International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Orchestrating an Ensemble of MapReduce Workflow with Budget and Deadline constraints in Heterogeneous Clouds

Harsha Daryani

Computer Department, Pune University, Pune, Maharashtra, India

Abstract: ***Cloud computing provides enchanting option for businesses to lease a suited size MapReduce cluster, use resources as a service, pay up only for resources that were used. A major challenge in such an environment is to enhance the consumption of MapReduce clusters to understate their cost. One of the way for obtaining this goal is to make execution of MapReduce jobs on the cluster optimum. This paper is considering MapReduce framework, Hadoop File System, the various work on scheduling in MapReduce. In addition to that, task level scheduling algorithms to address budget and deadline restraints for MapReduce workflow is considered. This has been done on heterogeneous machines in clouds. Heterogeneity is demonstrated in the "pay-as-you-go" model where the machines with varying performance would have varying service rates.***

Keywords: MapReduce scheduling, Cloud computing, Hadoop, batch workloads

1. Introduction

Due to their ample on-demand computing resources and elasticized billing models, Cloud computing is becoming as a promissory platform to address various issues like data processing and task computing and granularity problems[1]. The data stored on web is raising drastically with respect to time, this motivates the cloud computing concept. MapReduce portrayed by its noteworthy simplicity, fault tolerance, locality and scalability, is emerging as a popular programming model to automatically parallelize extensive data processing.

1.1 MapReduce Framework

MapReduce is programming model which is used for large datasets application in cloud computing environment. Knowledge of Workload Characterization in MapReduce is useful for both service providers in cloud and users in cloud. For better scheduling decisions, service providers use this knowledge and for learning aspects of job impacts users can use this knowledge.

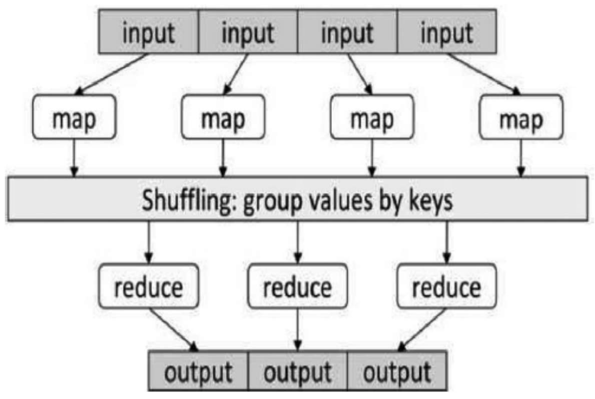

Figure 1: The MapReduce Framework

In MapReduce Framework, input is provided by users. The input is regarded as *map* function that operates a key/value pair to yield a batch of intermediate key/value pairs, a *reduce* function that unify all intermediate values that relate with the same intermediate key. Numerous real world tasks are representable in this model [2]. Figure.1 shows The MapReduce Framework.

MapReduce runs on a huge cluster of machines and is highly ascendable: an emblematic MapReduce computation operates a large number of terabytes of data on thousands of machines. It is designed in such a way that it abstracts the messy details of computation like data distribution, load balancing, task granularity, backup tasks in a library, and lets users focus on programming model.

1.2 Hadoop

Hadoop has been widely used nowadays by many companies including AOL, Amazon, Yahoo, Facebook. Apache Hadoop [3] is an open source effectuation of the Google's MapReduce. Hadoop abstracts the details of parallel processing that includes partitioning of data to different processing nodes, restarting the failed tasks, and combining the results after computation. Moreover, this model allows developers to write out parallel processing code that emphasizes on their computation problems, instead of parallelization issues.

1.3 Hadoop Distributed File System (HDFS)

HDFS [4] is inspired by Google File System (GFS). GFS is a patented distributed file system originated by Google and specially planned to provide effective, trustable access to data using a large number of clusters of commodity servers. Files are partitioned into chunks of 64MB. However, it is usually affix to, or read, and only seldomly overwrites or shrivel. HDFS stores large files over a number of machines. It obtains reliability by duplicating the data across multiple servers. Likewise to GFS, multiple copies of data are storage on multiple compute nodes to offer reliable and rapid computations. The data is also put up over HTTP, providing access to entire content from a web browser.

Figure.2 shows HDFS consisting of one NameNode and many DataNodes in a group of linked computers. NameNode is responsible for associating data blocks to DataNodes and also for carrying file system operations such as, opening, closing, renaming files, etc.

After getting instructions from NameNode, DataNodes carries out tasks such as creating, deleting, and replicating data blocks. NameNode takes decision about replication of data blocks. For a standard HDFS, block size is allotted to be 64MB and replication factor to be 3.

