# Towards Energy Proportional Cloud for Data Processing Frameworks

Hyeong S. Kim   Dong In Shin   Young Jin Yu   Hyeonsang Eom   Heon Y. Yeom
*School of Computer Science and Engineering*
*Seoul National University*
*Seoul, Korea, 151-744*
hskim@dcslab.snu.ac.kr

## Abstract

Energy efficiency in cloud computing is becoming more and more important for IT operators of data centers. Several effort to use low power machines in the data center level has been explored. Also, data processing frameworks such as MapReduce and Hadoop are frequently used to process data intensive jobs. However, there have not been an extensive study on the impact of low power computers on such data processing frameworks. Actually, development of low power computers is demanding the architectural paradigm shift for cloud applications. In this paper, we evaluate Apache Hadoop on low power machines and study the feasibility of them in cloud systems. We also propose AnSwer (Augmentation and Substitution), an energy saving method to reduce energy consumption by introducing low power machines. In AnSwer, augmentation and substitution complement each other to prevent data loss and to improve overall power consumption.

## 1 Introduction

Recent advances in cloud computing is driving the heavy use of world-wide data centers in various ways. A great deal of services, (Infrastructure as a Service, Platform as a Service, Software as a Service, Data Storage as a Service) are emerging by the advance of cloud computing techniques. A large number of companies have been already benefited by delegating the operation of IT services to the cloud service providers. Along with this progress, energy efficiency is becoming more and more important for future computing environment. The cost of operating data center computers is rapidly increasing. Environmental Protection Agent (EPA) recently reported that 1.5% of the total US energy use in 2006 was used to power data centers, and this is expected to nearly double by 2010 [1]. Hamilton reported that the data centers of Amazon.com are facing highly increased power demand and this accounts for 59% of the total budget with three year amortizations [15], and also insisted that power distribution is already fairly efficient. Therefore, we should keep our attention on reducing the power delivered to the servers, which accounts for 59% of the total power supply. There are also environmental concerns which is yielding the regulations for environment protection. For example, at the recent UN climate summit, it is widely expected that industrialized nations will agree on a more comprehensive, actionable climate agreement to succeed the Kyoto Protocol.

The opportunity is here that there are still much room to reduce the power consumption of data center in many ways. Another report states that it can be expected that savings of the order of 20% can be achieved in server and network energy consumption with respect to current levels [2]. Much effort has been done to improve the energy efficiency of the cloud computing [6, 8, 25]. As one of such effort, Barroso et al. have proposed the concept of energy proportional computing [7]. They analyzed the Google's commodity servers and observed that the traditional servers lack the energy proportional property, which means the servers consume significant amount of power even if the servers are in idle state. They also indicate that for modern distributed systems, the aggregate cost/performance ratio of an entire system is much more important [5]. Due to this, we advocate the powering down techniques to view the global cloud system as an energy proportional computing system.

Among other cloud applications, we focus on the data processing frameworks. Data processing frameworks such as MapReduce [11], Pig [22], and Sector/Sphere [13, 14] are instances of the most widely used applications in cloud computing systems. The frameworks are especially optimized for the data intensive applications. To minimize the damage incurred by the significant amount of data transfer, they place each computing task as closely to the node which occupy the input file as possible. This property significantly reduces the data transfer overhead, which consequently greatly improves the overall running time. To support this, those systems use user level file systems such as Hadoop File System (HDFS), Google File System (GFS) [12] and Sphere. They usually do not distinguish the data node with compute node, where the data node is also a part of computation nodes. This becomes a problem when considering suspending[1] servers, which may cause temporal data unavailability. There are two conflicts that makes it hard to power down servers for data processing frameworks.

- Data locality

  As we have noted, the data processing frameworks usually treat the data node and the compute node in the same manner. Therefore, we cannot ignore the data placement policy when we suspend partial servers.

- Performance

  If we suspend partial servers, we might experience performance problems in data processing frameworks. This is because the suspended servers are not only used for the distribute storage, but also needed for the data processing.

