Phase and Resource Information-Aware Scheduling for MapReduce Jobs Scheduling

Preethi Ravi

Department of Computer Science and Engineering
K.Ramakrishnan College of Technology, Samayapuram
Tiruchirappalli, Tamil Nadu, India

preethiir92@gmail.com

***Abstract* - MapReduce is a model for the data-intensive computation. However, despite recent efforts towards designing efficient scheduler to perform MapReduce, the available solutions focuses on scheduling at the task-level offers sub-optimal performance in executing jobs. This is because tasks can have highly varying resource requirements during their lifetime, which makes it difficult for schedulers to effectively utilize the available resources to reduce job execution time. To address this limitation, PRISM, a fine-grained resource-aware MapReduce scheduler divides the tasks into phases, where each of the phase has a constant resource usage profile, and performs scheduling at the phase level. The importance of the Phase-level scheduling is demonstrated by showing the resource usage variability within the lifetime of a task using a wide-range of MapReduce jobs. The phase-level scheduling algorithm improves execution time and resource utilization without introducing any stragglers with the help of Virtual Space.**

Index Terms* - *MapReduce, scheduling, resource utilization, execution time, Virtual Space.

I. INTRODUCTION

The development of the data-driven decision making had paved the development of MapReduce [3], which is most widely used in data-intensive computation. The various phases of the MapReduce are (a)Map step (b) Shuffle step and (c)Reduce step. Job scheduler is the major component of MapReduce system. The job scheduler's major role is to create a schedule of one or more jobs that reduces the job completion time and maximizes resource utilization.

The development of the data-driven decision making had paved the development of MapReduce [3], which is most widely used in data-intensive computation. The various phases of the MapReduce are (a)Map step (b) Shuffle step and (c)Reduce step. Job scheduler is the major component of MapReduce system. The job scheduler's major role is to create a schedule of one or more jobs that reduces the job completion time and maximizes resource utilization.

Hadoop NextGen and Hadoop Yarn [1], uses the RAS (Resource-aware job scheduler)[8], that specifies a fixed size container for each task in terms of resource requirement, The assumption made in RAS is that the run-time resource consumption of the task is stable over its lifetime [8]. The major problem with the fixed resource requirement scheduling are excessive resource contention and the scheduling with the low resource utilization. The multiple phases of MapReduce task can be given as a) data transfer b)processing c)storage. Phase could be defined as the sub-procedure, that has the distinct purpose and the resource consumption will also be in uniform manner. The another important criteria to be noted is, the phases in the same task can have resource demand in different manner.

The main objective of the phase-level scheduling along with the Virtual Space is to reduce the delay that occurs during the "pause" stage of the node manager and also to increase the job running time and the resource utilization when compared with the task-level scheduling. Phase based scheduling algorithm [6]. In order to overcome the issue of delay during the job execution due to the "pause: stage, the virtual space is allocated to the node manager. thus the average job running time will be improved when compared with the phase based scheduling algorithm without virtual space.

II. BACKGROUND

This section provides an overview of phase-base scheduling algorithm and Hadoop MapReduce.

A) Scheduling process

PRISM (Phase and Resource Information-aware scheduler [6] for MapReduce cluster), performs resource-aware scheduling at the level of task phases. The phase-level scheduling algorithm is designed with the aim of achieving high job performance and resource utilization. Given a job's phase-level resource requirement and its current progress information, the scheduler decides whether to start a new task, or allow a paused task to begin its next phase and then informs the node manager about the scheduling decision. This "pause" stage will increase the job execution and running time.

The main goal of the Fair scheduling [4] is, if a job takes t seconds to complete the access to a cluster, then it should not require more than Jt seconds of execution time when the cluster is shared among J concurrent job is high, then they compete for bandwidth, which in turn complicates performance prediction.