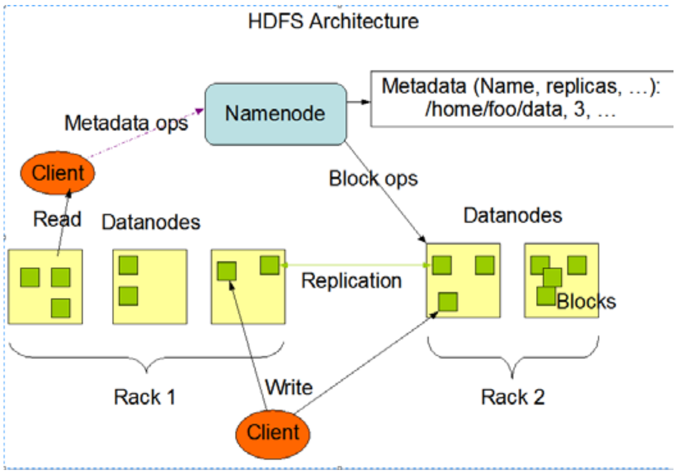

Figure 2: Hadoop Distributed File System

2. Scheduling in Hadoop MapReduce

2.1 Default FIFO Scheduler

The default Hadoop scheduler functions using a FIFO queue. As the jobs arrives, it is partitioned into a number of tasks, then they are loaded up in the queue and allotted to free slots as they turn obtainable on TaskTracker nodes. Originally, each job uses complete cluster, so other jobs had to hold off for their turn. This is the major disadvantage of this scheduler. The problem of sharing resources evenhandedly between clients requires a better scheduler.

2.2 Fair Scheduler

The Fair Schedular was designed at Facebook [5] to contend access to their Hadoop cluster, then it can be released to the Hadoop community. Its goal is to provide every user with fair share of cluster resources over time. Moreover, users may allocate jobs to pools, where each pool is assigned with a guaranteed nominal of Map and Reduce slots. The unused slots in idle pools can be assigned to other pool, which gives the benefit of sharing excess capacity. It supports preemption, i.e. when a pool didn’t received its fair share over a time period, then the scheduler can discard tasks in pools which are running in over capacity. This scheduler overcomes the challenges of Default FIFO Scheduler by providing each of its users a fair share of cluster capacity.

2.3 Capacity Scheduler

Capacity Scheduler [6] originally designed at Yahoo focus on a usage scenario where there are large numbers of users. However, these large number of users need to be provided with a fair assignment of computation resources. The Capacity Scheduler assigns jobs based on the users submitting to queues with configured numbers of Map and Reduce slots. This scheduler has the benefit of dispensing cluster capacity amongst users, in spite of dispensing amongst jobs, as it was in the Fair Scheduler.

2.4 Dynamic Priority Scheduler

Dynamic Priority Scheduler was proposed by Thomas Sandholm and Kevin Lai [7]. It provides capacity apportioning dynamically amongst co-occurrent users depending on the priorities of the users. Automatic capacity assignment and redistribution is sustained in a controlled task slot resource market. This method provides users with adequate proportion of Map or slots. Moreover, Hadoop MapReduce supports division of larger jobs to smaller jobs to assure that fewer co-occurrent tasks runs by using the same amount of resources. This framework imposes the difficulty scheduling to satisfy the SLA and deadline requirements.

2.5 Resource Aware Scheduling

Fair Scheduler and Capacity Scheduler mentioned above provides fair allocation of capacity amongst users and jobs without including resource accessibility. Resource Aware Scheduling [9] is becoming a research challenge in Cloud Computing. The scheduling technique in Hadoop is centralized, and worker initiated. The decisions of scheduling are interpreted by a master node, known as the JobTracker, on the contrary, the worker nodes, known as TaskTrackers are accountable for task execution. JobTracker keeps a queue of currently running jobs, positions of TaskTrackers in a cluster, and also the details of tasks assigned to each TaskTracker. Every Task Tracker node is configured with an utmost number of obtainable computation slots. Although each node can be configured to signify the existent processing ability and disk channel speed up available on cluster machines. In this method, each Task Tracker node administer resources like CPU utilization, disk channel I/O. These are the basic resources that must monitored completely to ameliorate the load balancing on cluster.

3. Budget and Deadline driven Scheduling

Budget and Deadline driven scheduling [1] is accomplished on a fine-grain level i.e. task level. Jobs i.e. coarse-grain in this case, is subdivided into smaller tasks. The batch of jobs is orchestrated as a k-stage workflow. Two optimization problems, relying on whether the constraints are based on pecuniary budget or on completion within deadline has been described.

A set of MapReduce jobs can have multiple levels of MapReduce computation, each level runs either map or reduce tasks in collimate, with enforced synchronization amongst them. Therefore, the batch of MapReduce jobs is viewed as multiple synchronized stages, each comprising of an ensemble of sequential and/or parallel map/reduce tasks.

