Energy Efficiency for MapReduce Workloads: An In-depth Study

Boliang Feng[1] **Jiaheng Lu**[1] **Yongluan Zhou**[2] **Nan Yang**[1]

[1] School of Information and DEKE, MOE
Renmin University of China,
59 Zhong Guan Cun Avenue, Beijing, PR China 100872,
Email: {blfeng, jiahenglu, yangnan}@ruc.edu.cn

[2] Department of Mathematics and Computer Science
University of Southern Denmark,
Campusvej 55, DK-5230 Odense M,
Email: zhou@imada.sdu.dk

Abstract

Energy efficiency has emerged as a crucial optimization goal in data centers. MapReduce has become a popular and even fashionable distributed processing model for parallel computing in data centers. Hadoop is an open-source implementation of MapReduce, which is widely used for short jobs requiring low response time. In this paper, we conduct an in-depth study of the energy efficiency for MapReduce workloads. We identify four factors that affect the energy efficiency of MapReduce. In particular, we make experiments over four typical MapReduce workloads that represent different kinds of application scenarios and measure the energy consumption with varied cluster parameters. Our key finding is that with well-tuned system parameters and adaptive resource configurations, MapReduce cluster can achieve both performance improvement and good energy saving simultaneously in some instances, which is surprisingly contrast to previous works on cluster-level energy conservation.

Keywords: Energy Efficiency, Performance, MapReduce, Hadoop, Data Center

1 Introduction

Growing demands for large-scale data storage and processing have shifted the bulk of workloads to data centers that typically employ thousands of servers (Yahoo 2008). Energy efficiency is a crucial optimization goal in such large clusters. An early report (EPA 2007) estimated that in US the data centers accounted for roughly 61 billion kilowatt-hours (kWh) (1.5% of the total U.S. electricity consumption) in 2006 at a total cost of 450 million. The number is expected to be doubled by 2011 (Koomey 2008). Furthermore, inadequate approaches of energy management not only result in reduced business competitiveness but also decrease the system reliability. For example, without adequate resource configuration, most servers would be idle most of the time.

Most recent works on energy conservation of larger clusters (Chase et al. 2001)(Elnozahy et al. 2002)(Rajamani & Lefurgy 2003) focused on exploiting low utilization periods and attempted to temporarily switch off some of the servers to reduce the cluster's energy consumption. This type of approaches improve the cluster's energy efficiency by dynamically adjusting the active server set according to the current workload. However, their energy savings are always at the expense of performance degradation since the computational resource is limited by such cluster-level strategy.

On the other hand, *MapReduce* (Dean & Ghemawat 2004) has recently been emerged as a promising programming model for parallel computing in data centers. Unfortunately, energy efficiency has attracted little attention in the design of most existing MapReduce platforms, such as *Hadoop* (Hadoop 2007). We believe that there would be great opportunities to develop more energy-efficient MapReduce clusters since original MapReduce project did not give considerable attentions on energy efficiency.

Actually, the MapReduce programming model imposes new challenges on cluster-level energy conservations. First, existing MapReduce platforms, such as Hadoop, usually perform automatic parallelization and distribution of computations. The aforementioned energy conservation techniques based on dynamic resource reconfiguration are in conflict with the intention of this design. Obviously, turning off a part of servers will have a negative impact on the parallelization and load balance of the cluster.

Second, the cluster configuration should take both computing performance and data storage requirements into account in a distributed system, which complicates the trade-off as well. In addition, MapReduce also provides services for other workloads such as web service and data analysis. Furthermore, MapReduce incorporates mechanisms to be resilient to failures for instance, machine crashes and software failures. Such mechanisms may make negative affects on energy efficiency.

Inspired by the above observations, we conduct an in-depth energy efficiency study of MapReduce workloads on Hadoop, which is a popular open-source MapReduce platform. We aim to address the following two questions. (1) Which factors affect the cluster-wise energy efficiency of a MapReduce platform? (2) Is there any opportunity to perform trade-off between the energy saving and the performance in a MapReduce platform?

To answer the first question, we consider the impact of the architectural design of MapReduce and identify four factors that affect the energy efficiency of MapReduce: CPU intensiveness, I/O intensiveness, replica factors of the underlying distributed file system as well as the file block size. The last two factors are storage-independent parameters. With regard to

the second question, we elaborately design experiments for an in-depth study. We identify four typical workloads of MapReduce that present different kinds of application scenarios by measuring the energy consumption with varied disparate cluster scales and other related factors.

The main experimental findings can be summarized as follows:

1. For CPU intensive and some map-only workloads, such as WordCount and GrepSearch, the key points to minimize energy consumption are to guarantee load balance and the adequacy of CPU resource. Our experiments show that with good load-balance strategies and resource configurations, the energy saving for GrepSearch is 13.0% with response time reduced by 19.1%.

2. We observe that the default merge and grouping algorithm of MapReduce is not efficient in this two cases: (1) I/O intensive workloads, (2)workloads with low map/reduce ratio [1]. Due to the lack of content-aware reducer assignment policy, the method to provide higher I/O throughput by adding more servers cannot always bring energy saving. The experimental results show that TeraSort with 10GB data set obtain less energy saving with the increase of the cluster size in certain cases.

3. While MapReduce is independent of the underlying distributed file system, the two factors: replica factor and block size are both important for energy efficient MapReduce. In general, relatively higher degree of replica factors would always save more energy in most instances. Furthermore, our results also show that GrepSearch with 20GB data set and 512MB block size can obtain 36.8% energy saving in comparison to default setting.

4. Due to the lack of content-aware reducer assignment policy, some methods for improving parallelism cannot obtain significant performance or energy saving improvement for low map/reduce ratio's workload like TeraSort. However, for map-only jobs or high map/reduce ratio's jobs (e.g., GrepSearch,WordCount), our results show that well-tuned MapReduce clusters can reduce energy consumption by 20.0%.

In summary, the MapReduce cluster can, in some cases, achieve performance improvement when the energy consumption is optimized. That is, the two optimization goals: performance and energy saving are coincident in some scenarios. And our results show that well-tuned MapReduce clusters can help saving energy significantly.

The remainder of this paper is organized as follows: Section 2 discusses the factors that affect the energy efficiency of MapReduce. In Section 3, we present the experiment design. An in-depth analysis of experimental results and recommendations are presented in Section4. We briefly review related work in Section 5. Section 6 concludes the paper with future directions.

