Budget based dynamic slot allocation for MapReduce clusters

[1]S. Janani, [2]G.R.Anantha Raman

[1]P.G. Scholar,[2]Assistant Professor,
Department of Computer Science and Engineering,
Adhiyamaan College of Engineering, Hosur,
Tamilnadu, India

Abstract: **MapReduce is one of the programming models for processing large amount of data in cloud where resource allocation is one of the research areas since it is responsible for improving the performance of Hadoop. However the resource allocation can be further improved by focusing on a set of mechanisms, that includes the budget based HFS algorithm where the fast worker node is identified first based on the budget, and then Dynamic Hadoop Slot Allocation (DHSA) is used for allocation of the slots, and Longest Approximate Time to End (LATE) is used to handle the speculative tasks that are identified. Finally the overall performance is compared with the existing mechanisms using the Hadoop simulation tool.**

Keywords: **LATE, DHSA, MapReduce, Speculative Tasks.**

I. INTRODUCTION

Cloud computing has dramatically transformed the way many critical services are delivered to customers for example, the Software, Platform, and Infrastructure as a Service paradigms, and at the same time has posed new challenges to data centers. The result is a complete new generation of large scale infrastructures Therefore, data analytics is one of the more prominent fields that can benefit from next generation data center computing, building new models to develop such applications, and mechanisms to manage them, are open challenges. An example of a programming model especially well-suited for large-scale data analytics is MapReduce, introduced by Google in 2004.

Map Reduce workloads usually involve a very large number of small computations executing in parallel. High levels of computation partitioning, While it was originally used primarily for batch data processing, its use has been extended to shared, multi-user environments in which submitted jobs may have completely different priorities. The Map and Reduce function are mainly used to perform filtering and summary over the data that is stored across the clusters. Figure 1 shows how the Map and Reduce operation takes place.

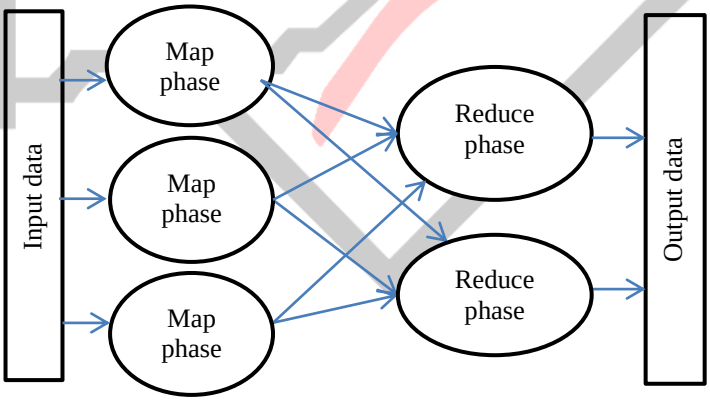

Fig.1 Map and Reduce operation

First, Since the jobs that are scheduled to run the tasks on the TaskTracker is not a fast worker node moreover running the jobs in such a TaskTracker exceeds the budget of the user .Second ,resources are statically configured by the administrator as map and reduce slots to complete the map and reduce operations the resources go underutilized and major performance goals like deadline and cost are not met which in turn increases the cost for the users as well as the bandwidth. Third, there is a possibility of slow running tasks known as stragglers due to unavoidable runtime contention for processor, memory and other resources that causes delay in the job execution. Moreover the straggled tasks are run in the form of backup tasks in the same node. That increases the computation time thereby making execution of the current node slower.

To overcome this challenges we proposed some mechanisms that mainly focuses on identifying the fastworker based on the budget by using the budget driven algorithm, further resource utilization is maximized in the fastworker node by using the dynamic allocation technique to meet their deadlines and straggled tasks are also run as backup task in the same node there by

maximizing the Data locality which in turn preserves the time and cost. The combination of these techniques along with the Hadoop Fair Scheduler (HFS) results in enhanced HFS which is implemented in the JobTracker and examined whether the overall performance of the Hadoop is improved.

II. RELATED WORK

In literature there was research work on the performance optimization of MapReduce jobs related to resource utilization are discussed as follows. Jorda Polo in 2011 proposed a utility based by allocating slots or resources dynamically to the map and reduce tasks until the deadlines of the job are met[1].

Joel Wolf in 2011 allocates the slots based on priority of the jobs. where first the scheme allocates the minimum no of slots to each job and there may be some idle slots that can be allocated to other jobs based on priority[2].

Xiaowei Wang proposed a method in which the resources are allocated according to the cluster load based on the weighting technique where the resource requirement is changing based on the completion rates on the map and reduce tasks[4].

