I J C T A, 9(34) 2016, pp. 199-209
© International Science Press

Distributed Scheduler of Multiple Map Reduce Jobs for Big Data

J. Anitha[1], B. Ida Seraphim[2] and B. Sowmiya[3]

ABSTRACT

The increasing size of the data-centers to support a high number of users, has led to many-fold increase in their energy consumption. Most of these data-center's contain Hadoop Map Reduce clusters of hundreds and thousands of machines, to process the infrequent batch and interactive with big data jobs Such applications are executed on large clusters requiring huge amounts of energy, making the energy costs a considerable fraction of the data center's overall costs. Here lowering the energy consumption when executing each Map Reduce jobs is a critical part for data centers. In this paper, we advise a framework for mending the energy efficiency of Map Reduce applications, while satisfying the (SLA) Service Level Agreement. Our project aims to improve the energy efficiency of Map Reduce clusters. To realize this we will perform a comprehensive energy characterization of Hadoop Map Reduce and create its energy-performance model. Then analysis done by using this model will help in configuring energy efficient Hadoop Map Reduce. The energy characterization will be used to derive examining for designing energy aware scheduling algorithm. That the assignments of map and reduce tasks to the machine slots in order to minimum the energy consumed when executing the application. We perform extensive experiments on a cluster of 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 an extensive simulation study to evaluate the work of the proposed algorithms. The energy aware scheduling will improve the energy efficiency of Map Reduce clusters thus help in minimizing the operational costs and energy cost of the data-center.

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

I. INTRODUCTION

Several businesses and organizations are faced with an ever-growing need for analyzing the unprecedented amounts of data. Such needs challenges existing approaches, and require novel approaches and technologies in order to cope with present challenges of big data processing. One of the leading challenges of processing data intensive applications is to minimum their energy costs. Electricity used in US data centers in 2010 accounted for about 2% of total electricity run-down nationwide [1]. In addition, the energy spend by the data centers is growing at over 15% annually, and the energy costs frame about 42% of the data centers' operating costs [2]. Considering that server costs are consistently falling, it should be no surprise that in the coming future a big percentage of the data centers' costs will be energy costs. The increase in energy consumption of data-centers has become an important issue of concern. The U.S. Environment Protection Agency (EPA) published a report expose the two-fold increase in energy consumption by the data centers from 2000 to 2006 and displaying same increase again from 2007 to 2011 [3]. The Hadoop MapReduce [4] clusters formulate a higher part of today's data-centers as organizations use MapReduce for efficiently processing their huge volume of data. Thus reduces the energy consumed by the Hadoop clusters will contribute significantly towards improving the data-centers energy efficiency in general. The MapReduce programming model breaks a computation into small tasks and executes them across multiple machines for greater performance of big data jobs [5]. The, most commonly used implementation of Hadoop MapReduce

[1,2,3] Department of Computer Science and Engineering, S.R.M University, Chennai, India, *E-mails: anijoe@gmail.com; idaseraphim87@gmail.com; arul_sowmiya@gmail.com*

framework that replicates the data on multiple machines for providing fault tolerance on the commodity hardware cluster. The Hadoop clusters consisting of several hundreds and thousands of machines are created in data centers to support top level workloads, large data volumes and high fault-tolerance. A high peak-to-mean ratio of workloads and numerous-copies of data makes these clusters energy inefficient because they are under-utilized and consume a high power most of the time.

Therefore, it is critical for the data centers to minimize their energy consumption when providing services to customers. Big data applications run on large clusters within data centers, where their energy costs make energy efficiency of executing such range applications an important concern. MapReduce [6] and its open-source implementation, Hadoop [7], have emerged as the leading computing platforms for big data analytics. For scheduling multiple MapReduce jobs, Hadoop originally worked on a FIFO scheduler. To overcome the issues with the waiting time in FIFO, Hadoop then employed the Fair Scheduler [8]. These two schedulers, however, do not consider improving the energy efficiency when executing large MapReduce applications. Improving energy efficiency of MapReduce applications leads to a significant reduction of the total cost of data centers. In this paper, we design MapReduce scheduling algorithms that improve the energy efficiency of working on each individual application, while satisfying the service level agreement (SLA).