Delay scheduling [8] address the conflict between the locality and fairness. The main concept is, according to fairness the job that should be scheduled next cannot launch a local task, instead it lets other jobs to launch task. The process of reassign the task could be done by killing tasks to make room for new jobs, and waiting to finish the task and then assigning slots for new jobs.

B) MapReduce Process

Hadoop Online Prototype (HOP) [2] explains a modified MapReduce architecture that allows the data to be

pipelined between operators. The continuous queries that enables the MapReduce programs such as event monitoring and stream processing is also supported by HOP.

In Hadoop system the Job Tracker will take care of scheduling a new map task. If a new task is started after the failure of the reduced task, then all the input data that was sent to the failed reduced attempt must be again sent by the new reduce instance. The MapReduce process is divided as the map task and reduce task. There are large number of the commodity machines available in the Hadoop clusters.

In cloud the workload can have varied heterogeneous capabilities [8], [9]. Heterogeneous execution of machines has great impact on resource sharing. The job schedule in Hadoop assumes machines of clusters are homogeneous and the processing of tasks are done linearly.

In this case of the task that have heterogeneous resource requirement, it should be decided that how much resources should be allocated to task to achieve high performance. This problem of the performance bottlenecks that arises due to the homogeneous assumption of Hadoop clusters could be overcome by MR Orchestrator [10] that resolves these bottleneck through coordinated, fine-grained and on-demand resource allocation.

III. PHASE-LEVEL SCHEDULING WITH VIRTUAL SPACE

The existing system is the PRISM that uses the phase-level scheduling [6]. PRISM uses the phase-level scheduling algorithm to achieve high job performance and resource utilization. This could be achieved by considering the resource demand at phase level.

The problem with the phase-level scheduling algorithm is, when the task has completed the execution of phases, and if no sufficient resources are available then the subsequent phases may not be scheduled immediately. This could be alternatively explained as, the phase execution will be "paused" when no sufficient resources are available.

The "paused" execution will avoid contention of resource by delaying task completion [3]. The running time of the jobs will be improved while comparing with task-level resource-aware scheduler but the higher fairness or scheduling cannot be achieved. In order to overcome this problem the virtual space is used. To eliminate the "pause" stage and to reduce the job execution time.

The two components of the resource manager are (a)phase-base scheduler (b) Job progress monitor. When the request for the job execution is received by the phase-base scheduler, then the jobs are divided into small tasks by the phase-base scheduler.

The node manager along with a threshold amount of space will be allocated to each task by the phase-based scheduler. The execution of the job will be paused and the request for the additional space from the phase-base scheduler will be done by the node manager. this "pause" stage will always result in long job execution time.

In order to overcome this problem of long job execution time, the virtual space will be allocated to node manager. the virtual space will have the additional space apart from node

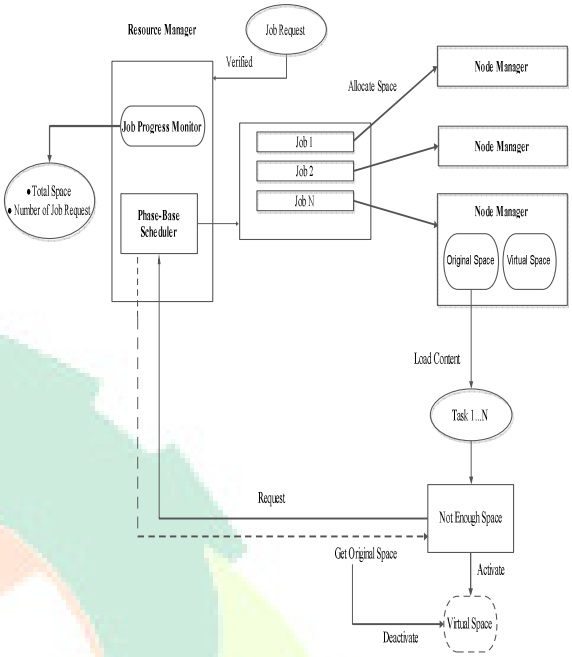

Fig. 1 Architecture of Phase- based scheduler with virtual space to reduce job execution time and improve resource utilization

manager. During the "pause" stage of the job execution, the space available with the virtual space will be utilized to avoid the long job execution.