Fig. 1 shows the problem definition. Suppose that when the data center falls short of power supply, operators decide to turn off several servers. If they turn off the *server4* and *server5* of Fig. 1, then two blocks (gray and half white half zigzag) will be temporarily unavailable. Also, in the system's viewpoint, the powering down two servers inevitably leads to the degraded performance even if they are willing to sacrifice performance for the improved power efficiency. We augment existing servers with low power computers.

There have been a couple of solutions which consider the redundancy to address this problem. To address this short-fall in Hadoop's data-layout strategy, Leverich et al. proposed a new invariant for during block replication: at least one replica of a data-block must be stored in a subset of nodes we refer to as a covering subset [17]. They used the covering subset to prevent data loss in case of suspending partial servers. In this way, they provided the ability to reduce power consumption. Harnik et al. address this by introducing the *full coverage set* [16]. They did not consider the data processing frameworks, but focus on the data placement policy of generic distributed file systems. The problems of these solutions is threefold. First, suspending servers reduces the number of active servers, which inevitably damages the processing power of the cluster. Since one of objectives of previous work is to maximize the nodes which are considered safe to be turned off, there will be a significant performance degradation of the whole system. Second,

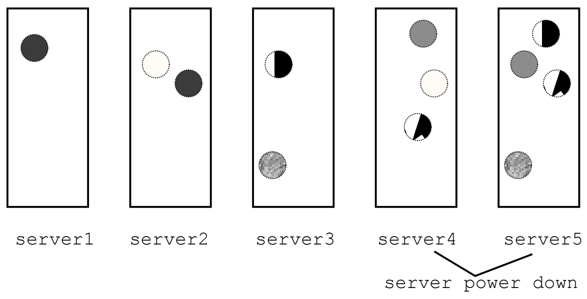

Figure 1: Some chunks becomes unavailable when partial servers are suspended

they still use high end or at least commodity computer systems to replace or augment servers which are suspended for a given time period. Along with the first problem, we cannot put many servers in place of suspended servers since they also require much power. We overcome this by introducing low power servers instead of existing servers. Since their power consumption is much smaller than that of servers, we can put a relatively large number of low power computers in place of servers. This can greatly enhance the availability and reliability of data because the number of decreased servers is smaller than that of when using existing high end servers. Finally, the primary concern of those solutions is to avoid potential data unavailability for distributed storage systems. However, for data processing frameworks, not only the replicas that are going to completely disappear, but also the replicas which are partially lost are important since the data processing frameworks use those replica map information to schedule the compute tasks. If the distribution of replicas is biased towards the servers, the low power servers will have to acquire the data from remove locations to process each compute task. Our solution addresses these problems and lies between two extremes, in essence, we use both of these techniques to complement each other. We propose AnSwer (Augmentation and Substitution), which is a rack-based augmentation method to complement the power eating servers. Similar to our work, recent studies have shown that employing low-power, low-cost machines for data centers is a good solution to improving the energy efficiency [15, 19]. In this paper, we answer the following questions for the data processing frameworks.

- Feasibility of low power computers instead of commodity servers

  We study the feasibility of augmenting the high performance servers with the low-power, and low-cost machines for running data processing frameworks. We show the per-server performance of 4 types of machines for running sort, and other benchmark tools with Apache Hadoop [4].

- Architectural shift by integrating energy efficient machines with traditional servers

  The current data centers are already full of a large number of servers or commodity computers. If data centers are willing to exploit low power machines for economical, and environmental motivations, how can those energy efficient servers can be integrated with the existing servers?

This paper is organized as follows. Section 2 presents techniques related to cloud and energy efficiency solutions. Section 3 shows several proof that low power computers perform well enough to partially replace the existing high end servers in running data processing frameworks. Section 4 proposes our scheme and Section 6 concludes our work.

## 2 Related Work

There have been many studies on the improvement of the energy efficiency for storage systems [10, 18, 21, 23, 27–31]. Most of them deal with the energy efficiency of RAID storage systems. Among them, *Diverted Accesses* is the first effort to exploit redundancy to conserve energy in storage systems [24]. Exploiting redundancy is a nice technique to inactivate idle servers in storage systems. It segregate the storage system into two sets, original and redundant copies, respectively. Our approach is similar to their work in that we leverage the inherent redundancy policy of data processing frameworks to conserve system's overall energy.

