JouleMR: Towards Cost-Effective and Green-Aware Data Processing Frameworks

Zhaojie Niu, Bingsheng He, Fangming Liu

Abstract—Interests have been growing in energy management of the cluster effectively in order to reduce the energy consumption as well as the electricity cost. Renewable energy and dynamic pricing schemes in smart grids are two major emerging trends in energy markets. However, current data processing frameworks are not aware of the efficiency of each joule consumed by the data center workloads in the context of these two major trends. In fact, not all joules are equal in the sense that the amount of work that can be done by a joule can vary significantly in data centers. Ignoring this fact leads to significant energy waste (by 25% of the total energy consumption in Hadoop YARN on a Facebook production trace according to our study). In this paper, we propose *JouleMR*, a cost-effective and green-aware data processing framework. Specifically, we investigate how to exploit such *joule efficiency* to maximize the benefits of renewable energy as well as dynamic pricing schemes for MapReduce framework. We develop job/task scheduling algorithms with a particular focus on the factors on joule efficiency in the data center, including the energy efficiency of MapReduce workloads, renewable energy supply, dynamic pricing and the battery usage. We further develop a simple yet effective performance-energy consumption model to guide our scheduling decisions. We have implemented JouleMR on top of Hadoop YARN. The experiments demonstrate the accuracy of our models, and the effectiveness of our cost-effective and green-aware optimizations outperform the state-of-the-art implementations over Hadoop YARN.

Index Terms—Data Processing, Cost-Effective Optimization, Green-aware Optimization, Hadoop YARN

1 Introduction

As emerging computing infrastructures, data centers consume up to 3% of all global electricity production while producing 200 million metric tons of CO2 in 2014 [2]. Effective energy management becomes more and more important for the data centers to reduce the energy consumption as well as electricity cost. Renewable energy and dynamic pricing are two major emerging trends in the energy management of data centers. In order to leverage the renewable energy, many data centers have been built, being powered at least partially by renewable energy (or green energy, such as solar and wind). They are re-designed and integrated with the intelligence of smartly drawing power from multiple sources, including *green* energy from renewable and non-polluting sources and *brown* energy from traditional polluting sources [34]. Under this context, green-aware systems that integrate the awareness of renewable energy has become a hot research topic (e.g., [26], [18], [16], [49], [14], [9], [8]). In modern smart grids, the electricity price can be varying over time. Moreover, the variations of the price are substantial, as much as a factor of 10 between peak and non-peak periods [44]. Taking a special consideration on the dynamic pricing, cost-aware systems are proposed to exploit this variations for significant economic benefits [44], [32].

Data processing frameworks (such as MapReduce [13]) are becoming more and more important workloads in data centers, which contribute to significant portions of energy consumption as well as electricity cost. Particularly, we consider each job can be delayed by a bounded amount of time and we have the opportunity to leverage green energy and the dynamic pricing by designing green-aware and cost-effective scheduling algorithms. Overall, this paper investigates whether and how we can reduce the brown energy consumption and electricity cost for data processing frameworks in the data center with both brown and renewable energy supply.

Considering leveraging renewable energy and dynamic pricing schemes in data processing frameworks, we find that the key challenge is that they are both time-varying. Take renewable energy as an example. The renewable energy sources are intermittent due to daily/seasonal effects and the prices of the electricity markets are varying on an hourly basis. Thus, the renewable energy supply or the low-price electricity may not match the workload demands, which results in severe under-utilization of renewable energy in a non-green-aware system or significant financial burden in a non-cost-effective system. Green-aware systems (e.g., [14], [9], [18]) and cost-effective systems (e.g., [44], [32]) have been developed to address the mismatch in the context of data centers. The core ideas behind those studies are similar: they delay workloads according to jobs' deadline to match the renewable supply or the low-price electricity.

While those green-aware algorithms and systems can exploit the green energy and dynamic pricing *at a coarse-grained level*, they fail to capture the efficiency of each joule, which leads to severe waste of renewable energy and brown energy. We define the concept of *joule efficiency* to be the amount of work that can be done by a joule. Not all joules are equal in the sense that the joule efficiency of the energy can vary significantly, depending on when the energy comes and how the energy is used in computing. Ignoring joule efficiency can lead to significant energy waste (by 25% of the total energy consumption of Hadoop YARN on a

- *Zhaojie Niu is with LILY, Interdisciplinary Graduate School, Nanyang Technological University, Singapore.*
- *Bingsheng He is with School of Computing, National University of Singapore, Singapore.*
- *Fangming Liu is with School of Computer Science & Technology, Huazhong University of Science and Technology, China.*

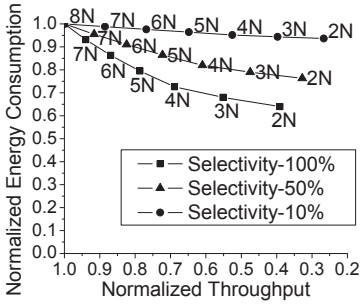


Fig. 1: Energy consumption and performance tradeoff of Hadoop YARN in our local cluster. "xN" means the result is obtained by running on x nodes. Selectivity is defined to be the ratio between the size of the output data of the map phase and the entire input data size of the benchmark (500GB in the compressed format).

Facebook production trace according to our study in Section 5).

We have observed a number of key factors resulting in joule efficiency of data processing frameworks in data centers. First, data processing frameworks, particularly MapReduce, have non-linear performance-energy consumption speedups when the number of machines scales. Figure 1 shows the normalized performance and energy consumption for running three MapReduce jobs. All the three jobs are WebdataScan operations in the GridMix benchmark, with different selectivities in our local cluster. More details of the experimental setup can be found in Section 5. We can see that running on fewer nodes is actually more energy efficient (the total energy consumed for the job is lower) than running on more nodes. Therefore, one joule of energy can result in different amounts of work done, depending on the workload characteristics and the number of machines involved in the job. The largest difference is around 35%, which shows the significant energy waste if the non-linear performance-energy consumption phenomenon is ignored. Second, battery has become an integral component for many data centers [45], [1], [3]. Charging/discharging causes inherent energy loss. Ideally, a green-aware and cost-effective system should be aware of not only the green energy supply and the dynamic pricing, but also joule efficiency to maximize the effectiveness of utilizing green energy as well as dynamic pricing.

In this paper, we propose *JouleMR*, a cost-effective and green-aware MapReduce framework. With the slack time on each job, we develop job/task scheduling algorithms with special considerations on the factors on joule efficiency in the data center, including the energy efficiency of MapReduce workloads, renewable energy supply, dynamic pricing and the battery usage. Since finding the optimal scheduling solution is a NP-hard problem, JouleMR embraces a series of simple and effective heuristics for optimizations. We further develop a performance-energy consumption model. The model guides us to make the scheduling decision so that the deadline can be met and joule efficiency can be accurately estimated for scheduling the job/task at appropriate time.

We have implemented JouleMR on top of the latest Hadoop (Hadoop YARN 2.6.0). We evaluate JouleMR in two complementary approaches. First, we run real experiments with JouleMR on a local cluster with micro benchmarks. Second, to allow more comprehensive studies, we have developed a simulator that replays the traces from the Facebook Hadoop cluster. Our experimental results demonstrate that 1) Our performance-energy consumption model can accurately predict the performance and energy consumption of our local cluster, 2) JouleMR significantly reduces the brown energy on both real experiments and simulations compared with

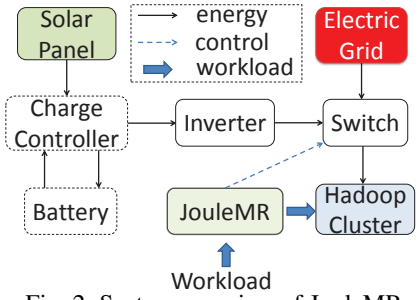


Fig. 2: System overview of JouleMR

GreenHadoop (the state-of-the-art green-aware Hadoop) [18] (up to 35% and 28% reduction, respectively). 3) JouleMR reduces the electricity cost on both real experiments and simulations compared to GreenHadoop (support dynamic pricing) (by 30% and 36% reduction, respectively).

The remainder of this paper is organized as follows. Section 2 formally defines the problem and describe the overall design of JouleMR. Section 3 presents our detailed design of green-aware scheduling, followed by the cost-effective scheduling in Section 4. Next, we show our experiment results in Section 5. We review the related work in Section 6 and conclude this paper in Section 7.

2 Problem Definition and Overall Design

2.1 Problem Definition

We consider reducing the brown energy usage of data processing frameworks (particularly MapReduce) in a single data center. The data center is powered by both brown energy and green energy sources, with battery supports.

In the MapReduce framework, a set of machines are involved for data processing: a master node and many slave nodes. The master node is the central coordinator responsible for workload scheduling and computational resource management. A slave node is responsible for executing map or reduce tasks and data storage. A MapReduce job consists of many map and reduce tasks which are executed on slave nodes in parallel.

User constraints such as deadlines are important requirements for the performance of data processing environments such as Hadoop. Thus, JouleMR allows users to specify the slack of a MapReduce job: the slack of each job is defined according to its specified deadline. JouleMR can leverage the slacks of these jobs to perform green-aware and cost-effective optimizations. We note that the deadline is "soft", mainly representing the quality of service.

The system overview of JouleMR is illustrated in Figure 2. The system comprises of a Hadoop cluster, a charge controller, an inverter, batteries and a switch. As the input voltage from the renewable energy source varies dynamically, the charge controller is used to prevent any overcharging. The charge controller also monitors the charging/discharging operations. With the charge controller, surplus green energy is automatically charged into battery.