Balaji Palanisamy in 2011 proposed a locality scheduling technique to increase the performance of map and reduce phases and how the network traffic is reduced in order to maximize datalocality. However this technique is different from conventional MapReduce that uses the separate cloud that deals with data and vm placement[3].

B.Thirumala Rao in 2013 used a resource configurator that adjusts the CPU resources without Violating the completion time by dynamically increasing or decreasing the VM to maximize the datalocality and resource configurator is used that allocates the required no of map and reduce slots to complete the task[5].

Mohammad Hammoud in 2013 proposed a LARTS(Locality Aware Reduce Task Scheduler) where the slots are allocated dynamically rather than statically and at the sametime it handles the speculative tasks and maximizes data locality[6].

Jian Tan in 2013 discussed about the joint optimization of MapReduce that causes resource starvation and unfavourable data locality due to delay scheduling and then he proposed a coupling scheduler that couples the map and reduce jobs to mitigate the starvation moreover random peek scheduling and wait scheduling to optimize the data locality by making reduce tasks run close to the intermediate slave nodes.

Xiangping Bu, in 2013 discussed about the inference that causes the performance degradation of map and reduce tasks and proposed a inference scheduling algorithm that predicts when the tasks slows down and further adaptive delay scheduling algorithm is proposed that improves the delay scheduling algorithm by adjusting delay intervals that are ready to run jobs and mainly focuses on server locality[7].

Xuanjia Qiu in 2014 an efficient scheduling mechanism that enables efficient utilization resources and to reduce the task outsourcing cost, by using a time-slotted system where each time slot, makes the users to complete their task with in time.

Zhenhua Guo in 2014 discussed about the unutilized idle slots of the tasks and proposed a stealing method to allocate the idle slots to the pending tasks by using fair scheduler and capability scheduler for resource allocation. speculative execution problem is also overcome by using BASE(Benefit Aware Speculative Execution) where the speculative tasks are launched only when they are expected to complete earlier than the original tasks[11].

R.manopriya in 2014 proposed a Johnson algorithm where the job tracker schedules the jobs into different tasks and allocate the tasks to slots when the pool contains excess of the slots are allocated to other pools that are dependent hence this approach minimizes the execution time[10].

Abishek verma in 2014 proposed a deadline based Hadoop scheduler that allocates the number of map and reduce slots to each tasks within completion time and it allocates and deallocates the resources which are unused to someother jobs to complete their tasks moreover the jobs are ordered based on the"Earliest Deadline Policy"[9].

Yang Wang and Wei Shi proposed a budget driven algorithm by using global greedy budget and multiknapsack problem that overcomes the problem of parallel transactions across the clusters by allocating the slots dynamically [13].

III. PROPOSED WORK

The proposed system overcomes the problem of resource utilization by identifying the fast worker based on their budget by using budget driven algorithms .Second, dynamically allocating resources to complete the task with in the deadline by using the Deadline and dynamic slot allocation and also making the stragglers to run in the fast worker node. Figure 2 shows the overall architecture of the proposed system

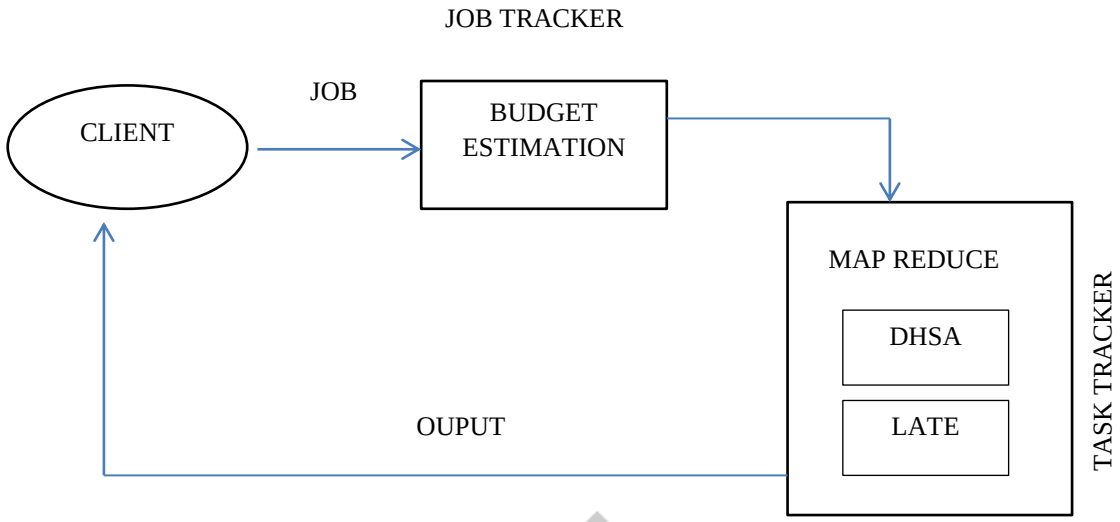

Fig. 2 Overview of Budget Driven Algorithm

1. Budget Estimation

The JobTracker identifies the TaskTracker based on the total estimation time ,cost ,local and nonlocal maptasks, ,inputs from its logs. Once the fast worker node is identified from the trace file, the job scheduler schedules the jobs by using budget based driven algorithms.