IV. PHASE-LEVEL SCHEDULER DESIGN

This section describes in detail the design of Phase-base scheduling algorithm with virtual space

A) Algorithm Description

The following steps will describe the working of the phase scheduling algorithm.

(a)When a task has to be scheduled, the scheduler replies to the heart beat message with scheduling request.

(b)The node manager then launches the task execution. When there is no enough space available to execute the job that is scheduled, then the execution of the job will be paused.

(c)Now the node manager will load the content that is paused due to the insufficient resources into the virtual space.

(d)when the actual memory that is demanded by the node manager from phase-base scheduler is allocated.

(e)The node manager will deactivate the virtual space and load the content into original space.

(f)After finished executing a particular phase, the task asks permission to start the next phase from the node manager

(g)Then the node manager forwards the request for the permission to scheduler through heart beat message.

(h)If the phase-level resource requirement and the current progress information is known, the scheduler decides whether to start the execution of new task or paused task to

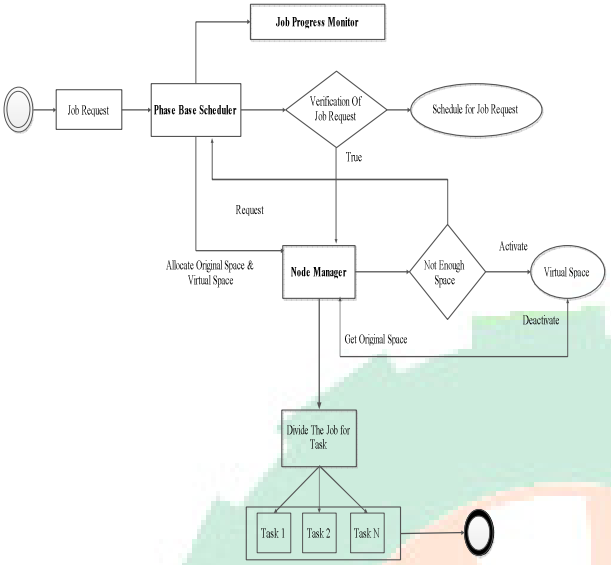

Fig. 2 Overall workflow of the phase-base scheduler with the virtual space in executing the task and managing pause stage using virtual space

begin next phase, and then this scheduling decision will be informed to node manager.

(i)When the execution of all task is completed, the task status is received by node manager.

(j)Then this task status will be forwarded to scheduler by node manager

B) System Design Rationale

The entire working of the phase-base scheduler can be divided into following modules : (1) Phase based scheduling, (2) Verifying of Job Request, (3) Allocation by Node manager, (4) Monitoring Job progress, (5) Activate virtual space and (6) Deactivate virtual space.

[1] Phase based scheduling

This phase base scheduler is the master node, which is located inside the Resource manager. The resource manager can also be called as Job tracker, which is used for dynamically allocating the space. The job request is dynamically scheduled to achieve high job performance and resource utilization. The job progress monitor is present inside the resource manager, whose job is to capture the phase-level progress information. These information includes : (i) Total space in resource manager, (ii) Number of job request arrived, (iii) Space available in node manager.

