International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 05 | May 2018 www.irjet.net p-ISSN: 2395-0072

Energy Efficient MapReduce Task Scheduling on YARN

Sushant Shinde[1], Sowmiya Raksha Nayak[2]

[1]P.G. Student, Department of Computer Engineering, V.J.T.I, Mumbai, Maharashtra, India
[2]Assistant Professor, Department of Computer Engineering, V.J.T.I, Mumbai, Maharashtra, India

---

Abstract - *Modern industries adopting the big data technologies have started shifting the data center activity from many smaller data centers to a few lager data centers. For such hyperscale data centers, energy cost is one of the major challenges in providing computational infrastructure. MapReduce, which is a distributed processing platform, is accepted by many enterprises and have started using it through cloud services with large computing clusters. Therefore, minimizing the energy consumption of each execution of the MapReduce job is of much importance for data centers. Most of the existing practices either focus on make-span improvement or consider improving energy efficiency at data center component level. In this paper we present an algorithm for improvement of energy efficiency at application execution level proposing the efficient task assignment of a MapReduce on cluster. This algorithm takes advantage of YARN architecture of treating resources generically and considers the energy consumption differences of different task placements on machines for energy efficient assignment of tasks. With the adoption of proposed task assignment technique in the MRAppMaster, Hadoop YARN shows considerable amount of improvement in energy savings in MapReduce job executions.*

Key Words: Energy efficiency, MapReduce, YARN, scheduler, resource allocation.

1. INTRODUCTION

Over the last few years most of the modern industries have started adopting the big data technologies for uncovering insights and hidden patterns to explore new opportunities in business by enhancing customer experience, finding future strategies, reducing cost of existing systems or by enhancing security. Massive size, high diversity and unstructuredness possessed by big data, presents unique storage and computational challenges like scalability, fault tolerance, storage bottleneck and timeliness resulting into requirements of new computational paradigms.

MapReduce is a popular platform for distributed processing of such a huge data set on large clusters of machines capable of processing data reliably in a parallel manner. With the acceptance of MapReduce as a computational platform, many enterprises started using it through multi-user cloud services with large computing clusters.

The trend of 'hyperscale shift' [2] shows the shift of the data centre activity from many smaller data centres to a few larger data centres. For such hyperscale data centres, energy cost is one of the major challenges in providing computational infrastructure. As per the US Data Centre Energy Usage Report [2], in 2014 data centres in U.S. consumed an estimated 70 billion kWh which is 1.8% of total U.S. electrical consumption. Current study shows the increase in the consumption by about 4% from 2010 to 2014. Expected to continue this increase in near future estimates consumption of 73 billion kWh energy in 2020.

Servers, storage, network and infrastructure are the factors those influence the energy consumption of the data centres. Storage devices are becoming more efficient resulting into reduced energy consumption along with the reduced power consumption by network ports. Thus, servers are the major factor for energy consumption of today's data centres.

The energy efficiency practices that many data centres have started following include maximizing the efficiency of each type of facility in data centre. While these practices considers data centre level components, very little attention is given for improving the energy efficiency at application execution level such as the efficient task scheduling of a MapReduce job on a cluster.

Apache Hadoop [4], which is an open-source implementation of Google's MapReduce framework, has been upgraded to Hadoop 2 or NextGen Hadoop by separating the cluster resource management capabilities from computational logic like MapReduce. With this split it allows fine-grained resource management resulting in better cluster utilization and improved scalability. The resource management capabilities are known as YARN. MapReduce task scheduling in Hadoop 1 had to consider the distinction between the map slots and the reduce slots, leading to under-utilization of cluster resources. In YARN there is an improvement in utilization by considering the cluster resources in the form of generic containers instead of separate slots for separate type of tasks.

There are previous efforts of using the job profiling information when taking decisions of task placements on the nodes with the intention of energy efficiency, but these have not given much attention to the improvement in resource allocation of Hadoop YARN.

