FIDOOP: PARALLEL MINING OF FREQUENT ITEMSETS USING CURA AND IWD ALGORITHM

Raja.R ,
Assistant Professor,
Velammal Institute Of Technology,
Chennai,India.

Sathya Narayanan.R.K,
Velammal Institute Of Technology,
Chennai,India.

Vasanth.P ,
Velammal Institute Of Technology,
Chennai,India.

Dinesh Babu.P,
Velammal Institute Of Technology,
Chennai,India.

Abstract-The growing trend in cloud computing is the combination of Big Data and Big Data analytics that has been driven with rapid evolution of data center technologies towards more cost-effective solutions. Existing system Cura, for provisioning cost-effective MapReduce services in acloud. First, Cura is designed to provide a cost-effective solution to efficiently handle MapReduce production workloads that have a significant amount of interactive jobs. Second, unlike existing services that require customers to decide the resources to be used for the jobs, Cura leveragesMapReduce profiling to automatically create the best cluster configuration for the jobs. While the existing models allow only a per-job resource optimization for the jobs, Cura implements a globally efficient resource allocation scheme that significantly reduces the resource usage cost in the cloud. Third, Cura leverages unique optimization opportunities when dealing with workloads that can withstand some slack.This project presents a cost-effective resource management framework called (IWD) Intelligent Waterdrop algorithm that aim to minimize the infrastructure cost in the datacentre. IWDis a new technique that was used to solve job shop scheduling problem in Cloud.The IWD may results into better performance than existing workflow scheduling algorithms.

Keywords:MapReduce , Intelligent waterdrop , Frequent Data Mining,CURA.

I.INTRODUCTION

FREQUENT itemsets mining (FIM) is a core problem in association rule mining and sequence mining.Existing parallel mining algorithms for frequent itemsets lack a mechanism that enables automatic parallelization, load balancing, data distribution, and fault tolerance on large clusters. As a solution to this problem, we design a parallel frequent itemsets mining algorithm called FiDoop using the MapReduce programming model. To improve FiDoop's performance, we develop a workload balance metric to measure load balance across the cluster's computing nodes.This proposed algorithms are used to dynamically create an optimal schedule to finish the submitted jobs in a High Performance Computing environment showing promising results which achieves significantly lower resource usage costs for the jobs.The resource management techniques include cost-aware resource provisioning, VM aware scheduling and online virtual machine reconfiguration.

