Improved Modeling Method for Rebalanced Mapreduce Processing in Public Cloud's Optimal Resource Provisioning

Sathishkumar. N. S
PG-Information Technology
Jayam College of Engg & Tech
Dharmapuri, India.

Siva Kumar. C
Associate Professor
Jayam College of Engg & Tech
Dharmapuri, India.

***Abstract*— Due to the massive improvement in the usage of data's in the real world, it becomes more burdens to handle and process it effectively. The Map reduce is the one of the more developed technology which is used to handle and process the big data/largest tasks. Map reduce is used to partition the task into sub partitions and map those partitions into the machines for processing. This process need to be done by the considering the minimization of cost and meeting deadline to improve the user satisfaction. In the previous work, CRESP approach is used which focus on allocating the map reduces tasks in the machine with the consideration of reduction of cost and deadline. However this method does not concentrate on the skew and stragglers problem which can occur while handling the largest task. In our work, we try improve the performance of resource allocation strategy by considering the skews and stragglers problem in mind. This problem of skews and stragglers are handled by introducing the partitioning mechanism. The partitioning mechanism will improve the failure of task allocation strategy.**

Index Terms—Cloud Computing, Hadoop, Map Reduce, Micro Partitioning.

I. INTRODUCTION

In the Development of Cloud Computing, Sensor Networks, Grid Computing the huge amount of Datasets could be collected from the users, Applications and Environment. For Instance, nowadays users are capable of storing huge amount of Datasets in a Datacenter, where we go for usage of the Big Data. Map Reduce is a technique which is used to work with the Big Data. Big Data is nothing but a collection of huge datasets which is difficult to process using on-hand database management tools or traditional data processing applications.

On the other hand, Cloud analysts in most Organizations such as research institutions, Government institutions have no opportunity to access more private Hadoop/Map Reduce clouds. Based on the requirements Amazon introduced Elastic Map Reduce which runs on Hadoop Clusters.

Apache Hadoop is the open source Software for Storing and Processing Large amounts of datasets on Clusters of Hardware. Requirements for running Hadoop cluster on Public and a Private cloud varies promptly. First for each job clusters could be allocated starting from the virtual nodes to make use of "pay-as-you-use" economic Cloud model.

It is difficult to maintain a constant Hadoop cluster as private Hadoop clusters because data processing requests are generally entering in continuously. Thus on demand Hadoop cluster has become a convenient choice for many users. Such an on-demand cluster is created for a long-running job, where no multi-user or multi-job resource competition happens within the cluster. Second, it is responsibility of user to set the number of virtual nodes for the Hadoop cluster.

The Optimization of the Resource Provisioning could involve about 2 factors. Provisioning the virtual machine nodes which could consider about the monetary cost and finishing of the job which considers about the time cost. Resource provisioning could be done based on the cost which depends on the time required for resources to be used. It is complicated to use other constraints such as deadline or monetary budget to reduce the cost for usage of resources. We propose a method to help users to decide about Allocation of running Map Reduce programs in public clouds. This method, Micro Partitioning is used to distribute the work load among the nodes.

However, running a Hadoop cluster on top of the public cloud has different requirements from running a private Hadoop cluster. First, for each job normally a dedicated Hadoop cluster will be started on a number of virtual nodes to take advantage of the ''pay-as-you-use'' economical cloud model. Because users' data processing requests are normally coming in intermittently, it is not

economical to maintain a constant Hadoop cluster like private Hadoop cluster owners do. Meanwhile, current virtualization techniques allow a virtual cluster to be provisioned or released in minutes.

Thus, on-demand Hadoop clusters have become an appropriate choice for most users who have ad-hoc Hadoop jobs. Typically, such an on-demand cluster is created for a specific long-running job, where no multi-user or multi-job resource competition happens within the cluster.[2] second, it is now the user's responsibility to set the appropriate number of virtual nodes for the Hadoop cluster. The optimal setting may differ from application to application and depend on the amount of input data. An effective method is needed to help the user make this decision.

II. THE CRESP APPROACH

This method, Cloud Resource Provisioning (CRESP) for Map Reduce Programs is based on the time cost model. Considering the time cost model and the parameters, providing the users could solve optimization problems. We analyze cost model in terms of Input datasets, Specific complexity of the application and the resources that are available for the system. In this approach we consider the combination of the white-box and machine learning approaches. Map Reduce programs have different time and the logical complexity. The cost functions may be differing from application to application.