Harnik et al. [16] propose a method that obtains full coverage without modifying the data placement functions of existing storage systems by introducing auxiliary nodes. Their primary contribution is that they do not modify the data placement policy of the existing storage systems. However, their primary concern is the problem of data unavailability for the generic distributed storage systems. Hadoop has been studied by many researchers in terms of performance, reliability, and availability. Recently, considering energy efficiency in Hadoop is being actively explored. Leverich et al. [17] introduce an additional invariant for the Hadoop's data placement policy. They use Hadoop's global knowledge to place data blocks in one of fixed servers, which is called the covering subset. In this way, they prevent the temporal data loss caused by the suspended servers. Their solution is to use other power consuming devices to reduce the overall power. Chen et al. [9] present an analytical framework for software energy efficiency, including efficiency of systems such as MapReduce. They experimentally analyzed the Hadoop's energy consumption in many cases. They also provide a quantitative model to provision the servers depending on the current power drain.

Several effort to use low power machines in the data center level has been explored. Hamilton et al. have proposed an Athlon-based sled server design, called the Cooperative Expendable Micro-Slice Servers (CEMS) [15]. Each server consists of dual core AMD processor, and Mini-ITX board. In addition, Each sled has six servers, six disks and a single shared power supply. They show a reasonable performance considering the power consumption and total cost of ownership (TCO). Lim et al. also analyze the performance of low power computers with Internet application workloads and propose a new server architecture with low power computers for warehouse-computing environments. There are ARM CPU based servers that are running Web servers as well. LinuxArmOrg [20] is such a project which built server blade systems of Marvell MV87100. By contrast, there have been quantitative analysis on the microarchitectural study of small cores [26]. They prototyped a search system with Atom-based multicore machines and analyzed the such behaviors as power consumption and microarchitectural activity. FAWN(a Fast Array of Wimpy Nodes) [3] is a cluster of cost-effective components, such ad low-power, efficient embedded CPUs and the flash storage.

Finally, Ranganathan has studied a large number of researches regarding energy efficiency and proposed several approaches [25] for future research. Two directives inspired our approach. First, we spend somebody else's power to energy efficiency at the expense of the low power machines. Second, we spend power to save power. We substitute part of the servers with low power machines.

## 3 Feasibility of Low Power Machines for Data Processing Frameworks

### 3.1 Server Measurements

Our primary concern is to augment high performance or commodity systems with low-power machines for data processing frameworks. Therefore, we have to consider whether utilizing low power computers performs efficiently enough to augment existing high end servers. We quantitatively analyze the performance of Hadoop jobs under various computing power environments. We have set up four distinctive classes of computers, two for server environments, and two for low power computing environments. First two classes represent high-end server and commodity server environment, and last two low power computer environment respectively. The servers which are used in our evaluation are shown in Table 1. *Svr1* and *Svr2* are typical server machines whereas *Low1* and *Low2* are small machines whose dimensions are only *215x210x55mm* and *101x115x27mm* respec-

Table 1: Server specifications

| Name | CPU | Cores | Memory | CPU TDP | Measured Power Consumption | Cost | Remarks |
|---|---|---|---|---|---|---|---|
| Svr1 | Intel Xeon X5450 3.00 GHz | 2 x 4 | 16 GB DDR2 | 120 W | Peak/360 W, Idle/228 W | $3,200 | prepackaged server |
| Svr2 | Intel Core2 Quad Q9550 2.83 GHz | 4 | 8 GB DDR2 | 95 W | Peak/125 W, Idle/69 W | $1200 | |
| Low1 | Intel Atom 330 1.60 GHz | 2 | 2 GB DDR2 | 8 W | Peak/33 W, Idle/25 W | $390 | Zotac ION motherboard |
| Low2 | Intel Atom Z530 1.60 GHz | 1 | 1 GB DDR2 | 2 W | Peak/12 W, Idle/7 W | $360 | fitPC2 |

