www.ioirp.com
International Journal of Innovative Research in Technology, Science & Engineering (IJIRTSE)
ISSN: 2395-5619, Volume – 1, Issue – 8. October 2015

Resource Provisioning Framework for MapReduce Jobs Schedule

Preethi R
Department of Computer Science and Engineering
K.Ramakrishnan College of Technology, Samayapuram
Tiruchirappalli, India
e-mail id: preethiir92@gmail.com

***Abstract*—MapReduce has become a popular model for the data-intensive computation over the recent year. MapReduce can significantly reduce the running time of data-intensive computation. The job scheduling problem becomes significantly easier to solve if we can assume that all map tasks and reduce tasks have homogenous resource requirements in terms of CPU, memory, disk and network bandwidth. Existing solutions focuses on scheduling at the task-level. But unfortunately, the task-level scheduling leads to inefficient job schedules with low resource utilization and long job execution time. The phase-level scheduling achieves higher resource utilization when compared to task-level schedulers. PRISMconsists of three main components: a phase-based scheduler that act as the master node, local node managers that coordinate phase transitions with the scheduler and the job progress monitor that captures the phase level progress information. Thus the average job running time will be improved in phase-level scheduler when compared to the task-level resource-aware schedulers.**

Keywords— Cloud computing, MapReduce, scheduler, resource allocation, execution time.

I. INTRODUCTION

The shifting towards data-driven decision making has fueled the development of MapReduce[3], a parallel programming model that is most widely used in the data-intensive computation. MapReduce is a programming model, implemented for processing and generating large data sets with a parallel, distributed algorithm on a cluster. The map( ) procedure performs filtering and sorting operations. For example, sorting student first name into queues, one queue for each name and the reduce( ) procedure provides the summary of the operations that is performed. For example, yields the number of counts. The main component of MapReduce system is its job scheduler. The main role of the job scheduler is to create a schedule of one or more jobs that minimizes job completion time and maximizes resource utilization.

The job scheduling problems becomes significantly easier to solve if the map and reduce task have the homogeneous resource requirement in terms of CPU, memory, disk and network bandwidth. Hadoop MapReduce version 1.x makes the above assumption and this uses the slot-based resource allocation scheme[2], [5]. In the slot-based resource allocation scheme, the physical resources on each machine are captured by the number of identical slots that can be assigned to tasks.The problem with that slot-based resource allocation is that the run-time resource consumption varies from task to task and from job to job.

Hadoop MapReduce version 2 (also known as Hadoop NextGen and Hadoop Yarn[1]) uses the RAS (Resource-aware job scheduler) [8], to MapReduce framework. RAS specify a fixed size for each task in terms of resource requirement and it also assumes, run-time resource consumption of the task is stable over its life time [8]. However, this is not true in most of the Hadoop job scheduler.The disadvantages of fixed resource requirement scheduling are: excessive resource contention and low utilization by scheduling. MapReduce task can be divided into multiple phases of data transfer, processing and storage

II. PRISM: PHASE-LEVEL SCHEDULING ALGORITHM

The Hadoop Distributed File System (HDFS) is the existing system that uses the task-level scheduling algorithm [6]. The task-level scheduling algorithm suggests that the task run-time resource usage allocates a fixed sized container for each task can lead to inefficient scheduling decisions. The problems in allocating a fixed size container are: (i) if the resource allocated to a task is higher than current resource usage, then the idle resources are wasted, and (ii) if the resource allocated to the task is much less than actual task resource demanded, then the performance bottleneck occurs. In order to overcome the problem with the existing task-level scheduling algorithm, the technique called the PRSIM that uses the phase-level scheduling is designed.

PRISM, a Phase and Resource Information-aware Scheduler for MapReduce clusters uses the phase-level scheduling algorithm with the aim of achieving high job performance and resource utilization. By considering the resource demand at the phase level, it is possible for the scheduler to achieve higher degrees of parallelism while avoiding resource contention.

The phase-level scheduling scheme allocates the resources according to the phase that each task is executing. The main issues in the phase-level scheduling scheme is, if a task has completed a phase, the subsequent phase of the task may not be scheduled immediately, if the machine does not have sufficient resources. The solution to the issues in the phase-level scheduling is, the execution of the phase is "paused" to avoid resource contention, by delaying the completion of task [3]. Thus the average job running time will be improved when compared to task-level resource-aware scheduler.