2 Factors Affecting Energy Efficiency of MapReduce

In this section, we start our analysis by briefly reviewing the MapReduce programming model and the execution flow of a MapReduce job and then conduct an energy efficiency profile for MapReduce.

2.1 MapReduce Preliminary

MapReduce is a programming framework and an associated implementation for processing and generating large data sets. It was originally developed by Google and now MapReduce programs are becoming popular applications in data center at companies like Yahoo!, IBM and Facebook. It has several preeminent features in distributed data management. Its inherently parallel can simplify the complexity of running distributed data processing functions across multiple nodes in a cluster, since MapReduce allows a programmer with no specific knowledge of distributed programming to create his MapReduce functions running in parallel across multiple nodes in the cluster. Also, MapReduce platform can offer fault tolerance that is entirely transparent to programmers, which means it can run on clusters with cheap commodity machines. And in Yahoo!, the MapReduce program is running on cluster with up to 10000 cores (Yahoo 2008).

The MapReduce programming model divides a program into tasks, of which there are two types: map() and reduce(). Both the two functions are defined with respect to data structured in <key, value> pairs. The Map function is applied in parallel to take one pair of data with a type in the input dataset and then produces a list of intermediate pairs for Reduce call. After that, the MapReduce framework collects all pairs with the same key from all lists and groups them together, thus creating one group for each one of the different generated keys. The Reduce function is then applied in parallel to each group, which in turn produces a collection of values. Each Reduce call typically produces either one value or an empty return. The returns of all calls are collected as the desired result list:

$$\begin{aligned} Map(k_1, v_1) &\rightarrow list(k_2, v_2); \\ Reduce(k_2, list(v_2)) &\rightarrow list(v_3) \end{aligned} \quad (1)$$

The node of jobtracker divides the input data of one job into appropriate size splits and the framework assigns one split to each Map function, which is known as job initialization. For a map task, jobtracker takes account of the load balance, parallelism and the tasktracker's network location of the system, picking a task whose input split is as close as possible to the tasktracker. Once the map function is completed, jobtracker simply takes the next in list of yet-to-be-run reduce tasks as the reducer. And intermediate data will be shuffled to different reduce tasks according to their values. Overall, MapReduce are executed in four stages: job initialization, map process, shuffle and reduce process.

Hadoop is a successful implementation of the MapReduce framework. Optimizing replica placement distinguishes HDFS (HDFS 2007) from most other distributed file systems. Generally, each file will have 3 replicas in default to improve data reliability, availability, and network bandwidth utilization. For our work, we select Hadoop as the implementation of MapReduce.

2.2 Energy Efficiency Factors

In this section, we consider the impact of the architectural design of MapReduce and identify four factors that affect the energy efficiency of MapReduce: CPU

[1] Low map/reduce ratio means that the job's most execution time is spend on map stage.

intensiveness, I/O intensiveness, replica factors and block sizes.

2.2.1 Energy Consumption Model for MapReduce.

We present a mathematical model for energy analysis of MapReduce workloads in Formula (2). This model gives an overall view of the energy consumption during the process of a MapReduce job.

$$Energy = PT = P_i T_i + P_m T_m + P_s T_s + P_r T_r \quad (2)$$

where, energy is equal to power (P) multiplies time (T). $P_i T_i$ is the energy consumption of job initialization. $P_m T_m$ and $P_r T_r$ represent the energy consumption during the map stage and reduce stage respectively. And we use $P_s T_s$ to denote the energy consumption of intermediate data shuffling. Based on this model, there are four factors that affect the total energy consumption of a MapReduce job.

2.2.2 CPU Intensiveness Workloads

For the CPU intensive MapReduce workloads (e.g., WordCount), the power consumptions of map and reduce phases are high (Tsirogiannis et al. 2010). However, from the performance standpoint, the intuitive idea to add more CPU resources by increasing the number of servers can decrease the running time. On the other hand, this way would increase the power consumption of the cluster. Therefore, a good balance of this trade-off needs theoretical analysis and experimental testing, since energy consumption is equal to the product of power and time.

2.2.3 I/O Intensiveness Workloads

For I/O intensive MapReduce workloads, partial tasks should wait for the data to be ready for processing. Meanwhile, the idle CPUs may lead to the waste of energy. That is not energy efficient and implies long response time especially for the map and reduce phases. The inherent parallel of MapReduce programming model can take good use of distributed resources in map stage, which can reduce the time of map by well-tuned parameters. On map side, HDFS's rack-aware replica placement policy also helps improve the I/O efficiency and system throughput by distributing the stresses on I/O across all the nodes. In the current implementation of Hadoop, MapReduce can not produce a fine-grained reduce tasks assignment policy to avoid large size of intermediate data shuffle. According to Equation (2), there would be large energy consumption ($P_s T_s$) for such workloads. Therefore, the best way to provide higher I/O throughput is to add more servers.

2.2.4 Replica Factor.

Although the MapReduce programming model is designed to be independent of storage system, HDFS is closely related to MapReduce processing. HDFS's rack-aware replica placement policy that improves data reliability, availability, network bandwidth utilization and system throughput, distributes 3 block replicas across the cluster.

If we do not take storage resources into account, then, intuitively, higher degree of replica factors will provide more choices for MapReduce tasks assignment, which improves load balance and parallelism of the system and decrease the response time T_m and T_r. In Section 4, we will experimentally investigate if this intuition is true and how data replication affects the energy efficiency of MapReduce workloads and discuss the opportunity for replica placement optimization.

2.2.5 Block Size.

Block Size is another important factor of the implementation of Hadoop, which makes affect on energy efficiency for MapReduce as well. For MapReduce, Hadoop divides the input to a MapReduce job into fixed-size pieces (usually equals to block size) called input splits. Hadoop creates one map task for each split. Having many splits means the time (T_m) taken to process each split is small compared to the time processing the whole input. And, the process would be better load-balanced as the splits become more fine-grained. On the other hand, if splits are too small, then the overhead of managing the splits and of creating map tasks ($P_i T_i$) begins to dominate the total job energy consumption. This characteristics' effect on energy efficiency for MapReduce workloads will also be discussed in Section 4.