In this paper an attempt is made to take advantage of YARN architecture of treating resources as a bundle of memory and CPU cores on which any type of task can run. While

considering the energy consumption differences of different tasks placements on machines, as demonstrated in the work of Mashayekhy et al. [3], the main contribution of this paper is to use these differences for dynamic task placements where the dependency between the map and reduce phase of a job is considered and the tasks are ranked as per this dependency. This rank is used in progressive phase of task assignment where as soon as a container gets free of executing a map or a reduce task it can be further utilized by either next map or next reduce task depending upon the execution progress.

Thus the improved task scheduling algorithm for YARN results into better energy savings for a MapReduce job execution on cluster.

These algorithms can be treated as secondary scheduling strategies and can be incorporated with other higher, multi-user scheduling strategies like Fair and Capacity scheduling.

2. RELATED WORK

2.1 Energy-efficient resource management in data centers

BEEMR architecture proposed by Chen et al. [5], splits the cluster into interactive and batch zones. Interactive zone serves interactive data analysis and uses a pool of dedicated machines which are kept fully powered. Energy saving is achieved by serving batchable jobs by batch zone which is kept in low-power state in-between the batches. Cardosa et al. [6] uses space-time trade-off in achieving energy efficiency by, 1) co-placing virtual machines with complementary resource requirements and thus reducing spatial wastage and 2) co-locating virtual machines with closely matched resources which allows the physical machines to be emptied at around the same time and hence can be suspended resulting into improved machine utilization. Wirtz et al. [7] focuses on energy efficiency of computational intensive workloads. It considers the number of compute nodes and DVFS scaling to improve resource allocation. It demonstrates that frequency scaling has large impact on computationally intensive workloads and thus can be used to scale down the voltage and hence saving energy. GreenHDFS proposed by Kaushik et al. [8], logically partitions the data centre into Hot and Cold zones. It relies on inherent heterogeneity in the access pattern and each cluster zone has different temperature characteristics. GreenHDFS considers the dormancy of a file to which temperature is inversely proportional. Thus the coldness of the cluster can be increased with dormancy of the files resulting into better energy savings.

All these strategies can be categorized as data centre level energy saving techniques and do not exploit MapReduce phases of execution for energy conservation.

2.2 Efficient resource allocation and scheduling in MapReduce

Much of the work in improvement of MapReduce resource allocation is done in the perspective of reducing the execution time or improving the make-span of a MapReduce application. Realizing the existence of the synchronization barrier between the two phases of the MapReduce job execution and using it to maximize the parallelization and thus getting speedup is the key of SMapReduce [9]. But it does not pay attention to the energy conservation while allocating more resources to the overlapped section of Map phase in YARN. HaSTE [10] presents a new YARN scheduler which is aimed at efficiently utilizing the resources in YARN and reducing the make-span of the jobs. It dynamically schedules (prioritizes) tasks based on each task's fitness: the gap between resource demand of tasks and the resource capacity of nodes, and urgency: refers to the importance of a task as per the dependency between map and reduce phases. Though our paper recognizes the similar dependency among phases, considering it for reducing the energy conservation has not done in previous works. Kurazumi et al. [11], focuses on the under-utilization of the cluster due to I/O waiting by the tasks which are not data-local. It calculates the I/O wait percentage for each CPU and adds or removes map slots accordingly. Ibrahim et al. [12], focus on reducing the non-local tasks and balancing the number of map tasks across f=different nodes by considering the probability of scheduling a map tasks on a given machine depending on the replicas of the input data. Speedup is gained by improving data locality in execution. None of these works focuses on energy consumption of MapReduce execution.

2.3 Energy-Efficient resource allocation in MapReduce

2.4

Energy aware load management framework proposed by Shao et al. [13], employs a prediction module which predicts the number of running workload tasks in near future by continuously sampling past and current records. The control module side-by-side employs node state control strategy by turning on the proper number of nodes as per the predicted value. Turning off these nodes for a specific duration results into energy saving. SLA aware energy efficient scheduling proposed by Li et al. [14], for YARN uses job profiling of jobs to get performance characteristics of different phases of a MapReduce application. DVFS based controller is used for YARN resource provision thus utilizing the slack time for system energy optimization.

Mashayekhy et al., is the first work towards energy efficient scheduling in which energy consumption differences in task placement are considered. Even this work exploits the job profile information for efficient task placements.

