Elastic Power-aware Resource Provisioning of Heterogeneous Workloads in Self-Sustainable Datacenters

Dazhao Cheng, Jia Rao, Changjun Jiang, and Xiaobo Zhou

Abstract—While major Cloud service operators have taken various initiatives to operate their datacenters with renewable energy partially or completely, it is challenging to effectively utilize the renewable energy since its generation depends on dynamic natural conditions. In this paper, we propose and develop an elastic power-aware resource provisioning approach (ePower) for heterogeneous workloads in self-sustainable datacenters that completely rely on renewable energy. We aim to maximize the system goodput and control the system power consumption with respect to green power supply. ePower takes challenges and advantages of dynamic power supply, heterogeneous workload characteristics and QoS requirements, and automatically optimizes elastic resource allocations to workloads. The core of ePower design is a novel power-aware simulated annealing algorithm with fuzzy performance modeling for the efficient search of an optimal resource allocation. We have implemented ePower in a university cloud testbed hosting Gridmix2 and RUBiS benchmark applications. We utilize real weather data traces to simulate the green power generation and supply in the experiments. Experimental results demonstrate ePower can achieve near-to-optimal system performance while being resilient to dynamic power availability. It outperforms a representative resource provisioning approach for heterogeneous workloads by at least 24% in improving system goodput and 35% in reducing QoS violations.

Index Terms—Sustainable computing, Green datacenters, Heterogeneous workloads, Dynamic power supply, Power-aware Resource provisioning, Simulated annealing

1 Introduction

Today, major cloud service operators have taken various initiatives to operate their datacenters with renewable energy partially or completely [1]. Google, Facebook, and Apple have started to build their own green power plants to support the operation of their sustainable datacenters. Researchers envision that datacenters at cluster level can be completely powered by renewable energy, e.g., solar and wind, and be self-sustainable [2], [3], [4]. Most green power plants use wind turbines and/or solar panels for power generation. Unlike traditional energy resources, the availability of renewable energy varies widely during the times of a day, seasons of the year, and the geographical locations of the power plants. Such intermittency makes it hard for green datacenters to effectively use renewable energy.

On the other hand, the power demand of a datacenter is highly dependent on the resource requirements of hosted workloads. Datacenters are commonly shared by many users for quite different uses. An important trend is to co-locate heterogeneous workloads, transactional workloads and batch jobs, on the shared server infrastructure in datacenters for resource utilization efficiency [5], [6], [7]. Such mixed application hosting poses challenges and also opens up opportunities in the resource management in sustainable datacenters.

First, it is difficult to control system power consumption under dynamic power supply rather than a static power budget. Most previous studies assume a static power budget as the constraint for resource provisioning. However, the green power supply in a sustainable datacenter is often time-varying and highly depends on the natural weather conditions. It is challenging to effectively match the power supply and demand.

Second, it is difficult to accommodate heterogeneous workloads in a unified provisioning scheme as these workloads could have distinct resource requirements and performance goals. For transactional workloads, the goal is to maximize request throughput under a certain response time bound. It requires that resources allocated to transactional workloads be sufficient during the execution of short-lived requests or clients will observe significant decline in service quality. Batch jobs concern the job completion time in a relatively longer term. It is required that the aggregate resource allocation during the lifetime of batch jobs should guarantee job completions before individual deadlines.

Dynamic green power supply and mixed application hosting also create opportunities for joint performance-power optimizations in self-sustainable datacenters: (1) Although transactional workloads are sensitive to resource allocations, their traffic intensities vary wildly

- *D. Cheng, J.Rao and X. Zhou are with the Department of Computer Science, University of Colorado, Colorado Springs, CO, 80918, USA. E-mail: {dcheng,jrao,xzhou}@uccs.edu.*
- *C. Jiang is with the Department of Computer Science, Tongji University, 4800 Cao-An Road, Shanghai, 201804, China. Email: cjjiang@tongji.edu.cn.*
- *X. Zhou is the corresponding author.*

overtime [8], [9]. Matching resources to the actual demands could achieve significant power savings. (2) Batch jobs can be temporarily delayed and compensated later without violating the deadline. (3) Given the dynamic power supply, a careful planning of resource allocations to heterogeneous workloads can maximize the overall system performance.