III. PHASE-LEVEL SCHEDULING ALGORITHM WORKING

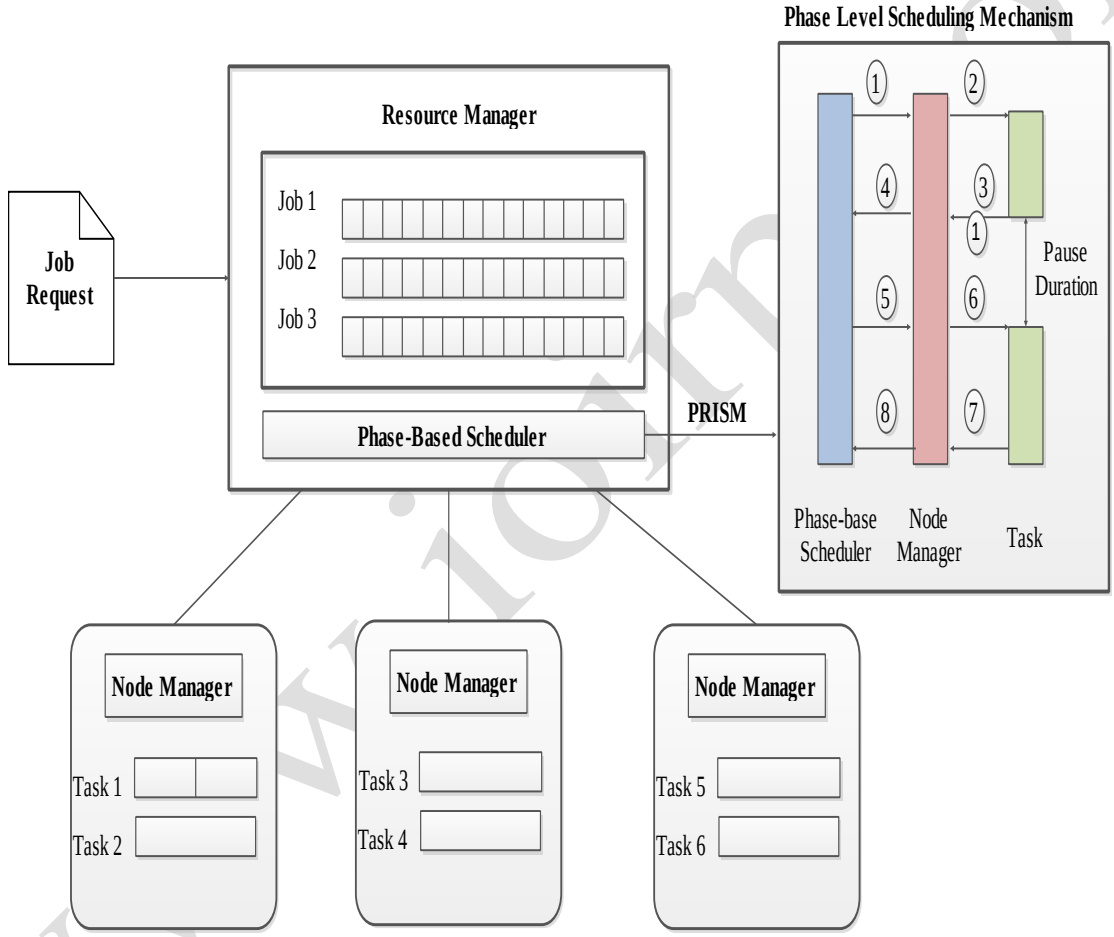

Figure 1.1 System architecture

Figure 1.1.will describe about the system architecture. The resource manager consists of the:(i) Phase-base scheduler and (ii) Job progress monitor. When the job request is received by the Phase-base scheduler, the phase-base scheduler divides the job into small tasks. For each task, a node manager is allocated; the node manager is allocated a threshold amount of space by the phase-base scheduler. When there is no enough memory, node manager will pause the job execution and request for the additional space from the phase-base scheduler. Once the additional space is gained, it starts the job execution. The job progress monitor monitors the total space and the number of job request arrived.

