Dynamic Load Aware Scheduler of Map Reduce Tasks for Cloud Environments

Adepu Sree Lakshmi, N. Subhash Chandra, M. Balraju

Abstract--- Most of the current day applications are data and compute intensive which led to invention of technologies like Hadoop. Hadoop uses Map Reduce framework for parallel processing of big data applications using the computing resources of multiple nodes. Hadoop is designed for cluster environments and has few limitations when executed in cloud environments. Hadoop on cloud has become a common choice due to its easy establishment of infrastructure and pay as you use model. Hadoop performance on cloud infrastructures is affected by the virtualization overhead of cloud environment. The execution times of Hadoop on cloud can be improved if the virtual resources are effectively used to schedule the tasks by studying the resource usage characteristics of the tasks and resource availability of the nodes. The proposed work is to build a dynamic scheduler for Hadoop framework which can make scheduling decision dynamically based on job resource usage and node load. The results of the proposed work indicate an improvement of up to 23% in execution time of the Hadoop Map Reduce applications.

Keywords--- Hadoop, Map Reduce Scheduler, Capacity Scheduler, Load Aware, Big Data.

I. INTRODUCTION

Technologies like Hadoop [1] enabled for large scale data processing using Map Reduce parallel computing framework Map Reduce. Map Reduce framework divides the job into two phases of execution namely Map and Reduce phase. The input data of the job is divided into multiple chunks and each chunk gets copied to nodes in accordance to specified replication factor. Hadoop Distributed File System (HDFS) module of Hadoop performs the distribution of input data to multiple nodes. HDFS is the distributed file system that acts like data management framework while Map Reduce is the processing framework of Hadoop.

Map task runs the user specified Map function on the chunk of input data copied to HDFS of the data node and generates the intermediate key value pairs. Reduce task processes these intermediate key value pairs with the user specified Reduce function and generate the final output. Hash function is used to partition the intermediate key value pairs from Map tasks to Reduce tasks. Map Reduce has become popularly used framework for parallel processing due to its features like scalability, automatic parallelism, fault tolerance mechanisms and simplicity.

Despite of its success in large scale data processing there are few limitations in the current implementations of Hadoop. Hadoop schedulers behave in a static way for allocation of tasks to the data nodes. Hadoop provides First in First out (FIFO), Fair and Capacity scheduler (CS) [5, 29] in the package. FIFO schedules the jobs in the order of job submission time and priorities.

Fair scheduler creates pools for multiple users and each pool is assured fair share of resources and more jobs are executed concurrently. Capacity Scheduler uses multiple queues/pools in a hierarchical way where each pool is guaranteed some fraction of physical resources in the cluster. All the schedulers provided in Hadoop package are static in nature where the scheduling decisions are not based on the resource requirements of the job [25]. The description of the schedulers available for Hadoop framework is provided in the paper [6].

Though Hadoop has good performance in static cluster environments, but when executed in virtual environments like cloud its performance is affected by the virtual machines(VMs) running on physical machine [3,30]. Cloud has now become a popular choice for infrastructure due to its flexible way of using the resources and pay as you use strategy. Executing Hadoop jobs on public cloud has become very common choice.

In cloud environment multiple virtual machines run on a physical machine. The performance of one virtual machine affects other virtual machines as the entire resource requests go through the hypervisor. If the resource usage of a Map task is studied then scheduling decision can be done based on the job resource usage and node resource availability so that tasks with dissimilar resource usages are scheduled on VMs running on one physical machine.

To overcome the limitations of static way of scheduling the tasks to data nodes running on virtual machines, a dynamic load aware scheduler is proposed which can dynamically schedule the Map/ Reduce tasks. When Map/Reduce tasks are scheduled on virtual machines, the resource usage behavior of jobs and resource availability of virtual machines are considered in the decision of scheduling of Map/ Reduce tasks of different jobs to achieve an optimized execution time.

A virtual machine executing on a node that is IO heavy is more suitable for a task which needs less amount of IO. The scheduling of Map and Reduce tasks is done in consideration with the resource characteristics of the virtual machines and the resource usage behavior of Map/Reduce task.

Virtual machine load characteristics include load on virtual machine in term of its CPU usage and IO usage, number of tasks currently running on the VM. The runtime

Manuscript received September 16, 2019.

Adepu Sree Lakshmi, Associate Professor, Geethanjali College of Engineering & Technology, Hyderabad, India.(e-mail: adepu.sreelakshmi@gmail.com)

Dr.N. Subhash Chandra, Professor, CVR College of Engineering.

Dr.M. Balraju, Professor, Swamy Vivekananda Institute of Technology. T.N, India.

of a task scheduled on a virtual machine is affected by the load on other virtual machines running on the same physical machine, as all the IO communication and CPU communication goes through the hypervisor, and these overheads increase as the number of VMs per bare-hardware node increases. Job characteristics include whether job is CPU intensive or IO intensive? All jobs do not use the resources in the similar way [4]. Few jobs like word count needs lot of CPU processing than the IO data transfer where as some other jobs like pi, grep needs lot of IO data transfer than CPU processing.