tively. We can take advantage of these low power machines in terms of not only power consumption but also space savings. To quantify the performance and fairly compare the result regarding the power consumption, we first measure the power consumed at each server. For accurate measurement, we directly measured the power consumption by inserting measuring toolkit between the power supply unit and the power socket. The last column of Table 1 shows the power consumption when each server is at idle/peak load. Svr1 consumes more than 200 W even if it is just sitting around. Since Svr1 is equipped with two quad-core processor, the power consumption of CPU accounts for more than two thirds of the overall power consumption. It is even bigger than the power consumed by Svr2 at peak load. Low power nodes spend negligible amount of powers during idle time as expected. It is intriguing that the power consumption of Low2 is similar to the power consumed when the Svr2 has just plugged in without powering on. This numbers show that low power machines spend extremely low power at any time.

## 3.2 MapReduce Performance

To measure the performance of Hadoop at various computing environments, we ran Apache Hadoop [4] jobs on each server class. We selected two workloads, *sort* and *gridmix*, which are both present in the source code tree of Hadoop. TeraSort (sort) is one of the most famous benchmarks which is used to measure the performance of distributed systems. We generated 100,000,000 random key/value pairs (which amounts to approximately 10 GB). The sort consists of 150 map tasks and 1 reduce task. gridmix is an open set of applications for benchmarking and other cluster related evaluation. The current application set of gridmix has been critical in the recent performance enhancements in Hadoop. There are six applications in gridmix, most of them are I/O extensive jobs. For gridmix benchmark, we synthetically generated 2 GB of input files. Also, we used all the included workloads of gridmix, such as streamSort, javaSort, webdataScan, combiner, monsterQuery and webdataSort, for all kinds of size classes - Small (3 map tasks and 1-15 reduce tasks), Medium (30 map tasks and 7-170 reduce tasks) and Large (100 map tasks and 70-370 reduce tasks). For both of the benchmark, to study the performance of each server alone, we executed Hadoop jobs on a single server and measured the running time. For both of the benchmarks, the maximum number of concurrent map/reduce tasks on each server are set equal to the number of cores it has. For instance, Svr1 can concurrently run 8 map/reduce tasks (totally 16 tasks) at maximum. In case of hyper-threading-enabled processors, such as Low1 and Low2, we regard each thread as a single core. Therefore, Low1 and Low2 can concurrently 4/4 and 2/2 map/reduce tasks at maximum, respectively. In this way, servers can use most of their resources efficiently.

Table 2 summarizes the normalized running time and performance/Watt for each server class. One big difference between sort and gridmix is that gridmix has multiple concurrent jobs which are to be executed at the same time. For sort benchmark, we submitted only one job for each setting. Therefore, gridmix represents the situation where a lot of job requests are issued in a short time. As you can see, for all the cases, Svr1 performs the best of all, though the difference between Svr1 and Svr2 is very small. As expected, Low1 and Low2 exhibit two or three times increased running time. However, if we consider power consumption, we can see interesting results. To compute the Perf/Watt, we used the power consumption measured in the previous section (we use the two thirds of the range between power of idle and peak to consider the utilization), and computed Perf/Watt, the running time divided by the power consumed. Although the low power machines increased the running time significantly, they are very power-efficient. For instance,

Table 2: Comparison of standalone performance. The values of this table are normalized to the Svr1's performance.

| | sort | | gridmix | |
|---|---|---|---|---|
| | running time | Perf/Watt | running time | Perf/Watt |
| Svr1 | 1 | 1 | 1 | 1 |
| Svr2 | 1.1 | 3.3 | 1.1 | 3.2 |
| Low1 | 2.5 | 25.5 | 1.4 | 14.1 |
| Low2 | 3.7 | 113.3 | 2.1 | 65.9 |

Table 3: Performance contribution of each server class. The values of this table are normalized to the Svr1's performance.

| | gridmix with small | |
|---|---|---|
| | running time | Perf/Watt |
| Svr1 | 1 | 1 |
| Svr2 | 1.1 | 3.4 |
| Low1 | 1.3 | 18.2 |
| Low2 | 1.3 | 40.6 |

Low2's Perf/Watt is 113 times better than that of high end Svr1 for sorting application. Since the sorting generates only one reduce task, this improvement may come from the under utilization of Svr1. So we checked the Perf/Watt in running gridmix application. Although the number is decreased from 113 to 65, Low2 still shows significant performance per unit power consumption.