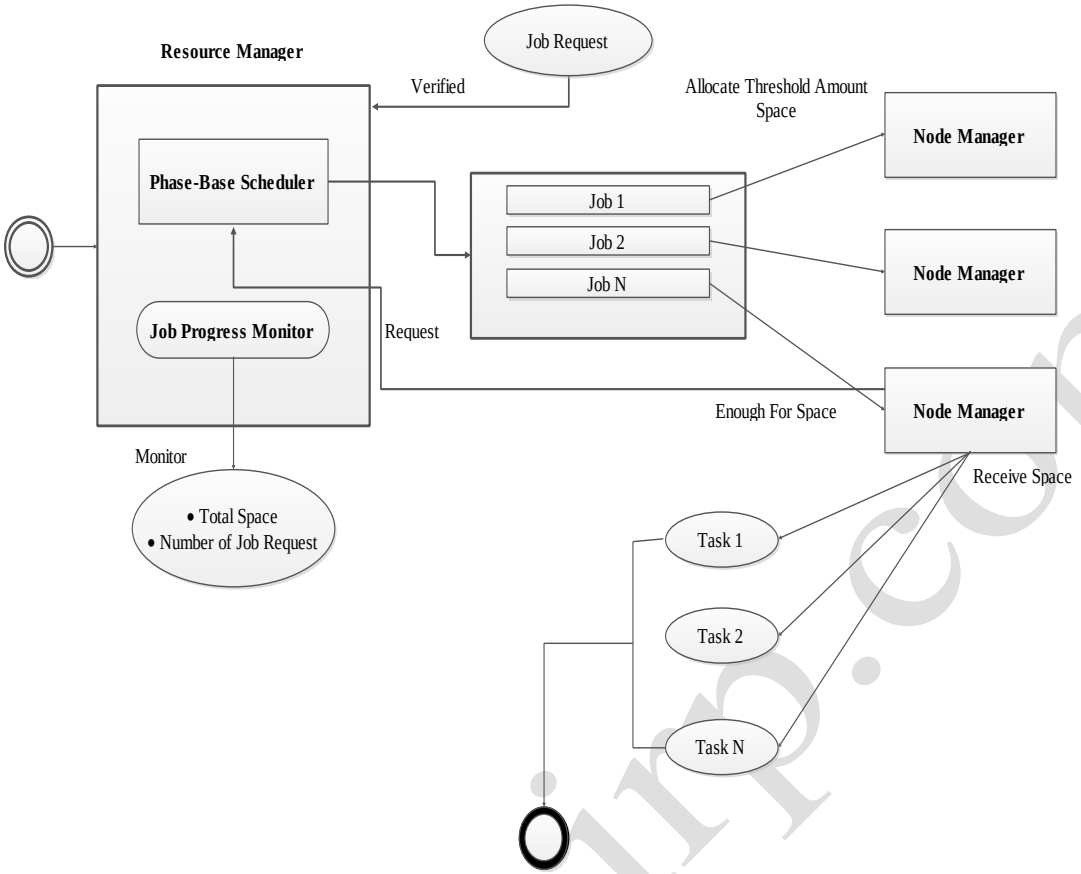

Figure 1.2 Workflow execution

Figure 1.2.Defines the overall workflow of the phase-level scheduling algorithm.The following steps will demonstrate the working of the phase-level scheduling algorithm:

(i) When a task needs to be scheduled, the scheduler replies to the heartbeat message with a task scheduling request.
(ii) The node manager then launches the task. If there is enough space available to execute the task, then the execution of the phase will be carried out. On the other hand, if there is not enough space available to execute the scheduled job then the node manager will "pause" the execution of the phase and request the phase-based scheduler for additional space. Once the additional amount of memory is allocated then the scheduler will resumes the execution of the paused job.
(iii) Each time a task finishes executing a particular phase (e.g. shuffle phase of the reduce task), the task asks the node manager for a permission to start the next phase (e.g. reduce phase of the task).
(iv) The local node manager then forwards the permission request to the scheduler through the regular heartbeat message.
(v) Given a job's phase-level resource requirements and its current `progress information, the scheduler decides whether to start a new task, or allow a paused task to begin its next phase (e.g., the reduce phase), and then informs the node manager about the scheduling decision.
(vi) Finally, once the task is allowed to execute the next phase, the node manager grants the permission to the task process.
(vii) Once the task is finished, the task status is received by the node manager.
(viii) Then the node manager forwards the task status to the scheduler.

If the phase-level scheduling is needed to be done, then the phase-level resource requirement of each job is required. PRISM is ideal for the jobs that are executed repeadtly with same input size. (i.e.) common in production environment [7].