3 Experimental Design

In this section, we introduce the performance metrics, appropriate workloads and the cluster setup in our energy efficiency studies of MapReduce, which is important for the analysis of software performance.

3.1 Metrics

Metrics are critical for our analysis on energy efficiency characteristics of MapReduce workloads. In this paper, we consider both energy consumption and performance. Generally, the energy cost in data centers is measured in kWh (kilo*Watt*hour) or Wh (Watt*hour). The performance, namely execution time, is a key issue for data center as users are "spoiled" and have not the patience for a long delay in the execution of their jobs in many scenarios. In addition, the degree of replica factor is also a key parameter in distributed file system. Therefore, a bundle of metrics are collected including response time, replica factor, number of machines, and energy consumption. These metrics can help us understand the relationship between the energy cost and the response time.

To measure the overall Energy Efficiency (EE) of a system, we use the following equation:

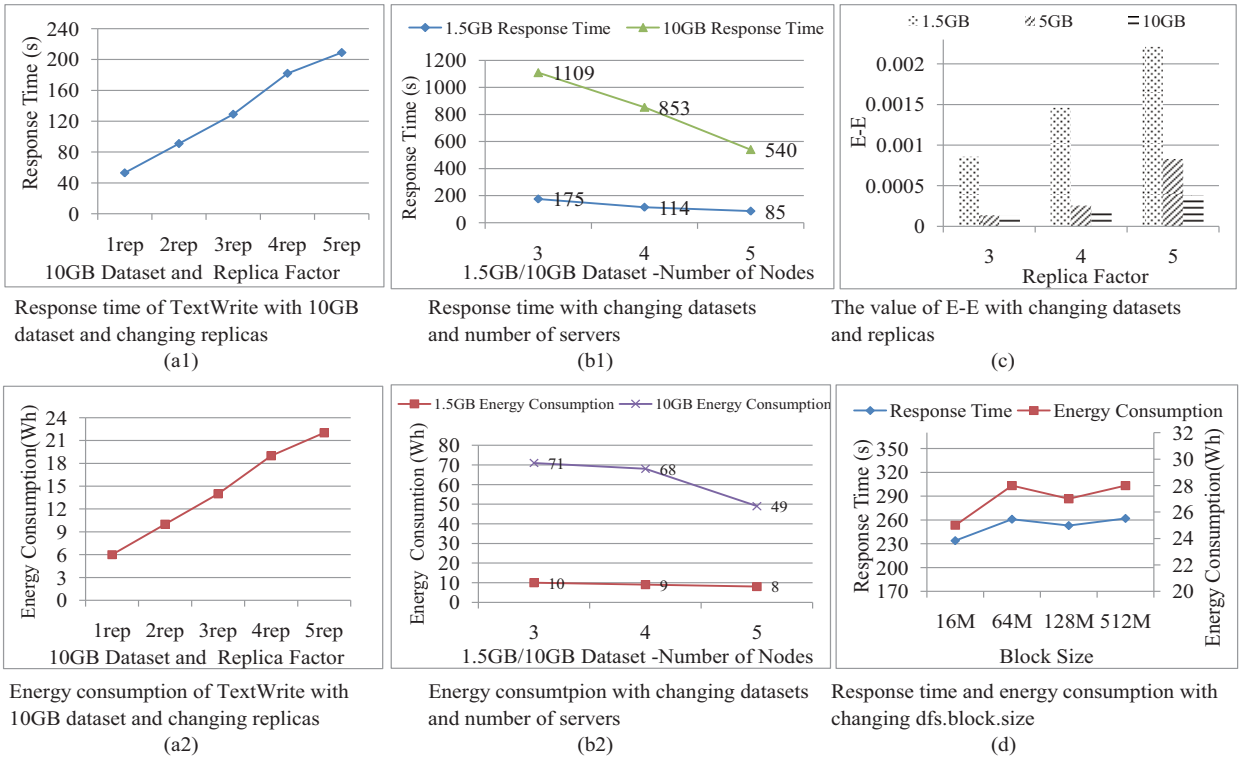
$$EE = \frac{Size\ of\ workload}{Energy \times response\ time} \quad (3)$$

This metric takes into accounts both the energy consumption and the execution time with the varying sizes of workloads.

3.2 Cluster Setup

We used a small cluster of six nodes on high-rate network traffic feed. Each datanode/tasktracker with a 2.4GHZ Intel Core Duo processor running Linux 2.6.31, 2GB RAM and 500GB SATA disk.

The master node that hosts the namemode and jobtracker is equipped with a Quad-core processor, 4GB RAM. All the nodes are connected to each other via 1000Mbps ethernet ports. In the average, the disk performance that we measured is approximately 100MB/s. This node consumes 41W at idle, 67W at



Response time of TextWrite with 10GB dataset and changing replicas (a1)

Response time with changing datasets and number of servers (b1)

The value of E-E with changing datasets and replicas (c)

Energy consumption of TextWrite with 10GB dataset and changing replicas (a2)

Energy consumtpion with changing datasets and number of servers (b2)

Response time and energy consumption with changing dfs.block.size (d)

Figure 1: TextWrite

peak utilization, 1.6W at hibernate. We measured the energy consumption for the whole system. And, system energy is measured by using a plug-in power meter, whose minimal accuracy is 1Wh.

We ran Hadoop version 0.20.2. and java version 1.6.0. The property `mapred.child.java.opts` is set to "-Xmx1024m" and `io.sort.mb` is increased to 250. Other relevant parameters such as number of mappers and reducers per job are defined as follows. The number of mapper equals to the number of splits of the input files. In our experiments it is around 100 mappers/node. And the number of reducer is set around to 0.9 multiply the reduce task capacity, which will change according to the cluster scale.

3.3 Workloads

To comprehensively analyze the energy efficiency of MapReduce, we select the following four typical benchmarks .

1. *TextWrite*: This program writes a large unsorted random sequence of words from a word-list. This job is network-intensive.
2. *GrepSearch*: This program is a map-only CPU intensive job by matching regular expressions from input files. It is high CPU utilization in map stage.
3. *TeraSort*: It is essentially a well-known benchmark, sorting the official input datasets. It is CPU bound in the map stage and I/O bound in the reduce stage.
4. *WordCount*: This program is a balance between CPU-intensive and I/O intensive jobs. It reads the text input files, breaks each line into words and distributes them to multiple machines and finally counts them.