In this paper, we propose and develop an elastic power-aware resource provisioning approach (ePower) for self-sustainable datacenters that completely rely on renewable energy. We aim to maximize the overall system performance in terms of datacenter goodput of heterogeneous workloads by effectively prioritizing the power budget of workloads with respect to dynamic green power supply. ePower takes advantages of time-varying traffic of transactional workloads and delay-tolerance of batch jobs to optimize the overall system performance under dynamic power constraints.

ePower employs a novel power-aware simulated annealing (SA) algorithm to search the optimal resource allocations online. The SA algorithm prioritizes transactional workloads when possible but also aims to avoid batch job deadline miss. Based on the predictions of a fuzzy performance model, it performs efficient search in the space of all possible allocation combinations. ePower is adaptive to both workload and power changes and works automatically without human interventions.

We do not run a green powered datacenter. Instead, we utilize weather data to simulate the real green power generation and supply in the experiments. Given the weather data from NREL [13], we implement solar and wind energy prediction models [25] at the granularity of 10 minutes to obtain the green power traces.

Specifically, our contributions are as follows:

- We propose an elastic power-aware resource provisioning approach for heterogeneous workloads in self-sustainable datacenters that completely rely on renewable energy. It maximizes the system performance in terms of system goodput and respects the time-varying green power supply, different workload characteristics and QoS requirements.
- We develop a novel power-aware SA algorithm. The core of the SA algorithm is the design of a cooling schedule that considers dynamic power supply and the characteristics of heterogeneous workloads. To support SA-based resource allocation, we further develop a self-learning fuzzy performance model to predict application performance during searches in simulated states.
- We have implemented ePower on our university cloud testbed and performed extensive evaluations with the Gridmix2 and RUBiS benchmark applications. We utilize real weather traces to simulate the green power supply in the experiments. Experimental results demonstrate that ePower can achieve near-to-optimal system performance while being resilient to dynamic power availability. It outperforms a representative approach for heterogeneous workloads provisioning in a datacenter [6] by at least 24% system goodput improvement. It reduces response time bound violation of RUBiS workload and completion time bound violation of Gridmix2 workload by at least 35%.

The rest of this paper is organized as follows. Section 2 gives motivations on joint optimization of power and performance. Section 3 describes the design of ePower. Section 4 presents the details of the proposed simulated annealing algorithm. Section 5 presents the fuzzy performance modeling. Section 6 gives the testbed implementation. Section 7 presents the experimental results and analysis. Section 8 reviews related work. Section 9 concludes the paper.

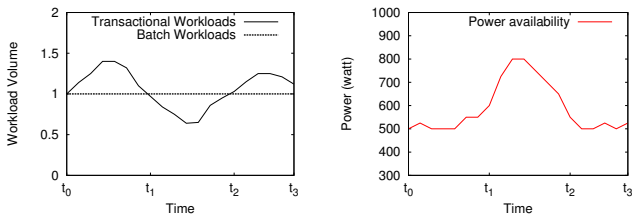
2 Motivation

To improve resource utilizations, datacenters consolidate workloads, ranging from transactional applications (e.g., e-Commerce website) to batch jobs (e.g., MapRedce data analytics), on the same physical hardware. These workloads are inherently heterogeneous with different QoS and resource requirements. A transactional workload comprises short client requests and its performance is measured based on the throughput of requests finished within a response time target. A MapReduce batch job is usually a long-running program with an expected completion time. While responsiveness is most important to transactional workloads as slow responses turn away potential customers, the delay in processing batch jobs is more tolerable and may be compensated later.

Another salient difference between the workloads is in their resource requirements. While the resource requirements of batch jobs is relatively stable during execution, the need of transactional workloads is quite dynamic [3], [6], [10] due to time-varying client traffic. Given a fixed total amount of resources, it is not trivial to determine the optimal resource allocations to these workloads. The dynamic power supply in a sustainable datacenter further complicates the problem. The varying power availability changes the amount of resources available to these workloads making an optimal allocation scheme both workload and power dependent.

