase of constructing the job analysis model for obtaining the job characteristics. The evolution task is directly related to the VM instances of the particular resource in order to generate a task classification model. These classification methods are referred to as lazy classification methods, because they wait for knowledge of the VM instance in order to create a locally optimised model, which is specific to the VM instance. The advantage of such methods is that they can be directly tailored to the particular VM instance, and can avoid the information loss associated with the incompleteness of any classification model. Slot configuration on the VM instance pipelined under priority based schedulers. Figure 5 represents the Performance Evaluation of the DODAR approach against AJS approach in terms of Job Elapsed Time.

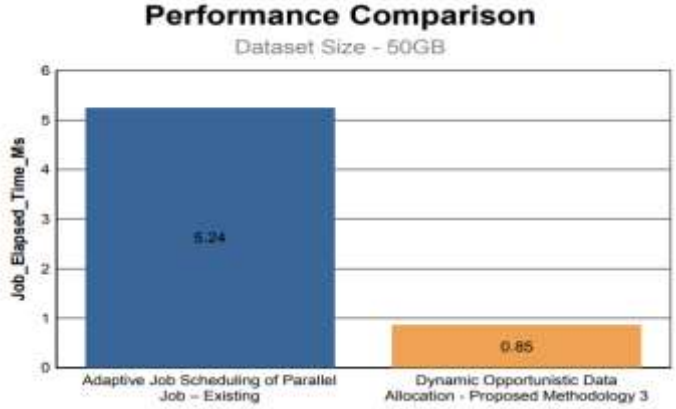

Figure 5: Performance Evaluation of the DODAR model against AJS approach in terms of Job Elapsed Time

Slot configuration of Hadoop cluster for batch processing of job with map and reduce provides the effective task scheduling. In addition, it provides the ability to identify and separate the data into structured format. Data classification on task is carried out using machine learning algorithm and business intelligence techniques to allocate deadline constraints to the VM instance to obtain the execution with less computation time. The data classification model to the evolution task process with the following

- Identifying and keeping frequently used resources in disk/memory cache
- Data arranging depends on content/file type, size and time of information
- Sorting for security reasons by arranging data into confined, public or private data types

The dynamic configuration of data parameters for the Hadoop clusters to the evolution task is computed with fitness measures on considering the job parameters. The optimal function estimates the present map and reduces workloads in the cluster. It also provides estimation to decide and adjust the slot ratio between maps and reduce tasks.

Figure 6: Performance Evaluation of the DODAR model against AJS approach in terms of VM Utilization

The figure 6 provides the Performance Evaluation of the DODAR approach against AJS approach in terms of VM Utilization for evolution task. Dynamic slot assignments in heterogeneous environments are carried out with two primary criteria's. One is splitting criteria and the second is pruning technique. Decision tree splits the attributes based on different univariate criteria. Each of them has different theoretical approach as a splitting mechanism. The second step of decision tree is pruning technique. Employing firmly stopping criteria tends to generate small and under-fitted decision trees.

VII. CONCLUSION

In this work, the dynamic data allocation and redistribution model for heterogeneous Hadoop clusters has been designed and implemented considering multiple constraints. The multiple constraints job tracker, slot assigner, workload monitor and job scheduler towards the job information collection. The incremental data acquisition or incremental crawling techniques in order to improve data collection performance using MapReduce architecture has been analysed in detail. The processing framework for the MapReduce paradigm in the big data distributed data management has been done. Incremental processing framework performs the key value pair level computation for new data updates to the already established clusters instead of performing re-computation to the entire scale of data. It eliminates the mean error and complexity related issues.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] Brian Cho, M Rahman, T Chajed, I Gupta, C.Abad, N Roberts and P Lin, *"Natjam: Design and Evaluation of Eviction Policies for Supporting Priorities and Deadlines in Mapreduce Clusters"*, in Proc. Of the Annual Symposium on Cloud Computing, pp. 1–17, 2013.

[13] Carlo Mastroianni, Michela Meo and Giuseppe Papuzzo, *"Probabilistic Consolidation of Virtual Machines in Self-Organizing Cloud Data Centers"*, IEEE Transactions on Cloud Computing, Vol. 1, No. 2, pp. 215–228, 2013.

[14] Jiong Xie, *"Improving Performance of Hadoop Clusters"*, A dissertation submitted to the Graduate Faculty of Auburn University, Auburn, Alabama, 2011.

[15] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz and I. Stoica, "Improving mapreduce performance in heterogeneous environments," in Proc. of USENIX Symposium on Operating Systems Design and Implementation, pp. 29-42, 2008.

[16] Smitha Sundareswaran, Anna Squicciarini and Dan Lin, *"Ensuring Distributed Accountability for Data Sharing in the Cloud"*, in IEEE transaction on Dependable and Secure computing , Vol.9, No. 4, 2012.