I J C T A, 9(15), 2016, pp. 7597-7602

Distributed Scheduler of Multiple-Map Reduce Jobs for Big Data

Paripelly Shiva Rama Krishna* and J. Anitha**

ABSTRACT

In order to support a large user base, the size of data centers has increased and this has lead to increase in their energy consumption. Most of these data centers contain Hadoop Map Reduce clusters of hundreds and thousands of machines, to process the infrequent batch and interactive with big data jobs, the applications need to get executed on large bases requiring large amount of energy, this adds to the overall data center's cost. While executing each Map Reduce, we have to make sure that lowering of overall resource consumption occurs. In this paper, the energy efficiency of Map Reduce applications is restored using a framework, while satisfying the (SLA) Service Level Agreement. In order to achieve this, we will perform a complete energy characterization of Hadoop Map Reduce and create its energy usage- model. Then analysis done by using this replica will assist in configuring energy well-organized Hadoop Map Reduce. The energy characterization will be worked to derive examining for designing energy aware scheduling algorithm. That the assignments of a map and reduce tasks to the machine slots to a minimum energy consumed when executing the application. We perform extensive experiments on a cluster of a system to determine the energy consumption and execution times for several workloads from the HiBench benchmarking tool includes TeraSort, PageRank, and K-means Clustering, and then use this data in a graph model and then study to evaluate the work of the proposed algorithms. In order to improve the energy efficiency of Map Reduce clusters the energy-aware scheduling is used thus helps in minimizing the energy cost of the data center and operational costs.

Keywords: Big data; Map Reduce; benchmarking; minimizing energy consumption; scheduling; Hadoop

1. INTRODUCTION

Organizations and businesses are facing with an ever growing challenge of analyzing large amounts of database. The task is very challenging and asks for novel approaches and technologies to cope up with the present condition. Data intensive application processing should be done with minimum energy costs that are the daunting task currently to be undertaken. Data reveals that in 2010, data centers in US consumed nearly 2% of total electricity used nationwide. Also, the energy that is spent by data centers is growing at over 15% annually and about 42% energy costs frame the data centers' operating costs. As the server costs are consistently falling, cost of energy will involve a large proportion of the total data center costs'. The increase in energy consumption of data centers has become major issue of concern. The U.S. (EPA) published a report exposing the two-fold increase in energy usage by the data centers from 2000 to 2006 and displaying same increase again from 2007 to 2011. As organizations use MapReduce for efficiently processing their huge volume of data, the Hadoop MapReduce clusters form a major part of today's data centers. The lesser energy consumed by the Hadoop node clusters will contribute significantly towards improving the data centers resource capability in general. The MapReduce programming model breaks a data processing into small tasks and executes them across multiple systems for greater performance of big data jobs. To support top level workloads, large data volume, and high fault tolerance in data centers, the Hadoop node clusters consisting of several hundreds and thousands of machines are created in data centers. A high peak-to-mean

* Department of Computer Science and Engineering SRM University Chennai, India, *Email: shivaram0456@gmail.com*

** Department of Computer Science and Engineering S.R.M University Chennai, India, *Email: anitha.jo@ktr.srmuniv.ac.in*

ratio of workloads and numerous copies of data sets make these clusters energy inefficient because they are under-utilized and consume a high-level power most of the time.

Therefore, optimization of energy utilized by data center is needed. Big data Hadoop environment runs on large clusters within data centres. MapReduce and its open-source implementation, Hadoop, have evolved as the leading platform for computing platforms for big data analytics. Hadoop works on a FIFO scheduler, for scheduling multiple MapReduce jobs. Hadoop then employed the Fair Scheduler in order to overcome the issues with the waiting time in FIFO. However, these schedulers do not consider developing the energy efficiency when executing large MapReduce. Making MapReduce applications energy efficient leads to a significant reduction of the total cost of data centers. Also, we design MapReduce scheduling algorithms that improve the energy efficiency of working on each application while satisfying the service level agreement (SLA).

1.1. Evaluations for scheduling of different Map-Reduce Algorithms

The overall proficiency of any approach is measured using efficient technique results were used. The algorithms discussed above were evaluated on various aspects. As seen, the experiments in multiple user environments are not done by many papers and no consideration of increasing number of nodes in map-Reduce clusters is done in most of the papers.