We also measured the impact of each server class on Hadoop by evaluating gridmix benchmark with three fixed servers. The specifications of the three fixed servers are the same as Svr2. For all the experiments, we constructed four nodes, where three of them were fixed as noted. Fixing these three machines, we conducted experiments with varying server class. We used the same input data as we used in the previous experiment. The size of input is 2 GB and we selected several workloads. We used part of the gridmix applications, *monsterQueries* and *webdataSort*. All the workloads were set to use Small, which amounts to 3-10 map tasks and 5-70 reduce tasks. The number of total jobs was 17. Since the jobs are executed in parallel, the system is highly utilized with concurrent map/reduce tasks. We have shown the results in Table 3. The table says that the difference of running time is not significant and low power computers use power more effectively than the high end servers. When the cluster had Low1 or Low2 machine, its performance is degraded compared to that of homogeneous Svr1's by 30%. We consider that the power save mode is intended for the light usage pattern, so the increased running time can be compensated by reduced cost through low QoS levels. With this result, we can also indirectly show that replacing high end servers with low power computers does not incur significant performance degradation.

With this quantitative analysis, we summarize the results as follows. First, using low power computers is still effective for data processing frameworks. Second, replacing part of the high end servers with low power does not incur severe performance degradation.

## 4 Low Power Servers for Data Processing Frameworks

As we have noted in Section 2, suspending partial servers leads to inevitable data loss and performance degradation if we do not carefully handle data blocks. We categorize two kinds of existing solutions for this problem and use them for the data processing frameworks.

**prePSM** prePSM uses a pre-configured node set to proactively prepare for the system-wide power save mode.

**postPSM** postPSM is a reactive solution where the system adjusts the configuration after the system is intended for the power save mode.

prePSM is well suited for the short-term and dynamic services because in prePSM, the pre-configured node set enables the system to be responsive to the server configuration change. Even if the system enters the power save mode, prePSM produces a small amount of data transfer since it pro-actively replicates data blocks. However, prePSM might suffer from the energy waste induced by the augmenting servers when the target system is a long-running service. On the other hand, postPSM is optimized for the long-running services due to the fact that it uses the extra servers only when necessary. However, postPSM may spend significant network bandwidth and time to enter the power save mode. Therefore, we provide a hybrid solution to this problem by mixing these two solutions.

### 4.1 Augmentation and Substitution (AnSwer)

We use a rack-based solution as Hadoop is a completely rack-based framework. The current implementation of Hadoop is highly specialized for the rack-based clusters. Both of the scheduling and the replica placement policy utilizes the rack-based cluster environment to protect the data even if a whole rack is failed. Along with that, we pre-configure low power nodes within a local rack. The advantage of using rack-based solution is as follows. First property is reliability. As Hadoop places replicas in at least two different racks, our rack-based solution can provide the reliability in the same manner. A rack of low power servers can be regarded as a separate rack for reliability. Therefore, replicas of suspended servers can be safely placed at the remote rack of low power servers. Second property is the administrative effort. The nodes added during the power save mode are interim nodes and therefore, they should be easily maintained by administrators. Based on the quantitative analysis of low power computers for data processing frameworks, AnSwer uses two auxiliary techniques, augmentation and substitution with low power computers. Augmentation minimizes the data transfer caused when auxiliary nodes replace the existing servers. Substitution reduces the impact on data processing performance by replacing high end servers with low power nodes.

### 4.1.1 Augmentation

The need for augmentation is to minimize the data transfer caused by the data blocks residing at the nodes to be suspended. We augment the existing servers with pre-configured low power nodes to store file blocks at the time of block allocation. We use the invariant on Hadoop's data placement policy. We augment the system with rack-based low power computers. Since Hadoop version 0.21, they provide user-specified block allocation policy, which can easily be implemented with a single Java class.