(A) Evaluations done for different Map-Reduce Scheduling Algorithms

The efficacy i.e. the effectiveness and sufficiency of any approach can be proved by showing the results of its extensive evaluation. The algorithms discussed above were also evaluated in respective papers with different rigors. The Table 3 below summarizes the various evaluation aspects covered by each of the paper. As seen in the table the experiments with mix workload were not performed by most of the papers, which is required in the current scenarios where data centers are shared multiple users for running different types of jobs. Also it was observed that not many papers exclusively performed the tests for studying the overheads of the proposed algorithms and their scalability with increasing number of nodes to thousands as seen in current Map-reduce clusters.

(B) Energy Efficiency in Hadoop Map-Reduce Systems

The data centers have been growing size to support large number of online user and process their huge for different analytical purposes. Their operational costs including power, cooling etc has been increasing accordingly. The report from U.S. Environment Protection Agency (EPA) showed that the energy consumed by data centers has doubled from 2000 to 2006 and projected that it is expected to double again from 2007 to 2011 [EPA07]. So, the increasing operational costs of growing data centers have attracted a lot of attention over past decade. For their massive data processing many industries use map-reduce for processing huge volumes of data spread across large number of clusters consisting of hundreds and thousands of machines. These clusters consist of commodity hardware, medium size servers and a few higher performance servers in different ratios. Such large-scale infrastructure consumes lot of power. Thus power management for MR clusters has also become important. The data centers have large number of machines as they are sized for handling peak loads which occur rarely, so the machines lay underutilized consuming power.

II. PRELIMINARIES

(A) HDFS

A Hadoop distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster.

Daemons in Hadoop core

1. Name node
2. Data node HDFS
3. Secondary name node
4. Job tracker
5. Task tracker MR

Daemon process: process which works in background and has no controlling terminal. In HDFS this nodes store's data are known as data nodes (slave nodes). And name nodes (master node) is responsible for management of file that are distributed across the cluster (metadata)

Now check how the file is stored in HDFS. Files are broken down into smaller chunks are called as blocks and then this broken chunks or blocks are replicated. Then blocks are distributed over the cluster. This process of replication and distribution is managed by the name node. Block size is the minimum size of the data can be read or write to a file system. Its default size is 64mb. These changes can be done in HDFS-site.xml here we can set replication factor and block size as well.

(B) MapReduce

MapReduce is a Hadoop framework using which we can write applications to process huge amount of data, in parallel, on several clusters of commodity hardware in a reliable manner.

- Generally MapReduce paradigm is based on sending the computer to where the data stays!
- MapReduce program carry out in three stages, namely map stage, shuffle stage, and reduce stage.

Map task: In Map task each block (HDFS) has one Map jobs and this will given to shuffle stage where portioning and sorting will be done. Then this output is given to reduce task where merging will be done like sum of the value.

(C) YARN

yarn (yet another resource negotiator) enhances the performances of the Hadoop compute cluster. Yarn resource manager focuses exclusively on scheduling making it clear to manage large Hadoop cluster. The major works of yarn is to improve resource management, scheduling, monitoring and improving scaling. Here scheduling Hadoop cluster part will be done in yarn-site.xml.

(D) HiBench suite

We perform wide-ranging experiments on a Hadoop cluster to determine the energy consumption and execution time for several workloads. There are several benchmarking tools which are used give the results in graphical representation.

Micro benchmarks

For job based tasks (in the contrast to streaming based tasks), HiBench provide following workloads:

Sort

This workload sorts its *text* input data, which is generated using RandomTextWriter.

Word Count

This workload calculates the occurrence of each word in the input data, which are generated using RandomTextWriter. It is representative of another typical class of real world MapReduce jobs extracting a small amount of interesting data from large data set.

TeraSort

TeraSort is a standard benchmark created by Jim Gray. Its input data is generated by Hadoop TeraGen example program.

Sleep (sleep)

This workload sleeps an amount of seconds in each task to test framework scheduler.

1) SQL

Scan (scan), Join (join) and Aggregate (aggregation)

This workload is developed based on SIGMOD 09 paper "A Comparison of Approaches to Large Scale Data Analysis" and HIVE396.It contains Hive queries (Aggregation and Join) performing the typical OLAP queries described in the paper. Its input is also automatically generated Web data with hyperlinks following the Zipfian distribution.

2) Web Search Benchmarks

PageRank (pagerank)

This workload benchmarks PageRank algorithm implemented in SparkMLLib/ Hadoop (a search engine ranking benchmark included in pegasus 2.0) examples. The data source is generated from Web data whose hyperlinks follow the Zipfian distribution.