If such behavior of job can be analyzed during the execution of initial Map/Reduce tasks, that knowledge can be used in scheduling the remaining Map/Reduce tasks of the job to efficiently schedule the tasks based on the resource availability of the node.

Job finishing time plays a crucial role in cloud computing environments as the users should pay per use and most of the cloud models follow hourly model for pricing. This dynamic loadaware scheduler of Map Reduce tasks can reduce the makespan of all big data applications submitted by users to a cloud thereby reducing the rent to be paid for cloud infrastructure.

Task scheduling was explored by many researchers and different strategies were proposed. Optimized and efficient way of scheduling were aimed with different goals like scheduling based on interference [26], reducing partitioning skew [24], cost effective way to create best cluster configuration [28] profit oriented [31], prediction based and locality aware task scheduling [22].

Comparison of different scheduling algorithms can be found in [27]. Few works were proposed to design schedulers which can schedule the Map Reduce tasks based on different criteria related to resource usage. Dyscale [7], a Map Reduce job scheduler for heterogeneous multicore processers was designed by Feng Yan et al. Dyscale tries to configure cores statically into categories like slow core and fast core and uses the information in scheduling.

This work is based on the assumption that the slots are fixed as Map slot and Reduce slot which is not suitable for Hadoop MRv2. K. Arun Kumar et al. proposed context aware scheduler (CASH) [8] where scheduling decisions are done in consideration with job characteristics and node characteristics. CASH classifies the nodes statically as computational or IO good based on the performance of CPU and IO.

The work was demonstrated on a simulation environment and not on the realistic Map Reduce environment. They neglected the IO bound shuffle phase where Map outputs are moved to the Reducers. It is completely a static approach of classifying the node which does not consider the current load on the node.

Prism [9] is a fine grained resource aware scheduling for Map Reduce. It performs resource aware scheduling at the level of task phases. This work differentiates the resource usage at different phases and accordingly allocates the resources required at each fine grain.

This paper also did not consider the node load levels during the resource estimation. Most other works [10, 11, 12, 13, 14] have configured the nodes based on the resource availability but all these works categorized the nodes statically as either the node as CPU good or IO good based on the performance of the node. This work differs in the way the nodes are classified dynamically based on the load on the virtual machines whether the node is ready for IO heavy or CPU heavy tasks based on the estimated load on the node.

The proposed dynamic loadaware scheduler is built using the following contributions

1. Task resource usage tag determination model using the counters provided by the initial Map/ Reduce tasks of the job.
2. Node load prediction model that predicts the future CPU and IO load on the node using the Linear Regression technique based on previous CPU and IO utilizations of the node and determine the node resource tag.
3. Dynamic load aware scheduling algorithm that dynamically schedules the Map / Reduce tasks on to the nodes based on the resource requirements and node resource availability in terms of CPU and IO intensiveness. The tasks which are observed to be CPU intensive are prioritized to schedule on the nodes which have less CPU utilization and similarly for IO utilization parameter. This can reduce the running time as the jobs with different resource requirements are scheduled on the node.

Proposed dynamic load aware scheduler is implemented using Hadoop 2.5.2 and the results indicate an improvement of 23% in the makespan times of the jobs submitted to Hadoop cluster.

This paper is organized as follows: Section 2 provides the motivation behind our work, Section 3 discusses the details of design and implementation of the proposed scheduler, Section 4 gives the evaluation results of the proposed scheduler compared with the capacity scheduler provided in Hadoop, and finally the conclusion is given in Section 5.

II. HADOOP INTERNALS

The scheduling process in Hadoop MRv2 YARN is performed by two components i) Resource Manager and ii) Application Master.

Resource Manager allocates resources to the submitted jobs with respect to the scheduler specified by the property yarn.resourcemanager.scheduler.class in file yarn-site.xml. Application Master is created for each job submitted and is responsible for negotiating the required resources from Resource Manager and assigns them for the Map / Reduce tasks of the job.

The resource management in YARN is done using Containers which represents a resource unit instead of slot based resource management as done in MRv1.

Every data node has the Node Manager module which is responsible for containers management, monitoring their resource usage and communicates the same to the Resource Manager through heartbeat messages.

The application workflow in Hadoop framework is given as

Step 1: client submits an application to the Resource Manager.

Step 2: Resource Manager allocates a container for Application Master.

Step 3: Application Master registers with Resource Manager.

Step 4: Application Master requests Resource Manager for containers with resource specifications required to run the tasks.

Step 5: Application Master requests node managers to launch containers.

Step 6: Map task/Reduce task is executed in the container.

Step 7: client can monitor application status by contacting to Resource Manager/Application Master.

Step 8: Application Master unregisters itself from Resource Manager once all tasks finish execution.

The Fig.1 indicates the detailed steps in the job execution of Hadoop using Resource Manager and Application Master. Once the Application master is started it assigns the Map tasks to multiple node managers based on the response received from Resource Manager. Node Managers create containers to execute the assigned Map/Reduce task. Node Manager communicates regularly with Resource Manager using Heart Beat Mechanism.