### 4.1.2 Substitution

Previous studies do not consider the compensation for the performance degradation caused by the suspended servers. Therefore, Augmentation alone is not sufficient to effectively and efficiently exploit the low power computers. We mitigate this by substituting high end servers with low power machines with a specified ratio. As we have noted in Section 3, the replacement does not incur severe performance degradation. Therefore, we replace the existing high end servers with low power computers by a given ratio,

$$\alpha = \frac{\text{the number of replaced high end servers}}{\text{the number replacing low power computers}}.$$

For instance, if $\alpha = 1$, then the high end servers are replaced with the same number of low power computers. If $\alpha = 2$, then two high end servers will be replaced with one low power computers.

## 4.2 Energy Proportionality

Table 4 shows the notations used in the following sections. We first consider the power consumption. The net power reduction is as follows,

Table 4: Model

| | |
|---|---|
| $PS(x)$ | Power saved with $x$ high-end servers powered down |
| $PI(x)$ | Power increased with $x$ low power computers powered up |
| $NPS(x)$ | Net power saved. Equals to $PS(x) - PI(x)$ |
| $Q(s)$ | # of tasks per unit time for server class $s$ |
| $P_{he}, P_{lp}$ | power consumption of a high end and low power computers, respectively |
| $\alpha$ | substitution ratio |

$$PS(x) = x \times P_{he}$$

$$PI(x) = \frac{x}{\alpha} \times P_{lp}$$

$$NPS(x) = PS(x) - PI(x) = \frac{x}{\alpha} \times (\alpha P_{he} - P_{lp})$$

For a given $\alpha$, $NPS(x)$, the net power saved, is proportional to the number of replaced servers.

Second, we explore the AnSwer's impact on performance. The performance diminished and increased by suspending the high end servers and adding low power servers is calculated as follows.

$$P_{dim}(x) = x \times Q(\text{high})$$

$$P_{inc}(x) = \frac{x}{\alpha} \times Q(\text{low}),$$

where $Q(\text{high})$ and $Q(\text{low})$ denote the number of tasks per unit time for high end and low power computer respectively. Also, $P_{dim}(x)$ is usually greater than $P_{inc}(x)$. The performance loss when replacing existing servers with low power servers is as follows.

$$P_{loss} = P_{dim}(x) - P_{inc}(x) = xQ(\text{high}) - \frac{\text{x}}{\alpha}\text{Q(low)}$$

Suppose that we are using energy-proportional computing. Then,

$Q(\text{low}) = \beta P_{lp}, Q(\text{high}) = \beta P_{he}$.

If we rewrite the performance loss, we can get,

$$P_{loss} = x\beta P_{he} - \frac{x}{\alpha}\beta P_{lp} = \frac{\beta x}{\alpha}(\alpha P_{he} - P_{lp}) = \beta NPS(x)$$

We can conclude that the performance loss is proportional to the net power saved, which means the system has the property of energy proportionality.

## 5 Discussion

In this section, we discuss the practical problems when we actually suspend partial servers. As noted, these problems are not considered in the existing papers. We think that there are two specific problems, i) which nodes to choose and ii) where to migrate lost replicas

## 5.1 Node Selection

In order for the power save mode of data processing frameworks to work properly, one of the most important tasks is to reasonably decide which nodes to suspend. The following factors are primary obstacles that tackle the power save mode in data processing frameworks. First, there is neither a general metric to distinguish the energy efficiency of various servers nor accurate power measuring units. Defining the amount of work that is done by a server considering the power consumption is a tough job because there are a large number of parameters to consider. Server capability such as CPU and memory could be one of such parameters. However, even if the system consists of homogeneous servers, the server's energy efficiency still depends on a variety of parameters, such as what workload as used, what kind of tasks (map/reduce) was used, etc. Second, the amount of data to be migrated heavily depends on the node selection method. A possible solution is to consider the performance and the energy efficiency simultaneously. First, we can profile the power usage of each server with a suite of workloads, I/O intensive and CPU intensive may be good candidates. We can draw a power usage curve of each server class and use this power distribution for power measurement. On the other hand, for performance, we can select considerably slow nodes as candidates. This method has two implications. It is well known that few slow nodes could damage the overall performance severely [9]. Suspending these servers can induce positive effects on the system's overall performance. Also, the word "slow nodes" self-explains the energy inefficiency of such nodes. Therefore, those are good candidates to suspend. In Hadoop's context, these nodes are ones that mostly incurs speculative executions.