Nutch indexing

Large scale search indexing is one of the most important uses of MapReduce. This workload tests the indexing subsystem in Nutch, a famous open source (Apache project) search engine. The workload uses the automatically generated Web data whose hyperlinks and words follows the Zipfian distribution with corresponding parameters. The dict used to generate the Web page texts is the default linux dict file /usr/share/dict/linux.words.

3) Machine Learning

Bayesian Classification (bayes)

This workload benchmarks NaiveBayesian Classification implemented in SparkMLLib/ Mahout Examples.Massive machine learning is another important use of MapReduce. This workload tests the Naive Bayesian (a popular classification algorithm for knowledge discovery process and data mining) trainer in Mahout 0.7, which is an open source (Apache project) machine learning library. The workload uses automatically to generated documents whose words follow the zipfian distribution. The dict used for text generation is also from the default linux file /usr/share/dict/linux.words.

Kmeans clustering

This workload tests the Kmeans (a well known clustering algorithm for knowledge discovery and data mining) clustering in Mahout 0.7/SparkMLlib. The input data set is generated by GenKMeansDataset based on Uniform Distribution and Guassian Distribution

4) HDFS Benchmarks

1 Enhanced DFSIO (dfsioe) Enhanced DFSIO tests the HDFS throughput of the Hadoop cluster by generating a large number of tasks operating writes and reads simultaneously. It measures the average throughput of

each map task, and the aggregated throughput of HDFS cluster. Note: this benchmark doesn't have Spark corresponding implementation.

(E) Hierarchical scheduling algorithm

In shared environment, the Hadoop Map-Reduce first selects a user whose job needs to be scheduled. Then from the list of jobs for the selected user it selects the job to be scheduled. Once the job is selected its map task, reduce task or speculative task is scheduled. Various decision making algorithms can be used at each level depending on the work to be achieved. Multiple scheduling algorithms exists for each hierarchy level as shown in Figure.

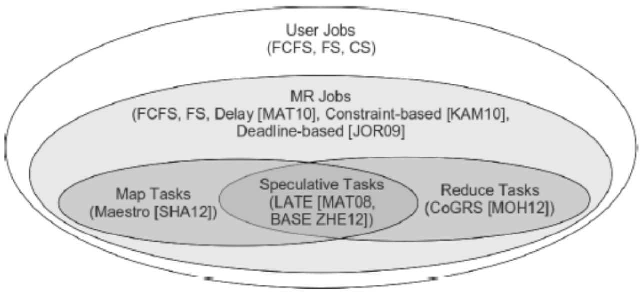


Figure 1: Hierarchical Scheduling policies proposed for Map-Reduce Clusters

1) User jobs

a) Fair Scheduling Algorithm

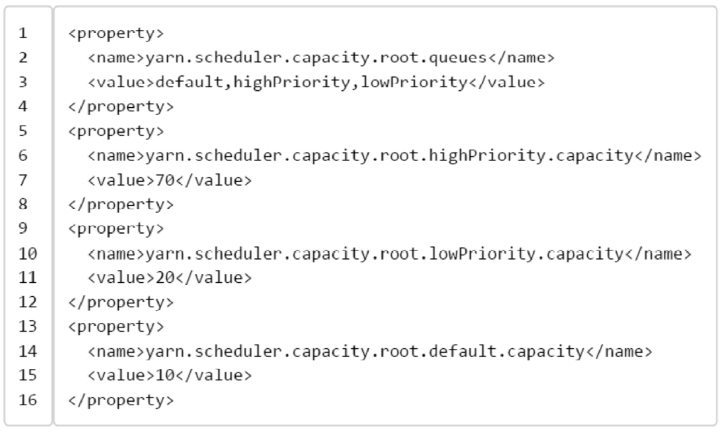
The objective of Fair scheduling (FS) algorithm is to do an equal distribution of compute resources among the users/jobs in the system [APHAD]. In case of multiple users, one pool is assigned to each user [APHAD]. The scheduling algorithm divides resources equally among these pools.

b) Capacity Scheduling Algorithm

The objective of Capacity scheduling is to maximize the resource utilization and throughput in multi-tenant cluster environment. The design of capacity scheduling algorithm is very similar to Fair scheduling [APHAD]. Here instead of pools, queues are used. Each queue is assigned to an organization and resources are divided among these queues. Additional security mechanisms are built for control access to the queues, so that each organization can access only its queue and cannot interrupt with other organization's queues, jobs or tasks.