These works can be categorized as cluster-level strategies where different aspects of map and reduce tasks are considered. Inspired from EMRSA algorithms our algorithms use Energy-to-Length Ratio (ELR) for prioritizing the tasks for a free container. EMRSA is not much adapted to Hadoop YARN while our algorithms take advantages of genericness of the containers provided by YARN architecture and thus significant utilization improvement is possible resulting into better energy saving by balancing the resource allocation among different phases of the MapReduce job execution.

3. BACKGROUND AND MOTIVATION

3.1 MapReduce

MapReduce [1] is a software framework for processing huge data sets in a distributed manner over a cluster of commodity hardware. MapReduce can be viewed as a specialization of the "split-apply-combine" strategy for data processing. The objective of this programming model is to speed up the data processing by parallelizing the execution of the job across multiple nodes.

Apache Hadoop, which is an open-source implementation of MapReduce model, along with its storage part Hadoop Distributed File System (HDFS), is designed to scale up to thousands of nodes each with dedicated computation and storage capabilities. HDFS is a distributed file system that splits the file into blocks and distributes them across nodes in a cluster, thus providing a high-throughput access to data for running applications. MapReduce transfers the packaged code into nodes to process data parallelly. The data-locality optimizes the spatial efficiency where nodes manipulate the data they have access to, and results into faster and efficient processing. Due to the chunk replication, Hadoop system is considered as a fault-tolerant and thus reliable data processing. Thus, the key advantage of the Hadoop MapReduce framework over some existing parallel paradigms(e.g. grid computing and GPU) are fault tolerance and high-throughput data processing via MapReduce processing and HDFS.

3.2 MapReduce Task Scheduling

MapReduce creates multiple tasks and executes them on multiple nodes. As there many combinations of tasks and machines are possible, there arise a problem of deciding which machine should execute which task. Here come the different scheduling strategies in picture. A scheduling policy can be developed keeping different objectives in mind like, considering user's priorities of job selection, considering data locality for faster execution, improving resource utilization, reducing network congestion, improving the reliability of job execution and so on. Taking the different objectives into account, achieving a balance between them is an NP-hard problem. Hence many scheduler designers have proposed different heuristics for different objectives.

Important thing one must pay attention to is that in-appropriate scheduling of tasks across machines may fail to exploit the true potential of the parallelization.

While the most common objective of the scheduling policy is to minimize the completion time of a parallel application by properly allocating the tasks, our objective is to minimize the energy consumption of a single job execution on cluster with improvement in cluster utilization and considering the energy consumption differences of different tasks on different machines.

3.2.1 MapReduce scheduling in Hadoop

Based on the objective, a scheduling policy can be designed to run at different levels:

- User level: Fair, Capacity scheduler
- Job level: FCFS, Fairness- based, SLA-based
- Task level: map task level(replica-aware), reduce task level(locality-aware), speculative task level(latency-aware) scheduling

3.2.2 Resource sharing schedulers in YARN

Fair scheduler offers equal distribution of resources among different jobs when there are different types of jobs are ready to run. It overcomes the drawback of long jobs blocking small jobs, associated with FCFS scheduling. By having limits on running/pending tasks and jobs from a single user, Capacity scheduler provides minimum capacity guarantee for each user.

These schedulers do not consider the impact of task scheduling of MapReduce jobs on the system energy consumption. Our solution considers the energy consumption of each task and also pays attention to the dependency between map and reduce phase while allocating resources to the tasks resulting into energy efficient job execution.

3.2 Improvements in resource utilization in Hadoop

In Hadoop v1, user submits MapReduce jobs to the JobTracker where it pushes work to the available nodes in the cluster. The TaskTrakers on each machine, run the map and the reduce tasks on map and reduce slots respectively.

The inflexible 'slot' configuration of nodes either as Map or Reduce results into under-utilization of the cluster when more map or reduce tasks are running. Also, there is a limit on the nodes per cluster due to single JobTracker, resulting into scalability bottleneck. Taking these limitations into account Hadoop has evolved into NextGen YARN. In YARN JobTracker of old Hadoop is split into two components: 1] ResourceManager: globally manages assignment of resources by keeping track of NodeManagers and available

resources, allocating these resources to applications, and 2] Application Master: one for each application, manages application life-cycle and asks for appropriate resource containers to run tasks.