For an illustration, Figure 1(a) plots the resource requirements of representative transactional and batch workloads in various time intervals. We profiled the transactional resource usage from replaying the Wikipedia trace [11] and the batch resource usage from the Facebook analytic workload [12]. It illustrates that the batch workload volume is relatively stable during the course of execution after the job is submitted. But for transactional workload, the workload volume (e.g., throughput) is quite dynamic. The unit on Y-axes is the dynamic workload volume of the transactional workload trace normalized to its beginning workload volume. The workload volume means throughput for RUBiS workload, and the size of input data for Hadoop workload. Figure 1(b) plots the dynamic availability of green power

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2428695, IEEE Transactions on Computers



(a) Resource requirements. (b) Dynamic power supply.

Fig. 1. Dynamic workload resource requirements and green power supply.

supply, which was derived from the weather condition records in the Measurement and Instrumentation Data Center [13]. From Figure 1, we can see that both the transactional workload and power supply show significant variations overtime. We identify three resource provisioning scenarios:

1) **Prioritizing transactional workloads:** During the interval $[t_0, t_1]$, the transactional workload increases while the green power supply is relatively stable and low. Based on the projection of high power availability in the next interval, in this interval more resources could be allocated to the transactional workload temporarily delaying the batch job.
2) **Compensating batch jobs:** During the interval $[t_1, t_2]$, green power supply increases while the transactional workload stays at a low level. The delayed batch job processing can be compensated with more resource allocation to the batch job for its speedup.
3) **Competing:** During the interval $[t_2, t_3]$, green power supply decreases and the transactional workload increases. The optimal resource allocation depends on complex interactions between the workloads. While the responsiveness of transactional workloads is important, an approaching batch job deadline should also be considered. The objective is to allocate resources in a way that maximizes the overall datacenter performance.

These challenges motivated us to develop a provisioning scheme that automatically optimizes the elastic resource allocations to heterogeneous workloads considering application QoS requirements, time-varying workload traffic, and dynamic green power supply. To this end, we propose ePower, an elastic power-aware datacenter resource manager.

3 ePower Design

ePower is a power-aware resource provisioning manager for heterogeneous workloads that maximizes the overall system performance in the presence of dynamic green power supply. The key insight is that the processing of long-running batch jobs can be temporarily delayed (or slowed down) in order to prioritize transactional workloads. ePower also aims to ensure that batch jobs are able to meet their completion deadlines.

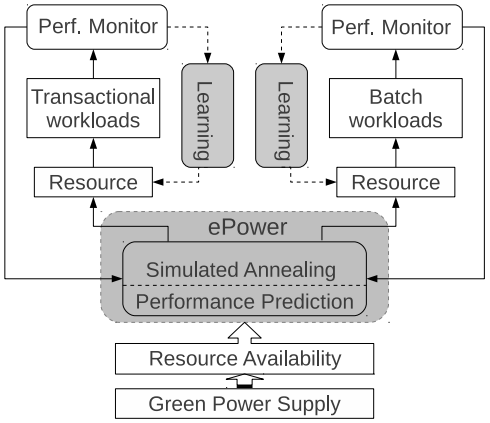


Fig. 2. The architecture and components of ePower.

Figure 2 shows the major components and their interactions of ePower. It takes the availability of green power supply as input and calculates the corresponding resource availability under such a power budget. Based on the amount of available resources, ePower determines the optimal resource allocations to the workloads.

The key to the ePower design is the automatic resource provisioning scheme that combines fuzzy performance modeling and simulated annealing resource optimization. Every control interval, ePower searches for a resource allocation with respect to overall system performance maximization. The simulated annealing component randomly picks up a resource allocation and queries the fuzzy performance model for the prediction of workload performance. If the predicted system performance is significantly better than the current one, ePower accepts the new allocation. At the end of each control interval, the performance of both workloads is fed back to ePower to update the performance model.

In this section, we define a metric to quantify the system performance with heterogeneous workloads and formulate the resource provisioning as an optimization problem. We elaborate the design of the simulated annealing optimization and fuzzy performance modeling in Section 4 and Section 5, respectively.

3.1 Quantifying System Performance

Although heterogeneous workloads have individual measures of client-perceived performance, such as request response time and job completion time, a unified metric is needed for cloud providers to quantify the benefit of resource allocation. We define system *goodput* as the total useful work delivered to users at a certain period of time. Specifically, it is the amount of effective data throughput completed by transactional workloads or batch jobs that meet their corresponding service level objectives (SLOs). Similar metrics have been used to quantify system performance for transactional workloads [14] and batch jobs [15], [16], respectively.