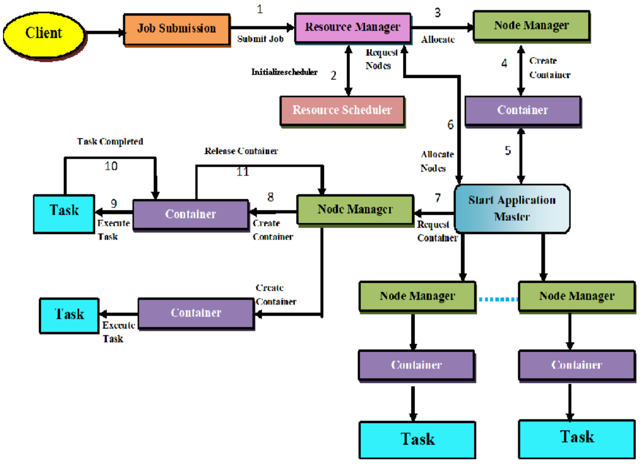

Fig. 1: Hadoop Job Execution Steps

The proposed work is to construct a dynamic capacity scheduler that can take scheduling decisions using job resource usage characteristics and virtual machine resources availability. All the Map tasks of a particular job apply same Map function on its own data split.

The data chunk size for each Map task expect for the last chunk is same. Hence once the task resource usage behavior is understood after completion of few tasks, then that knowledge is used in scheduling so that the remaining tasks which are CPU heavy are not scheduled on the data node that is less CPU free.

If a datanode is observed as executing tasks that are CPU intensive then a task that is less CPU intensive is selected for that node. The capacity scheduler of Hadoop framework is being reframed with the proposed dynamic loadaware algorithm. The makespan of all jobs submitted to Hadoop cluster is reduced when the job resource usage characteristics and the current load on the node are used in scheduling decisions.

III. DYNAMIC LOAD AWARE SCHEDULER FOR HADOOP

The design of the proposed scheduler includes the three contributions as specified in section 1. First contribution is to determine the job resource usage characteristics using the statistics related to CPU and IO usage of the initial Map / Reduce tasks. Secondly the node resource availability is predicted based on the previous history of CPU and IO utilizations of the node.

Finally the dynamic load aware scheduler is designed which schedules the Map Reduce tasks based on job resource usage and node resource availability. The initial Map Reduce tasks enable to understand the resource usage behavior of the job which can be used in scheduling the remaining tasks of the job.

Determination of Job Tag

Map Reduce tasks are assigned with a tag indicating the resource usage intensiveness in terms of CPU and IO. CPU tag indicates CPU intensiveness and IO tag indicates IO intensiveness of the job. Tag value is determined based on the execution counters of initial Map tasks that finished the execution. The counter values of CPU_MILLI_SECONDS, GC_TIME of the task are used to determine CPU tag and the counter values of HDFS_BYTES_READ, HDFS_BYTES_WRITTEN, FILE_BYTES_READ, FILE_BYTES_WRITTEN are used to determine the IO tag.

CPU utilization is calculated by taking the percentage of CPU time using (CPU_MILLI_SECONDS + GC_TIME)/elapsed time. If percentage of CPU utilization is greater than 50% then the task is categorized as CPU heavy otherwise as CPU light. IO utilization is determined by calculating the IO throughput as (HDFS_BYTES_READ+FILE_BYTES_WRITTEN+HDFS_BYTES_WRITTEN+FILE_BYTES_READ) /elapsed time, where the number of bytes is taken in MB. If IOthroughput >5 then the job is considered to be IO heavy, otherwise as IO light.

Job is associated with a tag once the initial Map tasks are completed depending on their CPU utilization and IO utilization as given in Table 1. The detailed implementations for determination of job tag for Map/Reduce tasks resource usage characteristics can be found in our previous paper [15].

Table 1: CPU and IO Tag Description

Tag	Tag value	Interpretation
00	0	CPU light and IO light process
01	1	CPU light and IO heavy process
10	2	CPU heavy and IO light process
11	3	CPU heavy and IO heavy process

Determination of Node Tag

To design the proposed scheduler, prediction of node resource usage status is done in terms of CPU usage and IO usage. The node is associated with a tag depending on the prediction of node load and is communicated to Resource Manager through the heart beat message of Node Manager.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Node load prediction in terms of CPU and IO is done using linear regression [23, 16, 17, 18] where the current CPU (or IO utilization) is the dependent variable and previous time interval CPU (or IO utilization) is considered as independent variable. The Linear model for the prediction used is in the form of equation 1.

$$U_t = \beta_0 + \beta_1 U_{t-1} \quad (1)$$

This model estimates the next CPU utilization of a virtual machine by considering the previous CPU utilizations. If Ut-1is CPU utilization at time t-1 and Ut is CPU utilization at time, t then there is a linear relation between Ut and Ut-1as given in Eq. (1) where least squares method is used to determine the relationship between the CPU utilizations at different time slots by determining the coefficients β0, β1 as given in Eq. (2) & Eq. (3)

$$\beta_0 = \frac{(\Sigma Ut)(\Sigma(Ut-1)^2) - (\Sigma Ut-1)(\Sigma(Ut-1xUt))}{n(\Sigma Ut-1^2) - (\Sigma Ut-1)^2} \quad (2)$$

$$\beta_1 = \frac{n((\Sigma(Ut-1xUt) - ((\Sigma Ut-1)x(\Sigma Ut)}{n(\Sigma Ut-1^2) - (\Sigma Ut-1)^2} \quad (3)$$

Similarly linear regression is applied for IO utilization where Ut-1is IO utilization at time t-1 and Ut be IO utilization at time, t using Eq. (1).