Scheduling policy on which we are working goes as a pluggable piece of code in a scheduler, which is a sub-part of the ResourceManager along with the ApplicationsManager which takes care of running ApplicationMasters. This separation of application specific tasks from resource management, allows ResourceManager to focus on better resource management resulting into improved scalability. NodeManager on each slave node manages user processes on that machine and provides computational resources to them in terms of containers. ResourceManager along with NodeManager forms the computational fabric of the cluster.

One of the major architectural features of the YARN is that it treats the cluster resources as a combination of memories and CPU cores. These combinations are known as the containers and allow a more precise control over the cluster.

Our paper takes the advantage of the genericness of these containers on which any type of the tasks can be run one after another. Containers can be seen as a request to hold resources on YARN cluster and there is no need to configure the system resources as a fixed number of map slots and reduce slots. This feature allows us to consider dependency between the map and the reduce phase while allocating resources to the tasks resulting into better utilization of the cluster. This opportunity along with energy consumption differences of tasks helps us to reduce the energy consumption of a MapReduce job execution.

4. ENERGY EFFICIENCY IN MAPREDUCE TASK SCHEDULING

In MapReduce job execution, input data blocks read by map tasks are processed into intermediate results, available to the reduce tasks as input. Map phase can be divided into three sub-phases: map, sort and spill phase. Reduce phase also consists of three sub-phases shuffle, sort and reduce. Shuffle phase transfers intermediate results to reducers. As soon as a single map task has finished its execution, the intermediate outputs are transferred to nodes on which reduce tasks wants them. Thus, shuffle phase run concurrently with the running map tasks. Here we can see that, though a sub-phase of reduce phase can start running in parallel to remaining of the map phase, actual task of reducing cannot be started before the end of the all map tasks.

4.1 System Considerations

The number of map tasks is driven by the number of input blocks which may depend on the input file size and the block size configured for the system. The number of reduce tasks is usually application specific and characterizes the job as reduce-heavy or map-heavy.

YARN cluster resources are considered as a fixed number (C) of containers, where each container is a combination of memory and CPU cores. Each task, Map or Reduce, runs on one container resource at a time. We are considering a big data job where number of Map tasks (M) and Reduce tasks (R) are larger than the available number of containers. As we are considering the heterogeneous cluster, a container may execute some tasks faster than others. Similarly, energy consumption of the tasks execution varies with the task-container combination. The time of execution of a task represents its 'length'. Thus, the Energy to Length Ratio (ELR) can be seen as the major decisive factor in prioritizing the tasks to run on the available container, where the task with lowest ELR are given preference.

4.1.1 Energy-Efficient Task Scheduling Problem

Consider X_{ij} variable denoting the assignment of task i to container j. We want to minimize the energy consumption of a job with $M \cup R$ tasks. This problem can be stated as,

Minimize

$$\sum_{j \in C} e_{ij} X_{ij} \forall_{i \in M \cup R}$$

Subject to,

$$\sum_{j \in C} X_{ij} = 1, \forall_{i \in M \cup R}$$

Where e_{ij} denotes the amount of energy required by task i to execute with container j, and

$$X_{ij} = \begin{cases} 1, if\ task\ i\ is\ assigned\ to\ container\ j \\ 0, otherwise \end{cases}$$

The constraint ensured that every task gets assigned to a container.

4.1.2 Prioritizing the tasks to run on the free container

The tasks to be run on a container when it's get free are prioritized by ranking tasks, 1] As per the ELR, and 2] As per the phase progress. When there are no tasks running on the cluster, each container is ready with the list of tasks which are sorted based on the ELR ratio for that particular container. As the execution of the job moves forward, the phase progress score of each task is taken into account along with the ELR for sorting the remaining tasks.

5. SCHEDULING ALGORITHM

Task scheduling takes place as shown in the following Progressive task assignment.

5.1 Progressive task assignment

After all the containers are assigned with tasks in initial container assignments, the execution of the tasks moves forward. As soon as any container finishes the execution of the task it signals its availability to the ResourceManager.