We can set these scheduler algorithms in yarn-site.xml.

For example:



```
<property>
  <name>yarn.scheduler.capacity.root.queues</name>
  <value>default,highPriority,lowPriority</value>
</property>
<property>
  <name>yarn.scheduler.capacity.root.highPriority.capacity</name>
  <value>70</value>
</property>
<property>
  <name>yarn.scheduler.capacity.root.lowPriority.capacity</name>
  <value>20</value>
</property>
<property>
  <name>yarn.scheduler.capacity.root.default.capacity</name>
  <value>10</value>
</property>
```

Figure 2: Setting capacity scheduler in yarn-site.xml.

2) Job Level

The job level scheduling policies available with default Hadoop Map-Reduce are First-Come First serve (FCFS) and Priority based. These may not be efficient for all the scenarios and requirements. Therefore some new policies have been proposed for selecting a job for meeting the different users/administrators requirements. The preliminary algorithms, their improved versions and some newly proposed scheduling algorithms are described in this section.

(a) First-Come First-Serve Scheduling Algorithm

The objective of FIFO scheduler to schedule jobs based on their priorities in first-come first serve order. FIFO is the default Hadoop scheduler which includes five priority levels as well.When the scheduler receives a heartbeat indicating that a map or reduce slot is free:

- o It scans through list of jobs to find a job with highest priority and then oldest submit time. If this job has a pending task of the slot type (map/reduce) then that job is selected. Else next job in this order is picked. This goes on till a matching task (type as per slot) is found.
- o Next if it's a map slot, then to achieved data locality that the scheduler consults NameNode and picks a map task in the job which has data closest to this free slave (on the same node, otherwise on the same rack, or on a remote rack).
- o If it's a reduce slot, any of the reduce task is scheduled on the node. Hadoop MapReduce doesn't wait for all map tasks to finish for scheduling a reduce task, so the map task execution and shuffling of intermediate data can run in parallel to have better turn-around time. This is known as early-shuffle.

(b) SLA-based Scheduling Algorithms

The huge enterprise data is now being stored in distributed file systems for higher reliability and availability. This data is required as input not only for long batch jobs but also for many users jobs which many be online jobs or interactive jobs. One such algorithm is the constraint based Hadoop scheduling algorithm proposed by Kamal et. al. in [KAM10]. This algorithm maintains the minimum number of map/reduce tasks, provided by the job execution cost model, in the cluster to meet the SLA. The job cost execution cost model determines the minimum number of map/reduce tasks required to meet deadlines based on the deadline, arrival time, input data size and map/reduce execution costs.

3) Task Level

MR programming model creates two types of tasks map and reduce which need to scheduled in respective slots. One more type of task created by MR during runtime is speculative task to handle straggler map/ reduce tasks. These needs to be scheduled on scheduled in a map or reduce slot depending on the type of task it is duplicating. Different scheduling algorithms have been proposed for selecting a task for the given slot type available to achieve different objectives like data locality, performance improvement and improve resource utilization. The section describes some of the scheduling algorithms proposed for each of the task type.

(a) **Map-Task Level:** The default MR scheduling algorithm uses data locality criteria to select a map task for a given node. Initially Job-Tracker assigns the map tasks to the slaves depending on the Task-Tracker slots capacity, considering data locality. At run time, when a Task-Tracker reports an empty slot to process map task, the Job-Tracker checks for map tasks in its pool and consults the storage meta-data service to get the hosted chunks. It selects a map task for this node in following order of preference (1) a map task that has local chunk on that node, else (2) a map task that has data on another node within the same rack, else (3) a map task that has data on another node outside the rack. The map tasks pool has three kinds of map tasks: failed map tasks which have highest priority, normal map tasks and speculative map tasks with lowest priority.

Replica-aware scheduling algorithms

The experiments done by Shadi et. al. [SHA12] shows a high percentage (23%) of non-local map tasks executions with default map scheduling algorithm. This impacted task execution time and resulted in large percentage of speculative tasks executions (55%) out of which only 50% were successful. All this increased the job response time. It was shown that the local map tasks cause speculation with a much lower probability as compared to non-local tasks. Also it was observed that non-local executions caused significant imbalance of the successful map tasks among different identical nodes.