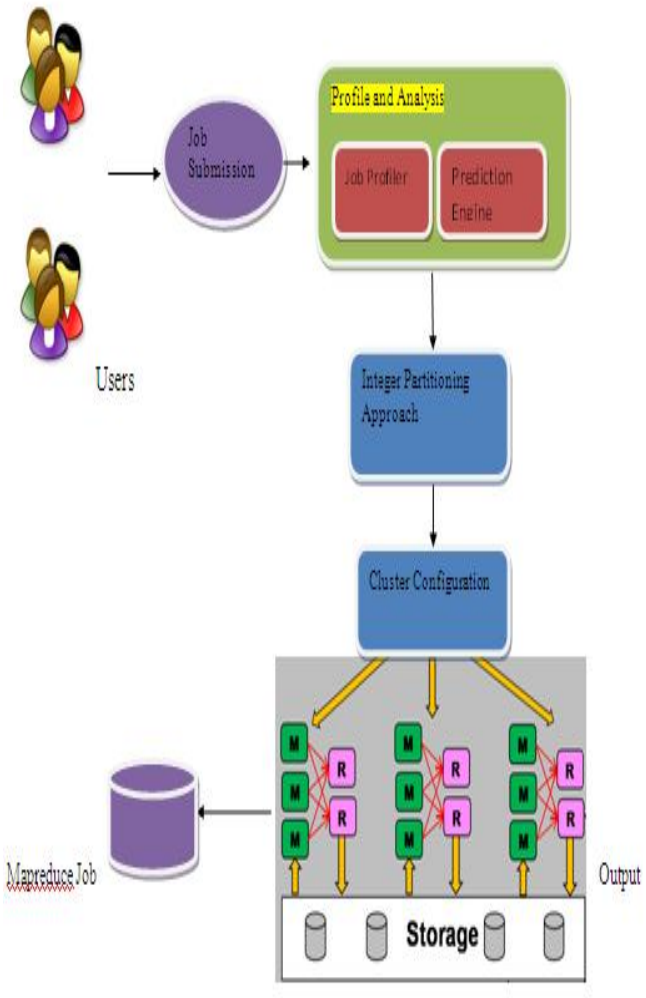
2. RELATED WORK

MapReduce is a promising parallel and scalable programming model for data-intensive applications and scientificanalysis. A MapReduce program expresses a large distributedcomputation as a sequence of parallel operations on datasets of key/value pairs. A MapReduce computation has two phases,namely, the Map and Reduce phases. The Map phase splits the input data into a large number of fragments, which areevenly distributed to Map tasks across the nodes of a cluster to process.Hadoop—one of the most popular MapReduce implementations is running on clusters where Hadoopdistributed file system (HDFS) stores data to provide highaggregate I/O bandwidth. At the heart of HDFS is a singleNameNode—a master server that manages the file systemnamespace and regulates access to files. The Hadoop runtimesystem establishes two processes called JobTracker andTaskTracker. JobTracker is responsible for assigning andscheduling tasks; each TaskTracker handles Map or Reduce tasks assigned by JobTracker**.**CURAleverages the mapreduce profiling that automatically creates the best cluster configuration for the jobs. Cura resource management techniques include cost-awareresource provisioning, VM aware scheduling and online virual machine reconfiguration.A price matching formation is proposed and it consistof a BackPropagation Neural Network based price prediction algorithm and price matching algorithm. In each round of the auction, Consumers and providers submit their bidding and asking prices**.** The cost-aware relevant challenges of WFS in cloud computing are classified based on Quality of Service (QoS) performance, system functionality and system architecture, which ultimately result in a taxonomy set.

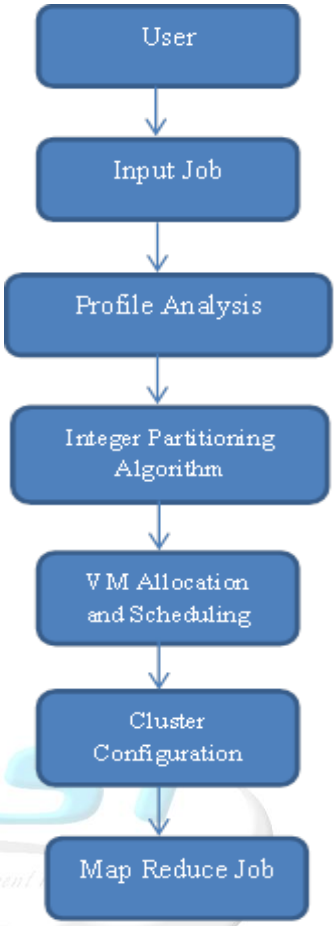
2.1 PROPOSED METHODOLOGY

The cluster configuration for the jobs which uses MapReduce profile are automatically created by IWD and leverages the deadlines by delaying the execution of certain jobs and allows cloud providers to optimize its global resource allocation. Also, IWD uses a unique secure VM allocation that ensures fast response time. The resource management techniques include cost-aware resource provisioning, VM aware scheduling and online virtual machine reconfiguration. This

proposed algorithms are used to dynamically create an optimal schedule to finish the submitted jobs in a High Performance Computing environment showing promising results.achieves significantly lower resource usage costs for the jobs.The IWD may results into better performance than existing workflow scheduling algorithms. Very less work has been done in exploring intelligent water drops based algorithm for scheduling workflow applications and minimizing cost.Our experimental results using Facebook-like workload traces show that our techniques lead to more than 80 percent reduction in the cloud compute infrastructure cost with upto 65 percent reduction in job response times.The resource management techniques include cost-aware resource provisioning, VM aware scheduling and online virtual machine reconfiguration.



Flow Diagram:

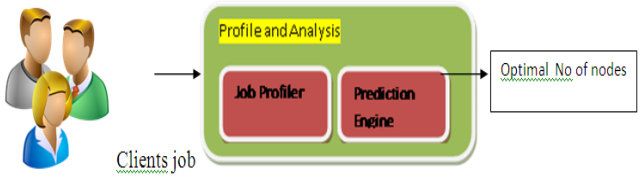


2.2 MODULES

There are four modules in our proposed system, Job Profiling and Analysis Phase, Integer Partitioning Approach, VM allocation and Scheduling, Cluster configuration for Mapreduce operation.