Result: X: remaining task assignment
j: Container available for execution;
T: Sorted unassigned Map and Reduce tasks $i \in M \cup R$ based on $Rank_{ij}$;
if T is empty **then**
return;
end
$i = T.dequeue()$;
$X_{ij} = 1$;

Algorithm -1: Progressive Assignment

For this free container the next task is chosen based on the rank which is calculated as follows,

$$Rank_{ij} = normalized\left(\frac{1}{ELR_{ij}}\right) + normalized(PS_{ij})$$

Where $Rank_{ij}$ is the rank of task i w.r.t container j and $ELR_{ij} = e_{ij}/t_{ij}$. PS_{ij} denotes the Progress Score of task i w.r.t container j and is calculated depending on the type of the task as,

$PS_{ij}^{m} = 0$
$PS_{ij}^{r} = 0$
if $\left[\frac{Run^{r}}{Run^{m}+Run^{r}}\right] < Comp^{m}/|M|$ **then**
$PS_{ij}^{r} = 1$
else
$PS_{ij}^{m} = 1$
end if

In above calculation, a reduce task has a preference over map task if the ratio of currently running reduce task to the total running tasks is less than the current progress of the map tasks. Here, Run^{m} and Run^{r} denotes the number of currently running map and reduce tasks respectively, while $Comp^{m}$ are the number of map tasks which are completed till this moment.

6. EVALUATION

The performance evaluation of the proposed algorithm is done by performing extensive experiments on Hadoop YARN cluster.

6.1 Experimental Setup

We used HiBench benchmark suit and performed experiments to measure energy and run-time for a number of MapReduce benchmark workloads of HiBench. The experimental Hadoop YARN cluster consist four nodes, one of which is a Master node. The Master node is configured with 16GB memory, 4 3.2GHz Intel quad-core processors and a Hard Drive of 1TB. The three slaves: slave01, slave02, and slave03, are composed of 8GB memory, 4 3.2GHz Intel quad-core processors and a Hard Drives of 1TB each. Thus the cluster has 40GB memory, 16 processors and 4TB of storage in total.

jRAPL [15], which is a framework for profiling Java programs executing on CPUs, is used for energy measurement. RAPL used by jRAPL is a set of low-level interfaces which can monitor energy consumption data of different hardware levels.

The performance of our algorithm is evaluated based on two metrics: Energy Consumption and Execution Time.

Several clustering and sorting workloads provided by HiBench suit are run and energy profiled. We run a single job at a time and calculate its start time and finish time to calculate the job's execution time. In similar fashion, we calculate energy consumption of that job with the help of jRAPL.

We implemented our algorithm on top of the Hadoop YARN 2.9.0. The default job scheduler adopted by the YARN Resource Manager, schedules the entire job for execution without any delay.

We consider different combinations of map and reduce task numbers. As this paper focuses on task scheduling of MapReduce applications only, we consider the Application Master component specific to MapReduce application, known as MRAppMaster. MRAppMaster is responsible for assigning tasks to the available containers.

6.2 Experimental Result

The performance of the algorithm along with Hadoop V1 and YARN (with existing MRAppMaster) is analysed in this subsection. Figure 1 plots the energy consumption of the jobs scheduled by the above considered configurations: proposed, Hadoop V1, YARN; for micro-benchmark TeraSort. Figure 2 shows the energy consumption of the jobs scheduled by these system configurations for Bayesian Classification Machine Learning benchmark. Results in Figure 1 show that the proposed algorithm is able to find

task assignments requiring an average of 21% less energy compared to those obtained by Hadoop YARN and an average 34% less compared to those obtained by Hadoop V1.