(b) **Reduce-Task Level:** Once even a single map task of a job is completed, Hadoop MR starts scheduling its reduce job so that the map task execution and shuffling of intermediate data can run in parallel to have better turn-around time. This is known as early-shuffle. Hadoop MR randomly selects the reduce task of the selected job for scheduling on the available reduce slot. Very few improvements have been suggested in literature for reduce task scheduling. Some of the modifications suggested were to meet jobs SLA as discussed in above section 3.3.2.3. A reduce task scheduling algorithm is described below.

Locality-aware scheduling algorithms

As the reduce task involves collecting data from the map nodes, some network congestion and performance problems can occur due to random reduce task scheduling by the Hadoop Job tracker. Consider the case when the node providing the input to a given reduce task is not near-by it, then this can cause lot of data shuffling and network traffic. Another problem could be of partitioning skew, i.e. the intermediate keys frequencies and distribution across nodes can be different causing high volume of data for some reduce tasks. Both these would impact the application performance. To address these issues, a locality-aware skew-aware Center of Gravity reduce task scheduler (CoGRS) is proposed [MOH12]. Here a reduce task is scheduled on node considering nearness to its feeding nodes and the skew in its partitions. In Hadoop tree-style network topology the bandwidth between two nodes is dependent on their relative locations in the network topology. So nodes that are on the same rack have higher bandwidth between them than those that are o_-rack. The bandwidth between nodes is represented as distance; distance between 2 nodes is measured by adding distances to a common ancestor and distance of a node to its parent is said to be 1.

(c) **Speculative-Task Level:** In Hadoop, if a node is available but is performing poorly, a condition that we call a straggler, MapReduce runs a speculative copy of its task (also called a backup task) on another machine to finish the computation faster. The goal of speculative execution is to minimize a jobs response time. A speculative task is run based on a simple heuristic comparing each tasks progress to the average progress. Hadoop monitors task progress using a parameter called Progress Score which has value between 0 and 1.

Latency-aware scheduling algorithms

Zaharia et al. address the problem of robustly performing speculative execution to maximize performance. The proposed Longest Approximate Time to End (LATE) algorithm [MAT08] is based on three principles: prioritize tasks to speculate, select fast nodes to run on, and cap speculative tasks to prevent thrashing. To realize these principles LATE algorithm uses following parameters:

_ SlowNodeThreshold

_ SpecultiveCap

_ SlowTaskThreshold.

III. RELATED WORK

Given its characteristics, improving the energy efficiency of Hadoop MapReduce clusters without impacting it's their quality is a challenging objective which our research wants to achieve. We plan to achieve this goal by creating:

- o The predictive models to perform recommendations for the energy efficient cluster and job configurations.
- o An energy-adaptive scheduling algorithm for efficient scheduling of MapReduce jobs and tasks.

The performance of jobs is affected by the underlying hardware characteristics and platform configuration settings. And the energy consumed by the MapReduce jobs is an action of the response time and the hardware characteristics. So, we will develop methods that will consider the crucial parameters of all the layers of MapReduce architecture. To do so we are following the steps described below.

(A) Empirical characterization of energy and performance of Hadoop MapReduce cluster

The project has started with an depth study of energy used by map-reduce clusters for different types of workloads, data volumes and configuration settings at all layers of Hadoop MapReduce. The configuration parameters, at various layers of Hadoop MapReduce, identified for this study are shown in Table 1. The energy is defined as product of Power and Time. So along with other parameters that impact the Hadoop MapReduce performance, we are also analyzing the impact of the relatively newer hardware power management capabilities on the MapReduce energy consumption by changing the cpu frequency of machines.

(B) Energy and performance models for MapReduce

Then create the energy and performance models for MapReduce framework to orderly predict the energy and performance of jobs with various configuration settings. We plan to use the multi-variate regression modeling on the data collected from the energy characterization of the Hadoop MapReduce to create these models. The parameters to be added in model will be determined by doing the fractional factorial analysis [14] of results of the energy characterization done using the max and min possible values of all the parameters mentioned above. Then create and validate the stochastic Markov chain models [15] for the MapReduce systems to predict the performance and energy, using the data collected from energy characterization.

(C) Energy aware MapReduce job/task scheduling algorithm

The execution order and distribution of the tasks on the nodes are to be managed by the scheduling algorithms which drive the performance of the jobs. Many scheduling algorithms have been designed to improve the work performance and resource management of map-reduce clusters. We plan to study how they perform of these scheduling algorithms would impact the Map-reduce energy efficiency. We also plan to explore if clubbing them with different power management and energy efficiency techniques can make a better energy efficiency improvement method. We will analyze the energy characterization and above study results to derive the heuristics for designing an energy aware MapReduce task and job scheduling algorithm.