## 5.2 Replica Reconciliation

Replica reconciliation is a crucial procedure which incurs actual data transfer and consumes much time. A naive approach is to restore all the replicas that is going to be lost during the power save mode. For example, suppose that the administrator decides to suspend 10% of the cluster and each server has 10 GB chunks on average. A naive approach will produce data transfer of 10 GB times the number of nodes to suspend totally. Therefore, if we process the replica reconciliation aggressively, the process of suspending partial servers may lead to significant bulk data transfer in a short period. By contrast, if we migrate chunks way too much carefully, the time to reach the stable power save mode may be too long. The current implementation of HDFS's fault tolerance works in such a limited way. When the namenode detects a failure of a datanode, the namenode acquires the list of chunks that were replicated by the dead datanode and initiates the replication process. In this procedure, the amount of replica is limited by the predefined parameter and consequently consumes much time to reconciliate all the replicas. A possible solution is to migrate partial replicas, not complete replica set. We can temporarily adjust the replication factor of each chunk during power save mode. The bottom line is that it is strongly recommended to balance this tradeoff.

## 6 Conclusion and Future Work

Energy efficiency in cloud computing is becoming more and more important for IT operators of data centers and several effort to use low power machines in the data center level has been explored. We consider the problems that arise when we suspend existing servers, In this paper, we evaluate Apache Hadoop on low power machines and study the feasibility. We also propose AnSwer (Augmentation and Substitution), an energy saving method to reduce energy consumption by introducing low power machines. As our future work, we will implement AnSwer in Hadoop and experimentally study AnSwer in depth to quantify the impact on performance and power savings. Furthermore, we will use other benchmark tools to study the behavior of data processing frameworks in various ways.

## References

[1] Report to Congress on Server and Data Center Energy Efficiency. Tech. rep., U.S. Environmental Protection Agency, 2007.

[2] Data Center Energy Forecast. Tech. rep., Accenture, 2008.

[3] ANDERSEN, D., FRANKLIN, J., PHANISHAYEE, A., TAN, L., AND VASUDEVAN, V. FAWN: A fast array of wimpy nodes. In *Proc. 22nd Symposium on Operating Systems Principles* (2009), ACM New York, NY, USA.

[4] APACHE SOFTWARE FOUNDATION. Hadoop. `http://hadoop.apache.org/`.

[5] BARROSO, L. The Price of Performance: An Economic Case for Chip Multiprocessing. *Queue* (2005).

[6] BARROSO, L., AND HÖLZLE, U. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan & Claypool, 2009.

[7] BARROSO, L. A., AND HÖLZLE, U. The Case for Energy-Proportional Computing. *Computer 40* (12 2007), 33–37.

[8] BERL, A., GELENBE, E., DI GIROLAMO, M., GIULIANI, G., DE MEER, H., DANG, M., AND PENTIKOUSIS, K. Energy-Efficient Cloud Computing. *The Computer Journal* (Aug. 2009).

[9] CHEN, Y., KEYS, L., AND KATZ, R. Towards Energy Efficient MapReduce. Tech. Rep. UCB/EECS-2009-109, EECS Department, University of California, Berkeley, Aug. 2009.

[10] COLARELLI, D., AND GRUNWALD, D. Massive arrays of idle disks for storage archives. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing* (2002), IEEE Computer Society Press Los Alamitos, CA, USA, pp. 1–11.

[11] Dean, J., and Ghemawat, S. MapReduce: Simplified data processing on large clusters. In *USENIX Operating Systems Design and Implementation* (2004), USENIX, pp. 1–13.

[12] Ghemawat, S., Gobioff, H., and Leung, S. The Google file system. *ACM SIGOPS Operating Systems Review 37* (2003), 43.