The data center has a switch connected with both green sources and brown sources (e.g., public grids). Research has been devoted to improve the effectiveness of this kind of switch [35]. When the power demand is higher than green power supply and/or the battery supply, it immediately draws power from brown sources.

Data centers may have centralized batteries to provide an uninterrupted power supply (UPS) for power failures. More recently, data centers also incorporate batteries at the rack- or even

server-level, with higher energy efficiency than centralized battery. Previous studies [43] have demonstrated that batteries can be used to reduce the energy consumption cost in the scale of data centers. In order to leverage the batteries as an energy source to reduce the energy consumption as well as cost without significantly hurting their lifetime, similar to GreenSwitch [16], we limit the depth of discharge to a percentage of the maximum capacity at all times. The cost of the batteries is amortized during their lifetime. In our study, we estimate this component as the amortized cost multiplied by the scheduling time. Without loss of generality, we model the battery with two parameters $< C_p, e >$, where C_p is the effective capacity of the battery that can be used for discharging and e is the charging efficiency of the battery ($0 \leq e \leq 1$). For a given charging power Δx, we estimate the energy storage increment as $\Delta x' = \Delta x \cdot e$. Here, we assume the charging efficiency as constant for simplicity. JouleMR can be extended to handle other battery models with variable efficiency. Due to the inherent charging/discharging loss, the battery is either charged or discharged at each time.

Optimization Goals. We have a green-aware optimization goal and a cost-effective optimization goal which are aimed at the two major trends in energy management. Given the MapReduce workload with predefined slacks, JouleMR dynamically schedules the workloads according to the green energy supply and joule efficiency. The green-aware optimization goal is to minimize the total amount of brown energy usage, given the constraint that all workloads are completed before their predefined deadlines. Besides, we extend our system to leverage the dynamic pricing. The cost-effective optimization goal is to minimize the electricity cost, given the jobs' deadline constraint.

2.2 Overall Design

Supply driven execution (SDE) is widely used in green-aware and cost-effective systems in the data center with multiple energy sources [26], [18], [49], [14], [9], [8]. By delaying the workload, the mismatch between the green supply (or low-price brown energy) and the workload demand can be resolved. However, SDE treats each joule of energy the same without considering the phenomenon of joule efficiency caused by workload scheduling. Thus, SDE can severely waste energy due to its obliviousness to joule efficiency.

Let us study two simple examples shown in Figure 3. If we execute job A on one machine, its execution time is 4 minutes and energy consumption is 2000 J. On the other hand, if we execute it on two machines, its execution time is 3 minutes and energy consumption is 3000 J. Due to the non-linear speedup feature of job A, increasing the computing resources by a factor of two allows us to delay the execution, but does not reduce the execution time by half. Thus, more energy is consumed if we increase the number of machines. SDE cannot fit all for different green distributions. In the above case of Figure 3, SDE tends to consume less brown energy. On the contrary, in the case below, executing the job earlier can actually consume less brown energy. This example shows SDE fails to capture joule efficiency on non-linear performance and energy consumption of data processing frameworks. We can easily give some examples from other joule efficiency (e.g., battery) to show that SDE does not always reduce the brown energy consumption.

There are a number of technical issues for JouleMR. First, SDE may defer too many workloads. Due to the limited processing capacity of a Hadoop cluster, it may cause excessive deadline violations. Second, we should have a systematic way of scheduling MapReduce jobs by considering joule efficiency: when to schedule the job and how many resources should be allocated to the job in order to utilize the brown as well as green energy efficiently. Third, integrating battery into system can optimize the intermittent green energy usage and the pricing scheme, with the overhead in charging/discharging.

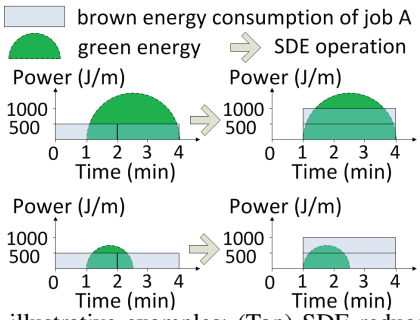


Fig. 3: Two illustrative examples: (Top) SDE reduces the brown energy consumption; (bottom) SDE increases the brown energy consumption.

We propose a MapReduce framework aware of joule efficiency (JouleMR), and develop effective cost models and efficient optimization algorithms to address the above-mentioned technical issues. The overall design of JouleMR is described in Algorithm 1. When a new job comes to the system, it is initially put into a job waiting queue. For the jobs in the job waiting queue, we need to conduct the performance and energy consumption estimations on the job (Line 2). With these estimations, we determine the most energy-efficient execution configuration that satisfies the deadline (Line 3). The configuration includes the number of servers and the amount of physical resources per server. Then, we adds the job to the job queue (Line 4). In our implementation, we use a *multi-queue* structure to express the energy-efficient execution plan of the workload.

Algorithm 1 Overall design of JouleMR

```
1: if A new job J comes then
2:     Conduct the performance and energy consumption estimations on J;
3:     Determine the execution configuration that has the smallest energy consumption
       and satisfies the deadline;
4:     Add J and its energy-efficient configuration to the corresponding queue;
5: if Current time is the beginning of an epoch then
6:     Conduct predictions on the workload and the green energy supply;
7:     Generate the basic energy-efficient plan, P;
8:     Optimize P with green-aware or cost-effective job transformations;
9:     Optimize P with battery assisted energy shifting;
```

As MapReduce jobs are coming to the system in an ad hoc manner and the green energy is varying over time, it is hardly feasible to know all future information at one moment and make an optimal offline scheduling plan. The scheduling plan should be generated and adjusted at runtime to adapt to the variance of the workload and green energy. In this study, we perform the adjustment periodically for simplicity of algorithm design. JouleMR periodically generates the energy-efficient and green-aware (or cost-effective) scheduling plan (Lines 5-9) (the period is called *epoch*). At the beginning of each epoch, we perform predictions on workloads and the green supply. Given the energy-efficient execution configuration of each job, we first generate the basic energy-efficient plan according to the cluster capacity. Next, we apply a series of optimizations to reduce the brown energy consumption (or electricity cost) of the plan, including green-aware or cost-effective job transformations and battery assisted

energy shifting. The generation of the basic energy-efficient plan, green-aware or cost-effective job transformations and battery assisted energy shifting are orthogonal with each other, and we will evaluate their separate impact in Section 5.

JouleMR periodically schedules jobs/tasks and performs power management on servers according the optimized execution plan (the period is called *slot*). In our design, we consider fine-grained jobs/tasks scheduling and power management and an epoch consists of multiple slots. For each time slot, JouleMR decides the set of active servers and allocates them to jobs and executes them according to the execution plan. For all jobs to be scheduled in the execution plan of current slot, JouleMR adopts the deadline first scheduling algorithm, which gives priority to the jobs that are close to deadlines.

Finally, we note that the epoch/slot design improves the robustness of JouleMR to inaccurate energy usage predictions or other dynamic factors including hardware failure, I/O contention and workload imbalance [6]. JouleMR re-evaluates and reacts to the system state every epoch/slot. For example, if the current epoch is over-estimating the energy usage, it may be forced to use more brown energy when not necessary. However, scheduled jobs can finish faster than expected, causing JouleMR to adjust by scheduling more jobs from the job queue. In the following sections, we present the details on the green-aware and cost-effective scheduling of JouleMR in Section 3 and Section 4, respectively.

3 GREEN-AWARE SCHEDULING

To minimize the brown energy consumption of the cluster through green-aware scheduling, we first generate a basic execution plan with special considerations on energy efficiency of data processing frameworks and cluster capacity. Then, we further reduce the brown energy consumption of the basic energy-efficient plan by green-aware job transformations and battery assisted green shifting mechanisms.

3.1 Basic Energy-efficient Plan

At the beginning of an epoch, we perform online predictions on the workload (including job types and arrival rates, etc.) and on the green supply for the epoch.

Workload Prediction. Generally, it is difficult to have an accurate prediction on the workload, because of the diversifying job sizes [11]. Fortunately, we observe that small jobs and large jobs in real production traces have very different arrival patterns (see the experimental setup in Section 5.1). Particularly, the arrival rate of small jobs (# maps is less than ten in our experiment) can be predicted with relatively high accuracy by using some simple time series analysis [15] (prediction error is less than 8% in average for the real production trace from facebook). In our experiment, we apply exponentially weighted moving average algorithm based on the historical statistics data to predict the arrival rate for small jobs. The total energy consumption and execution time of these small jobs can be estimated accurately with similar time series prediction algorithms. In our experiments, the prediction error of energy consumption is less than 6% in average and prediction error of execution time is less than 4% in average.

As for large jobs, they come in a more ad hoc manner, and the energy consumption and execution time of large jobs vary significantly. Instead, we estimate large jobs individually using our cost model after they come. Since large jobs tend to have longer execution time, such an on-demand prediction has given sufficient optimization room for scheduling, as we observed in our experiments.

Green Supply Prediction. Most green sources such as solar and wind depend on weather. There are a number of existing weather learning methods and models (e.g., [18], [24]). This is not the focus of this paper. We simply adopt the previous approach used in GreenHadoop [18], which has been shown to achieve very good accuracy for solar energy. Additionally, we also explicitly study the impact of prediction errors in the experiments.

Execution Plan Generation. We now generate the basic energy-efficient plan for JouleMR. We note that, different from the SDE plan generated in GreenHadoop [18], the plan generated by JouleMR has two distinct features. First, our job execution configuration has the smallest energy consumption while satisfying the deadline. Second, the plan generation is guided by our energy consumption and performance model for MapReduce jobs, rather than heuristics [18].