However the scheduled tasks are run on the fast worker node by making use of the map and reduce slots that are available. The estimated budget based on time is shared and allocated to each and every map and reduce phases of the job.

2. Dynamic Hadoop Slot Allocation (DHSA)

MapReduce suffers from a underutilization of the various slots because the variety of map and reduce tasks varies over time, leading to occasions wherever the amount of slots allotted for map/reduce is smaller than the amount of map/reduce tasks. Our dynamic slot allocation policy is predicated on the observation that at totally different amount of your time there could be idle map (or reduce) slots, because the job income from map phases to reduce part. We will use the unused map slots for those full reduce tasks to boost the performance of the MapReduce work, and contrariwise. as an example, at the beginning of MapReduce work computation, there'll be only computing map tasks and no computing reduce tasks, i.e., all the computation work lies within the map-side. In that case, we will create use of idle reduce slots for running map tasks. Simply permitting reduce tasks to use map slots needs configuring every map slot to require additional resources, which will consequently reduce the effective variety of slots on every node, inflicting resource under-utilized throughout runtime.

However the dynamic slot allocation takes place inside and outside the pool based on the scenarios
Case 1: When *NM* <=*SM* and *NR* <= *SR*, the map tasks are run on map slots and reduce tasks are run on reduce slots, i.e., no borrowing of map and reduce slots.
Case 2: When *NM* >*SM* and *NR* <*SR*, we satisfy reduce tasks for reduce slots first and then use those idle reduce slots for running map tasks.
Case 3: When *NM* <*SM* and *NR* >*SR*, we can schedule those unused map slots for running reduce tasks.
Case 4: When *NM*> *SM* and *NR* >*SR*, the system should be in completely busy state, and similar to (1), there will be no movement of map and reduce slots. Thereby, the whole dynamic slot allocation flow is that, Whenever a heartbeat is received from a compute node, we first compute the total demand for map slots and reduce slots for the current MapReduce workload.

3. LATE

While processing the map and reduce tasks if the progress rate is found to be lesser than the original tasks then it is identified as a speculative task. The speculative task we should also only launch speculative tasks on *fast nodes* - not stragglers There are various reasons that cause stragglers, including faulty hardware and software mis configuration. We classify the stragglers into two:
Types, namely, *Hard Straggler* and *Soft Straggler*, defined as follows:
1) Hard Straggler: A task that goes into a deadlock status due to the endless waiting for certain resources. It cannot stop and complete unless we kill it.
2) Soft Straggler: A task that can complete its computation successfully, but will take much longer time than the Common tasks. For the hard straggler, we should kill it and run another equivalent task, or called a *backup task*, immediately once it was detected. In contrast, there are two possibilities between the soft straggler and its backup task:
(S1). Soft straggler completes first or the same as its backup task. For this case, there is no need to run a backup task.
(S2). Soft straggler finishes later than the backup task. We should kill it and run a backup task immediately.
To deal with the straggler problem, *speculative execution* is used in Hadoop. Instead of diagnosing and fixing straggling tasks, it detects the straggling task dynamically using heuristic. The identified speculative task is run as a backup in the fast worker node

in order to complete the tasks within the budget of each phase. Here the job tracker identifies the fast worker node also based on the availability of the free slots required to complete the tasks within the deadline.
The combination of the techniques results in the enhanced budget based HFS algorithm.

Budget driven algorithm

When jobj is added

Fetch FastWorker from logs of the JobTracker

When nonlocal maptask=0,computation time<=estimated time

Then

Deadline(D)= Computation time

For T=0 to T=D;T++

Loop

Case1:if the idle slots are available allocate it to the map task

Case2:when idle reduce slots are available allocate it to the reduce task

Case3:when case1 and case2 fails try reduce slots for pending tasks

LATE

Identifies the slow running tasks the progress rate(Tmr)

Where Tmr=D

Run the speculative task in the same node

End .However the computation cost is found to be less than the estimated cost due to the dynamic allocation slots.