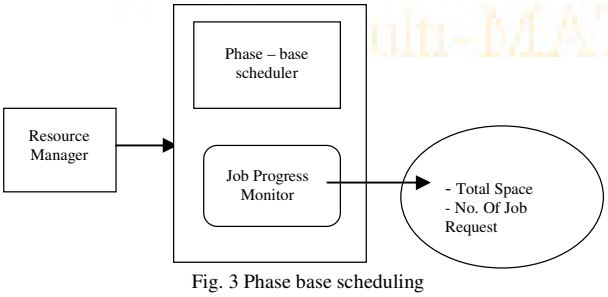

Fig. 3 Phase base scheduling

[2] Verifying Job Request

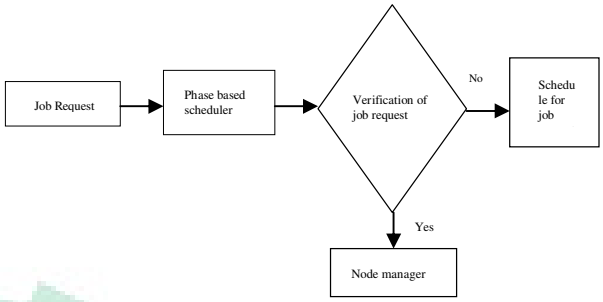

Fig. 4 Verifying of Job Request

Each job will be verified by the resource manager and after that it will be allowed by the phase base scheduler. These jobs are then dynamically received and allocated by the node manager. Details about the availability of space in resource manager will be send to phase base scheduler.

[3] Allocation by Node manager

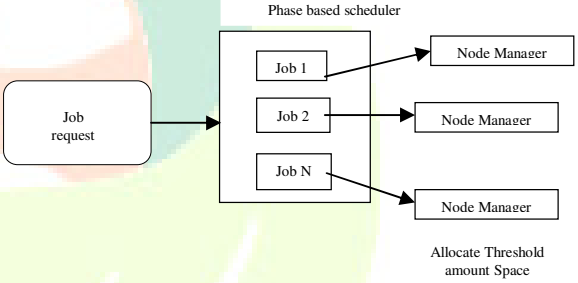

Fig. 5 Allocation by node manager

The process of launching and allocating resources for each task is done by Node manager, which monitors the progress of running task and available resources on the node and transmits a heartbeat message to resource manager. Some threshold amount of space has been allocated to the node manager for scheduling tasks.

When the sufficient amount of space is not available, then the node manager request the scheduler for additional space. The node manager can also be called as task manager.

[4] Monitoring Job progress

The Job progress monitor captures the phase-level progress information. Once the task has finished executing a particular phase, it asks permission to start next phase. If the task has finished executing, the node manager receives the task status and forwards it to scheduler.

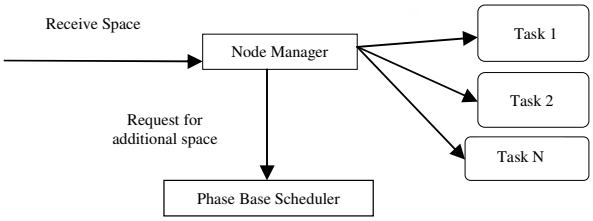

Fig. 6 Monitoring Job progress

[5] Activate Virtual space

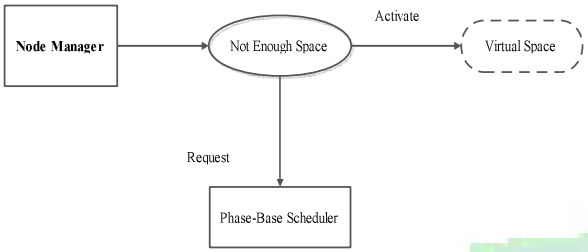

Fig. 7 Activate Virtual Space

The scheduler allocates the virtual space and original space to Node Manager before the process starts. When the user sends the content to node manager, he uploads the content in original space. When the space exceeds, the node manager request the scheduler for more space. Then the scheduler activates the virtual space by virtual mechanism. Virtual space is activated till the scheduler allocate original space to node manager. The node manager uses the virtual space for load the content before gets the original space from scheduler.

[6] Deactivate Virtual space

The scheduler allocates the original space. The allocated message is send to node manager. The node manager copies the content to allocated original space. After copying the content, the node manager deactivates the virtual space. Now the content will be available in the original space once the virtual space is deactivated.