In order to measure the system performance under different load pressures, we run each job with three different data sizes from 1.5GB to 20GB. And the replica factor is varied from 1 to 5. The block size ranges from 16MB to 1GB. For each job, we run three trials to find the average energy consumption and response time.

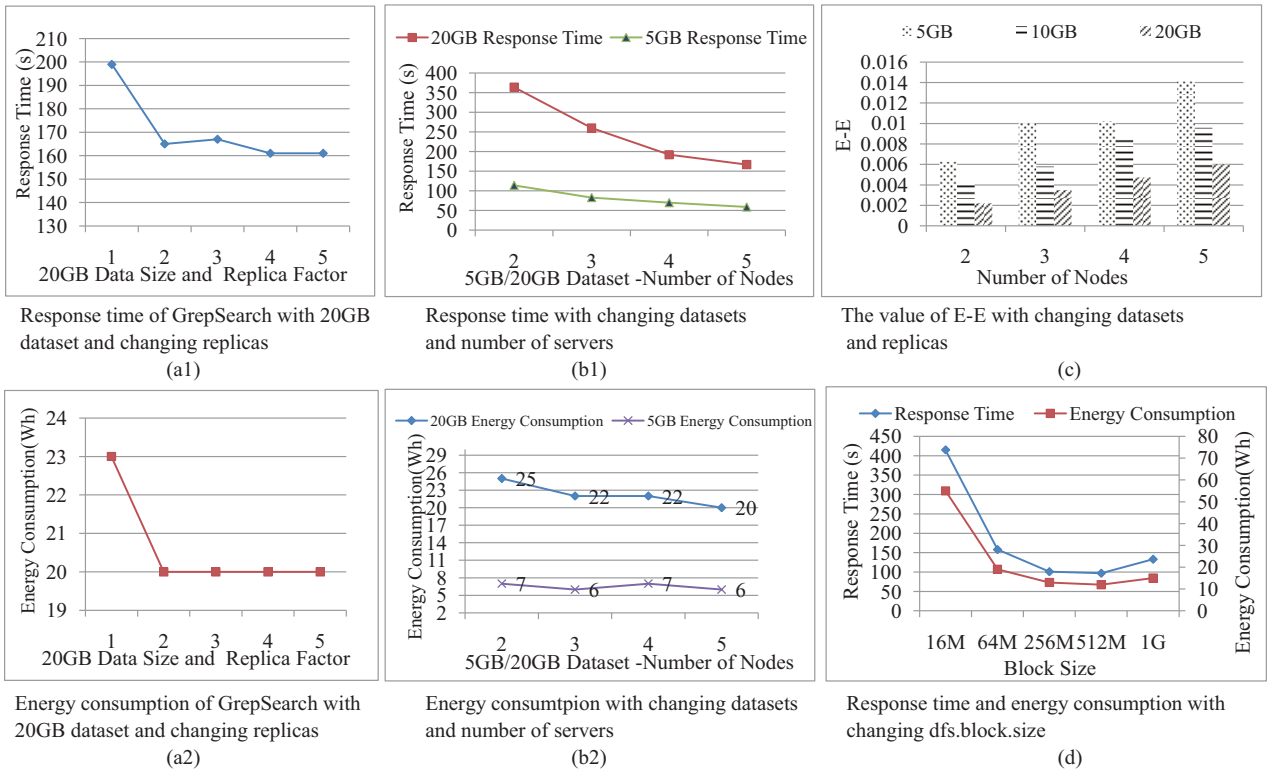
4 Analysis of Results

Our results suggest that energy saving and performance are not two divided optimization goals in some cases. In this section we analyze the experimental results. In general, our results come in contrast to recent work (Leverich & Kozyrakis 2010), which suggested that energy efficiency would already be at the cost of performance degradation. Our results suggest that energy saving and performance are not two contradict optimization goals in some cases.

4.1 TextWrite

HDFS creates multiple replicas (the default is 3) of data blocks and spreads them throughout a cluster to enable reliable, extremely rapid computations. We use map-only workload *TextWrite* to write large data files with varied workload size, cluster scale, replica factor and block size. It is network intensive workload for data transfer.

Fig.1 shows the results of this experiment. The replica factor is set from 1 to 5. And, the workload size ranges from 1.5GB to 20GB. As depicted in Fig.1.a, it has almost a linear growth of both latency and energy consumption with the replica factor increasing. Analogously, a larger workload size would result in proportional increase of the response time as well. Obviously, the reason is that higher degree of replica factor means more data should be copied



Response time of GrepSearch with 20GB dataset and changing replicas (a1)

Response time with changing datasets and number of servers (b1)

The value of E-E with changing datasets and replicas (c)

Energy consumption of GrepSearch with 20GB dataset and changing replicas (a2)

Energy consumtpion with changing datasets and number of servers (b2)

Response time and energy consumption with changing dfs.block.size (d)

Figure 2: GrepSearch

across the nodes. For this network intensive and I/O workload, it is totally not energy efficient. Some irrelevant components with high power consumption like processor are in low utilization during the runtime. They only consume the power but do no contribution to performance. This reason is also can be used to explain why the larger the data set is, the smaller the E-E is, as showed in Fig.1.c.

We also show the results of *TextWrite* with varied cluster size and workload size in Fig.1.b (replica factor is the default 3). We see that with more nodes in the cluster, it takes great improvement on the performance, around 51.3%. Meanwhile, energy consumption is decreased from 71Wh to 49Wh. We believe this behavior is reasonable. This can be explained that although more nodes means the increase of power consumption of the whole cluster, the response time reduction can trade-off the energy consumption, as the total energy consumption is the product of time and power.

Fig.1.d shows the results with varied block sizes. When the block size is as small as 16MB, the system can obtain both high performance and energy saving. The reason is that small block size enables more tasks to be processed in parallel. And when the block size is larger than 64MB, the parallelism of the system is reduced, so that energy consumption increases significantly.