[13] Grossman, R., and Gu, Y. Data mining using high performance data clouds: Experimental studies using sector and sphere. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (2008), ACM, pp. 920–927.

[14] Gu, Y., and Grossman, R. Sector and Sphere: the design and implementation of a high-performance data cloud. *Philosophical Transactions of the Royal Society A:* (2009), 2429–2445.

[15] Hamilton, J. CEMS: Low Cost, Low Power Servers for Internet-Scale Services. In *Conference on Innovative Data Systems Research* (2009), pp. 1–8.

[16] Harnik, D., Naor, D., and Segall, I. Low power mode in cloud storage systems. In *Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing* (2009), IEEE Computer Society.

[17] Leverich, J., and Kozyrakis, C. On the Energy (In) efficiency of Hadoop Clusters. In *HotPower* (2009).

[18] Li, D., and Wang, J. EERAID: energy efficient redundant and inexpensive disk array. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop* (2004), ACM, p. 29.

[19] Lim, K., Ranganathan, P., Chang, J., Patel, C., Mudge, T., and Reinhardt, S. Understanding and designing new server architectures for emerging warehouse-computing environments. In *Proceedings of the 35th International Symposium on Computer Architecture* (2008), IEEE Computer Society, pp. 315–326.

[20] LinuxArmOrg. `http://www.linux-arm.org/Main/LinuxArmOrg`.

[21] Narayanan, D., Donnelly, A., and Rowstron, A. Write off-loading: Practical power management for enterprise storage. *ACM Transactions on Storage (TOS) 4* (11 2008), 1–23.

[22] Olston, C., Reed, B., Srivastava, U., Kumar, R., and Tomkins, A. Pig Latin: A not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (2008), ACM, pp. 1099–1110.

[23] Pinheiro, E., and Bianchini, R. Energy conservation techniques for disk array-based servers. In *Proceedings of the 18th annual International Conference on Supercomputing* (New York, New York, USA, 2004), ACM New York, NY, USA, pp. 68–78.

[24] Pinheiro, E., Bianchini, R., and Dubnicki, C. Exploiting Redundancy to Conserve Energy in Storage Systems. In *Proceedings of the joint international conference on Measurement and modeling of computer systems - SIGMETRICS '06/Performance '06* (New York, New York, USA, 2006), ACM Press, p. 15.

[25] Ranganathan, P. A Recipe for Efficiency? Some Principles of Power-aware Computing. Tech. Rep. HPL-2009-294, HP Laboratory, Sept. 2009.

[26] Reddi, V., Lee, B., Chilimbi, T., and Vaid, K. Web Search Using Small Cores: Quantifying the Price of Efficiency. Tech. Rep. MSR-TR-2009-105, Microsoft, Aug. 2009.

[27] Storer, M., Greenan, K., Miller, E., and Voruganti, K. Pergamum: Replacing tape with energy efficient, reliable, disk-based archival storage. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies* (2008), USENIX Association Berkeley, CA, USA, pp. 1–16.

[28] Wang, J., Zhu, H., and Li, D. eRAID: Conserving Energy in Conventional Disk-Based RAID System. *IEEE Transactions on Computers 57* (2008), 359–374.

[29] Weddle, C., Oldham, M., Qian, J., Wang, A., Reiher, P., and Kuenning, G. PARAID: A gear-shifting power-aware RAID. *ACM Transactions on Storage (TOS) 3* (2007).

[30] Yao, X., and Wang, J. RIMAC: A Novel Redundancy-based Hierarchical Cache Architecture for Energy Efficient, High Performance Storage Systems. *ACM SIGOPS Operating Systems Review 40* (2006), 249–262.

[31] Zhu, Q., Chen, Z., Tan, L., Zhou, Y., Keeton, K., and Wilkes, J. Hibernator: helping disk arrays sleep through the winter. *ACM SIGOPS Operating Systems Review 39* (2005), 190.

## Notes

[1] In this paper, we do not distinguish powering-down and suspending servers. We can save energy not only by switching off servers, but also by using them for services which demand more powers.