5. Workload

Our experimental workload consists of a set of representative applications that are run concurrently. We can run the same application with different input datasets of varying sizes. A particular application reading from a particular set of inputs is called an application instance. Each application instance can be allocated varying number of map and reduce slots resulting in different job executions. The applications used in our experiments are as follows:

1. Word count: This application computes the occurrence frequency of each word in the Wikipedia article history dataset. We use three datasets of sizes: 32GB, 40GB and 43GB.

2. Sort: This applications sorts a set of records that
is randomly generated. The application uses identity map and identity reduce functions as the MapReduce framework does the sorting. We consider three in- stances of Sort: 8GB, 64GB and 96GB.

IV. SIMULATION

The simulation of the proposed is carried out with the help of Hadoop 0.19.0 with suitable workloads and performance evaluation is carried with comparison with the existing HFS algorithms that shows how the budget is reduced based on the time since the cost is directly proportional to the time.

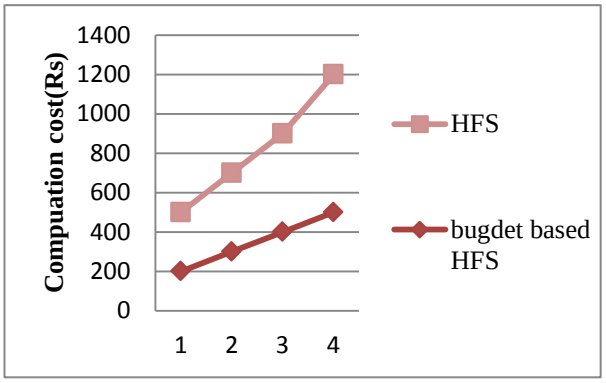

Fig 3 Budget based on different Datsets

This figure 3 how the performance of MapReduce is evaluated based on the different datasets like wordcount and sort.

V. CONCLUSION

The proposed algorithm increases the Hadoop performance by saving the cost for the users and organizations. Compared to the existing techniques the proposed techniques improve the performance by 30-40 percent. In future the budget driven HFS is enhanced by focusing on the energy saving of the clusters by switching off the idle nodes.

REFERENCES

.

[1] Jorda Polo, Claris Castillo, David Carrera, Yolanda Becerra, Ministry of Science and Technology of Spain and the European Union (FEDER funds)
[2] Joel Wolf, Andrey Balmin, Deepak Rajan," *CIRCUMFLEX*: A Scheduling Optimizer for MapReduce Workloads With Shared Scans**",** NSF grant 0910989.
[3] Balaji Palanisamy, Aameek Singh," Purlieus: Locality-aware Resource Allocation for MapReduce in a Cloud**",** ACM,2011
[4] Xiaowei Wang, Jie Zhang, Huaming Liao," Dynamic Split Model of Resource Utilization in MapReduce**",**ACM,2011
[5] B.Thirumala Rao, L.S.S.Reddy," Scheduling Data Intensive Workloads through Virtualization on MapReduce based Clouds",*Communications of the ACM*, 2011
[6] Mohammad Hammoud and Majd F. Sakr, Locality-Aware Reduce Task Scheduling for MapReduce, *Communications of the ACM*, 2011
[7] Xiangping Bu, Jia Rao, Cheng-Zhong Xu," Interference and Locality-Aware Task Scheduling for MapReduce Applications in Virtual Clusters",ACM,2012
[8**]** Balaji Palanisamy, *Member, IEEE*, Aameek Singh, *Member, IEEE* Ling Liu, *Senior Member, IEEE,"* Cost-effective Resource Provisioning for MapReduce in a Cloud",IEEE,2013
[9] Abhishek Verma, Ludmila Cherkasova," Deadline-basedWorkload Management for MapReduce Environments: Pieces of the Performance Puzzle",ACM 2013
[10**]** R.manopriya, C.P.Saranya," Optimal Resource Allocation and Job Scheduling to Minimise the Computation Time under Hadoop Environment", IJLTET, Vol. 3 Issue 3 January 2014
[11] Zhenhua Guo, Geoffrey Fox, Mo Zhou, Yang Ruan," Improving Resource Utilization in MapReduce", National Science Foundation under Grant No. 0910812.,2014
[12] Shanjiang Tang, Bu-Sung Lee, and Bingsheng He," DynamicMR: A Dynamic Slot Allocation Optimization Framework for MapReduce Clusters", IEEE Transactions on Cloud Computing, vol. 2, no. 3, July-September 2014
[13]. Yang Wang and Wei Shi," Budget-Driven Scheduling Algorithms for Batches of MapReduce Jobs in Heterogeneous Clouds", IEEE TRANSACTIONS ON CLOUD COMPUTING, 2014