The near future CPU and IO utilization of a particular node is determined using linear regression and is associated with a tag of two bits where MSB denotes CPU utilization and LSB denotes IO utilization.

If CPU utilization is greater than 50% then the node is tagged as CPU heavy and if IO utilization is greater than 50% then the node is tagged as IO heavy. The tag specifies the next CPU and IO utilization estimation based on previous history using linear regression as indicated in the Table 2.

Table 2: Node Tag Description

Node Tag	Node Tag Value	Description
00	0	CPU free IO free
01	1	CPU free IO busy
10	2	CPU busy IO free
11	3	CPU busy IO busy

Current CPU utilization and IO utilization of the node is determined using the command iostat –x. iostat is the command that reports the CPU and IO statistics. iostat with –x option is used to display the extended statistics where the columns %user, %nice, %system which indicates percentage of CPU utilization that occurred by executing at the user level, user level with nice priority and system level respectively and %utilization indicates the IO utilization.

The required statistics are extracted using awk script (stat.awk). `Iostat –x|awk –f stat.awk` is used to retrieve the required CPU utilization and IO utilization with an interval of 2 msec.

The values obtained are used for linear regression to predict the future CPU and IO utilizations of the node. The shell script that determine the near future CPU and IO utilization of a particular node using linear regression is being invoked in monitoring thread of Node Resurce Monitor Impl class of Hadoop package.

The monitoring thread runs continuously and sets the CPU tag and IO tag for the node.

Once the node status is determined, it is passed to Resource Manager through the heartbeat message. To enable the tag information to be embedded in heartbeat, the NodeHeartbeatRequest class is being modified to include a node heartbeat instance with CPU tag and ÏO tag.

To evaluate our proposed prediction model of the node load prediction using linear regression another bash script is written to monitor the real CPU and IO utilizations using iostat and the awk script. The observed values are compared with estimated values using Mean Absolute Error (MAE), Root Mean Square Error (RMSE).

Workloads are being varied to have CPU intensive tasks and IO intensive tasks and mixture of CPU and IO intensive tasks. Hi-Bench [19] Hadoop benchmarking tool is used to generate the workload for CPU intensiveness and IO intensiveness jobs.

The Mean Absolute Error (MAE), Root Mean Square Error (RMSE) are calculated for different sets of workloads generated by Hi-Bench suite. For each workload samples of 100 is used and when merged on the common field of time between estimated utilizations and real utilizations of CPU and IO, lead to 90 to 95 samples.

Stressing tool [20] is used to stress the node in terms of CPU and IO to check for evaluation of node prediction model. Four test cases are used to evaluate the prediction model.

Test Case1: No stress, Test Case2: Stress on CPU, Test Case 3: Stress on IO and Test Case 4: Stress on CPU and IO. Table 2 gives the error value calculations for CPU utilization and IO utilization for these test cases.

Table 3: Linear Regression for CPU and IO Utilization Prediction

Test case	CPU utilization prediction		IO utilization prediction	
	MAE	RMSE	MAE	RMSE
Test case-1	0.087196	0.087933	0.001465	0.003689
Test case-2	0.100976	0.100887	0.011254	0.037082
Test case-3	0.120874	0.122632	0.109223	0.111756
Test case4	0.185433	0.186654	0.190876	0.190437

The model is observed to be efficient in forecasting the future CPU and IO utilizations based on MAE and RMSE values.

Dynamic Load Aware Scheduler

The proposed work is to perform dynamic scheduling of Map Reduce tasks using node tag and job tag as given in the Algorithm1.

To implement the proposed work the following changes were made in Hadoop package:

1. Inclusion of Boolean variable named loadaware in yarn_server_common_service_protos.proto
2. All the modules modified in Hadoop package are being written in consideration with the loadaware variable to differentiate if the loadaware property is set by the user.

Algorithm 1 assignContainers(Resource clusterRes, FicaSchedulerNode node)

```
alloc=false;
turn=1;
count=0;
size=activeApplications.size();
for(FicaSchedulerAppapp:activeApplications)
        if(load-aware)
        count++;
        if node has data local for application app
        selectedapp=app
        n=nodetag of node;
        j=jobtag of app;
        if((j==0) or (j==-1))
            selectedapp=app
        if(match=true) store turn into file
        else
                if(j AND n=0)
                    selectedapp=app
                    match=true
                    if(count<size) continue
                        else
                        if (turn=1)
                            turn=2
                            return
                            NULL_
                            ASSIGN
                            MENT
                        end if
                    end if
                end if
        end if
        return selectedapp;
end for
```