We develop a multi-queue data structure to represent the execution plan of an epoch. Each queue corresponds to one time slot and it contains the jobs/tasks starts at the beginning of the time slot. Each epoch consists of N time slots and the slot length is t_s seconds, where one time slot corresponds to one queue. There are N queues for the current epoch in our design, expressed with $q_0, q_1, \ldots, q_{N-1}$, where $q_i (0 \leq i < N)$ maintains all jobs with the slack time less than $(i+1) \cdot t_s$. A separate queue q' is used to store all jobs with the slacks beyond the current epoch. Suppose a job with a slack of k is submitted after t seconds of the beginning of current epoch, it will be added to $q_{\frac{t+k}{t_s}}$ if $t+k$ is within the current epoch, otherwise it will be added to waiting queue q'. As the time goes by, at the beginning of each time slot, the slacks of jobs belongs to the current queue will decreases to $[0, t_s]$, and they all will be assigned with the decided number of servers which are scheduled to run until all tasks completes. At the beginning of each epoch, jobs with the slacks smaller than the end of the epoch are distributed to corresponding q_i, and jobs have not finished in the previous epochs will be distributed to q_0. Thus, the jobs for an epoch include the workload from q', the jobs that arrive during the current epoch with slacks before the end of the epoch and the jobs from the previous epochs.

After initiating each job into the corresponding queue, we ensure each slot can schedule all jobs/tasks allocated to it subject to the cluster capacity. Suppose the numbers of servers that are required in each time slot are $n_i (0 \leq i < N)$. If n_i exceeds the cluster capacity, we need to move the jobs running in current time slot with the earliest submission time to the queue j ($j < i$) with available resources. We repeat this process until all n_i are no larger than the cluster's capacity. Otherwise, the cluster is overloaded.

3.2 Green-aware Optimizations with Job Transformations

The MapReduce jobs consist of individually tasks which can be executed concurrently. Even the job has started, for all pending tasks, we can still dynamically adjust the number of concurrent running tasks at one moment by controlling the amount of resources allocated to the job as adjustment to the execution plan by job transformations. As observed in Figure 3, we can transform a job in a certain way so that it can reduce the brown energy usage. We first define four basic job transformation operations, and next present the optimization techniques that leverage the job transformations.

JouleMR supports four basic job transformations: 1) $Delay(j,i)$: it delays the starting time of job j for i time slots; 2) $Advance(j,i)$: it advances the starting time of job j for i time slots; 3) $Promote(j,i)$: it increases the number of machines allocated to the job j by one in time slot i, so that it can complete faster but usually at the cost of more energy consumption; and 4) $Demote(j,i)$: it reduces the number of machines allocated to the job j by one in time slot i, which is a dual operation of promotion. Promotions and demotions capture the non-linear performance and energy consumption in MapReduce jobs. Our cost model can predict the performance and energy consumption changes for the job transformations.

Those four operations can be used together to form a sequence of transformation operations. Additionally, the same operations can be used multiple times. For example, we should use promotions for x times if we want to add x machines to the job.

Given the generated basic energy-efficient plan of the current epoch and the predicted green energy distribution in each time slot, we need to perform the transformations for each job using the above four operations, and our goal is to minimize the total brown energy consumption in the current epoch among all jobs. We need to decide how many machines to be assigned in each time slot for every job under the deadline and resource capacity constraints. We can formulate this problem into a temporal knapsack problem [7]. We omit the formulation due to the space limitation. Since finding the optimal solution to that problem is NP-hard, we propose a series of simple and effective heuristics for optimizations. Particularly, we propose a novel two-phase heuristics-based approach to solve this non-trivial optimization problem. Those two phases are complementary with each other. Phase 1 algorithm utilizes the delay and advance operations to shift the basic energy-efficient plan as a whole without impact on its energy efficiency. Phase 2 algorithm leverages the demote and promote operations to change the shape of the basic energy-efficient plan at the cost of the energy efficiency.

Phase 1: In this phase, we perform advance and delay operations to reduce the brown energy consumption while preserving the joule efficiency of the basic energy-efficient execution plan of the current epoch. As all jobs in the basic energy-efficient execution plan are delayed as late as possible, any delay operations will cause deadline violations. Therefore, we first perform the advance operations and then consider the delay operations if the brown energy consumption can be further optimized.

The optimization algorithm for Phase 1 is described in Algorithm 2. First, we perform advance operations on the basic energy-efficient execution plan. We iterate the execution plan in ascending order (from the first time slot to the last time slot in the epoch). For each time slot, we consider all jobs that are submitted before that time slot while start after that. If the brown energy consumption of the current epoch can be reduced by advancing the starting time of any job, we perform the advance operation on that job. Note, the brown energy consumption of an epoch is estimated to be $\sum_{i=1}^{N} Max\{J_i - G_i, 0\}$, where N is the number of time slots in one epoch, J_i is the total energy consumption of all jobs running in time slot i (estimated using the cost model), and G_i is the predicted amount of green energy in time slot i. Then, we perform delay operations on the basic energy-efficient execution plan. We iterate the execution plan in reverse order (from the last slot forwarding to the first slot in the epoch). For each time slot, we consider all jobs start before that time slot. We iterate these jobs and perform delay operations on them if the brown energy consumption can be reduced.

Algorithm 2 Green-aware optimization of Phase 1

```
1: for each time slot t of current epoch in ascending order do
2:     J = jobs submitted before time t and start after time t;
3:     for each job j in J do
4:         u = brown energy consumption of the current execution plan;
5:         s = start time of job j;
6:         o = advance the start time of job j for s − t time slots;
7:         if o can be performed on the execution plan then
8:             v = brown energy consumption after performing operation o;
9:             if v ⩽ u then
10:                Advance(j, s − t);
11: for each time slot t of current epoch in reverse order do
12:    J = jobs start before time t;
13:    for each job j in J do
14:        u = brown energy consumption of the current execution plan;
15:        s = start time of job j;
16:        o = delay the starting time of job j for t − s time slots;
17:        if o can be performed on the execution time then
18:            v = brown energy consumption after performing operation o;
19:            if v ⩽ u then
20:                Delay(j, t − s);
```

Phase 2: Complementary to Phase 1, we consider promote and demote operations on the jobs if those transformations can further reduce the brown energy consumption. Instead shifting an execution plan of a job as a whole, Phase 2 algorithm can change the shape of the execution plan at the cost of energy efficiency.

The optimization algorithm for Phase 2 is described in Algorithm 3. We iterate the execution plan optimized after Phase 1 in ascending order. For each time slot t in the current epoch, we consider workload shifting through promote and demote operations to further reduce the brown energy consumption. We add all jobs, which will be scheduled in the current time slot, into a FIFO queue. We dequeue a job from the FIFO queue each time and relocate the machine assigned to the job in the current time slot to the slot with the most surplus green energy supply if the brown energy consumption can be reduced. The tasks to be scheduled on this machine are also shifted accordingly. After the shifting, we enqueue the current job in to the FIFO queue and repeats this process until the FIFO queue becomes empty. We choose the FIFO queue structure to make all jobs benefit from the green energy evenly, which also minimizes the total reduction of joule efficiency caused by the non-linear speedup feature of Hadoop.

Algorithm 3 Green-aware optimization of Phase 2

```
1: for each time slot m in the current epoch do
2:     Q = ∅; /* FIFO queue */
3:     for each job j in time slot m do
4:         Q.enqueue(j);
5:     while Q ≠ ∅ do
6:         u = brown energy consumption of the current execution plan;
7:         j = Q.dequeue();
8:         for each time slot n in slots with surplus green energy do
9:             o = demote job j at time m and promote job j at time n;
10:            if o can be performed on the execution plan then
11:                v = brown energy consumption after performing operation o;
12:                if v < u then
13:                    Promote(j, n);
14:                    Demote(j, m);
15:                    shift corresponding tasks from slot m to n;
16:                    u = v;
17:                    Q.enqueue(j);
18:                    break;
```

3.3 Battery Assisted Green Shifting

We can take advantage of battery to store the green energy for future use when the green energy supply is surplus. The (**basic** approach) adopted by the previous studies (e.g., [14], [16]) are as follows. The battery is charged only when there is surplus

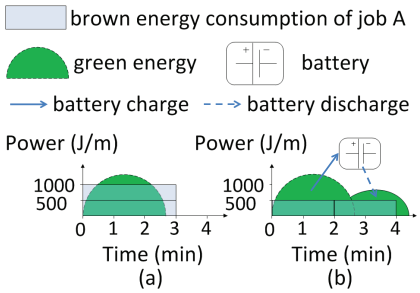


Fig. 4: An example of battery assisted green shifting

green energy (i.e., the green energy supply is higher than the total energy consumption). When the energy demand is higher than green supply, the battery is first discharged. The discharging stops when the battery capacity reaches a predefined threshold, only for emergency use (e.g., when external energy sources are unavailable), and the brown energy is used.

However, the basic approach does not consider *joule efficiency*. If we find that storing the green energy can do more work in the future (green shifting) rather than using it at present, we can proactively store it into the battery despite of the charging loss. This charging method can be feasible, since MapReduce jobs exhibit non-linear speedup and energy consumption behavior.

With the consideration of joule efficiency, the charging of battery needs careful designs. One example is illustrated in Figure 4. In Figure 4 (a), all green energy is utilized directly. Instead, Figure 4 (b), only partial green energy is used after a job transformation (demotion) and remainder is charged to the battery for future use. It results in significant brown reduction as energy is utilized more efficiently.