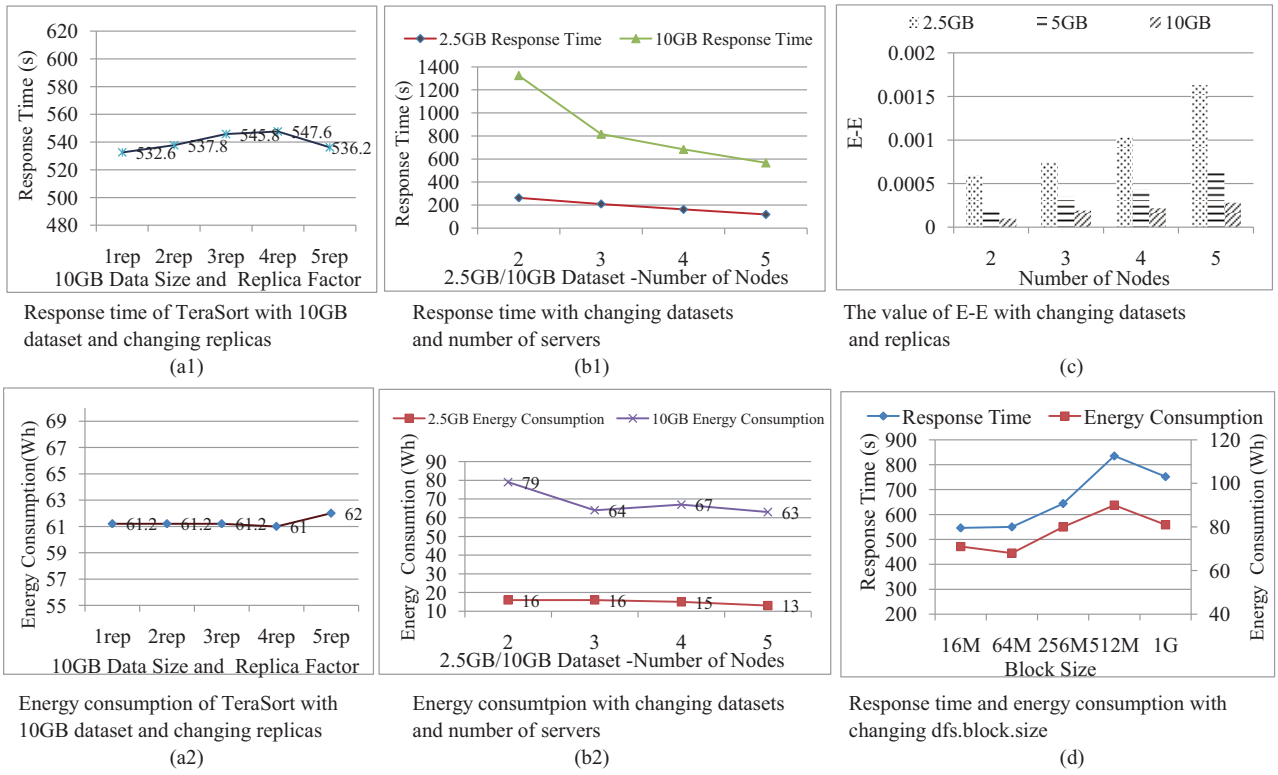
4.2 GrepSearch

GrepSearch is a map-only and scan-like job that searches files globally for lines matching a given regular expression. In our experiments, the file is a text file with words list and the regular expression is set to be '$\hat{}a$' (match the words that begin with 'a'). These experiments are also implemented with varied workload size, cluster scale, replica factor and block size. It is high CPU utilization in map stage.

Fig.2 shows the results of the GrepSearch workload. We have several interesting observations. No matter how large the workload is, the worst case always exists when the replica factor is 1. We think this reason is quite simple. Higher degree of replica factor means more choices for tasks assignment, improving the load balancing of the system. In other words, Jobtracker would have more choices to assign Map tasks, taking into account data location and request number of the targeted nodes. And, once the replica factor is larger than 2, the response time and energy consumption are approximately reduced by 19.1% and 13.0% respectively. This also prove that HDFS replica placement policy not only improve data reliability, availability, but also improve the parallelism of the GrepSearch workload.

Fig2.b and Fig2.c present the results of GrepSearch with varied cluster size and workload size. With workload size increasing from 5GB to 20GB, the value of E-E is reduced. We believe that long latency makes irrelevant components run in low utilization and thus waste the energy. On the other hand, more nodes in the cluster means more resources and better performance that make the system more energy efficient. As depicted in Fig.2.b, when the workload size is large (20GB), the system can achieve more energy saving (20.0%) with the cluster size increasing. However, when the workload size is as small as 5GB, the initial cost for each tasks can not be amortized, and resources are sufficient. Thus, there is no obvious energy saving with the increase of the cluster size.

Fig.2.d shows the results with varied block size. When the block size is as small as 16MB, it means around 1280 maps should be initialized. The overhead for the creation of map tasks can not be offset by the superiority of parallel processing. We can see that, by well-tuned block size, the system can obtain energy saving by as much as 36.8% with the response time



Response time of TeraSort with 10GB dataset and changing replicas
(a1)

Response time with changing datasets and number of servers
(b1)

The value of E-E with changing datasets and replicas
(c)

Energy consumption of TeraSort with 10GB dataset and changing replicas
(a2)

Energy consumtpion with changing datasets and number of servers
(b2)

Response time and energy consumption with changing dfs.block.size
(d)

Figure 3: TeraSort

reduced by 38.6%.

4.3 TeraSort

TeraSort is essentially a sequential I/O benchmark. It is CPU bound in map stage and I/O bound in reduce stage. We set the output file's replica factor to be 1 to avoid additional network cost. During the reduce stage, CPUs are almost idle, waiting for I/O intensive jobs. Again, we present the results of *TeraSort* with varied replica factor, workload size, cluster scale and block size in Fig.3.

TeraSort is I/O bound program waiting for memory swapped in both map and reduce stages that making WAIT CPU occupies a large proportion of the total CPU utilization. There is a random fluctuation in the measurements when we run TeraSort with different replica factors. We run this experiment 5 times and calculate the average value, which is showed in Fig.3.a. We can not see significant changes of performance with varied replica factor. We believe this behavior is due to the process of MapReduce framework. The two key factors that affect the energy efficiency of I/O intensive workloads are load balance and reduce tasks assignment. More replicas would improve the load balance of map tasks, but there also would be large data transfer among nodes in the shuffle stage. Meanwhile, a task with high latency will affect the progress of the whole job. Unfortunately, MapReduce could not give a more fine-grained performance optimization, since the process of data transfer are related to the value of the intermediate key-value pairs.