IV. CONMBINATION OF 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. Overhead is not basic, nor is it significant in this case. Waiting time and response time depend on the priority of the process. Higher priority processes have less waiting and response times. Deadlines can be easily met by giving higher priority to the earlier deadline processes.

Disadvantage: Starvation of lower priority processes is possible if large no of higher priority processes keep arriving continuously.

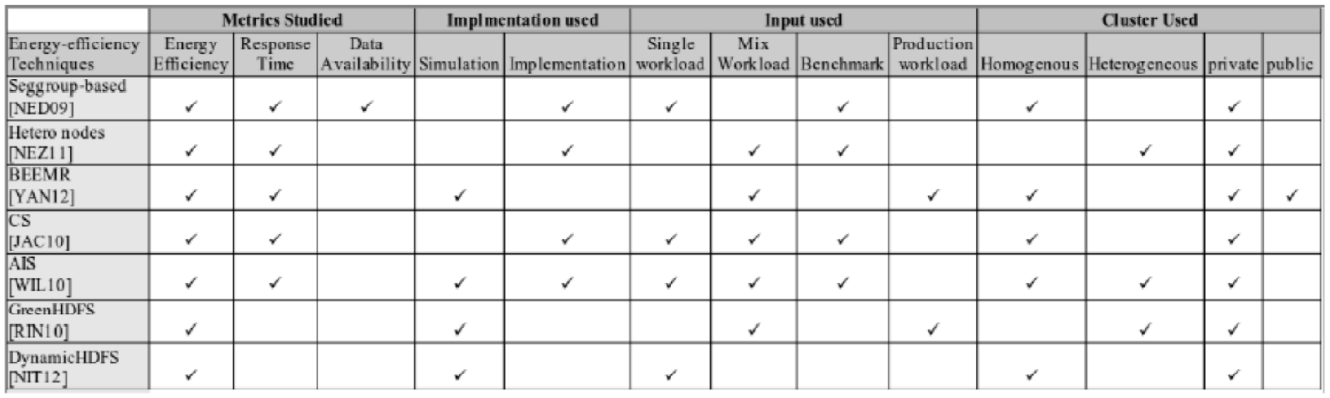
- The proposed architecture focuses on the shortcoming of simple round robin architecture which gives equal priority to all the processes (processes are scheduled in fcfs manner). Because of this drawback round robin architecture is not capable for processes with smaller CPU burst. This results in the increase in waiting time and response time of processes which output in the decrease in the system throughput.
- Introduced algorithm will be executed in two steps which will helps to minimize a number of performance parameters such as context switches, average waiting time and average turnaround time. The algorithm performs following steps:
- Step 1: Allocate CPU to every process in round robin fashion, according to the given priority, for given time quantum (say k units) only for one time.
- Step 2: After completion of first step following steps are performed:
- a) Processors are arranged in increasing order or their remaining CPU burst time in the ready queue. New priorities are assigned according to the remaining CPU bursts of processes; the process with shortest remaining CPU burst is assigned with 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.

V. COMPARISON OF MR ENERGY EFFICIENCY IMPROVEMENT TECHNIQUES

The various techniques studied above are compared against following key properties expected from them in Table 1 below.

- Energy efficiency - The energy efficiency metrics shows the percentage improvement reported by each of the paper over the native Hadoop Map-reduce. Though each one has different environments like simulation, private cloud and public cloud, different mechanism for power measurements like meter reading or counting number of active nodes, all have reported percentage energy efficiency improvement over the native Hadoop Map- reduce.
- Performance - As improving energy efficiency involves trade-off with performance, the proposed techniques are compared for their impact on application performance.
- Data availability - Data availability is an important aspect of Hadoop HDFS and MR systems, at the same time data replication is one of the factor for increased energy consumption by Hadoop MR. So various techniques are compared on how they handle this aspect by comparing them for impact on data availability and possibilities of getting hotspots criteria.
- Scalability - Considering the size of Map-reduce clusters and volume of workload they need to handle, any technique used need to be scalable to thousands of servers and high workload intensities. So the techniques are compared for their scalability.
- Overheads - Also the application performance should not get impacted much due to the optimization activities like improving energy efficiency, so the overheads introduced by each technique are listed.