1.2. Energy Efficiency in Hadoop Map-Reduce Systems

To handle a large number of user requests and to perform analytical evaluations, the size of data centers is being increased. Their operational costs are increasing, and the report from EPA showed that the energy utilized by data centers has been increased from 2000 to 2006 and projected that it is expected to double again from 2007 to 2011. So, the energy consumption of servers was regarded to be very important for research. Various machines in a data center are also considered to avoid utilization of any useful resource.

1.2.1. User jobs

a) Fair Scheduling Algorithm: In the system Fair scheduling (FS) algorithm performs an equal distribution of computing resources among the users/jobs. If more user is present then one pool is assigned to each user. The task of scheduling algorithm is to equally share the resources among these pools.

b) Capacity Scheduling Algorithm: The main objective of Capacity scheduling is maximizing throughput and resource utilization in an environment of multi-tenant type cluster. The design and work resembles

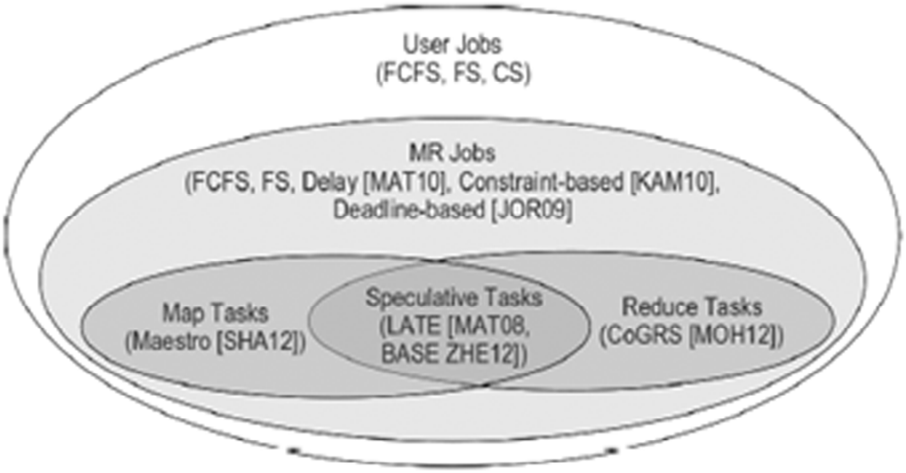

Figure 1: Map-Reduce Cluster's hierarchical scheduling policies

to Fair scheduling. Queues are also used here. Resource management and arrangement is done for every queue which are allotted to organization. Security is implemented to check that organization uses only the queue allotted to it and not using the queue, tasks or jobs allotted to other organization. The coding of algorithm is done in XML format in the Hadoop yarn. The Yarn manages the resource focuses exclusively on scheduling, and it makes easy to manage max Hadoop node cluster.

1.2.2. Job Level

The job level scheduling policy which is by default available in Hadoop MR are First Come First Serve, Priority based. The productiveness may not be same for all models and requirement, so for selecting a job for meeting the different administrator's needs new principles have been proposed.

a) FIFO Scheduling Algorithm: The FIFO scheduler schedules the job based on their priorities. This algorithm is the default Hadoop job level scheduling which also contains five priority levels. When the heartbeat is received by the scheduler from nodes to the HDFS indicating map/reduce slot is free:

- Scanning is done at first through the main list of jobs to find a job having the highest priority and the oldest submit time. If this job has a waiting in slot type, then that task is selected else next task in the order is picked. This process is repeated until the task is matched.
- Next if it is a map slot, then for the data locality which is achieved the scheduler consults NameNode which acts like meta data that gives direction for picking a map task for the job which is having data which is closest to this free slave.
- If it's a reduce slot, then to the node any reduce task is scheduled. Hadoop MapReduce doesn't wait for the completion of all the map tasks for a reduce task, so to have a better turn-around time the map task execution and shuffling of intermediate data can be made to run in a parallel. This process can also be called early-shuffle.

b) SLA-based Scheduling Algorithms: The large amount of organizational data is stored in distributed file systems for attaining higher reliability and availability. This input data is for long batch jobs and also for many users jobs. Algorithm used for long batch jobs and many user jobs is constraint-based Hadoop scheduling algorithm. Aim of this aalgorithm is to maintain minimum number of map/reduce tasks provided by the cost model of job execution to meet the SLA. The tenacity of the job cost model is to discover the minimum number of map/reduce tasks which are required to meet deadlines on the basis of the given time, arrival time, input data size and map/reduce execution costs.