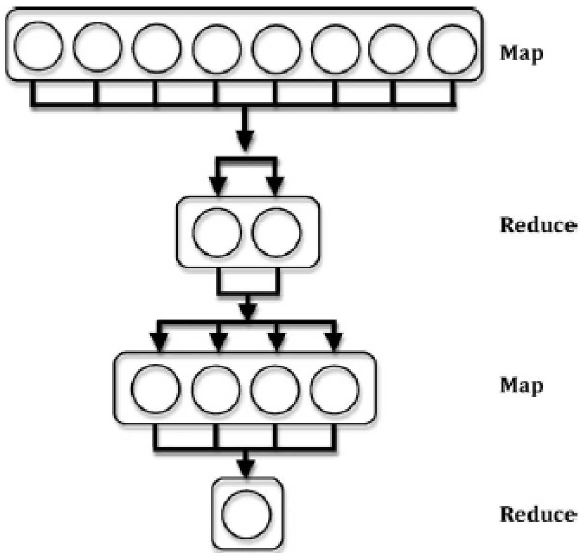

Figure 3: Four-Stage MapReduce Workflow

An example of workflow is depicted in figure 3. It constitutes 4 stages, with 8,2,4,1 map or reduce tasks. They are scheduled on distinct nodes for parallel processing. However, in heterogeneous clouds, distinct nodes may have disparate performance and/or configuration specifications, and hence they may have diverse service rates. This scheduling method focuses on two optimization problems:

1) Given a determinate budget B, how to effectively choose a machine from a probable set of machines for every task so that the entire scheduling length of the workflow comes out to be minimum without breaking the budget. To solve this problem, an effective in-stage greedy algorithm is devised for finding the minimum execution time within a given determinate budget for every stage. With the help of results of in-stage greedy algorithm, a dynamic programming (DP) algorithm is designed to achieve a global optimal solution.

2) Given a determinate deadline D, how to effectively chose a machine from set of probable machines for every task so that the entire pecuniary cost of the workflow comes out to be minimum without missing the deadline. This problem can be reduced to classical multiple-choice knapsack (MCKS) problem through a parallel transformation. The two problems can be effectively resolve if the budget B, and deadline D are polynomially restricted by the total number of tasks, and the total number of stages, in the workflow.

4. Conclusion and Future Scope

Job/task scheduling in MapReduce environs may pursuit diverse performance goals, and hence, might be driven by different SLOs. This paper summarizes the different scheduling policies of Hadoop Schedulers designed by diverse communities. Moreover, scheduling a set of MapReduce jobs as a workflow upon a set of heterogeneous machines in the cloud has been discussed. This scheduling methodology basically takes into account two constraints on budget and deadline. The scheduler described here, take into account resources such as CPU, memory, budget, deadlines etc. All the schedulers mentioned in the paper consider one or more problems while scheduling in Hadoop.

As Future Work, some advanced features in Hadoop like speculative task scheduling, dynamic pricing, redundant computation for fault tolerance can be considered in job/task scheduling. Enforcing the complete system in a real cloud computing environment (Eg. Amazon) can also be undertaken.

References

[1] Yang Wang and Wei Shi, “Budget-Driven Scheduling Algorithms for Batches of MapReduce Jobs in Heterogeneous Clouds”, IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL. 2, NO. 3, JULY-SEPTEMBER 2014.
[2] Jeffrey Dean and Sanjay Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters”, Commun. ACM Jan. 2008.
[3] Apache Hadoop, http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
[4] Hadoop Distributed File System, http://hadoop.apache.org/hdfs
[5] Hadoop’s Fair Scheduler http://hadoop.apache.org/docs/r1.2.1/fair_scheduler.html
[6] Hadoop’s Capacity Scheduler, http://hadoop.apache.org/docs/r1.2.1/capacity_scheduler.html
[7] Thomas Sandholm and Kevin Lai, “Dynamic Proportional Share Scheduling in Hadoop”, E. Frachtenberg and U. Schwiegelshohn (Eds.): JSSPP 2010, LNCS 6253, pp. 110–131, 2010. c_Springer-Verlag Berlin Heidelberg 2010.
[8] Abhishek Verma, Ludmila Cherkasova, and Roy H. Campbell, “Orchestrating an Ensemble of MapReduce Jobs for Minimizing Their Makespan”, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 10, NO. 5, SEPTEMBER/OCTOBER 2013.
[9] Mark Yong, Nitin Garegrat, Shiwali Mohan: “Towards a Resource Aware Scheduler in Hadoop” in Proc. ICWS, 2009, pp:102-109

Author Profile

Harsha Daryani received the B.E. degree in computer from Pune University. She is pursuing her M.E. degree Computer Engineering) from Pune University. Her interest’s areas are Distributed Systems, Data Mining, Android, and Ubiquitous Computing.