2.2.1.JOB PROFILING AND ANALYSIS

Users simply submit their jobs (or composite job workflows) and specify the required service quality with the respective deadline. Profile and analysis phase develops a performance model for the job in order to be able to generate predictions for its performance later on. When making scheduling decisions, performance predictions in terms of execution time are made based on the input dataset size, VM types, cluster sizes and job parameters. The output of the profiling phase is the optimal number of nodes that can perform the job. This optimal number of nodes is the basis for the cluster configuration for the Mapreduce job.



2.2.2 INTEGER PARTITIONING APPROACH

The Number no of nodes is processed to the Integer Partitioning Algorithm. An Integer partitioning algorithm that generates all possible combination of numberswhich result to the output of the profiling phase. The nodes are hence allocated toconfigure a cluster. In this approach, initially the switch has the number of active nodes available inthe network. Within thisavailable nodes only cluster configuration is going to made.

2.2.3 VM ALLOCATION AND SCHEDULING

There will be no of nodes to form a cluster. Once the optimal no of nodes chosen, the nodes which is active inside the switch is chosen for cluster configuration. In that nodes which has the lowest CPU utilization will get more priority than other nodes. So that VM is allocated for the Mapreduce jobs.

2.2.4 CLUSTER CONFIGURATION FORMAP REDUCE OPERATION

Once VM allocated, Cluster configuration has made to do the Mapreduce Job. Mapreduce job is performed using HadoopMapreduce Framework.

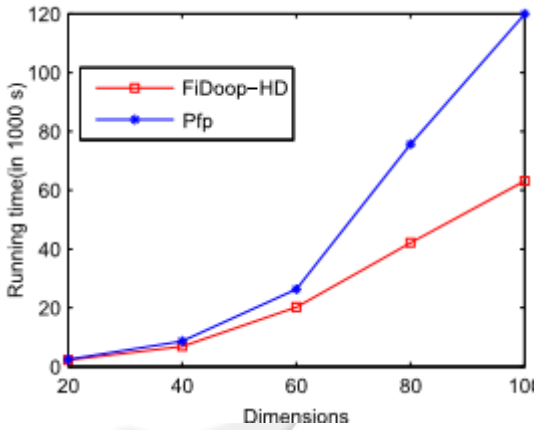
3. ALGORITHM

The proposed IWD algorithm,

Require:ResourceStatus[1..n]=3,
NodeList[1…n* m]=3,AvailableNodes=n*m,
allResourceSelected[1..n]=0
ifnumberOfNodes>AvailableNodes then
print "Cluster cannot be allocated due to insufficient Resource"
else
//check for all the permutation of the number of nodes in decreasing order such that we can get the permutation free in the mapping list
whilemappingList!=NULL do
break
// We get the best resources that can be used for our cluster, now we need to identify the nodes.
// Decrement the value of each resource by the number of nodes used on that resource, change the color accordingly
for i=1 to 4 do
ifNodeList[ResourceNumber*4 +i]==0
then
NodeList[ResourceNumber*4 + i]=1
end if
end for
allResourcesSelected[next]
end for
Call this function for all the request.
end while
end if

3.1 HIGH DIMENSIONAL OPTIMIZATION

The aforementioned analysis confirms that if the length of itemsets to be decomposed is large, the decomposition cost will exponentially increase. In this section, we conduct experiments to investigate the impact of dimensionality on FiDoop. It also presents an optimization algorithm called FiDoop-HD for high-dimensional data processing. When it comes to mining frequent itemsets, varying dimensionality leads to a wide range of itemset lengths. Our algorithm needs to decompose each itemset generated by pruning infrequent items for each transaction.