1.2.3. Task Level

MR programming model creates two types of map and reduce task which is needed to be scheduled in the respective slots. Another type of job created by MR during runtime is a tough to handle straggler map/ reduce tasks. These needs to be scheduled on schedules in a map or reduce slot which depends upon the type of task it is duplicating. There are many scheduling algorithms for selecting a job for the given free slot type which is free to carry objectives like improve resource usage, data locality andperformance growth.

a) Map-Task Level: The data-locality criteria is used by default MR scheduling algorithm to select a map for a given node. Initially, a Job-Tracker is present whose task is to assign the map tasks to the slaves depending on the slots capacity of the Task-Tracker and considering data locality. At run time, when an empty slot to process map task is reported by Task-Tracker reports then Job-Tracker checks the pool for the map tasks and also consults the metadata storage service to get hosted chunks. Finally the data blocks/chunks with the default size of 64mb or 128mb is present with the data node. Then the input record is converted as a key value pair.The map task has three different tasks: failed map tasks with the highest priority, normal map task and speculative tasks with the lowest priority.

- Replica-aware scheduling algorithms

  The experiment results in a high percentage (23%) of non-local map tasks executions with the default map scheduling algorithm. The result of this impacted execution time is max of speculative tasks executions (55%) out of which only 50% were considered meaningful. All of these increased the response time of the job. Its result shows the local map tasks cause much lower probability speculation when compared to non-local jobs. Also the observation shows that the non-local executions caused imbalance of successful map tasks among several identical nodes.

b) Reduce-Task Level: After the completion of the map task the output is given to the reduce and in case if reduce is not given then by default it is taken by reducer identifier, and then MR starts scheduling its reduce job in order to run the map task execution and shuffling of intermediate data to have a better turn-around time. It is also known as early shuffle. Hadoop MR randomly selects the reduce job task for scheduling for the available reduce slot. For Hadoop reduce job scheduling there are very few improvements that has been taken.

Locality-aware scheduling algorithms

Collecting the output data from the map nodes is the reduce task. Different problems like network congestion and performance problems may arise due to random reduce task scheduling by the Hadoop Job Tracker. The map task gets input by the data node and gives the output which is considered as the input to reduce which is not close to it, this causes various data shuffling and network traffic problems. One more problem could be of partitioning skew. Both these problem can impact the application performance. So in-order to address these types of problems a task scheduler is proposed known as a Center of Gravity reduce task scheduler.

1.3. Speculative-Task Level

In Hadoop, if a system node is available but is performing poorly i.e. Running slowly then the condition is called as straggler. MapReduce task is finish the computation faster by running a speculative copy of its task on another machine. The closest node is searched in-order to complete the given work.. The goal of speculative execution is maximizing the response time of the job. It runs on a simple heuristic in in which the comparison is made for each task to the average progress. Hadoop uses parameter called Progress Score to monitor the task improvement which has the value between 0 and 1.

Latency-aware scheduling algorithms:

This algorithm addresses the problem of performing a speculative execution to maximize the performance. The proposed Longest Approximate Time to End (LATE) algorithm has three principles: prioritize tasks to speculate, select fast nodes to run on, and cap speculative to prevent thrashing. Pparameters of Longest Approximate Time to End algorithm are: SlowNodeThreshold, SpecultiveCap and SlowTaskThreshold.

2. RELATED WORK

Developing the energy efficiency of Hadoop MapReduce clusters without impacting it’s their quality is a challenging objective for the researchers to achieve. The goal is achieved by creating

- The predictive models which is used to perform recommendations for the energy efficient cluster and job configurations.

- An energy-adaptive scheduling algorithm which performs the productive scheduling of MapReduce tasks.

The performance of job is dependent on the hardware mechanism and platform configuration settings and the energy consumption of the MapReduce jobs is an action performed by the response time and the hardware features. So, we will develop methods that will consider critical parameters of all the layers of MapReduce architecture.

2.1. Performance of Hadoop MapReduce cluster and Empirical characterization of energy

Depth study of energy used by map-reduce node group for different types of workloads, data volumes, and configuration settings at all layers of Apache Hadoop MapReduce marks the starting of the project. The product of Power and Time is the energy. Many parameters are present which impact the Hadoop MapReduce performance. Analyse the newer hardware capabilities of managing power on the MapReduce consumption of energy by changing the CPU frequency of machines.