A Dynamic scheduler algorithm as given in algorithm1 is designed by contributing to Hadoop framework which can schedule tasks based on node tags and job tags. For the initial Map tasks, the job tags are unknown (job tag=-1) then the scheduling is done by using Hadoop Capacity Scheduler. But once the job tag is determined after completion of few Map tasks, the job tag is used in scheduling decisions along with the node tag. Node tag is determined by the monitoring thread as given in section 2.2. The default Hadoop Capacity scheduler is used if the current job tag is 0 as the job is neither IO heavy nor CPU heavy. If the current job tag is not equal to 0 then a search of job is done that matches the node tag of the current node (AND operation of job tag and node tag is used to select the job whose resource usage is not same as the node resource usage heaviness). The selected app based on the tags is scheduled on the node if found, otherwise a NULL_ASSIGNMENT is returned. The number of turns allowed for returning NULL_ASSIGNMENT is only once. If a particular job A (turn=2) was skipped due to mismatch of the job tag and node tag and no jobs in the queue got scheduled, then job A gets scheduled.

Dynamic load aware scheduler checks for an appropriate job based on the job tag and node tag. If job tag is found to be 0, it is scheduled directly on the data node which is ready for the next task. Otherwise the AND operation of job tag and node tag can be used to get a job with dissimilar tags. If a job is observed to be CPU heavy (2) then a node with node tag 1 and 0 is more appropriate for the task.

IV. EVALUATION AND RESULTS

The proposed dynamic load aware scheduler for Hadoop is evaluated on private and public cloud environments.

Dynamic Load Aware Scheduler Evaluation on Private Cloud

The proposed scheduler is evaluated on private cloud setup using VMware tools: VMware ESXi server and VCenter. Four VMs are created with the following configurations OS: Ubuntu14.0, Memory: 4GBHard Disk: 200GB. Jobs WordCount(WC), Pi, RandomTextWriter(RTW), Sort, grep are used for evaluation of proposed work and to have accurate results each test case is executed for two times and the average execution time is considered. Capacity scheduler is configured with two queues of capacity configuration 50% each. Fig.2 to Fig.5 gives comparison of Capacity scheduler and proposed dynamic Capacity Scheduler Load Aware (CSLA) for data sizes 1GB, 2 GB and 3 GB respectively.

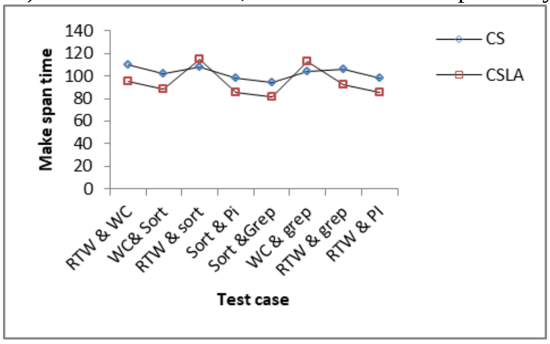

Fig. 2: CS Vs CSLA on Private Cloud for Data Size=1GB

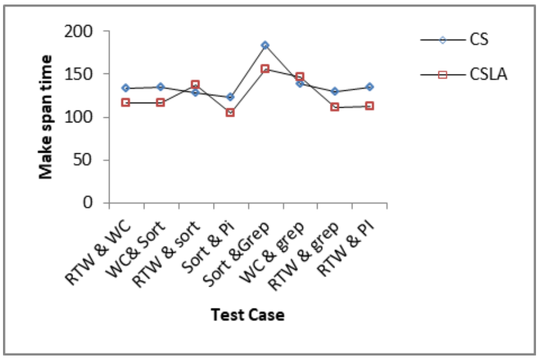

Fig. 3: CS Vs CSLA on Private Cloud for Data Size=2GB

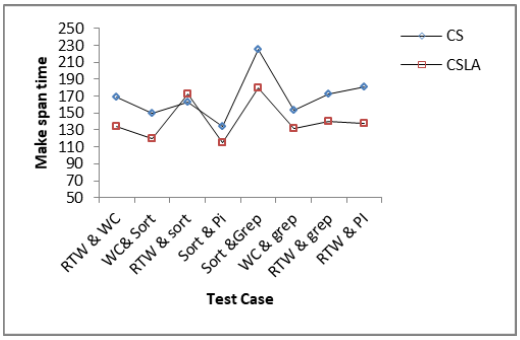

Fig. 4: CS Vs CSLA on Private Cloud for Data Size=3GB

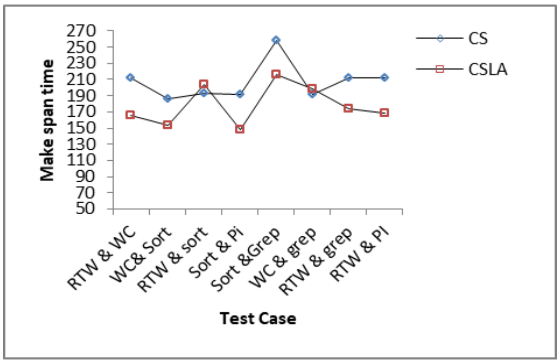

Fig. 5: CS vs. CSLA on Private Cloud for Data Size=4GB

The above results in Fig.2 to Fig.5 indicate a good performance improvement from 12% to 23% using Capacity scheduler load aware for jobs of size 1GB, 2GB, 3GB, 4GB when executed on private cloud of 4 machines except for the jobs which have similar resource usage intensiveness like RTW and sort when executed simultaneously. The Proposed model is also tested using test cases with combination of 4 jobs with two queues of 50% capacity each on four virtual machines. The set of four jobs with input data size of 3GB are submitted with two jobs to each queue. The makespan time for different combinations of jobs using Capacity scheduler and capacity scheduler load aware for four jobs with data size of 3GB on private cloud is indicated in Fig.6.