Formally, we define system goodput $G(k)$ at time

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2428695, IEEE Transactions on Computers

interval k as the aggregate effective data throughput:

$$G(k) = \sum E_i(k), i \in \mathbb{T} \cup \mathbb{B}, \quad (1)$$

where $\mathbb{T}$ and $\mathbb{B}$ are sets of transactional workloads and batch jobs, respectively. E_i is a job's effective data throughput. For heterogeneous workloads, E_i is uniformly defined as:

$$E_i(k) = \frac{\sum_{j=1}^{m} d(j) * \delta(j)}{l(k)}, j \in \mathbb{T} \cup \mathbb{B}, \quad (2)$$

where $d(j)$ is the data size of job j in a group of m jobs that finish during time period k, in which $l(k)$ is the length of interval k. Let t_j be the response time of job j. To count only useful work, decay function $\delta(j)$ discounts the data throughput from jobs with violated SLOs:

$$\delta(j) = \begin{cases} 1 & \text{if } t_j < t_{soft}. \\ 1 - \frac{t_j - t_{soft}}{t_{soft}} & \text{if } t_{soft} \leq t_j \leq t_{hard}. \\ 0 & \text{if } t_j > t_{hard}. \end{cases} \quad (3)$$

We use a SLO with two time bounds, t_{hard} and t_{soft}, in the decay function. While t_{hard} sets a hard deadline for job completion, beyond which no revenue is generated, t_{soft} is a soft deadline whose violation will incur reduction in revenue. We ensure that hard deadlines to be no long than 2 times of the soft deadlines so that the decay function never becomes negative. Accordingly, $\delta(j)$ considers the data delivered by job j as useful work if the observed job finish time t_j meets t_{soft}. Violations of t_{soft} and t_{hard} result in a linear decay in the counted throughput and zero work, respectively. Cloud providers could choose different decay functions for different types of tasks. In this study, we choose to define it uniformly for transactional and batch workloads and measure their performance using a unified metric, goodput.

The way of data size measurement in Eq. (2) is different for transactional jobs and for batch jobs. For transactional workload, ePower treats requests as individual jobs and estimates the transaction data size $d_t(j)$ as the size of the response sent back to the clients. For batch jobs, ePower uses the input data size to approximate the job size $d_b(j)$. Although the calculations for heterogeneous workloads are different, ePower uses a unified performance metric (i.e., goodput) to count the data delivered by both workloads as useful work to users in the datacenter. The definition of goodput encourages transactional workloads or batch jobs to complete and meet the SLOs. The decay function ensures that ePower only counts useful work to users while allocating the available resource to different workloads.

For long-running batch jobs that cannot be finished in one control interval, we include the data processed for the job so far in the effective data throughput and use $t_j(k)$, the elapsed execution time until interval k, for the observed job finish time. $\triangle p_j(k)$ refers to the progress of job j's execution in interval k and thus the data throughput is calculated as $d_b(j) \times \triangle p_j(k)$. Many batch jobs provide the interface to query the execution progress, such as the *JobTracker* in Hadoop.

3.2 The Optimization Problem

We formulate the resource provisioning for heterogeneous workloads as a combinatorial optimization problem with constraints:

$$\text{Maximize } \sum_{k=1}^{K} G(k) \quad (4)$$

$$\text{Subject to } P_t(k) + P_b(k) \leq P(k) \quad (5)$$

$$0 \leq t_{t,i} \leq t_{t,hard}, i = 1, 2, \ldots, m \quad (6)$$

$$0 \leq t_{b,j} \leq t_{b,hard}, j = 1, 2, \ldots, p \quad (7)$$

TABLE 1
Main notations.

Symbol	Meaning
k	The control time interval
K	Total number of control intervals
$G(k)$	System goodput in the k_{th} interval
$P_t(k)$	Power consumption of transactional workload
$P_b(k)$	Power consumption of batch workload
$P(k)$	Overall power supply in the k_{th} interval
$r_t(k)$	Resource allocation for transactional workload
$r_b(k)$	Resource allocation for batch workload
$s(k)$	Resource allocation solution in the k_{th} interval
$r_a(k)$	Total available resource amount in the k_{th} interval