IV. COMPONENTS

The various components of PRISM are: (A) Phase base schedulers, (B) Job Request Verification, (C) Node Manager Allocation and (D)Job Progress Monitor.The working of each of the components is elaborated in the following section.

A. Phase base scheduler

The Phase base scheduler is the master node that is located inside the Resource Manager. The resource manager is also called as the job tracker. This scheduler scheme has been setup for dynamically allocating the space. They dynamically schedule the job request using phase-level scheduling algorithm, with the aim of achieving high job performance and resource utilization. The Job progress monitor is also present inside the Resource Manager.

The main job of the Job progress monitor is to capture the phase-level progress information. This information will contain the details such as the: Total amount of space on resource manager, number of job request, node manager allocation space details etc. By considering the resource demand at the phase-level, it is possible for the scheduler to achieve high degree of parallelism by avoiding resource contention.

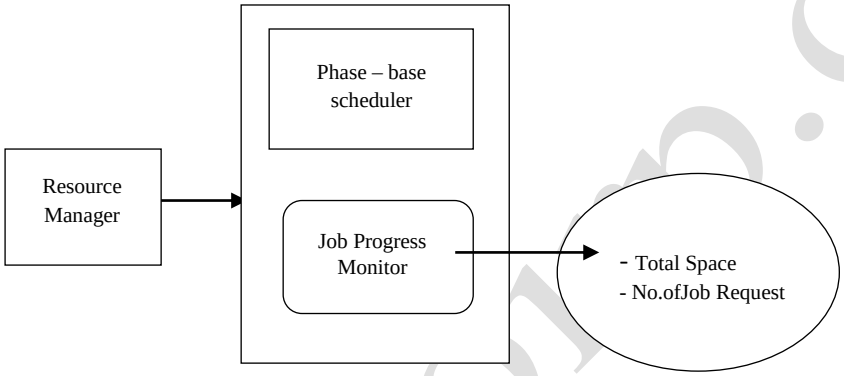

Figure 1.3 Phase base scheduler

B. Job request verification

The each job request has been verified by the resource manager and after completing the verification, the job will be allowed by the phase base scheduler. The entire jobs are dynamically received and allocate the node manager. The total amount of space in the Resource manager will be send to the phase base scheduler. If the verification of the job request is successful, it will be allowed to process further. Once the verification is finished successfully, the job request will be then send to the node manager, to allocate the space required for each job to execute.If the verification failed, then the failed job will be scheduled for job request.

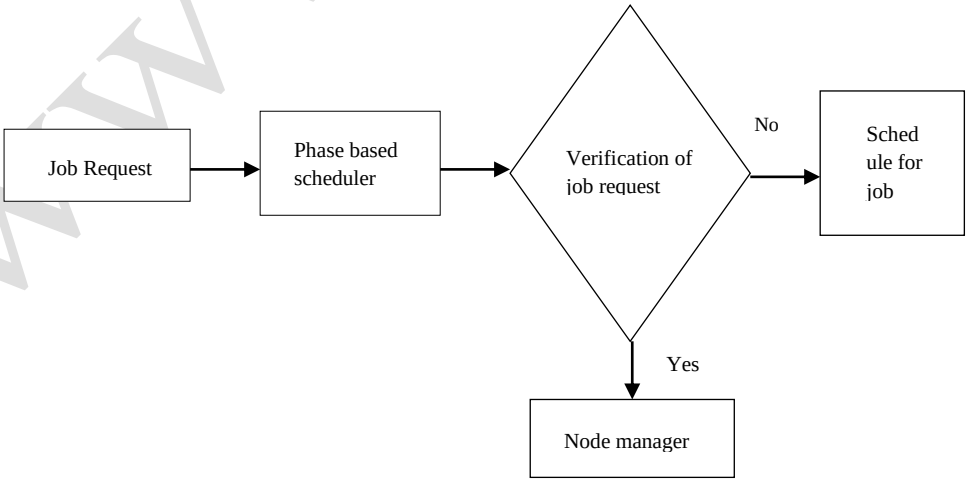

Figure 1.4 Job Request verification

C. Node manager allocation

The node manager is responsible for launching and allocating resources for each task, to accomplish this, node manager launches Java Virtual Machine (JVM) that executes the map and reduce tasks. Node manager monitors the progress of each running task and available resources on the node and transmits a heartbeat message to convey this message to resource manager.