A. ANALYZING MAP REDUCE TASKS

Map Reduce is the Combination of parallel and distributed processing. Large number of data's that could be processed which are defined as clusters. Reduce phase is executed after the execution of Map phase. The execution of Map and Reduce programs is done by using the concepts of Map/Reduce slot and Map/Reduce task. Slot is the unit used for running the tasks by allocating the resources that are available. Fixed number of slots could be allocated based on the capacity of the system.

Hadoop consists of four Components, Name Node which is the heart of Hadoop file system, Data Node which stores blocks of data's and retrieves them, Task Tracker is responsible for the allocation of the number of Map slots and Reduce slots, and Job Tracker is responsible for the allocation of the client jobs.

B. CALCULATION OF COST FOR MAP TASK

Map phase consists of the three stages Map, Read and Sort. First we consider the calculation of cost for the input that is given to the Map phase. The input that may be in the form of data blocks i (b) that could be from the local or remote disk. Second we consider the Map function f (b) that is given by the user. After that sorting of the data's Om (b) could be done and then the output will be in the form of (Key, Value) pairs which could be give to the reduce phase.

$\Phi_m = i(b) + f(b) + s(om(b),R)+\varepsilon_m$

C. CALCULATION OF COST OF REDUCE TASK

Reduce phase consists of the three stages Shuffle, Merge Sort, Reduce and Write Result. The execution that could be done in parallel in the reduce phase. In the execution of the reduce task the amount of data is proportional to the number of keys that are assigned. The keys given by the Map phase is distributed equally to the reduce Tasks. In the Shuffle stage the each and every reduce tasks will be assigned for its shares which could be given as k/R and the amount of data that could be given is

$b_R = M * o_m(b) * k/R$

In the Merge Sort stage the data's that could be simply merged because the sorting of the data's are done at the earlier stages.

The cost of Merge Sort could be given as it is ms (b_R) which depends upon b_R.

Learning the Model:

For the calculation of the cost function we concentrate on the number of input variables, M, m, R and then the parameter $\beta_{i..}$ First we randomly sample the variables which are considered for testing. Second collect the time and cost for the map and reduce tasks by setting the variables (M, m, R). m is nothing but the number of samples that are considered. Third the regression model is considered which is applicable for learning of the models with the transformed variables such as,

$X_1 = m/M$, $x_2 = MR/m$, $x_3 = R/m$, $x_4 = (M \log M)/R$, $x_5 = M/R$, $x_6 = M$, $x_7 = R$.

III. MICRO PARTITIONING

Micro Partitioning is the key technique is to run a large number of reduce tasks, splitting the map output into many more partitions than reduce machines in order to produce smaller tasks. These tasks are assigned to reduce machines in a "just-in-time" fashion as workers become idle, allowing the task scheduler to dynamically mitigate skew and stragglers. With large tasks, it can be more efficient to exclude slow nodes rather than assigning them any work. By assigning smaller units of work, jobs can derive benefit from slower nodes.

Micro-tasks can also help to mitigate skew. Increasing the number of hash partitions generally leads to smaller, more-even partitions because there is a lower probability of collision in the key's partitioning function. These tasks are assigned to reduce machines as workers become idle, allowing the task scheduler to dynamically mitigate skew and stragglers.

Running many small tasks lessens the impact of stragglers, since work that would have been scheduled on slow nodes is only small which can now be performed by other idle workers. By assigning smaller units of work, jobs can derive benefit from slower node.

DATASETS

We use four types of testing Datasets to test the samples. The datasets could be 1000 words randomly chosen is used as samples from the dictionary. Another dataset is samples generated from the Page Rank Program, next dataset is packages from the Hadoop.

The samples used are

Word Count: Used to calculate the number of words that are present in the input file that is given.

Tera Sort: Sorting of the data's that are done and then given for reducers.

Page Rank: The ranking for the accessing of the websites are given.

Table Join: The joining of the word count, Tera Sort is done.

The procedure for implementing the micro partitioning technique is as follows:

Step 1: Take the input samples

Step 2: Store the input samples in tire

Step 3: Build the two-level tire

Step 4: Count the occurrence of each prefix

Step 5: Using cut-point algorithm determine the cut-points (split points)

Step 6: Split points are obtained by dividing the sum of counter value by number of partitions+1

Cut points = sum of counters/number of partitions+1

Step 7: Using cut points send the keys to appropriate reducers

if (key<cutpoint1)

Send key to reducer 1

else if (key>=cut-point1&&key<cut-point2)

Send key to reducer2

else if (key>=cut-point2&&key<cut-point3)

Send key to reducer3

else

Send to the finished reducer

Step 8: Determine if there is any slow running node by comparing the performance of each node with other

Step 9: If there is any such node move the data that is processing on that node to the free node.