The notations used for the problem formulation are listed in Table 1. The objective function Eq. (4) is to maximize the overall system performance with respect to the system goodput. There are three constraints. Eq. (5) requires that the sum of the power consumption of transactional workloads $P_t(k)$ and batch workloads $P_b(k)$ is upper bounded by the power supply $P(k)$ in the k_{th} control interval. $P_t(k)$ and $P_b(k)$ are determined by the amount of the resource allocated to the two different workloads, which are controlled via dynamic resource provisioning. Eq. (6) and Eq. (7) require that any transactional request or batch job needs to meet its corresponding hard deadline or its data throughput will not be counted in $G(k)$, where m and p is the number of requests and jobs respectively. The resource availability in a datacenter is the overall amount of the CPU resource available for workloads, which is determined by the dynamic power supply for the datacenter. In this work, we focus on optimizing CPU resource allocations between Hadoop and RUBiS workloads under such availability constraint (i.e., Eq. (5)). Note that the availability of green power supply varies depending on the natural weather conditions. Thus, the system power budget $P(k)$ changes accordingly. Thus, the resource allocation solution should be updated over time. With limited power supply, decisions need to be made balancing the power budget of workloads. Since performance degradation of transactional workloads directly leads to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2428695, IEEE Transactions on Computers

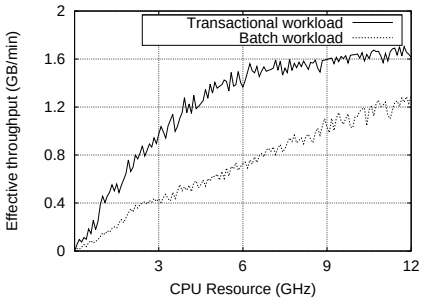


Fig. 3. The effective throughput under different resources.

reduced goodput and the processing of batch jobs may be compensated when more power supply is available, transactional workloads is to be prioritized. Care should also be taken to avoid the deadline miss of batch jobs.

4 Simulated Annealing Optimization

To optimize system goodput with the presence of dynamic power supply, we develop an efficient simulated annealing algorithm for the search of the optimal resource allocations. Simulated annealing is a robust optimization technique, which can deal with highly nonlinear models, chaotic, noisy data, and multiple constraints [17]. The original annealing process is to find the point with the lowest energy. To avoid trapping in the local optimum, the annealing process explores neighbour points according to a time-varying parameter T called the *temperature*. The larger the temperature, the higher the chance that neighbour points are visited. When $T = 0$, the procedure reduces to a greedy algorithm. The core of a simulated annealing algorithm is to design a *cooling schedule* that dynamically changes T to balance the exploration and exploitation.

The optimal resource allocation problem fits within the annealing process. The objective is to find the resource allocation that maximizes system goodput. Recall that system goodput is defined as the aggregate effective data throughput of transactional and batch workloads. In Figure 3, we plot the effective data throughput of the RUBiS transactional workload and the Gridmix2 batch workload when given different amount of resources. We can see that allocating resources to transactional workloads gives a higher margin in the overall goodput. Thus, the original simulated annealing algorithm is likely to favor (or prioritize) transactional workloads in resource allocation irrespective of the execution progress of batch jobs and the availability of green power supply.

Our objective is to design a cooling schedule that considers the varying power budget and avoids missing batch job deadlines. The optimization of system goodput favors transactional workloads. We define a batch job's capacity for resource migration as the inverse of the job's temperature. When the temperature is low, the annealing process seeks an optimal solution without compromising batch jobs. In the next, we elaborate the design of the power-aware simulated annealing algorithm.

4.1 Cost Function

The cost function is used to compare the quality of solutions in searching space. Originally, the annealing process aims to minimize the cost function. For the resource allocation problem, the objective is to maximize the system goodput. Accordingly, the cost function is defined as the inverse of the system goodput. For a solution $s(k)$, its cost function is defined as

$$C(s(k)) = \frac{1}{G(k)}. \tag{8}$$

4.2 Search Space

For the resource allocation problem, the search space is the set of possible allocations to the heterogeneous workloads. A solution (or a state) $s(k)$ in the search space is defined as a vector of resource allocations:

$$s(k) = (r_t(k), r_b(k)), \tag{9}$$

where $r_t(k)$ and $r_b(k)$ are the resource allocations for transactional workloads and batch workloads at the k_{th} interval, respectively. The total available resource is

$$r_a(k) = r_t(k) + r_b(k). \tag{10}$$

Based on the available power supply, ePower dynamically adjusts the total amount of resources. It builds a simple model, similar to the relationship model of CPU resource and its power consumption proposed in [18], to translate the power budget into an affordable amount of resources. The total resource available for hosting workloads is then capped by the predicted power supply value with some slackness. When power supply changes, resources are added to or removed from both workloads in proportion to their allocations in the last interval.