We develop our battery assisted green shifting as a further optimization for the execution plan optimized by job transformations. Our optimization is periodically performed at the beginning of each time slot. With the battery, we can delay workload to later slots for higher energy efficiency. As there is energy loss in charging/discharging, we need to assess the tradeoff between energy gain of battery assisted green shifting and the charging/discharging loss. Still, we can formulate this problem as a nonlinear integer programming problem with constraints (e.g., battery capacity and deadlines), which is NP-hard. Therefore, we propose a heuristic algorithm to perform the green shifting optimization on the execution plan after job transformations.

Green Shifting Algorithm: We iterate the execution plan in the ascending order. For each time slot, we consider performing demote and charging surplus green energy for future usage for each job in the current time slot. Similar to Algorithm 2, we firstly add all jobs J powered by green energy into a FIFO queue Q, and then dequeue a job from Q each time and perform demote as well as charging if brown consumption can be reduced. This process repeats until Q becomes empty. For each job j dequeued from Q, if demote will not cause deadline violation, we perform demote on job j. Demote operation on j causes less workload to be scheduled as surplus green is not fully utilized, The surplus green energy can be charged to battery and used to schedule the reduced workload for higher energy efficiency in the later time slots. However, not all of the surplus green energy can be utilized in the future because of energy loss in charging/discharging and energy waste when battery is at full capacity. Therefore, we need to compare the energy consumption of the reduced workload scheduled in the future for higher energy efficiency with the actual amount of green energy calculated by subtracting energy loss and waste from the amount of surplus green energy. If the actual amount of green energy is more than the energy consumption of reduced workload to be scheduled in the future, we commit the demote operation as well as charging surplus green for future usage, and enqueue j into Q for further optimization. Otherwise, we withdraw the demote operation.

3.4 Cost Model

Since we use the performance-energy consumption models at run time, we aim at developing a lightweight yet sufficiently accurate cost model. Since small jobs (i.e., with less than 10 map tasks) are predicted with time series algorithms, the model applies to large jobs only.

We have a number of design considerations. First, we need to integrate the hardware profile including servers and network switches to quantitatively reflect the relationship between utilization and energy consumption. We assume that the cooling energy consumption is strongly correlated to the total energy consumption of servers and network switches, and exclude the cooling energy consumption in our estimation for simplicity. Second, we need to have the total execution time and energy consumption of a MapReduce job, for the effectiveness of any scheduling decision. Third, the runtime overhead of the cost model should be low. There are a number of performance models for MapReduce (e.g., [23], [42], [22], [41]). We extend the prediction model used in Bazaar [23] according to our design principles for guiding the scheduling decisions of JouleMR. There is a tradeoff between runtime overhead and model accuracy. Our design decision leans towards simplicity with sufficient accuracy.

Overall, our performance-energy consumption model estimates the energy consumption and execution time, given the program for a MapReduce job and the execution configuration as input. Due to the space limitation, we present the details of the performance-energy consumption model in Appendix A of our supplementary file. Also we present the details of the server power management in Appendix B of our supplementary file.

4 Cost-effective Scheduling

In order to leverage dynamic pricing to reduce the electricity cost of the cluster, we provide cost-effective scheduling by extending JouleMR. We find that the algorithms proposed in green-aware scheduling (presented in Sections 3) can be extended to develop cost-effective optimizations with taking special consideration on the electricity cost. In the following, we mainly focus on the major differences to green-aware scheduling: cost-effective job transformations and battery assisted cheap brown shifting algorithm.

4.1 Basic Energy-efficient Plan

We use same approach used in green-aware scheduling to generate energy-efficient plan. Moreover, we take special consideration on the dynamic pricing which is able to guide our cost-effective scheduling. We apply the same type of variable grid energy pricing as used in GreenHadoop [18]. It assumes on-peak/off-peak pricing: \$0.13/kWh (on-peak) and \$0.08/kWh (off-peak). The peak grid power is charged at \$13.61/kW. Similar to GreenHadoop [18], we monitor the average brown power consumption every 10-minutes and define the maximum of these average as the peak brown power that has been reached so far. In our paper, we assume

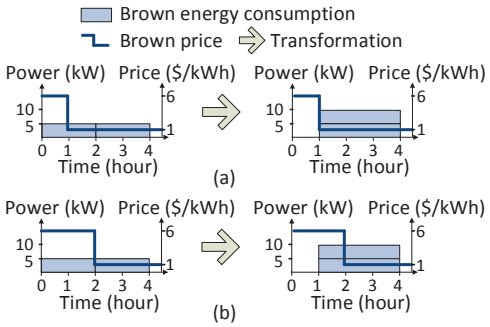


Fig. 5: Two illustrative examples: (a) Delay and promote operation reduces the electricity cost; (b) Delay and promote operation increases the electricity cost.

the self-generated green energy is free. However, our cost-effective optimization algorithm can be extended to handle the cases where renewable energy is charged [38].

By considering the joule efficiency, delaying workload and scheduling it with low-price brown energy cannot reduce the electricity cost all the time. Let us study two simple examples shown in Figure 5. If we execute the workload on one machine, its execution time is 4 hours and energy consumption is 20 kWh. On the other hand, if we execute it on two machines, due to the non-linear speedup feature of the workload, its execution time is 3 hours and its energy consumption is 30 kWh. In the case (a) of Figure 5, we delay the scheduling of the workload in order to utilize more low-price brown energy, then the electricity cost can be reduced. On the contrary, in the case (b), executing the workload earlier even when the price of brown energy is high leads to a smaller electricity cost due to the brown energy is utilized more efficiently. This example shows that joule efficiency must be considered in order to effectively reduce the electricity cost of the cluster.

4.2 Cost-effective Optimizations with Job Transformations

By considering brown energy prices and peak brown power charges, we perform further cost-effective optimizations on the basic energy-efficient execution plan. The basic idea of the cost-effective optimizations is scheduling workload with cheaper brown energy and shifting the peak brown consumption to the other time slots if the total electricity cost of the current epoch can be reduced.

Similar to the green-aware job transformations, we also utilize the 4 basic operators (Delay, Advance, Promote and Demote) to perform the job transformations on the basic energy-efficient plan. We extend the 2-phase algorithms (Algorithm 2 and Algorithm 3) to support the cost-effective optimizations by considering electricity cost of the execution plan in an epoch instead of the brown energy consumption. The electricity cost of the execution plan in an epoch is calculated as $\sum_{i=1}^{N} C_i$, where N is the number of time slots in one epoch and C_i is the electricity cost of time slot i. C_i is defined as follows:

$$C_i = \begin{cases} p_i^g \times J_i, & \text{if } p_i^g <= p_i^b \text{ and } G_i >= J_i \\ p_i^g \times G_i + p_i^b \times (J_i - G_i), & \text{elif } p_i^g <= p_i^b \text{ and } G_i < J_i \\ p_i^b \times G_i, & \text{elif } p_i^g > p_i^b \end{cases}$$

, where p_i^g is the price of green energy at time slot i, p_i^b is the price of brown energy at time slot i, J_i is the total energy consumption of all jobs running in time slot i (estimated using the cost model), and G_i is the predicted amount of green energy in time slot i. If the green energy is cheaper than the brown energy, we utilize the green energy with higher priority until it is used up. Otherwise, the brown energy is utilized.

4.3 Battery Assisted Cheap Brown Shifting

We develop our battery assisted cheap brown shifting as a further optimization for the execution plan optimized by the cost-effective job transformations. Our optimization is also performed at the beginning of each time slot. With the help of batteries, we can purchase the brown energy at off-peak period with lower price and use it at on-peak period. As there is energy loss during charging/discharging, we need to compare the electricity cost saving due to the price differences and the loss in the charging/discharging. Still, we can formulate this problem as a nonlinear integer programming problem with constraints, which is NP-hard. Therefore, we propose a heuristic algorithm to perform the cheap brown shifting on the execution plan.

Cheap Brown Shifting Algorithm: We iterate the execution plan in the ascending order. For each off-peak time slot, we consider purchasing the brown energy at lower price and charging it for future usage for on-peak hour. As not all of the purchased brown energy can be utilized in the future because of energy loss in charging/discharging. Therefore, we need to compare the electricity cost saving due to the on-peak/off-peak price differences with the loss due to the energy waste. If the benefit from the price differences is larger than waste during charging/discharging, we commit the purchasing and charging for future usage.

5 Experimental Evaluations

5.1 Experiment Setup

We perform two complementary sets of experiments to evaluate JouleMR. The first set of experiments is on a real deployment on a 10-node cluster using micro benchmarks including TeraSort in Hadoop APIs and GridMix[1] benchmarks. This small-scale experiment in the real cluster is to reveal micro-level details with full control of choices on scheduling optimizations and to evaluate the energy consumption and performance models. The second set of experiments is to perform simulations on the real-world trace from data centers (particularly from Facebook) to assess the effectiveness of JouleMR in large-scale systems.

In both sets of experiments in the local cluster and simulations, the battery and solar panels are simulated. The battery charging efficiency is set to be 0.82, according to our measurements on a Fujitsu lithium ion battery with 10.8 V voltage. The charging/discharging model for lithium ion battery can also be applied to lead-acid batteries but with different charging efficiencies. The lead-acid batteries also account for the fact that only percentage of the energy charged will be available for later use [16]. Our model is general for both kinds of batteries. By default, the battery capacities are 1 kWh and 20 kWh for the experiments in the local cluster and simulations, respectively. We use the real-world traces for solar energy from Measurement and Instrumentation Data Center (MIDC), because solar energy is widely available. By default, we use the two days (May 1-2, 2011), as illustrated in Figure 6. The solar energy supply is intermittent. We consider different solar power scales to evaluate the impact of different

1. http://hadoop.apache.org/docs/stable/gridmix.html

numbers of solar panels. By default, the peak solar power supply is provisioned to the peak power usage of the cluster. The default time slot size is one minute and the epoch size is 20 minutes.