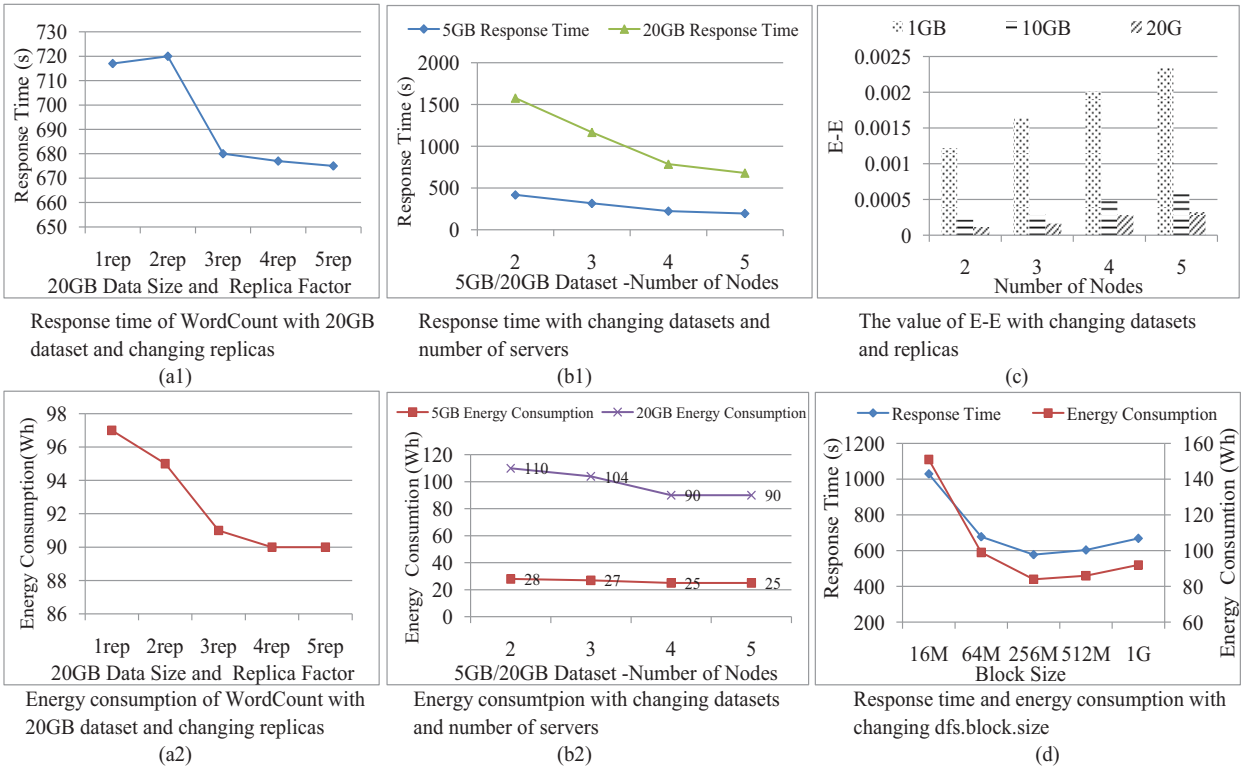
As depicted in Fig.3.b, with varied cluster size and workload size, *TeraSort* shows a similar result of EE as GrepSearch. A larger workload would increase the job runtime, and more IT components will lead to more energy waste. Generally, the intuitive way to provide higher I/O throughput is to add more servers. So with the expansion of the cluster size, the system achieves energy consumption reduction by 20.2%. Let us take a closer look at the results at the point when the number of nodes is 3. At this moment, the job needs few data transfer after map tasks according to the data location and partition results. That also explains why there is obvious reduction of response time reduced from 1325s to 816s. Fig.3.c shows the value of E-E with varied cluster size and workload size.

Fig.3.d shows the results with varied block size, which are quite different from the results of GrepSearch, because most cost of such workloads comes from the shuffle stage and reduce stage. And, small block size means fine-grained input splits which will improve the performance of reduce sort. When the block size is 1GB, we can see the decreased response time and energy consumption. We believe that 1GB size of splits brings less data shuffle after map processing.

4.4 WordCount

WordCount achieves a balance between CPU- and IO-intensive workload in both two stages: map and reduce. We set the output file's replica factor to be 1 to avoid additional network cost. Each map takes a line as input and breaks it into words. It then emits a key/value pair of the word and 1. Each reducer sums the counts for each word and emits a single key/value with the word and sum. It is really energy intensive, employing almost 95% of the potential CPU. We implement this experiment with varied replica factor, workload size, cluster scale and block size.

It is an interesting observation that increasing the degree of replica factor can always improve the system performance. When the replica factor is larger than 2, the system achieves a significant performance improvement by 6.2% with energy consumption reduced by 7.2%. The reason is that larger degree of replica



Response time of WordCount with 20GB dataset and changing replicas (a1)

Response time with changing datasets and number of servers (b1)

The value of E-E with changing datasets and replicas (c)

Energy consumption of WordCount with 20GB dataset and changing replicas (a2)

Energy consumtpion with changing datasets and number of servers (b2)

Response time and energy consumption with changing dfs.block.size (d)

Figure 4: WordCount

factors implies more opportunities for load balance, which is good for CPU intensive workloads. Furthermore, WordCount is different from TeraSort. Its two stages map and reduce account for nearly half of the whole process but TeraSort's map/reduce ratio is around 2/3.

With more nodes added in the cluster, both response time and energy consumption decrease by 56.8% and 18.2% with 20GB data set. We can see that increasing the number of servers can achieve certain energy saving. The increase of energy consumption due to the new nodes can be offset by the reduction of response time.

Fig.4.d shows the results with varied block size. As we excepted, *WordCount* shows similar behaviors to *GrepSearch* that small block size brings out high cost for tasks initialization that can not be amortized in the task execution time. On the other hand, a large block size such as 1GB will make negative effects on parallelism of the system, so that we can see the rising of both response time and energy consumption when the block size is 1GB.