The job request is allocated to each node manager for the phase base scheduling; some threshold amount of space has been allocated the node manager for scheduling the task. If particular amount space is not enough for the process of completing the content updating process, then it requests to scheduler and receives the additional space. Once the additional space is obtained, the content updating process will be finished successfully. The node manager is also called as task manager.

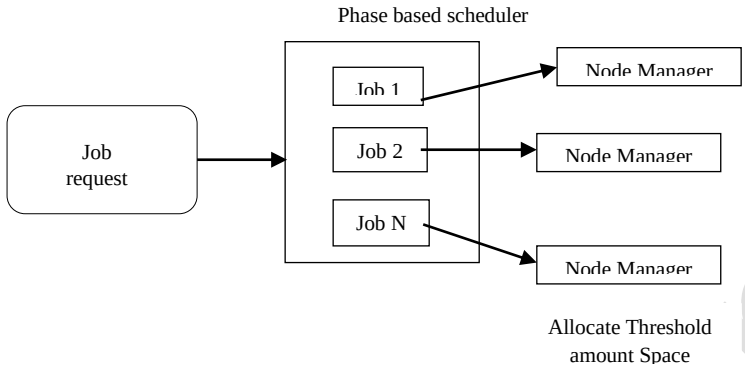

Figure 1.5 Node Manager Allocation

D. Job progress monitor

The main aim of the job progress monitor is to capture the phase-level progress information. The node manager after receiving the space, it launches the task. If the task finished executing a particular phase, the task asks node manager, permission to start next phase.

Once the task is finished, the node manager receives the task status and forwards it to the scheduler. If the phase-level scheduling need to be done, then the phase-level resource requirement of each job is required.

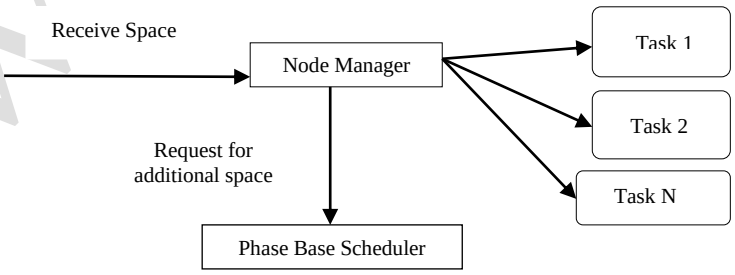

Figure 1.6 Job Progress Monitor

V. CONCLUSION

MapReduce is a popular programming model for data intensive computing. Existing work mainly focuses on designing task-level schedulers, and in which the execution of each task can be divided into phases with different resource consumption characteristics. To address this limitation, PRISM a fine-grained resource-aware scheduler that coordinates task execution at the level of phases is used. This phase-level job scheduling algorithm improves job execution without introducing stragglers. PRISM, a phase-level resource-aware scheduler offers high resource utilization and provides improvement in job running time compared to the current Hadoop schedulers.

The flexibility of phase-based scheduling should allow the scheduler to improve both resource utilization and job performance, which is still a challenging problem. This is because pausing the task execution at run-time may delay the completion of the current and subsequent task and increases the job completion time. To overcome this limitation, as a future work, the phase-based scheduler allocates some amount of cloud space for node manager. At the same time it also allocates the virtual type space using the virtual management mechanism (VCM). This virtual space will be used by the node manager in the pause duration. The"pause" time will result in delaying the execution, so to overcome this delay, in pause time, virtual space is used to load the job content. Once the Node Manager retrieve the original space from phase base scheduler, the node manager will shift the loaded content in the virtual space to the original space in node manager.

References

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

[2] T. Condie, N. Conway, P. Alvaro, J. Hellerstein, K. Elmeleegy, andR. 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] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. Cetin, andS. Babu, "Starfish: A self-tuning system for big data analytics,"in Proc. Conf. Innovative Data Syst. Res., 2011, pp. 261–272.

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

[6] Qi Zhang, Mohamed Faten Zhani, Yuke Yang, Raouf Boutaba, 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 provisioningframework for MapReduce jobs with performancegoals," in Proc. ACM/IFIP/USENIX Int. Conf. Middleware, 2011,pp. 165–186.

[8] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker,and I. Stoica, "Delay scheduling: A simple technique for achievinglocality and fairness in cluster scheduling," in Proc. Eur. Conf.Comput. Syst., 2010, pp. 265–278.