We implement JouleMR based on top of the latest version of Hadoop (Hadoop YARN 2.6.0). We consider Hadoop YARN 2.6.0 as the baseline, denoted as "Hadoop". Hadoop uses the default scheduler, and jobs are scheduled for executions without delay. We choose the state-of-the-art green-aware Hadoop [18] ("GreenHadoop") for comparison. Unlike JouleMR, GreenHadoop has not considered the joule efficiency of MapReduce or the energy efficiency of the battery usage. We have got the original source code of GreenHadoop and adapted GreenHadoop to run on Hadoop YARN. To evaluate the separate benefits of individual optimization techniques, we manually enable/disable certain optimizations in both real deployment and simulations. Overall, we define three optimization variants: "JouleMR(BE)", "JouleMR(JT)" and "JouleMR". The detailed configurations of optimization techniques in all compared algorithms are summarized in Table 1. JouleMR has all the optimizations on joule efficiency proposed in this paper. The difference between JouleMR(BE) and JouleMR(JT) measures the effectiveness of our job transformations. The difference between JouleMR(JT) and JouleMR measures the effectiveness of our battery assisted energy shifting optimization. Also, all algorithms use battery, either in the basic approach or with our proposed battery assisted energy shifting technique. In terms of the low and high utilization of the data center, JouleMR makes sure that the energy is utilized efficiently. We adaptively adjust the number of active machines according to the execution plan and the other machines are turned to low power state (i.e., ACPI S3), and use the replication method [18] to guarantee the data availability when a machine is set to the low power state. Our approach can work regardless of cluster utilization. We study the impact of cluster utilization in Section 5.4.

The runtime overhead of running JouleMR scheduler is small. In our experiment, that overhead for a typical MapReduce job is less than 0.1 millisecond on a single Intel Xeon X5675 CPU core. This is mostly ignorable for data-intensive workloads.

TABLE 1: Configuration of optimization techniques for all schedulers

Optimization / Scheduler	SDE	Energy-efficient Plan	Job Transformation	Basic Battery Usage	Green Shifting
Hadoop				✓	
GreenHadoop	✓			✓	
JouleMR(BE)		✓		✓	
JouleMR(JT)		✓	✓	✓	
JouleMR		✓	✓		✓

5.1.1 Micro Benchmark Setup

The local cluster consists of 10 nodes, each with two Intel Xeon X5675 CPUs (12 cores in total), 24 GB memory and 500G SATA disks. The idle and peak power consumptions of one machine are 150 w and 280 w, respectively. The machines are connected with 10Gb/sec Ethernet. One node is configured as master, and the others are deployed as slaves. We use multiple meters to measure the energy consumption of the whole system, including machines and networking.

We assemble a micro benchmark according to the daily pattern in Google search [37], as illustrated in Figure 6. The solid line represents the solar energy (kWh) while the dashed line represents the assembled workload (#jobs/hour). At the peak load point, each core of all the machines have one task (either map or reduce) to execute. We consider four kinds of jobs: Terasort and GridMix (WebdataSort, WebdataScan and APISort). By default, the input size of each job is around 80GB and the deadline is one hour. We consider a mixed workload, where each job is randomly generated from four kinds of jobs.

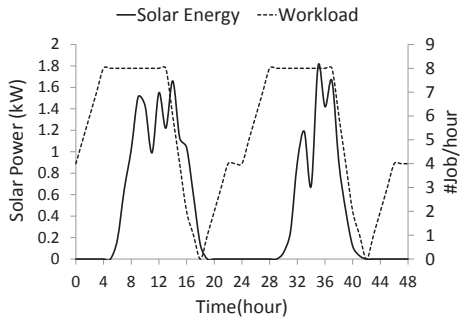


Fig. 6: A snapshot of solar trace obtained from MIDC, and the mixed workload simulating Google's search workload

We have run each experiment for five times. Variances among different runs are small, and we report averages.

5.1.2 Simulation Setup

We implement a trace-driven simulator by taking as input the solar traces and workload traces from data centers. We simulate the charging/discharging life cycles of battery and the power-performance tradeoff of the workload. In particular, we use a synthetic trace that models multi-user production workload in Facebook[2] with a 600-node cluster. We mimic the characteristics of its jobs using "loadgen". Loadgen is a configurable MapReduce job from the Gridmix benchmark included in the Hadoop distribution. The deadline of each job is computed based on its arrival time, expected execution time and slack. *By default, the slack is set to 100% of the expected execution time.* This setting is reasonable in practice, in the sense that users can tolerate a longer slack for a larger job. We also experimentally study other slack settings. The servers is configured with the same performance-energy consumption models as those in our local cluster. We apply two-level tree-structured networks in our simulation: intra-pod switch with 1Gb/sec bandwidth, and inter-pod switch with 10Gb/sec bandwidth. Each pod has 32 machines.

We analyze the arrival pattern and energy consumption statistics of jobs with different sizes. Table 2 summarizes the statistics of the Facebook trace for 7 categories of jobs in different sizes. For each category, the 2nd column lists the fraction of energy consumption and 3rd column lists the fraction of the jobs; the 4th column lists the range of the number map tasks; the 5th column lists the average number of reduce tasks. Small jobs (# Maps is smaller than 10) contributes to over 70% of the number of jobs, but their total energy consumption is less than 6%. In contrast, the total energy consumption of the jobs with more than 1000 map tasks accounts for over 79% of the total energy consumption of the entire trace. As discussed in Section 3.1, we develop a job size-aware prediction on performance and energy consumption. The arrival rate, energy consumption and execution time of small jobs is predicted with a time series algorithm, and large jobs are scheduled in an ad hoc manner.

2. https://github.com/SWIMProjectUCB/SWIM/wiki

TABLE 2: Facebook production trace characteristics

Cat.	% Energy	% Jobs	# Maps	# Aver Reds
1	6%	70.68%	[1,10]	4
2	4.7%	10.73%	(10,50]	5
3	4.6%	4.61%	(50,100]	13
4	5.03%	7.02%	(100,1000]	116
5	25.00%	5.36%	(1000,10000]	595
6	48.56%	1.53%	(10000,100000]	5683
7	6.11%	0.06%	(100000,1000000]	8831

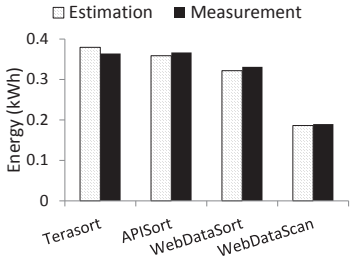


Fig. 7: Estimated and measured energy consumption of individual jobs

5.2 Green-aware Scheduling: Micro Benchmark Study

5.2.1 Model Accuracy

We evaluate the accuracy of our cost model. We present the evaluation on the total energy consumption, since we have observed similar results on the performance. We evaluate the accuracy of our energy cost model by comparing energy consumption of individual jobs in our micro benchmark. We choose four kinds of jobs: Terasort and Gridmix (WebDataScan, WebDataSort and APISort). In our experiment, the inputs are generated uniformly by using TeraGen included in the Hadoop distribution and the data generators included in Gridmix benchmark. Figure 7 shows the estimated and measured energy consumption for running each job in the local cluster. For all jobs, the difference between estimation and measurement is smaller than 5% of the measurement when the number of profiling tasks is larger than 10, which validates the accuracy of the energy consumption model. We also evaluate the average prediction for Terasort and Gridmix workload by varying the number of map task used for profiling. As we can see from the Table 3, our cost model achieves higher accuracy with the increase of map tasks to be profiled. Many dynamic factors including hardware failure, I/O contention and workload imbalance can result in inaccurate prediction, we study its impact to JouleMR in section 5.4 by manually simulating the prediction error.

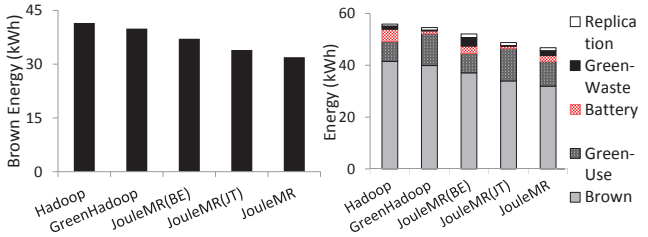
TABLE 3: Prediction error with different profiling overhead

# map task used for profiling	5	10	20	40
average prediction error	12.8%	4.9%	3.5%	2.9%

5.2.2 Overall Comparison

Figure 8(a) shows the brown energy consumption of different schedulers in running the micro benchmark. Overall, both GreenHadoop and JouleMR consume less brown energy than Hadoop, thanks to the alignment between power demand and green supply. Compared with GreenHadoop, JouleMR still consumes less brown energy, by embracing the energy-efficient optimizations. JouleMR(BE) reduces the brown consumption by 7.1% compared with GreenHadoop by considering energy efficiency. JouleMR(JT) applies job transformations to the basic energy-efficient plan and further reduces the brown consumption by 8.5% over JouleMR(BE). By integrating battery-assisted shifting, JouleMR reduces the brown consumption by 6% further over JouleMR(JT). We can see that the proposed approaches are orthogonal with each other in the optimization and result in accumulated energy reductions.