4.3 Cooling Schedule

The cooling schedule defines the way the temperature changes in the searching process. The temperature should be high to allow free explorations when the power supply is abundant and batch jobs have distant deadlines. The temperature should be set carefully when power supply is limited and batch job deadline is approaching. The setting ought to allow free competition between different workloads but guaranteeing batch job's completion.

4.3.1 Initial Temperature

As others [5], [19], we assume that there is only one batch job running in the system at a time and reset the temperature when a new batch job starts. The initial temperature T_0 is set to be 1 to encourage explorations. It indicates a zero resistance to resource depriving at the beginning stage of a batch job execution. Since batch jobs usually have long execution time, there are chances that the processing delay can be compensated when power supply increases or transactional workloads decrease.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2428695, IEEE Transactions on Computers

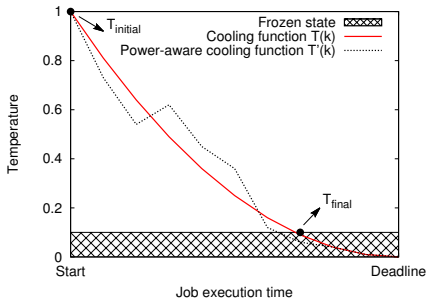


Fig. 4. Temperature cooling function.

4.3.2 Cooling Function

The cooling of the temperature serves two purposes. First, as the temperature cools down, the searching algorithm approximates a greedy algorithm and converges to the optimal solution. Second, a low temperature increases the batch job's resistance to processing delay so as to avoid a deadline miss. The cooling function $T(k)$ is defined as:

$$T(k) = \begin{cases} (1 - \frac{t_j(k)}{t_{soft}})^2 & \text{if } \frac{t_j(k)}{t_{soft}} < \sigma, \\ 0 & \text{if } \frac{t_j(k)}{t_{soft}} \geq \sigma. \end{cases} \quad (11)$$

where $t_j(k)$ refers to the elapsed execution time until interval k and σ measures the proximity to the soft deadline. The temperature decreases quadratically as batch job execution approaches its deadline. Once we are close enough to the deadline (i.e., exceeding the proximity measure σ), the temperature drops to zero freezing the exploration and speeding up the batch job.

The simulated annealing method is used at every control interval most independently. In the cooling schedule, if a batch job's execution crosses multiple control intervals, there is a connection among the SA runs, that is, the cooling temperature starts from that of the previous control interval.

4.4 Power-aware Cooling Schedule

The temperature of the annealing process needs to be adjusted when the power availability changes. If the power supply increases, the temperature should increase allowing more explorations. If power supply decreases, the temperature should drop accordingly avoiding the starvation of the batch job. The change of temperature restarts the search of optimal solutions, which requires several iterations to converge. We use the predicted power supply based on history data to proactively start the new search.

The next interval power supply $P(k+1)$ is predicted based on the last n observations of power supply (i.e., $P(k)$, ... $P(k-n+1)$). ePower adopts the Auto-Regressive Integrated Moving Average (ARIMA) model [20], [21] to predict the power series.

$$P(k+1) = a_1 P(k) + a_2 P(k-1) + ... + a_n P(k-n+1), \quad (12)$$

where $a_1, a_2, \ldots, a_n$ are coefficients obtained via model fitting. To incorporate power availability into the cooling function, we look h steps ahead in the power supply. Considering the long-term power supply allows early adjustment of the temperature. Let $P(k+h)$ be the h step prediction, we define the power impact function as:

$$I(k) = \frac{P(k+1)}{P(k)} * \tau \frac{P(k+2)}{P(k)} ... * \tau^{h-1} \frac{P(k+h)}{P(k)}, \quad (13)$$

where τ is the discount factor that devalues distant power predictions. The coefficients can be obtained by least square curve fitting.