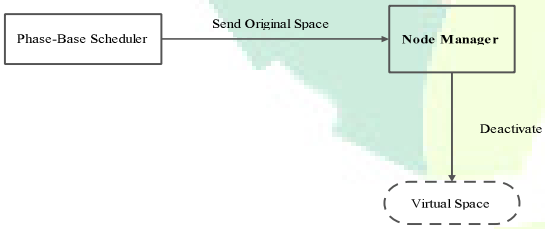

Fig. 8 Deactivate Virtual Space

V. CONCLUSION

The popular programming model for data intensive computing is MapReduce. PRISM is a fine-grained resource-aware scheduler that monitors the task execution at the level of phases of task execution. This phase-level job scheduling algorithm by using virtual space improves job execution without introducing stragglers.

This scheduler also offers high resource utilization and provides improvement in job running time when compared to the existing scheduler. The flexibility of phase-based scheduling allows the scheduler to improve utilization of resources and the performance in executing the jobs or tasks, which is a challenging problem.

The virtual space that uses the virtual mechanism is used by the node manager in the pause duration. When the Node Manager retrieve the original space it will shift the loaded content from virtual space to the original space.

REFERENCES

[1] The Next Generation of Apache Hadoop MapReduce [online]. Available:http://hadoop.apache.org/docs/current/hadoop-yarn-site/YARN.html, 2015

[2] T. Condie, N. Conway, P. Alvaro, J. Hellerstein, K. Elmeleegy, and R. Sears, "MapReduce online," in *Proc. USENIX Symp. Netw. Syst. Des. Implementation*, 2010, p.21.

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, Vol. 51, no. 1, pp. 107-113, 2008.

[4] Israd M., Prabhakaran V., Currey J., Wieder V., and Talwar K, "Quincy: Fair scheduling for distributed computing clusters," in *Proc. ACMSIGOPS Symp. Oper. Syst. Principles*, pp. 261-276, 2009.

[5] J. Polo, C. Castillo, D. Carrera, Y. Becerra, I. Whalley, M. Steinder, J. Torres, and E. Ayguade, "Resource-aware adaptive scheduling for MapReduce cluster," in *Proc. ACM/IFIP/USENIX Int. Conf. Middleware*, 2011, pp. 187-207.

[6] Qi Zhang, Mohamed Faten Zhani, Yuke Yang, Raouf Boutabai, and Bernard Wong, "PRISM: Fine-Grained resource Aware Scheduling for MapReduce," in *IEEE Transactions on Cloud Computing,* Vol. 3, no. 2, April/June 2015.

[7] A. Verma, L. Cherkasova, and R. Campbell, "Resource Provisioning Framework for MapReduce jobs with Performance goals," in *Proc. ACM/IFIP/USENIX Int. Conf. Middleware, 2011, pp. 165-186.*

[8] Savitri D.H, Narayana H.M, "PRISM: Allocation of Resources in Phase-level using MapReduce in Hadoop," in *International Journal of Research in Science and Engineering Vol. 1, Special Issue: 2.*

[9] Boutaba R., Cheng L., and Zhang Q., "On Cloud Computational models and the heterogeneity challenge," *Journal of Internet Services and Applications,* Vol. 3, no. 1, pp. 1-10.

[10] Bikash Sharma, Ramya Prabhakar, Seung-Hwan Lim, Mahmut T. Kandemir and Chitra R. Das, "MROrchestrator: A Fine-Grained Resource Orchestration Framework for Hadoop MapReduce," *Technical Report cse-12-001,* 2012.

[11] M. Zaharia, D. Borthaku, J. Sen Sarma, K. Elmelegy, S. Shenker, and I. Stoica, "Delay Scheduling: A Simple Technique for achieving locality and Fairness in Cluster Scheduling," in *Proc. Eur. Conf. Comput. Syst.,* 2010, pp. 265-278.