Figure 8(b) shows the energy breakdown. We divide the total energy into five categories: brown energy consumption, green energy consumed by the workload directly, green energy charged into battery, green energy wasted and data replication overhead. Green energy wasted includes energy loss during charging/discharging and the discarded part when battery is fully charged. GreenHadoop can utilize more green energy than Hadoop by aligning power demand with green supply. However, without considering joule efficiency, it uses more brown energy than JouleMR. Particularly, JouleMR(BE) utilizes every joule of energy more efficiently and consumes less brown energy than GreenHadoop. Job transformations in JouleMR(JT) use more green energy in an efficient manner. Finally, battery assisted green shifting techniques in JouleMR charges more green energy into battery rather than using it directly, because some green energy can be utilized with higher energy efficiency in the future and thus further reduces the brown energy.



(a) Brown energy consumption (b) Energy consumption breakdown

Fig. 8: Overall comparison of green-aware optimizations with the micro benchmark

We further study the replication overhead. Because we take advantage of ACPI S3 to put servers to a lower power state for energy saving (0.89 W power consumption per server in our experiment), some data blocks should be replicated to other servers for processing. We observed that the replication overhead is negligible (smaller than 20% of the energy saving by transiting the server to S3 when appropriate). Two more issues are worth further discussions. First, the state transition of a server is lightweight. The period of a server transiting from S3 back to the active state is around 7 seconds, and transiting from the active state to S3 takes only 1 second. Second, the network performance of our cluster is reasonably good. A data transfer of one HDFS chunk of 128MB takes less than 0.3 second in our local cluster.

5.3 Green-aware Scheduling: Large-scale Trace Study

5.3.1 Simulation Validation

We first evaluate the accuracy of our simulation. We use the SWIM workload replay tool to scale down Facebook production trace for our local cluster. With a similar scale-down approach in the previous study [18], we scale down the workload as follows: we reduce the data by a factor of 8, and eliminate the largest 1.9% of jobs. We schedule the scaled-down workload in our local cluster and validate the simulation result with the real-measured value. Figure 9 shows the measured and simulated energy consumption. The difference between simulation and measurement is small (less than 2 kWh), which validates the accuracy of our simulator.

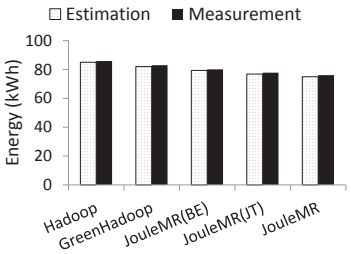
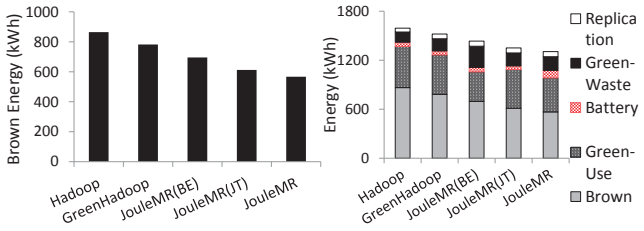


Fig. 9: Validation of the green-aware simulations on replayed Facebook trace

5.3.2 Overall Comparison

Figure 10(a) shows the brown energy consumption result on Facebook production workloads under five schedulers. The brown energy reduction of JouleMR is even larger than that in the local cluster, by 35% and 28% over Hadoop and GreenHadoop, respectively. Moreover, JouleMR has a lower total energy consumption, by 21% and 19% smaller than Hadoop and GreenHadoop, respectively. That shows the importance of integrating energy efficiency into the system.

The energy consumption breakdown is shown in Figure 10(b). In the experiments, we have observed similar results to the local cluster. We highlight with the following observations for simulations with the production workloads. First, the replication overhead is more significant than that of the local cluster, because data replications tend to be more frequent on a large-scale cluster. However, it is still kept in a reasonable low level (less than 5% of the total energy consumption). Second, the battery plays a more important role in the large-scale cluster especially for the variants of JouleMR, because green energy is utilized with higher efficiency and more surplus green energy can be charged into battery. Meanwhile, as higher green supply and battery capacity are provisioned in our simulation, Figure 10(b) shows more green energy waste for the simulation study than our micro benchmark. Third, we study the detailed energy consumption along the timeline. All variants of "JouleMR" significantly reduce the peak usage of brown energy, in comparison with GreenHadoop.



(a) Brown energy consumption (b) Energy consumption breakdown

Fig. 10: Overall comparison of green-aware optimizations with the Facebook production workloads

Figure 11 shows the detailed energy consumption for Facebook production trace for GreenHadoop, JouleMR(BE), JouleMR(JT) and JouleMR. We can see 1) All variants of "JouleMR" significantly reduce the usage of brown energy, in comparison with GreenHadoop. 2) Even though JouleMR(BE) consumes less green energy compared to GreenHadoop, the consumption of brown energy is still reduced by considering energy efficiency. 3) JouleMR(JT) applies job transformations to utilize green energy better, therefore, the brown consumption is further reduced. 4) JouleMR reduces the brown consumption the most, because battery-assisted optimization charges the current available green energy into battery if it can be utilized more efficiently.

5.4 Sensitivity Studies

We further conducted sensitivity studies on the parameters involved in the JouleMR design. Overall, JouleMR outperforms other schedulers in all different settings on those parameters. We present the results mainly with Facebook production trace.

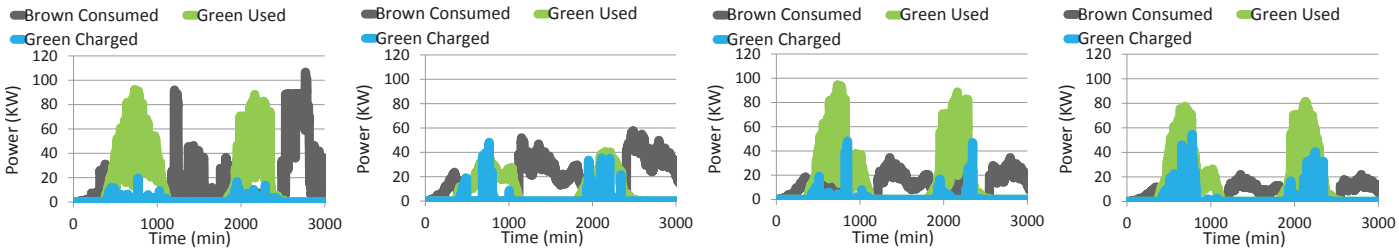
Impact of Battery Capacity. Figure 12 shows the result for varying the battery capacity. The brown energy consumption of all schedulers decreases significantly at the beginning and finally becomes stable later as the increasing of the battery capacity. The increasing of the battery capacity has more effect for the variants of JouleMR, because more green energy can be saved and charged into battery by applying optimizations considering joule efficiency. Among all optimizations, battery assisted green shifting benefits the most from the increase of battery capacity.

Impact of Slacks. Figure 13 shows the result of different slacks for Facebook production trace. We vary the slack in proportion to the default setting, in addition to setting the same deadline setting (one day) for all jobs used in GreenHadoop [18]. As expected, as the slacks increase, the opportunity of utilizing green energy becomes larger, and energy can be utilized with higher efficiency. Regardless of different deadline settings, JouleMR can reduce the brown consumption significantly compared to GreenHadoop, due to the optimizations considering joule efficiency. JouleMR(BE) performs worst and stays stable when the slacks is larger than 400%. The reason is that JouleMR(BE) only considers the energy efficiency rather than the supply of green energy. As the increase of the slacks, the execution plan generated by JouleMR(BE) varies for higher energy efficiency at the beginning, and remains unchanged when the slacks reaches a threshold.

Impact of Energy Estimation. Figure 14 shows the brown energy usage by introducing different degrees of prediction errors in the energy estimation. Specifically, given a prediction error e and the real energy consumption w, the estimation is randomly distributed in $[w, w(1+e)]$ in Figure 14(a) and is randomly distributed in $[w(1-e), w(1+e)]$ in Figure 14(b). The results demonstrate the robustness of our optimizations, if the prediction error is reasonable (less than 30% in our experiments). We observed similar results on the prediction errors of execution time.

Impact of Varying the Scale of Solar Energy. Figure 15 shows the result for varying the solar energy scale. The increasing solar energy scale can be achieved by using more effective solar panels and/or by installing solar panels of larger area. As the scale of solar energy increases, the brown energy consumption of all approaches decreases and the brown energy usages are getting closer due to more workloads can be scheduled with green energy. However, in practice, the peak power of green energy is provisioned at most to the peak power usage of the cluster. JouleMR consumes more energy than other approaches when the scale of solar energy is larger than 400%, because it does not aware the distribution of green energy and only executes following the generated plan.