For inputs containing few distinct keys, fine-grained partitioning may result in many empty reduce tasks that receive no data. These empty reduce tasks are unproblematic, since they can be easily detected and ignored by the scheduler. Jobs with few distinct keys are the most sensitive to partitioning skew, since there may not be enough other work to mask the effects of a straggling task created by a key collision in the hash partitioning function.

For jobs with large numbers of distinct keys, the impact of key collisions is small. Micro-tasks do not specifically address record size or computational skew, but these types of skew must be handled in an application-specific manner.

Micro-tasks help to achieve a more even assignment of indivisible units of work; techniques for reducing the units of work, such as map-side combining, are complimentary and orthogonal to this approach. Straggler is one the common performance issue in Map reduce system. Stragglers can impact job performance because a job's completion time is determined by the completion time of its final task. Several techniques have been developed to address stragglers. Here a new approach called micro partitioning is used for avoiding skew and stragglers during the reduce phase.

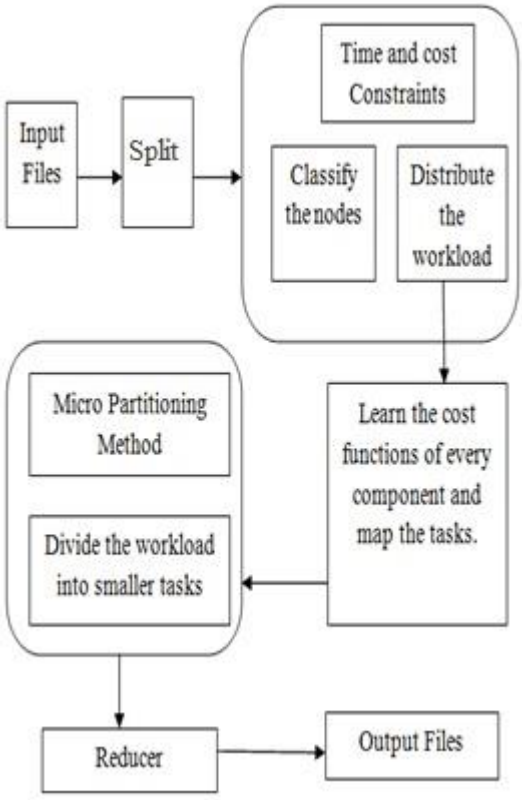

Fig.1 Architecture Diagram

A. *NODE CLASSIFICATION METHOD*

In the node classification algorithm, we use the comprehensive criterion to represent the data processing speed. This criterion should be the comprehensive processing capacity facing all memory intensity, IO intensity and CPU intensity jobs. In a heterogeneous environment, cluster usually contains nodes with different processing capacity, which means the speed of the processed data. Classification is done based on the processing capacity. There are two purposes of node classification with their processing capacity: one is to optimize the data distribution in order to improve the data locality; the other is to improve the evaluation accuracy of the task remaining time in heterogeneous environment.

In order to get the differences between nodes with their capacities, we divide the nodes to different levels. In the cluster, we take the slowest data processing speed nodes as the first level and the level factor is 1. To quantize the node's computing capacity simply by running a group of specific tasks. On a given cluster, we set each node with one Map slot and one Reduce slot. And all the map tasks are feed by the same input data. Then the benchmark jobs run on the cluster the node classification algorithm enables us to decide the nodes' level by their computing capacity. The data distribution strategy is that the size of each node's data is in proportion to the node's level.

Optimized Resource Allocation:

With the cost model we are now ready to find the optimal settings for some problems.

To find the best resource allocation for three typical situations: 1) with a certain limited amount of monetary budget; 2) with a time constraint; 3) and the optimal tradeoff curve without any constraint. In the following, we formulate these problems as optimization problems based on the cost model. In all the scenarios we consider, we assume the model parameters βi have been learned with sample runs in small scale settings. For the simplicity of presentation, we assume the simplified model T2 is applied.

Since the input data is fixed for a specific Map Reduce job, M is a constant. We also consider all general Map Reduce system configurations have been optimized via other methods and fixed for both small and large scale settings.