Accordingly, the power-aware cooling function $T'(k)$ is defined as:

$$T'(k) = \begin{cases} (1 - \frac{t_j(k)}{t_{soft}})^2 \times I(k) & \text{if } \frac{t_j(k)}{t_{soft}} < \sigma, \\ 0 & \text{if } \frac{t_j(k)}{t_{soft}} \geq \sigma. \end{cases} \quad (14)$$

Figure 4 plots the temperatures of the two cooling function discussed. The results are obtained from experimentations in a testbed that is described in Section 6.1. The solid and dotted plots represent the power-agnostic cooling function $T(k)$ and the power-aware cooling function $T'(k)$, respectively. Both cooling functions start with an initial value of 1, but decays differently thereafter. $T(k)$ decreases quadratically while $T'(k)$ is also affected by the dynamic power supply. Figure 4 also draws the frozen zone of the power-agnostic cooling function, beyond which the temperature is effectively zero.

Note that a batch job deadline miss is still possible due to limited power supply. ePower allows a batch job to finish after its hard deadline, but its throughput would not be counted in the system goodput. ARIMA modeling accuracy can affect ePower prediction accuracy of the power availability. If ePower overestimates the available power, it may reduce the goodput of Hadoop workload. If it underestimates the available power, it may reduce the goodput of RUBiS workload.

4.5 The Simulated Annealing Algorithm

The optimal solution of the simulated annealing algorithm is to find the resource allocation that minimizes the cost function (or maximizes goodput). To avoid becoming trapped in local optimum, the algorithm employs a random search guided by a state transition function.

At each interval, a simulated annealing algorithm randomly generates a neighbouring $s'(k)$ and tests if the searching should move to the new state. The acceptance of the transition is probabilistically decided by the Metropolis criterion [17] as it allows SA algorithm to explore more possible solutions:

$$pos[s(k), s'(k)] = exp[\frac{-[C(s'(k)) - C(s(k))]}{T'(k)}]. \quad (15)$$

Algorithm 1 shows the simulated annealing algorithm used in ePower. The algorithm takes current power supply $P(k)$ and resource allocation $s(k)$ as inputs and outputs the next interval resource allocation $s(k+1)$.

Algorithm 1 Simulated Annealing Algorithm of ePower.

1: **Inputs:** Power supply $P(k)$ and current state $s(k)$.
2: Update cooling function $T'(k)$ with $I(k)$;
3: /* Start searching from current state */
4: $s_{min}(k) = s(k)$;
5: **repeat**
6: /* Randomly generate a neighboring state */
7: $s'(k) = generate(s_{min}(k))$;
8: **if** $r'_b(k) > r_{b,min}(k)$ and $C(s'(k)) < C(s_{min}(k))$ **then**
9: Move to next state: $s_{min}(k) = s'(k)$;
10: Continue;
11: **end if**
12: **if** $r'_b(k) \leq r_{b,min}(k)$ and $pos[s_{min}(k), s'(k)] \geq random[0,1]$ **then**
13: Move to next state $s_{min}(k) = s'(k)$;
14: **end if**
15: **until** Number-of-iterations $\geq N$ (a stopping threshold)
16: $s(k+1) = s_{min}(k)$;
17: **Output:** Next interval provisioning $s(k+1)$

Starting from current state $s(k)$, the algorithm repeatedly searches for the state $s_{min}(k)$ minimizing the cost function. At each iteration, a neighbouring state $s'(k)$ is generated and tested. Since ePower favors the transactional workload, any move to a state that allocates more resources to the batch job (i.e., $r'_b(k) > r_{b,min}(k)$) is accepted as long as the cost function reduces (line 8). For a state that deallocates resources from the batch job, the decision on the move is made based on the state transition probability $pos[s_{min}(k), s'(k)]$, which takes the cost function and batch job deadline into consideration (line 12). The process restarts at the new state s_{min} and is repeated for N times. When the search terminates, the best state $s_{min}(k)$ achieved so far is assigned to $s(k+1)$, the resource provisioning in the next interval.

The search process simulates N moves in the solution space. At each move, a performance model is used to predict the system goodput at the new state. We empirically determined that an iteration of $N = 500$ steps provide a good tradeoff between the control accuracy and overhead. We prove that the proposed simulated annealing algorithm converges to an optimal solution given sufficient number of iterations in Appendix A. We discuss the design of the fuzzy performance model in Section 5.