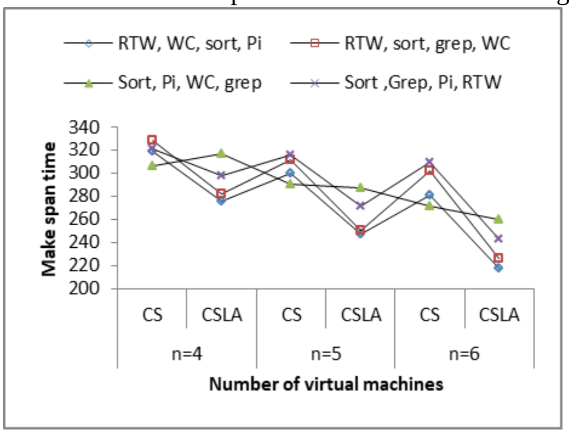

Fig. 6: CS vs CSLA for Four Jobs with Data Size=3G

The above results in Fig.6. indicate a good performance improvement of up to 23% using capacity scheduler loadaware for jobs of size 3GB when executed on private cloud of 4 machines except for the jobs which have similar resource usage patterns like Pi, WC, Sort, grep when executed simultaneously.

Dynamic Load Aware Scheduler Evaluation on Public Cloud

The proposed scheduler is evaluated on public cloud Hadoop cluster created on Amazon EC2 machines with basic OS as Ubuntu14.0 instances of type T2.medium by varying the number of instances for data nodes. Jobs RTW, WC, Pi, Sort, grep are used for evaluation. To get accurate results each pair of jobs is executed for two times and the average of execution time is used. Fig.7 to Fig.10 indicates the makespan for capacity scheduler and capacity scheduler load aware scheduler for test cases of data sizes of 1GB, 2GB, 3GB and 4GB.