In Map Reduce, the job's execution progress includes Map and Reduce stage. So, the job's completion time contains Map execution time and Reduce execution time. In view of the differences between Map and Reduces code, we divide the scheduling progress into two stages, namely Map stage and Reduce stage. In the aspect of the task's scheduling time prediction, the execution time of Map and Reduce is not correlative; their execution time depends on the input data and function of their own. Therefore, in this work the scheduling algorithm sets two deadlines: map-deadline and reduce-deadline.

In order to get map-deadline, we need to know the Map task's time proportion on the task's execution time. In a cluster with limited resources, Map slot and Reduce slot number is decided. For an arbitrary submitted job with deadline constraints, the scheduler has to schedule reasonable with the remaining resources in order to assure that all jobs can be finished before the deadline constraints. According to map-deadline, we can acquire the current map task's slot number it needs; and with reduce-deadline, we can get the current reduce task's slot number it needs.

In the task scheduling, to get map-deadline, we need to know the Map task's time proportion on the task's execution time. In a cluster with limited resources, Map slot and Reduce slot number is decided. For an arbitrary submitted job with deadline constraints, the scheduler has to schedule reasonable with the remaining resources in order to assure that all jobs can be finished before the deadline constraints. The scheduling strategy of MSTD is based on $\boldsymbol{\delta^m_j}$ and $\boldsymbol{\delta^r_j}$. The minimum Map and Reduce slot number required of job *J* can be denoted as δmj and $\boldsymbol{\delta^r_j}$ respectively. The symbol reflects that $\boldsymbol{\delta^m_j}$ Map tasks should be scheduled at present in order to meet job *J's* map-deadline, as well as to meet the reduce-deadline $\boldsymbol{\delta^r_j}$ Reduce tasks should be scheduled.

In the scheduling process, we take $\boldsymbol{\delta^m_j}$ and $\boldsymbol{\delta^r_j}$ as the basic criteria of priority allocation. At the beginning of the job be submitted, there is no data available, so the scheduler can't estimate the required slots or the completion time of tasks. In this case, the job's precedence is over than the others. In some scenarios, jobs may have already missed their deadline.

IV. RESULT ANALYSIS

Finally, the performance of the existing and the proposed approaches were illustrated and evaluated. The existing system based on cost analysis function, the optimized resource allocation in done. In the proposed system, the micro-partitioning method is also considered before allocating the tasks into the resources. Compared to existing method efficient scheduling is used in the proposed system and achieve high throughput.

A. *TIME CONSUMPTION*

Figure 2 compares the consumption of time in CRESP Approach with the Micro partitioning Mechanism. The time consumption is reduced in the Micro Partitioning mechanism.

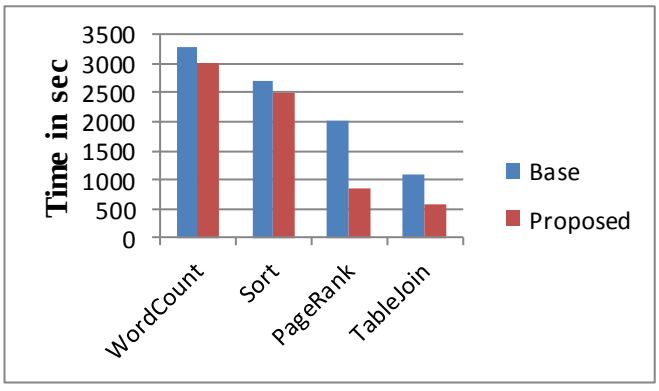

Fig.2 Time Consumption in ms

B. *MEMORY CONSUMPTION*

Figure 3 shows that usage of the size of memory is reduced in Micro Partitioning Mechanism when compared with the CRESP Approach.

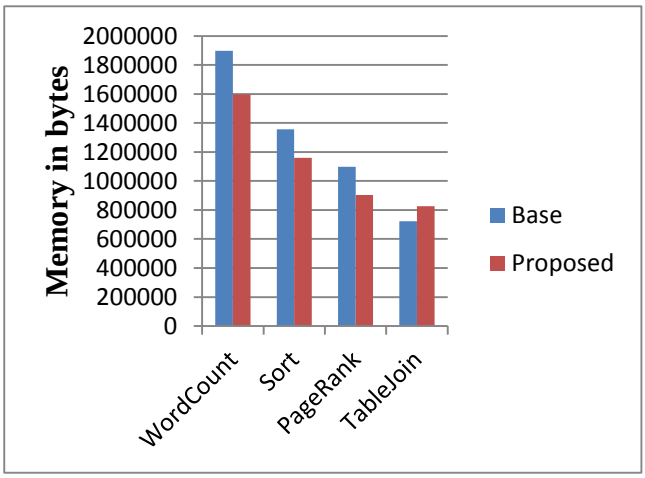

Fig.3 Memory consumption in bytes

C. COST CONSUMPTION

Figure 4 shows that cost is reduced in Micro Partitioning Mechanism when compared with the CRESP Approach.