4.5 Summary of Experiments

Based on the above experimental results, we give an overall observation combining these four factors in the following. And, we make several suggestions with regard to improving energy efficiency of MapReduce cluster.

4.5.1 CPU Intensive Workloads.

For CPU intensive and high map/reduce stage time ratio workloads (e.g., WordCount, GrepSearch), the best way to minimize energy consumption is to guarantee load balance and adequacy of CPU resources, for example, increasing the cluster scale. These methods are also make positive affect on performance. The experimental results show that WordCount with 20GB data set can obtain 14 Wh energy saving when the number of nodes increases from 3 to 5. And, the response time decreases by 56.8%.

There is no surprise that high degree of replica factors will always give more opportunities for load balancing. Meanwhile, it is a great challenge for task scheduler of MapReduce on how to decide which node is the best one for a specific map or reduce task. And we need an appropriate cluster-level reconfiguration strategy for matching the number of active nodes to the current needs of the workload.

4.5.2 I/O Intensive Workloads.

For I/O intensive workloads, fine-grained input splits are effective for shuffle and reduce stages, which is more energy efficient on condition that the energy cost for job initialization can be amortized. As MapReduce can not give a content-aware tasks assignment policy to avoid high cost of the shuffle process, larger scale of cluster can not give better parallelism and load balancing in some cases. It is worse for I/O intensive workloads, in which a large size of intermediate data should be shuffled, such as *TeraSort*. For example, we can see that TeraSort with 10GB data set running on 3 nodes consume less energy (64Wh) than running on 4 nodes (67Wh) in Figure 3. We believe that optimized replica placement policy of HDFS will help the map and reduce tasks assignment for energy efficiency and improved data partitioning algorithms and content-aware reduce tasks scheduling strategies are needed. And the ideal condition is that less data will be shuffled across the cluster before the reduce stage.

4.5.3 Replica Factor.

Most workloads running with higher degree of replica factors would obtain better performance and larger amount of energy saving especially for CPU intensive and high map/reduce ratio workloads. For these jobs, high degree of replica factor implies better load balance of the system, making a positive effect on both performance and energy efficiency. For I/O intensive workloads, this method can not provide significant energy saving.

4.5.4 Block Size.

Both the CPU intensive and high map/reduce ratio jobs show that a large block size will reduce the cost for tasks initialization, which will help improve the performance and energy efficiency of the system. But, for I/O intensive workloads such as *TeraSort*, a small block size means fine-grained splits, which will make positive effect on data shuffle and reduce process and bring the reduction on both response time and energy consumption.

4.5.5 Key Finding.

It is an interesting observation that the MapReduce cluster would obtain both energy saving and performance improvement in some scenarios. That is, energy saving and performance are not two opposite optimization goals in some cases for energy efficient MapReduce cluster. For example, by the well-tuned cluster setting, *Terasort* with 2.5GB data set can obtain energy saving by as much as 18.8% and response time decreases from 209s to 118s. And by well-tuned block size, *GrepSearch* with 10GB data set can obtain energy saving by as much as 36.8% with the response time reduced by 38.6%. And, with more nodes added in the cluster, energy consumption of *TeraSort* will decrease by 18.8% and response time decrease from 209s to 118s.

4.5.6 Recommendations.

Our findings are summarized into the following recommendations:

1. Workloads should be designed to be heterogeneous, taking advantage of as many as components of the system and allowing tasks on different components to overlap.
2. For write intensive MapReduce workloads, the system should power on nodes as fewer as possible without causing any loss of data dispersion and availability for predicted jobs.
3. For CPU intensive and high map/reduce ratio workloads, appropriate number of servers should be provided based on the workload size, ensuring load balance and adequate CPU resource.
4. For I/O intensive workloads, fine-grained input splits is effective for shuffle and reduce stages, which is more energy efficient on condition that the initialization cost can be amortized.
5. The default value of HDFS replica factor is appropriate for both performance and energy efficiency in most cases. In some cases, a lower HDFS replication level is better, but it is not possible when data durability is important.
6. The value of block size demonstrates a trade-off between the overhead for tasks initialization and parallelism of the system. Generally, the default value is too small for large files.
7. Improved data partitioning algorithms in map stage and content-ware reduce scheduling strategies are very important for energy efficiency, where more efforts on refinements and improvements are needed.

5 Related Work

Previous work on the energy efficiency of the large-scale data processing is committed to make a energy-proportional system (Barroso & Hölzle 2007). On the cluster-level energy management, all the works focus on dynamic cluster reconfiguration, aiming to reassign resource of the data center with a principal focus on energy management, and design workload distributed polices to shift the workloads around for meeting power consolidation (Chase et al. 2001)(Elnozahy et al. 2002)(Rajamani & Lefurgy 2003). Researchers seek to make balance among load balance, time duration and energy consumption, or give a extreme method for energy saving, do not take performance into account. However, energy-efficient MapReduce cluster has not attracted enough attentions. Recent work (Leverich & Kozyrakis 2010) attempts to scale down the Hadoop cluster to obtain energy saving, but they did not give comprehensive evaluation in practice. Two broad strategies for MapReduce energy management are compared in (Lang & Patel 2010), and a new strategy called AIS is proposed in their paper. Their results show that the computational complexity of the workload will affect the effectiveness of the two cluster-level strategies, but they do not give further discussion about the MapReduce Workloads. Chen et.al. (Chen et al. 2009) add energy as another performance metric to consider in the design space for computer systems and data centers and study on MapReduce operation variables. They characterize the performance of the Hadoop implementation of MapReduce under different workloads and then give some conclusion. In this paper, we explicitly raise two questions about energy efficient MapReduce workloads, and comprehensively analyze the factors that affect energy efficiency based on the energy consumption model and the MapReduce Framework. Then experiments are elaborately designed for comprehensive analysis on energy-efficient of MapReduce workloads on a cluster view.

On the application-level, efforts to develop energy-efficient data processing applications can also be found in (Graefe 2008)(Tsirogiannis et al. 2010)(Xu et al. 2010). Lang et.al. (Lang & Patel 2009) present proposal for considering energy efficient query processing in a database management system. (Chen, Ganapathi & Katz 2010) analyze how compression can improve performance and energy efficiency for MapReduce workloads. It is a trade-off between IO consumption and CPU consumption. For several workloads, their compression decision algorithm can help provide significant energy saving. However, they do not give a view on cluster-level optimization. (Chen, Ganapathi, Fox, Katz & Patterson 2010) present a statistics-driven workload generation framework that distills summary statistics from production MapReduce traces and realistically reproduces representative workloads for better MapReduce energy efficiency evaluation mechanisms.