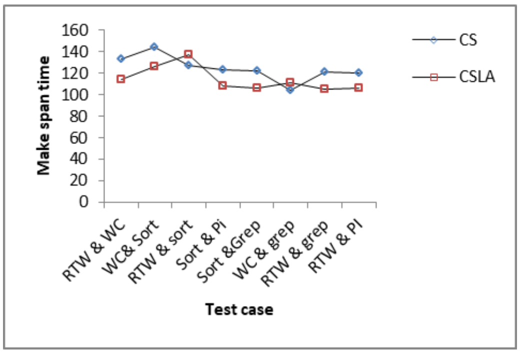

Fig. 7: CS vs. CSLA on Public Cloud for Data Size=1GB

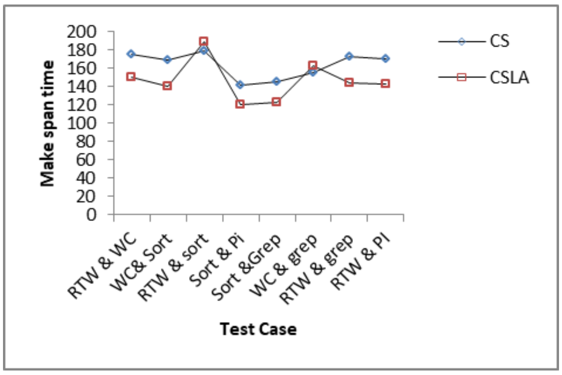

Fig. 8: CS vs. CSLA on Public Cloud for Data Size=2GB

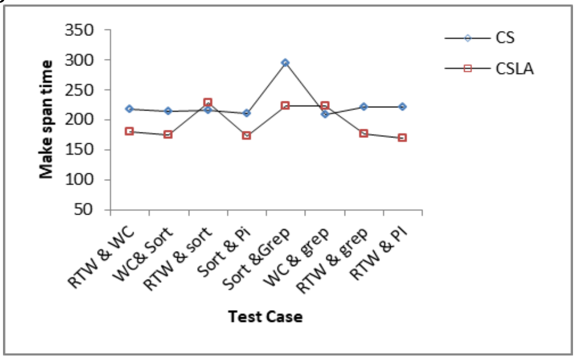

Fig. 9: CS vs. CSLA on Public Cloud for Data Size=3GB

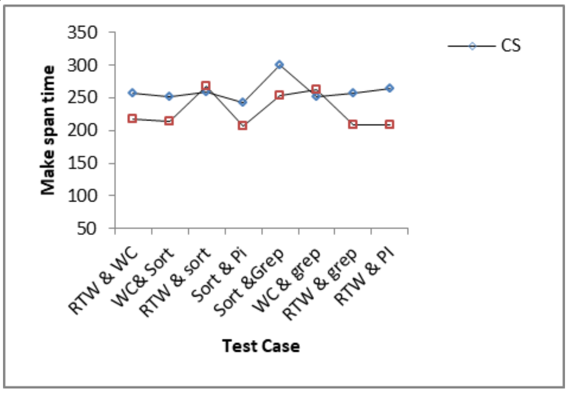

Fig. 10: CS vs. CSLA on Public Cloud for Data Size=4GB

The above results in Fig.7 to Fig.10 indicate a good performance improvement of up to 24% for jobs of size 1GB, 2GB, 3GB when executed on public cloud of 4 machines except for the jobs with similar resource usage intensiveness. Proposed model is also tested for combination of 4 jobs. The set of four jobs with input data size of 3GB are submitted with two jobs to each queue of capacity 50 and Fig.11 provides the comparison of makespan time for Capacity scheduler and capacity scheduler load aware on public cloud Amazon EC2.

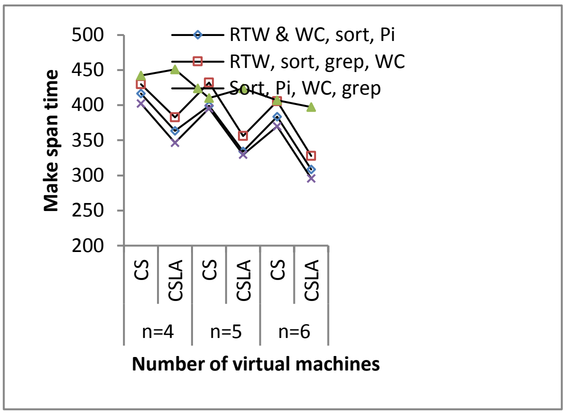

Fig. 11: CS vs. CSLA on Public Cloud for Four Jobs with Data Size 3G

The above results in Fig.11 indicate a good performance improvement using capacity scheduler load aware up to 20% for jobs of size 3GB when executed on public cloud of 4 machines except for the jobs with very similar resource usage intensiveness.

V. CONCLUSION

A dynamic load aware scheduler for Hadoop framework which can schedule the Map Reduce tasks based on resource requirements of the task and resource availability of the node is designed and implemented. The results indicate an improvement of up to 24% in execution time when scheduling is done using proposed dynamic load aware scheduler compared to default capacity scheduler of Hadoop.

The improvement in execution times increases as the number of machines used to execute the Hadoop jobs increases. The proposed dynamic load aware scheduler is more efficient for Hadoop clusters with more number of data nodes. This work can be further extended to have automatic Hadoop configuration parameter settings based on job resource usage characteristics to achieve optimized performance.