Impact of varying epoch size in scheduling. Figure 16 shows the result of different epoch sizes in scheduling. The brown energy consumption decreases slightly at the beginning with the increase of the epoch size (from 5 mins to 20 mins) as the scheduling plan benefits from the more future information which can be predicted accurately. After that, with the increase of the epoch size (from



(a) GreenHadoop (b) JouleMR(BE) (c) JouleMR(JT) (d) JouleMR

Fig. 11: Detailed energy consumption of different green-aware optimizations

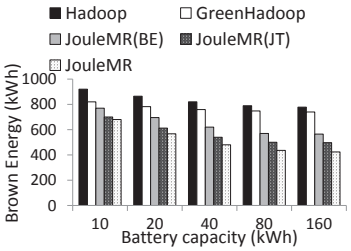


Fig. 12: The brown energy usage varying the battery capacity

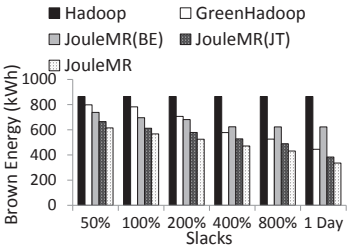


Fig. 13: The brown energy usage varying slacks

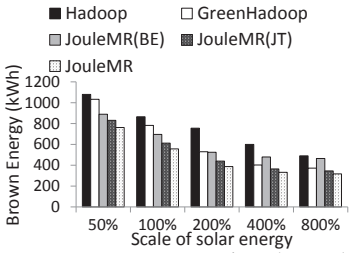


Fig. 15: The brown energy usage varying the scale of solar energy

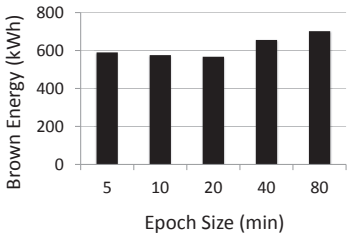
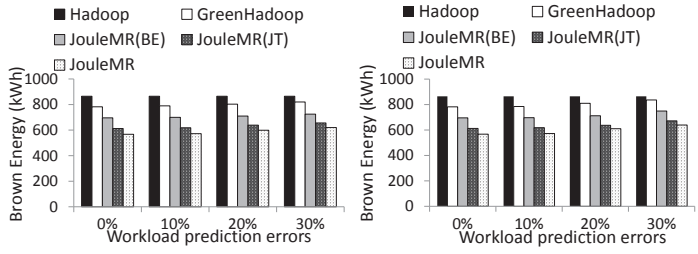


Fig. 16: The brown energy usage varying the epoch size

20 mins to 80 mins), the brown energy consumption increases significantly due to the less accurate prediction as well as less robustness. Thus, this paper chooses 20 minutes as a suitable epoch size.

Effects of JouleMR under highly variable solar energy. We use the real-world traces for solar energy in one period from MIDC. This trace contains the highly variant case due to the impact of the weather and other factors and the trace is shown in Figure 17(a). We synthesize a workload with the trace in the trace-driven simulator according to the length of this choosing period and schedule them with different schedulers. Figure 17(b) shows that JouelMR still performs efficiently even in the highly variably trace and the battery becomes more important when the solar energy becomes more variable.

Impact of cluster utilization. Figure 18 shows the impact of different data center utilizations by varying the cluster size. In term of different data center utilizations, JouleMR performs effectively compared to Hadoop and GreenHadoop. The improvement increases with the increase of the cluster size (i.e., with the cluster resource utilization decreases).



(a) Positive Error (b) Random Error

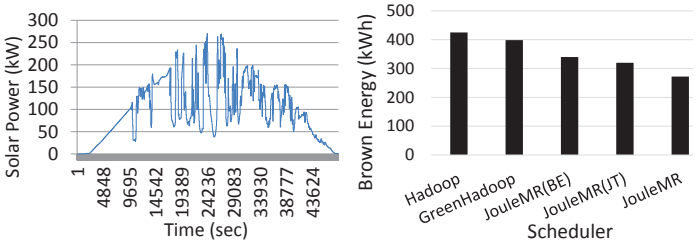
Fig. 14: The brown energy usage varying workload prediction errors

5.5 Results on Cost-effective Scheduling

We evaluate the cost-effective scheduling of JouleMR in real deployment and the large-scale simulation. Overall, JouleMR reduce the electricity cost compared to the state-of-the-art cost-aware systems through energy-efficient optimizations. Due to the space limitation, we present the details of the results in Appendix C of our supplementary file.

6 Related Work

Energy-efficient MapReduce. Recently, there have been many research studies on improving the energy efficiency of MapReduce/Hadoop. Roughly, we can categorize the related work in the following three categories.



(a) Highly variable trace (b) Brown energy consumption

Fig. 17: Brown energy usage when solar trace is highly variable

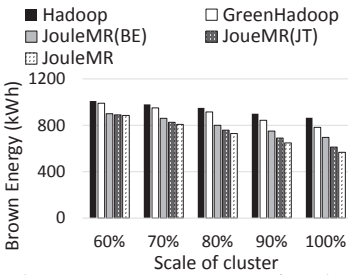


Fig. 18: The brown energy usage varying the cluster size

The first category is to turn down some machines in the cluster for energy saving. Different approaches have been developed to identify the machines to turn down [33], [26]. As MapReduce co-locates computation and storage in the same server, a main issue is how to place data replicas in HDFS for data availability. Many energy-aware replication strategies are proposed [47], [27], [25]. These data placement policies are complementary to JouleMR.

The second category is to reduce the energy waste according to the non-power proportionality of data centers. Lang and Patel [30] proposed an approach called All-In Strategy (AIS), and used all the nodes in the cluster to run a batch of workloads and then powered down the entire cluster. BEEMR [10] splits the cluster into two disjoint zones, namely interactive and batch zones. The interactive zone consists of a small, fixed percentage of cluster resources and is always fully powered on. The batch zone runs workloads in a batch fashion, and is put into the low power state after the batch is finished.

The third category is to leverage more energy-efficient hardware. Wimpy nodes (machines with low capability and low power consumption) have attracted considerate interests. The tradeoff on performance and energy consumption by wimpy nodes and normal servers have been explored for MapReduce (e.g., [30], [31]). JouleMR can deal with wimpy nodes, with proper performance and energy models.

Green-aware Algorithms and Systems. Due to the dynamic feature of green energy, renewable energy aware computing has become a hot research issue in data centers. Some studies have developed models and algorithms for predictions in order to minimize the cost of a data center. Chen et al. [8] proposed to schedule workloads across multiple geographically distributed data centers in order to utilize the renewable energy effectively. Preemption policies have been implemented on YARN [5], [12], which further promotes the elasticity of green-aware algorithms. However, checkpointing causes further I/O contention and preemption increases complexity of schedulers, needs further study. Recently, a number of real green-aware systems are developed for different workloads. Researchers have proposed to schedule batch jobs to maximize the usage of renewable energy [17], [18], [4]. Similar goals and techniques have been studied for interactive applications [29], [46], [48], [34]. For data centers that run a mix of interactive and batch workloads, Aksanli et al. proposed to adapt the amount of batch processing dynamically [4]. Goiri et al. has carried out a series of studies and a green data center prototype [16], [21] to manage deferrable and non-deferrable workloads at the presence of renewable energy. Chen et al. [9] proposed ReinDB that integrates renewable energy supply into database systems on a single server. Li et al. [36] utilizes data center power demand shaping technique to utilize onsite green energy efficiently. Some attention has been paid to leverage battery to store renewable energy [19], [28], without considering the energy efficiency of workloads. Few of the previous studies have paid attention to the joule efficiency problem, particularly in the MapReduce/Hadoop cluster. We conjecture the concept of joule efficiency can also be applied to other applications. GreenMR [40], [39] and GreenPar [20] consider joule efficiency when leveraging green energy. However, they do not study impact of on-peak/off-peak pricing on the cost in MapReduce/Hadoop cluster.

Cost-effective Optimization. Many online optimization algorithms are proposed to reduce the electricity cost in a single data center in the presence of workload and pricing uncertainties. Rahul et al [50]. developed an online control algorithm to minimize the electricity cost of the data center without any prior information on the workload and the pricing. Deng et al. [14] applied a two-state Lyapunov optimization to design an online control algorithm, SmartDPSS, which optimally schedules multi-source energy supply in a cost minimizing fashion. As large organizations need to operate multiple data centers, research has focused on the cost-effective energy management across multiple data centers. Qureshi et al [44]. leveraged the temporal and geographic variation of electricity prices to cut the electric bill for large-scale service. Le et al. [32] proposed to optimize the electricity cost of the data centers through intelligently distributing the computational workload among multiple data centers. In comparison with the previous studies, this paper leverages the joule efficiency in data processing frameworks to reduce their electricity cost.

7 Conclusions

This paper proposes JouleMR by integrating green-aware and cost-effective job/task scheduling into MapReduce. It considers the key aspects of joule efficiency in a MapReduce cluster, including energy efficiency of MapReduce workloads, renewable energy supply, dynamic pricing schemes and battery usage. An analytical model is developed to guide scheduling decisions. We have implemented JouleMR on top of Hadoop. Our results demonstrate that JouleMR significantly reduces the brown energy on both real experiments and simulations compared with GreenHadoop [18] (up to 35% and 28% reduction, respectively). Besides, JouleMR reduces the electricity cost on both real experiments and simulations compared to GreenHadoop (by 30% and 36% reduction, respectively).

There are a few interesting studies for extending this work. First, the current system does not support the prediction of the workload and the amount of green energy for very long epoch size. In the future, we are interested in developing more complex and robust time-series prediction algorithms to support long-term prediction. Second, we plan to design dynamic and fine-grained resource allocation model by extending the current reservation-based allocation mechanisms in Hadoop YARN to further improve the resource utilization.

8 Acknowledgment

This project is partially funded by a collaborative grant from Microsoft Research Asia and a MoE AcRF Tier 1 grant (T1-251RES1610) in Singapore. Zhaojie's work is in part supported by the National Research Foundation, Prime Ministers Office, Singapore under its IDM Futures Funding Initiative. Fangming's work is supported in part by a grant from National Natural Science Foundation of China (NSFC) under grant No.61370232.

REFERENCES

[1] Google throws open doors to its top-secret data center. http://www.wired.com/2012/10/ff-inside-google-data-center/all/.

[2] Industry outlook: Data center energy efficiency http://www.datacenterjournal.com/industry-outlook-data-center-energy-efficiency/.

[3] Microsoft reveals its specialty servers racks. http://www.datacenterknowledge.com/archives/2011/04/25/microsoft-reveals-its-specialty-servers-racks/.

[4] B. Aksanli, J. Venkatesh, L. Zhang, and T. Rosing. Utilizing green energy prediction to schedule mixed batch and service jobs in data centers. *ACM Special Interest Group on Operating Systems*, 2012.