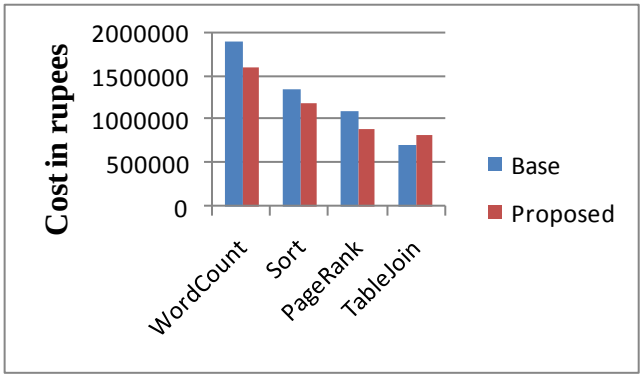

Fig.4 Cost in rupees

V. CONCLUSION

In this work, we study the components in Map Reduce processing and build a cost function that explicitly models the relationship among the amount of data, the available system resources (Map and Reduce slots), and the complexity of the Reduce function for the target Map Reduce program. The model parameters can be learned from test runs. Based on this cost model, we can solve a number of decision problems, such as the optimal amount of resources that can minimize the monetary cost with the constraint on monetary budget or job finish time. To improve the load balancing for distributed applications, Micro partitioning techniques is used. By improving load balancing, Map Reduce programs can become more efficient at handling tasks by reducing the overall computation time spent processing data on each node. In addition to that we use Map Reduce Task Scheduling algorithm for cost and time constraints for the efficient scheduling. For that we use node classification method and distribute the workload among the nodes according to the node capacity.

After that a micro partitioning method is used for applications using different input samples. This approach is only effective in systems with high-throughput, low-latency task schedulers and efficient data materialization.

VI. FUTURE WORK

In the future, we would like to implement the proposed task scheduler architecture and perform additional experiments to measure performance using straggling or heterogeneous nodes. We also plan to investigate other benefits of micro-tasks, including the use of micro-tasks as an alternative to preemption when scheduling mixtures of batch and latency-sensitive jobs.

REFERENCES

[1]. Candan KS, Kim JW, Nagarkar P, Nagendra M, Yu R (2010) RanKloud: scalable multimedia data processing in server clusters. IEEE MultiMed 18(1):64–77
[2]. Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrws M, Chandra T, Fikes A, Gruber RE (2006) Big table: a distributed storage system for structured data. In: 7th UENIX symposium on operating systems design and implementation, pp 205–218.
[3]. Dean J, Ghemawat Dean S (2008) Map Reduce: simplified data processing on large clusters. Common ACM 51:107–113.
[4]. Ghemawat S, Gobi off H, Leung S-T (2003) The Google file system. In: 19th ACM symposium on operating systems principles (SOSP).
[5]. Jiang W, Agrawal G (2011) Ex-MATE data intensive computing with large reduction objects and its application to graph mining. In: IEEE/ACM international symposium on cluster, cloud and grid computing, pp 475–484.
[6]. Jin C, Vecchiola C, Buyya R (2008) MRPGA: an extension of Map Reduce for parallelizing genetic algorithms. In: IEEE fourth international conference on escience, pp 214–220.
[7]. Kavulya S, Tany J, Gandhi R, Narasimhan P (2010) An analysis of traces from a production Map Reduce cluster. In: IEEE/ACM international conference on cluster, cloud and grid computing, pp 94–95.
[8]. Krishnan A (2005) Grid BLAST: a globus-based high-throughput implementation of BLAST in a grid computing framework. Concur Compute 17(13):1607–1623.
[9]. Liu H, Orban D (2011) Cloud Map Reduce: a Map Reduce implementation on top of a cloud operating system. In: IEEE/ACM international symposium on cluster, cloud and grid computing, pp 464–474.
[10]. Hsu C-H, Chen S-C (2012) Efficient selection strategies towards processor reordering techniques for improving data locality in heterogeneous clusters. J Super computes 60(3):284–300.