2.2. Energy & Performance models for MapReduce

Then we create the energy and performance models for MapReduce framework which is used to predict the energy used and performance of jobs with various Hadoop configuration settings.The plan to use the multivariate regression modeling on the data collected from the energy reading of the Hadoop MapReduce to create these models to control. The parameters added in a model which by getting output by doing the fractional factorial analysis of results of the energy characterization done using the max and min possible values of all the parameters mentioned above. Then create and verify the stochastic Markov chain models for the MapReduce systems to predict the performance and energy by making use of data collected from energy characterization.

2.3. Energy-aware MapReduce task scheduling algorithm

The scheduling algorithm is used to arrange the execution order and the distribution process of the jobs on the nodes which drives the performance of the jobs. Most of the scheduling algorithms task is improving the work performance and resource management of map-reduce clusters. Our plan is to study what will be the performance of these scheduling algorithms on the Map-reduce energy. We also want to explore the clubbing of scheduling algorithm with different power management and energy efficiency techniques or methods and check if it makes a better energy efficiency improvement. Studies will be done for energy characterization and above study results to derive the heuristics which can be used for designing an energy aware MapReduce task and job scheduling algorithm.

3. ROUND ROBIN AND PRIORITY ALGORITHM

The operating system assigns a fixed priority to every process, and the scheduler arranges the processes in the ready queue in the priority order. Lower priority processes get interrupted by incoming higher priority processes. Response time and waiting time both are dependent on the priority of the process. Higher priority processes have less waiting and response times. Deadlines have to be meet by giving higher priority to the earlier given time processes.

Disadvantage: Starvation of lower priority processes can be possible if there are large number of higher priority processes that keep on arriving one after the other continuously.

Proposed:

- The proposed architecture focuses on the shortcoming of simple round robin which gives equal priority to all the processes. Because of this drawback round robin is not capable of smaller CPU burst. This increase in waiting time and response time of processes which output in the decrease in the system throughput.

- The introduced algorithm will perform in two steps which will help to minimize the number of performance parameters such as average turnaround time, context switches and average waiting time. The algorithm performs following steps:
- Step 1: Allocate CPU to every process in the round-robin way, for the priority, the given time quantum (say k units) only for one time.
- Step 2: After the first following stages are performed:
- A) Processors are in increasing order or their remaining CPU burst time in the ready queue. In order to assign new priorities according to the remaining CPU bursts of processes, i.e., the process with shortest remaining CPU burst with the highest priority.
- B) The processes are executed according to the new priorities based on the remaining CPU bursts, and each process gets the control of the CPU until they finished their execution.

ACKNOWLEDGMENT

Due to the growing need for large data processing and the high-level adoption of MapReduce and its open source exercises Hadoop for such processing increase, improving MapReduce performance with energy saving objectives can have a more impact in reducing energy used in data centers. In this paper, we show that there is significant optimization opening regarding resource consumption within the MapReduce. We proposed a combination of two scheduling algorithm Round Robin and priority to save the energy of the data centers. Both the algorithms which are proposed give a very fast solution which makes them efficient to run in the real-time environment. The experiment was performed on the Hadoop cluster in order to determine the energy consumption of MapReduce benchmarking such as TeraSort and PageRank and then its data is used to evaluate the performance of Round Robin and Priority scheduling. The result shows that the proposed algorithms are capable of getting near optimal solutions leading to significant energy savings. In the future, plan can be made for the design and implementation of a distributed scheduler for multiple MapReduce jobs having primary focus on energy consumption.

REFERENCES

[1] B. EPA: EPA Report to Congress on Server and Data Center Energy Efficiency.U.S. Environmental Protection Agency (2007). http://www.energystar.gov/ia/partners/prod_development/downloads/EPA\ _Datacenter final report.

[2] Jain, Raj: 2kÀÛÜpFractional Factorial Designs. http://www.cse.wustl.edu/~jain/cse567-08/ftp/k_19ffd.pdf (2008)

[3] Trivedi, Kishore: Probability and Statistics with Reliability, Queuing, and Computer Science Applications.John Wiley and Sons, New York (2001)