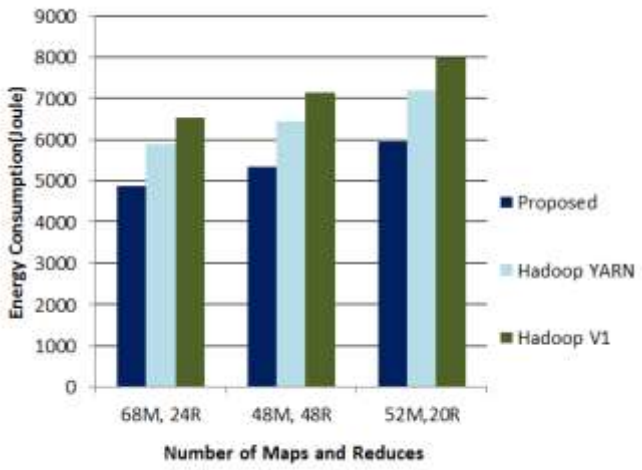

Figure -1: Performance on TeraSort: Energy Consumption

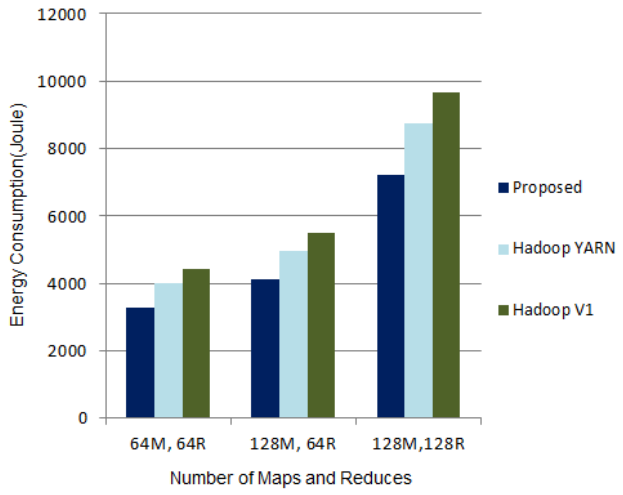

Figure -2: Performance on Bayesian Classification: Energy Consumption

Results in Figure 2 show that the proposed algorithm could assign the tasks resulting into an average of 19% and 22% less energy consumption compared to those obtained by Hadoop YARN and Hadoop V1, respectively.

Figure 3 and Figure 4 presents execution time of the algorithm. These results show that the proposed algorithm and the existing systems: Hadoop YARN and Hadoop V1 find the solution with almost same amount of time. Thus, the proposed algorithm can be used for scheduling tasks of big data application without negatively impacting the make-span of the applications.