Table 1
Summary of experiments done for evaluating proposed energy e_ciency techniques in various



	Metrics Studied			Implmentation used		Input used				Cluster Used			
Energy-efficiency Techniques	Energy Efficiency	Response Time	Data Availability	Simulation	Implementation	Single workload	Mix Workload	Benchmark	Production workload	Homogenous	Heterogeneous	private	public
Seggroup-based [NED09]	✓	✓	✓		✓	✓		✓		✓		✓	
Hetero nodes [NEZ11]	✓	✓			✓		✓	✓			✓	✓	
BEEMR [YAN12]	✓	✓		✓			✓		✓	✓		✓	✓
CS [JAC10]	✓	✓			✓	✓	✓	✓		✓		✓	
AIS [WIL10]	✓	✓		✓	✓	✓	✓	✓		✓	✓	✓	
GreenHDFS [RIN10]	✓			✓			✓		✓		✓	✓	
DynamicHDFS [NIT12]	✓			✓		✓				✓		✓	

VI. ACKNOWLEDGMENT

Due to the growing need for big data processing and the high level adoption of MapReduce and its open source implementation Hadoop for such processing is increased, 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 are significant optimization opening within the MapReduce in terms of reducing energy consumption. We proposed combination of two scheduling algorithm Round Robin and priority to save the energy of the data centers. Both proposed algorithms provide very fast solutions making them reasonable for execution in real-time settings. We performed experiments on a Hadoop cluster to determine the energy consumption of MapReduce benchmarking such as TeraSort, Page Rank, and K-means Clustering. We then used this data in an extensive simulation reading to analyze the performance of Round Robin and priority. The results showed that the proposed algorithms are capable of getting near optimal solutions leading to significant energy savings. In the future, plan to design and implement a distributed scheduler for multiple MapReduce jobs with the 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_Report_Congress_ Final1.pdf*

[2] Dean, Jeffrey, Ghemawat, Sanjay: MapReduce: simpli_ed data processing on large clusters. ACM Communications 51, 107{113 (2008).

[3] The Apache Hadoop Project. http://www.hadoop.org.

[4] Chen, Yanpei and Alspaugh, Sara and Borthakur, Dhruba and Katz, Randy: Energy e_ciency for large-scale MapReduce workloads with signi_cant interactive analysis. In: EuroSys pp. 43{56. ACM (2012).

[5] Kaushik, Rini T. and Bhandarkar, Milind: GreenHDFS: towards an energy-conserving, storage-e_cient, hybrid Hadoop compute cluster. In: HotPower pp. 1{9. USENIX (2010).

[6] Leverich, Jacob and Kozyrakis, Christos: On the energy (in)efficiency of Hadoop clusters. SIGOPS Oper. Syst.Rev. 44, 61{65 (2010).

[7] Maheshwari, Nitesh and Nanduri, Radheshyam and Varma, Vasudeva: Dynamic energy e_cient data placementand cluster recon_guration algorithm for MapReduce framework. Future Gener. Comput. Syst. 28, 119{127(2012)

[8] Vasi_c, Nedeljko and Barisits, Martin and Salzgeber, Vincent and Kostic, Dejan: Making cluster applications energy-aware. In: ACDC pp. 37-42. ACM (2009).

[9] Yigitbasi, Nezih and Datta, Kushal and Jain, Nilesh and Willke, Theodore: Energy e_cient scheduling of MapReduce workloads on heterogeneous clusters. In: GCM pp. 1:1{1:6. ACM (2011).

[10] Lang, Willis and Patel, Jignesh M.: Energy management for MapReduce clusters. In: VLDB Endow. pp. 129{139. VLDB Endowment (2010).

[11] B. Feng, J. Lu, Y. Zhou and N. Yang: Energy E_ciency for MapReduce Workloads: An In-depth Study. In: Australasian Database Conference pp. 61{70. ACS (2012).

[12] Li, Wenjun and Yang, Hailong and Luan, Zhongzhi and Qian, Depei: Energy Prediction for MapReduce Workloads. In: DASC pp. 443-448. IEEE (2011).

[13] Martin, Alain J.: Towards an energy complexity of computation. Inf. Process. Lett. 77, 181{187 (2001).

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

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

[16] Wirtz, Thomas and Rong Ge: Improving MapReduce energy efficiency for computation intensive workloads.