[5] G. Ananthanarayanan, C. Douglas, R. Ramakrishnan, S. Rao, and I. Stoica. True elasticity in multi-tenant data-intensive compute clusters. In *ACM annual Symposium on Cloud Computing*, 2012.

[6] G. Ananthanarayanan, S. Kandula, A. G. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris. Reining in the outliers in map-reduce clusters using mantri. In *USENIX Symposium on Operating Systems Design and Implementation*, 2010.

[7] M. Bartlett, A. M. Frisch, Y. Hamadi, I. Miguel, S. A. Tarim, and C. Unsworth. The temporal knapsack problem and its solution. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. 2005.

[8] C. Chen, B. He, and X. Tang. Green-aware workload scheduling in geographically distributed data centers. In *Cloudcom conference on Cloud Computing worldwide*, 2012.

[9] C. Chen, B. He, X. Tang, C. Chen, and Y. Liu. Green databases through integration of renewable energy. In *Conference on Innovative Data Systems Research*, 2013.

[10] Y. Chen, S. Alspaugh, D. Borthakur, and R. Katz. Energy efficiency for large-scale mapreduce workloads with significant interactive analysis. In *European Conference on Computer Systems*, 2012.

[11] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz. The case for evaluating mapreduce performance using workload suites. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems*, 2011.

[12] B. Cho, M. Rahman, T. Chajed, I. Gupta, C. Abad, N. Roberts, and P. Lin. Natjam: Design and evaluation of eviction policies for supporting priorities and deadlines in mapreduce clusters. In *ACM annual Symposium on Cloud Computing*, 2013.

[13] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 2008.

[14] W. Deng, F. Liu, H. Jin, and C. Wu. Smartdpss: Cost-minimizing multi-source power supply for datacenters with arbitrary demand. In *International Conference on Distributed Computing Systems*, 2013.

[15] S. S. Elnaffar. *Towards workload-aware dbmss: identifying workload type and predicting its change*. PhD thesis, 2004.

[16] Í. Goiri, W. Katsak, K. Le, T. D. Nguyen, and R. Bianchini. Parasol and greenswitch: Managing datacenters powered by renewable energy. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2013.

[17] Í. Goiri, K. Le, M. E. Haque, R. Beauchea, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini. Greenslot: scheduling energy consumption in green datacenters. In *International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011.

[18] Í. Goiri, K. Le, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini. Greenhadoop: leveraging green energy in data-processing frameworks. In *European Conference on Computer Systems*, 2012.

[19] S. Govindan, A. Sivasubramaniam, and B. Urgaonkar. Benefits and limitations of tapping into stored energy for datacenters. In *Annual international symposium on Computer architecture*, 2011.

[20] M. E. Haque, I. Goiri, R. Bianchini, and T. D. Nguyen. Greenpar: Scheduling parallel high performance applications in green datacenters. In *ACM International Conference on Supercomputing*, 2015.

[21] M. E. Haque, K. Le, Í. Goiri, R. Bianchini, and T. D. Nguyen. Providing green slas in high performance computing clouds. In *International Green Computing Conference*, 2013.

[22] H. Herodotou and S. Babu. Profiling, What-if Analysis, and Cost-based Optimization of MapReduce Programs. *International Conference on Very Large Data Bases*, 2011.

[23] V. Jalaparti, H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Bridging the tenant-provider gap in cloud services. In *ACM annual Symposium on Cloud Computing*, 2012.

[24] S. Jebaraj and S. Iniyan. A review of energy models. *Renewable and Sustainable Energy Reviews*, 2006.

[25] J. K. Jerry Chou and D. Rotem. Exploiting replication for energy-aware scheduling in disk storage systems. In *International Conference on Distributed Computing Systems*, 2004.

[26] R. T. Kaushik and M. Bhandarkar. Greenhdfs: Towards an energy-conserving storage-efficient, hybrid hadoop compute cluster. In *International conference on Power aware computing and systems*, 2010.

[27] J. Kim and D. Rotem. Energy proportionality for disk storage using replication. In *International Conference on Extending Database Technology*, 2011.

[28] V. Kontorinis, L. E. Zhang, B. Aksanli, J. Sampson, H. Homayoun, E. Pettis, D. M. Tullsen, and T. S. Rosing. Managing distributed ups energy for effective power capping in data centers. In *Annual international symposium on Computer architecture*, 2012.

[29] A. Krioukov, C. Goebel, S. Alspaugh, Y. Chen, D. E. Culler, and R. H. Katz. Integrating renewable energy using data analytics systems: Challenges and opportunities. *IEEE Technical Committee on Data Engineering*, 2011.

[30] W. Lang and J. M. Patel. Energy management for mapreduce clusters. *International Conference on Very Large Data Bases*, 2010.

[31] W. Lang, J. M. Patel, and S. Shankar. Wimpy node clusters: what about non-wimpy workloads? In *International Workshop on Data Management on New Hardware*, 2010.

[32] K. Le, R. Bianchiniy, M. Martonosiz, and T. Nguyeny. Cost-and energy-aware load distribution across data centers. In *Workshop on Power-Aware Computing and Systems*, 2009.

[33] J. Leverich and C. Kozyrakis. On the energy (in) efficiency of hadoop clusters. *ACM SIGOPS Operating Systems Review*, 2010.

[34] C. Li, A. Qouneh, and T. Li. iswitch: Coordinating and optimizing renewable energy powered server clusters. In *Annual international symposium on Computer architecture*, 2012.

[35] C. Li, W. Zhang, C.-B. Cho, and T. Li. SolarCore: Solar energy driven multi-core architecture power management. In *High Performance Computer Architecture*, 2011.

[36] C. Li, R. Zhou, and T. Li. Enabling distributed generation powered sustainable high-performance data center. In *High Performance Computer Architecture*, 2013.

[37] D. Meisner, C. M. Sadler, L. A. Barroso, W.-D. Weber, and T. F. Wenisch. Power management of online data-intensive services. In *Annual international symposium on Computer architecture*, 2011.

[38] M. J. Neely, A. S. Tehrani, and A. G. Dimakis. Efficient algorithms for renewable energy allocation to delay tolerant consumers. In *International Conference on Smart Grid Comm*, 2010.

[39] Z. Niu and B. He. A study of big data computing platforms: Fairness and energy consumption. In *2016 IEEE International Conference on Cloud Engineering Workshop*, 2016.

[40] Z. Niu, B. He, and F. Liu. Not all joules are equal: Towards energy-efficient and green-aware data processing frameworks. *The IEEE International Conference on Cloud Engineering*, 2016.

[41] Z. Niu, S. Tang, and B. He. Gemini: An adaptive performance-fairness scheduler for data-intensive cluster computing. *International Conference on Cloud Computing Technology and Science*, 2015.

[42] B. Palanisamy, A. Singh, L. Liu, and B. Langston. Cura: A cost-optimized model for mapreduce in a cloud. In *International Symposium on Parallel & Distributed Processing*, 2013.

[43] D. S. Palasamudram, R. K. Sitaraman, B. Urgaonkar, and R. Urgaonkar. Using batteries to reduce the power costs of internet-scale distributed networks. In *ACM annual Symposium on Cloud Computing*, 2012.

[44] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs. Cutting the electric bill for internet-scale systems. In *ACM SIGCOMM computer communication review*, 2009.

[45] D. Schneider and Q. Hardy. Under the hood at google and facebook. *Spectrum, IEEE, 48(6):63-67,*, 2011.

[46] N. Sharma, S. Barker, D. Irwin, and P. Shenoy. Blink: managing server clusters on intermittent power. In *ACM SIGARCH Computer Architecture News*, 2011.

[47] B. Shi and A. Srivastava. Thermal and power-aware task scheduling for hadoop based storage centric datacenters. In *International Conference on Green Computing*, 2010.

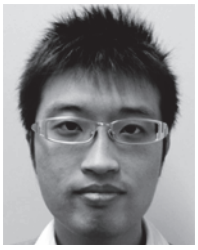
[48] R. Singh, D. Irwin, P. Shenoy, and K. K. Ramakrishnan. Yank: Enabling green data centers to pull the plug. In *USENIX conference on Networked Systems Design and Implementation*, 2013.

[49] C. Stewart and K. Shen. Some joules are more precious than others: Managing renewable energy in the datacenter. In *Workshop on Power-Aware Computing and Systems*, 2009.

[50] R. Urgaonkar, B. Urgaonkar, M. J. Neely, and A. Sivasubramaniam. Optimal power cost management using stored energy in data centers. In *SIGMETRICS*, 2011.

Zhaojie Niu is a Ph.D candidate in the Interdisciplinary Graduate School (IGS), Nanyang Technological University, Singapore. He received his master's and bachelor's degrees from Huazhong University of Science and Technology(HUST), China, in Jan 2012 and July 2009 respectively. He is interested in big data processing, resource management in the data center and distributed systems.

Bingsheng He received the bachelor degree in computer science from Shanghai Jiao Tong University (1999-2003), and the PhD degree in computer science in Hong Kong University of Science and Technology (2003-2008). He is an Associate Professor in School of Computing, National University of Singapore. His research interests are high performance computing, distributed and parallel systems, and database systems. He has served in editor board of international journals, including IEEE Transactions on Cloud Computing (IEEE TCC) and IEEE Transactions on Parallel and Distributed Systems (IEEE TPDS).



Fangming Liu received his B.Engr. degree in 2005 from Department of Computer Science and Technology, Tsinghua University, Beijing, China; and his Ph.D. degree in Computer Science and Engineering from Hong Kong University of Science and Technology in 2011. He is a professor in the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. His research interests include cloud computing and datacenter networking, mobile cloud, green computing, software-defined networking and virtualization technology