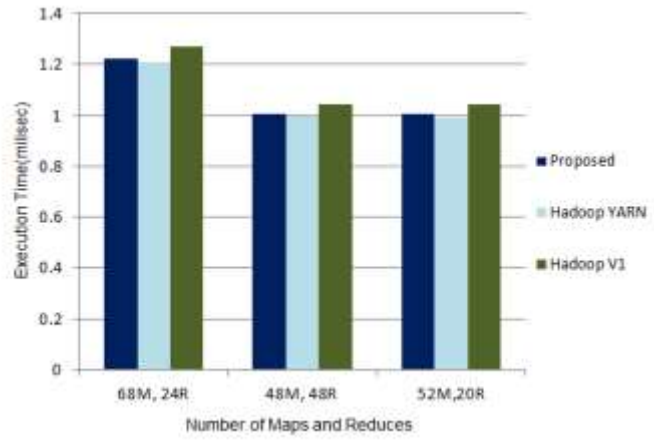

Figure -3: Performance on TeraSort: Execution Time

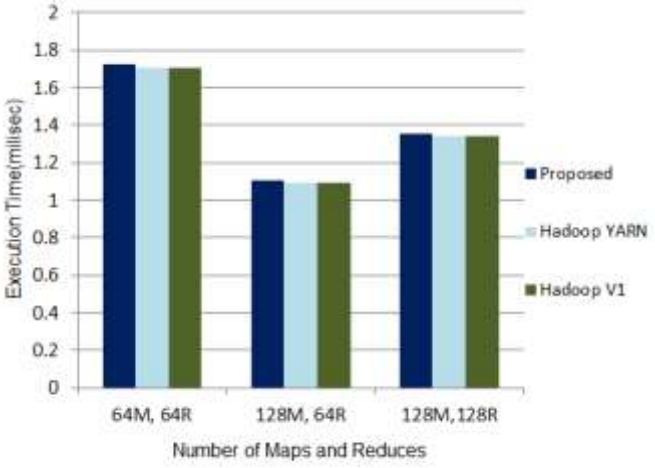

Figure -4: Performance on Bayesian Classification: Execution Time

6. CONCLUSION

With the acceptance of MapReduce as a computational platform and executing big data applications on large clusters, reducing energy consumption through efficient assignment in MapReduce jobs can have a significant impact on energy cost of data centres. The proposed algorithm assigns the individual tasks to the available containers so as to consume least amount of energy and make a full utilization of cluster resources through the map and reduce phases of the MapReduce job. This task assignment is done without negatively impacting the make-span of the application and thus, making it suitable to be used for scheduling big data applications.

REFERENCES

[1] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in Proc. of the 6th USENIX

Symposium on Operating System Design and Implementation, 2004.

[2] Arman Shehabi, Sarah Josephine Smith, Dale A. Sartor, Richard E. Brown, Magnus Herrlin, Jonathan G. Koomey, Eric R. Masanet, Horner, Nathaniel, Inês Lima Azevedo, Lintner, William, "United States Data Center Energy Usage Report", 2016

[3] L. Mashayekhy, M. Nejad, D. Grosu, Q. Zhang and W.S. Shi."Energy-Aware Scheduling of MapReduce Jobs for Big Data Applications." Transactions on Parallel and Distributed Systems, IEEE, 2015.

[4] Apache. Apache Hadoop YARN. [Online]. Available: http://hadoop.apache.org/docs/r2.9.0/hadoop-yarn/hadoop-yarn-site/YARN.html

[5] Y. Chen, S. Alspaugh, D. Borthakur, and R. Katz, "Energy efficiency for large-scale mapreduce workloads with significant interactive analysis," in Proc. of the 7th ACM European Conf. on Computer Systems, 2012.

[6] M. Cardosa, A. Singh, H. Pucha, and A. Chandra, "Exploiting spatio-temporal tradeoffs for energy-aware mapreduce in the cloud," IEEE Transactions on Computers, 2012.

[7] T. Wirtz and R. Ge, "Improving mapreduce energy efficiency for computation intensive workloads," in Proc. of the IEEE International Green Computing Conference and Workshops, 2011.

[8] R. T. Kaushik, M. Bhandarkar, and K. Nahrstedt, "Evaluation and analysis of greenhdfs: A self-adaptive, energy-conserving variant of the hadoop distributed file system," in Proc. 2nd IEEE Int'l Conf. on Cloud Computing Technology and Science, 2010.

[9] F Liang, F C.M. Lau, "SMapReduce: Optimising Resource Allocation by Managing Working Slots at Runtime," in Proc. 29th IEEE International Parallel and Distributed Processing Symposium, 2015.

[10] Y. Yao, J. Wang, B. Sheng, J. Lin and N. Mi, "HaSTE: Hadoop YARN Scheduling Based on Task-Dependency and Resource-Demand," in Proc. of the IEEE International Conference on Cloud Computing, 2014.

[11] S. Kurazumi, T. Tsumura, S. Saito, and H. Matsuo, "Dynamic processing slots scheduling for i/o intensive jobs of Hadoop mapreduce," in Proc. of the 3rd IEEE International Conference on Networking and Computing, 2012.

[12] S. Ibrahim, H. Jin, L. Lu, B. He, G. Antoniu, and S. Wu, "Maestro: Replica-aware map scheduling for mapreduce," in Proc. 12th IEEE/ACM Int'l Symp. on Cluster, Cloud and Grid Comp., 2012.

[13] Y. Shao, C. Li, W. Dong, and Y. Liu, "Energy-Aware Dynamic Resource Allocation on Hadoop YARN Cluster," in Proc. of the 18th IEEE International Conference on High Performance Computing and Communications, 2016.

[14] P. Li, L. Ju, Z. Jia, and Z. Sun, "SLA-Aware Energy-Efficient Scheduling Scheme for Hadoop YARN," in Proc. of the 17th IEEE International Conference on High Performance Computing and Communications (HPCC), 2015.

[15] jRAPL. A framework for profiling energy consumption of Java programs [Online]. Available: http://kliu20.github.io/jRAPL/