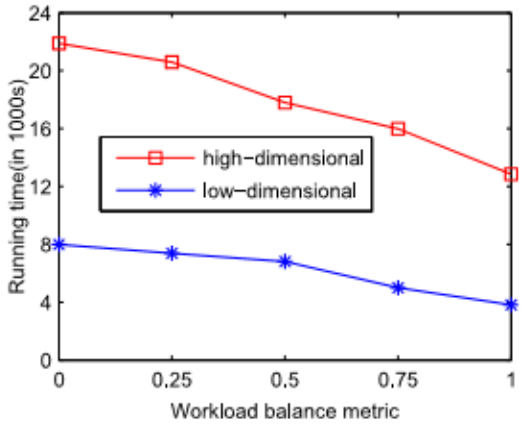


4.EXPERIMENTAL RESULTS

We evaluate the performance of FiDoop on our in-house Hadoop cluster equipped with 16 data nodes. Each node has an Intel Pentium-4 3.0 GHz processor, 512 MB main memory, and runs on the Ubuntu 10.10 operating system, on which Java JDK 1.6.0_37 and Hadoop 1.1.1 are installed. All the data nodes in the cluster have Gigabit Ethernet network interface cards connected to Gigabit ports on the switch; the nodes can communicate with one another using the secure shell protocol.

4.1 LOAD BALANCING

In this group of experiments, we measure FiDoop's workload balance metric on a low- and high-dimensional datasets. In the case of high-dimensional dataset, we test our algorithms using the celestial spectral dataset. Recall that our analysis shows that the load balancing mechanism of the third MapReduce job substantially improves the performance of FiDoop. Section V-A formally introduces the workload balance metric. We measure FiDoop's workload balance metric when the celestial spectral dataset is used as an input.



I. Load-balancing performance of FiDoop.

5. CONCLUSION

To solve the scalability and load balancing challenges in the existing parallel mining algorithms for frequent itemsets, we applied the MapReduce programming model to develop a parallel frequent itemsets mining algorithm called FiDoop. FiDoop incorporates the frequent items ultrametric tree or FIU-tree rather than conventional FP trees, thereby achieving compressed storage and avoiding the necessity to build conditional pattern bases. FiDoop seamlessly integrates three MapReduce jobs to accomplish parallel mining of frequent itemsets. The third MapReduce job plays an important role in parallel mining; its mappers independently decompose itemsets whereas its reducers construct small ultrametric trees to be separately mined. We improve the performance of FiDoop by balancing I/O load across data nodes of a cluster.

6.REFERENCES

[1] M. J. Zaki, "Parallel and distributed association mining: A survey," IEEE Concurrency, vol. 7, no. 4, pp. 14–25, Oct./Dec. 1999.

[2] I. Pramudiono and M. Kitsuregawa, "FP-tax: Tree structure based generalized association rule mining," in Proc. 9th ACM SIGMOD Workshop Res. Issues Data Min. Knowl. Disc., Paris, France, 2004, pp. 60–63.

[3] R. Agrawal, T. Imieli´nski, and A. Swami, "Mining association rules between sets of items in large databases," ACM SIGMOD Rec., vol. 22, no. 2, pp. 207–216, 1993.

[4] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," DataMin. Knowl. Disc., vol. 8, no. 1, pp. 53–87, 2004.

[5] S. Kotsiantis and D. Kanellopoulos, "Association rules mining: A recent overview," GESTS Int. Trans. Comput. Sci. Eng., vol. 32, no. 1, pp. 71–82, 2006.

[6] R. Agrawal and J. C. Shafer, "Parallel mining of association rules," IEEE Trans. Knowl. Data Eng., vol. 8, no. 6, pp. 962–969,DEC 1996

[7] A. Schuster and R. Wolff, "Communication-efficient distributed mining of association rules," Data Min. Knowl.Disc., vol. 8, no. 2, pp. 171–196, 2004.

[8] M.-Y. Lin, P.-Y.Lee, and S.-C. Hsueh, "Apriori-based frequent itemset mining algorithms on MapReduce," in Proc. 6th Int. Conf. Ubiquit. Inf. Manage. Commun. (ICUIMC), Danang, Vietnam, 2012, pp. 76:1–76:8. [Online]. Available: http://doi.acm.org/10.1145/2184751.2184842

[9] D. Chen et al., "Tree partition based parallel frequent pattern mining on shared memory systems," in Proc. 20th IEEE Int. Parallel Distrib. Process.Symp. (IPDPS), Rhodes Island, Greece, 2006, pp. 1–8.

[10] L. Liu, E. Li, Y. Zhang, and Z. Tang, "Optimization of frequent itemset mining on multiple-core processor," in Proc. 33rd Int. Conf. Very Large Data Bases, Vienna, Austria, 2007, pp. 1275–1255.