On the node architecture, several technologies such as PowerNap (Meisner et al. 2009), Intel's Speed-

Step and AMD's PowerNow! can give us fine grained control on energy management. Recent discussions on fundamentally new cluster server design are highly relevant to ideas of cluster-level energy management (J.Hamilton 2009)(Lang et al. 2010)(Vasudevan et al. 2009). Energy-efficiency benchmarks for MapReduce are discussed in (Poess & Nambiar 2008)(Rivoire et al. 2007). Meanwhile, related industrial standard about energy efficient data center has been proposed, such as PUE (PUE 2005) and DCiE (DCiE 2005). Finally, note that, although Hadoop Mapreduce framework has become a popular and even fashionable application in data center, energy efficient MapReduce cluster has not been well studied in the previous works.

6 Conclusion and Future Work

In this paper, we conducted an in-depth energy efficiency study for MapReduce workloads in an open source implementation Hadoop. In order to comprehensively analyze the opportunities for energy saving, we identified four factors that affect the energy efficiency of MapReduce. Then, we chose four typical workloads of MapReduce and measured the energy consumption with varied cluster scales and factors. The experimental results showed that with well-tuned system parameters and adaptive resource configurations , MapReduce cluster can achieve both significant energy saving and performance improvement simultaneously in some instances.

Based on the exciting finding in this paper, in the future, we plan to verify the results of this paper in a larger size of clusters. Furthermore, more benchmarks of MapReduce should be introduced in our experiments, such as web search and machine learning applications. In addition, we intend to investigate the effects of changing other parameters of MapReduce and Hadoop, such as IO parameters like the number of parallel reduce copies, memory limit, and file buffer size. We also intend to work on the content-aware assignment policy for reduce tasks to increase data locality, which is important to increase parallelism of MapReduce clusters.

Acknowledgement

This paper is partially supported by the Research Funds of Renmin University of China (No: 11XNJ003) , NSF China (Project No. 61170011) and Beijing Natural Science Foundation (Project No. 4112030).

References

Barroso, L. A. & Hölzle, U. (2007), 'The case for energy-proportional computing', *IEEE Computer* **40**(12).

Chase, J. S., Anderson, D. C., Thakar, P. N. & Vahdat, A. M. (2001), Managing energy and server resources in hosting centers, *in* 'SOSP'.

Chen, Y., Ganapathi, A. S., Fox, A., Katz, R. H. & Patterson, D. A. (2010), Statistical workloads for energy efficient mapreduce, *in* 'UC Berkeley Technical Report'.

Chen, Y., Ganapathi, A. S. & Katz, R. H. (2010), To compress or not to compress compute vs. io tradeoffs for mapreduce energy efficiency, *in* 'ACM SIGMOD workshop on Green networking'.

Chen, Y., Keys, L. & Katz, R. H. (2009), Towards energy efficient mapreduce, *in* 'UC Berkeley Technical Report', pp. number UCB/EECS–2009–109.

DCiE (2005), 'http://en.wikipedia.org/wiki/data center infrastructure efficiency'.

Dean, J. & Ghemawat, S. (2004), Mapreduce: Simplified data processing on large clusters, *in* 'OSDI', pp. 137–150.

Elnozahy, E. N., Kistler, M. & Rajamony, R. (2002), Energy-efficient server clusters, *in* 'PACS', pp. 179–196.

EPA (2007), 'Report to congress on server and data center energy efficiency public law 109-431', *Public Law* pp. 109–431.

Graefe, G. (2008), Database servers tailored to improve energy efficiency, *in* 'EDBT', pp. 24–28.

Hadoop (2007), 'http://wiki.apache.org/hadoop/poweredby'.

HDFS (2007), 'http://hadoop.apache.org/hdfs/'.

J.Hamilton (2009), Cooperative expendable microslice servers (cems): Low cost, low power servers for internet-scale services, *in* 'CIDR'.

Koomey, J. (2008), 'Worldwide electricity used in data centers', *Environmental Research Letters* **3**(12).

Lang, W. & Patel, J. M. (2009), Towards eco-friendly database management systems, *in* 'CIDR'.

Lang, W. & Patel, J. M. (2010), Energy management for mapreduce clusters, *in* 'VLDB Endowment'.

Lang, W., Patel, J. M. & Shankar, S. (2010), Wimpy node clusters: What about non-wimpy workloads?, *in* 'DaMoN'.

Leverich, J. & Kozyrakis, C. (2010), 'On the energy (in)efficiency of hadoop clusters', *SIGOPS* **44**(1), 61–65.

Meisner, D., Gold, B. T. & Wenisch, T. F. (2009), Powernap: eliminating server idle power, *in* 'ASPLOS', pp. 205–216.

Poess, M. & Nambiar, R. O. (2008), Energy cost, the key challenge of today's data centers: A ppower consumption analysis of tpc-c results, *in* 'PVLDB', pp. 1229–1240.

PUE (2005), 'http://en.wikipedia.org/wiki/power usage effectiveness'.

Rajamani, K. & Lefurgy, C. (2003), On evaluating request-distribution schemes for saving energy in server clusters, *in* 'ISPASS', pp. 29–43.

Rivoire, S., Shah, M. A., Ranganathan, P. & Kozyrakis, C. (2007), Joulesort: a balanced energy-efficiency benchmark, *in* 'SIGMOD'.

Tsirogiannis, D., Harizopoulos, S. & Shah, M. A. (2010), Analyzing the energy efficiency of a database server, *in* 'SIGMOD', pp. 231–242.

Vasudevan, V., Franklin, J., Andersen, D., Phanishayee, A., Tan, L., Kaminsky, M. & Mararu, I. (2009), Fawndamentally power-efficient clusters, *in* 'USENIX'.

Xu, Z., Tu, Y.-C. & Wang, X. (2010), Exploring power-performance tradeoffs in database systems, *in* 'ICDE', pp. 485–496.

Yahoo (2008), 'http://developer.yahoo.com/blogs/hadoop'.