REFERENCES

1. Hadoop The definitive guide, O'Reilly & yahoo press, Tom White
2. J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Communications of the ACM - 50th anniversary issue: 1958-2008, vol. 51, no. 1, January 2008, pp.107-113.
3. K. Chen, J. Powers, S. Guo and F. Tian "CRESP: Towards Optimal Resource Provisioning for MapReduce Computing in Public Clouds", *IEEE Transactions on Parallel and Distributed Systems,* Vol. 25, Issue: 6, 2014, pp. 1403 – 1412.
4. Lei Yang, Yu Dai, Bin Zhang, "Map Reduce scheduler by characterizing performance interference", China Communications, Volume 13, Issue 10, November 2016, pp. 253-262.
5. Open source Apache Hadoop, http://Hadoop.apache.org/core/
6. A.Sree Lakshmi, Dr. M. Balraju, Dr.N.Subash Chandra, "Scheduling of Parallel applications using Map Reduce on Cloud: a Literature Survey" ", *International Journal of Computer Science and Information Technologies,* Volume 6, Issue 1 , 2015, pp. 112-115.
7. Feng Yan, Ludmila Cherkasova, Zhuoyao Zhang, Evgenia Smirni, "Dyscale: A Map Reduce job scheduler for heterogeneous multicore processors" , *IEEE Transactions on Cloud Computing,* Vol. 5, issue 2, June 2017, pp. 317-330.
8. K.Arun Kumar, Vamshi Krishna, Kaladhar Voruganti, G. V. Prabhakara Rao, "CASH: context aware scheduler for Hadoop", ICACCI '12 Proceedings of the *International Conference on Advances in Computing, Communications and Informatics,* August 2012, pp.52-61.
9. Qi Zhang, Mohamed Faten Zhani, Yuke Yang, Raouf Boutaba, Bernard Wong "Prism, Fine grained resource-aware scheduling for MapReduce", *IEEE Transactions on Cloud Computing,* volume 3, issue 2, June 2015, pp. 182-191.
10. K. Chen, J. Powers, S. Guo and F. Tian "CRESP: Towards Optimal Resource Provisioning for MapReduce Computing in Public Clouds", *IEEE Transactions on Parallel and Distributed Systems,* Vol. 25, Issue: 6, 2014, pp. 1403 – 1412.
11. Yi Yao,Jiayin Wang, Bo Sheng, Jason Lin, NingfangMi , "HaSTE: Hadoop YARN Scheduling Based on Task-Dependency and Resource-Demand", 7th *IEEE International Conference on Cloud Computing,* July 2014, pp.184-191.
12. Radheshyam Nanduri, Nitesh Maheshwari, Reddy Raja, Vasudeva Varma, "Job Aware scheduling algorithm for Map Reduce Framework", 3rd *IEEE international conference on Cloud Computing Technology and science*, Nov 2011, pp. 724-729.
13. Quan Chen, Daqiang Zhang, MinyiGuo, Qianni Deng, and Song Guo. "Samr: A self-adaptive MapReduce scheduling algorithm in heterogeneous environment." *Computer and Information Technology, International Conference,* July 2010, pp. 2736– 2743.
14. Shyam Deshmukh, Dr. J. V. Aghav, Rohan Chakravarthy "Job Classification for MapReduce Scheduler in Heterogeneous Environment", *International Conference on Cloud & Ubiquitous Computing & Emerging Technologies,* November 2013, pp.26-29.
15. A.Sree Lakshmi, Dr. M. Balraju, Dr.N.Subash Chandra, "Determination of resource usage characteristics for Hadoop Map Reduce tasks", *International Journal of Computer Engineering & Technology (IJCET),* Volume 9, Issue 1, January-February 2018, pp. 113–119.
16. FahimehFarahnakian, PasiLiljeberg, JuhaPlosila, "LiRCUP: Linear Regression Based CPU Usage Prediction Algorithm for Live Migration of Virtual Machines in Data Centers", 39th *Euromicro Conference on Software Engineering and Advanced Applications,* September 2013, pp. 357 – 364.
17. Jina Wang, Yongming Yan, Jun Guo, "Research on the Prediction Model of CPU Utilization Based on ARIMA-BP Neural Network", *MATEC Web of Conferences,* Volume 65, January2016.
18. Peter A. Dinda and David R. O'Hallaron, "Host load prediction using linear models", Cluster Computing 3, Baltzer Science Publishers BV, Dec 2000, Volume 3, Issue 4, pp. 265–280.
19. https://github.com/intel-Hadoop/HiBench
20. http://kernel.ubuntu.com/~cking/stress-ng/
21. Mohd Usama, Mengchen Liu, Min Chen, "Job schedulers for Big data processing in Hadoop environment: testing real-life schedulers using benchmark programs", Digital Communications and

Networks, August 2017.

22. Hui Zhao, Qinghua Zheng, Weizhan Zhang and Jing Wang, "Prediction-Based and Locality-Aware Task Scheduling for Parallelizing Video Transcoding over Heterogeneous MapReduce Cluster", *IEEE transactions on circuits and systems for video technology,* Vol. 28, No. 4, April 2018.
23. Marina Kudinova1, Anna Melekhova, Alexander Verinov, "CPU Utilization Prediction Methods Overview", CEE-SECR '15, October 22-24, 2015, Moscow, Russian Federation, ACM.
24. Zhihong Liu, , Qi Zhang, , Reaz Ahmed, Member, IEEE Raouf Boutaba, Fellow, IEEE, Yaping Liu, and Zhenghu Gong Dynamic Resource Allocation for MapReduce with Partitioning Skew, *IEEE transactions on computers,* Vol. 65, No. 11, November 2016.
25. Shaowei Liu, Kaijum Ren, Kefeng Deng and Junqiang Song , "A Dynamic resource allocation and task scheduling strategy with uncertain task runtime on IaaS clouds" , Sixth *international conference on Information Science and Technology,* China, May 6-8 2016.
26. Lei Yang, Yu Dai, Bin Zhang, "Map Reduce Scheduler by Characterizing Performance interference", China Communications, Oct 2016.
27. NIDHI TIWARI, SANTONU SARKAR, UMESH BELLUR, MARIA INDRAWAN, "Classification Framework of MapReduce Scheduling Algorithms" ACM Comput. Surv. 47, 3, Article 49, April 2015.
28. Balaji Palanisamy, Aameek Singh, and Ling Liu, "Cost-Effective Resource Provisioning for MapReduce in a Cloud", *IEEE transactions on parallel and distributed systems,* Vol. 26, No. 5, May 2015.
29. Jian Li, Tinghuai Ma, Meili Tang, Wenhai Shen and Yuanfeng Jin, "Improved FIFO Scheduling Algorithm Based on Fuzzy Clustering in Cloud Computing", MDPI-Information, Feb 2017.
30. Qutaibah Althebyan1,Yaser Jararweh2, Qussai Yaseen3, Omar AlQudah2 and Mahmoud Al-Ayyoub ," Evaluating Map Reduce tasks scheduling algorithms over cloud computing infrastructure", Concurrency and Computation: Practice and Experience 2015, Published online in Wiley Online Library.
31. Xu-qing Chai1, Yong-liang Dong and Jun-fei Li, "Profit-oriented task scheduling algorithm in Hadoop cluster", *EURASIP Journal on Embedded Systems, a Springer open Journal.*