5 Fuzzy Performance Modeling

ePower uses the self-learning fuzzy model for the prediction of system goodput in the simulated annealing search process. The fuzzy model characterizes the complex behaviors of the heterogeneous workloads. It is able to describe complex non-linear relationships between different workloads and their resource allocations by a set of linguistic rules in a heterogeneous workload environment. Its ability to give good performance prediction over a wide range of operating conditions is essential for optimization and control.

5.1 The Fuzzy Model

ePower adopts a fuzzy model to describe complex behaviors of the resource allocation and achieved application performance. For brevity, we use $E(k)$, $r(k)$, and $w(k)$ to represent the effective throughput, resource allocation, and workload level of heterogeneous workloads, respectively. The model is of the input-output Nonlinear Auto Regressive model with eXogenous inputs (NARX) as follows.

$$E(k+1) = R(r(k), w(k), \xi(k)). \tag{16}$$

R is the relationship between the input variables and the output variable. The regression vector $\xi(k)$ contains a number of lagged outputs and inputs of the previous control interval. It is represented as

$$\begin{aligned}\xi(k) = [(E(k), E(k-1), \cdots, E(k-n_p)),\\ (r(k), r(k-1), \cdots, r(k-n_r))]^T\end{aligned} \tag{17}$$

where n_p and n_r are the number of lagged values for outputs and inputs. Let ρ denote the number of elements in the regression vector $\xi(k)$, that is, $\rho = n_p + n_r$.

R is the rule-based fuzzy model that is consisted of Takagi-Sugeno rules [22]. A rule R_i is represented as

$$\begin{aligned}R_i : \text{IF } \xi_1(k) \text{ is } \Omega_{i,1}, \xi_2(k) \text{ is } \Omega_{i,2}, \cdots, \text{ and } \xi_\rho(k) \text{ is } \Omega_{i,\rho}\\ r(k) \text{ is } \Omega_{i,\rho+1} \text{ and } w(k) \text{ is } \Omega_{i,\rho+2}\\ \text{THEN } E_i(k+1) = \zeta_i \xi(k) + \eta_i r(k) + \omega_i w(k) + \theta_i.\end{aligned} \tag{18}$$

Here, $w(k)$ is the workload volume. Ω_i is the antecedent fuzzy set of the i_{th} rule, which is composed of a series of subsets: $\Omega_{i,1}, \Omega_{i,2}, \cdots, \Omega_{i,\rho+2}$. ζ_i, η_i and ω_i are parameters, and θ_i is the offset. Their values are obtained by offline training.

Each fuzzy rule describes an operating space of the nonlinear system model. The spaces have some overlaps. So each output contains several fuzzy rules. The output $E(k+1)$ is computed as the weighted average value by the rules. That is,

$$E(k+1) = \frac{\sum_{i=1}^{L} \gamma_i (\zeta_i \xi(k) + \eta_i r(k) + \omega_i w(k) + \theta_i)}{\sum_{i=1}^{L} \gamma_i}. \tag{19}$$

In Eq. (19), L is the number of rules for the output. γ_i is the degree of fulfillment for the i_{th} rule. The value of γ_i is the product of the membership degrees of the antecedent variables in that rule. Membership degrees are determined by fuzzy membership functions associated with the antecedent variables. The model output in Eq. (19) is expressed in the form of

$$E(k+1) = \zeta^* \xi(k) + \eta^* r(k) + \omega^* w(k) + \theta^*. \tag{20}$$

The aggregated parameters ζ^*, η^*, ω^* and θ^* are the weighted sum of vectors ζ_i, η_i, ω_i and θ_i respectively.

5.2 Online Self-learning of the Fuzzy Model

Due to high workload dynamics, we design an online self-learning module to adapt the fuzzy model. The self-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2428695, IEEE Transactions on Computers

learning module aims to minimize the prediction error $e(k)$ of the fuzzy model, $e(k) = E(k) - \hat{E}(k)$. Here, $E(k)$ is the measured output value of the control system and $\hat{E}(k)$ is the model's predicted value for $E(k)$.

The fuzzy model consists of many rules. If $e(k) \neq 0$, we apply a recursive least squares (RLS) method to adapt the parameters of the current fuzzy rule. We express the fuzzy model output in Eq. (19) as follow:

$$E(k+1) = \phi(k)X(k) + e(k) \tag{21}$$

where $e(k)$ is the error between the actual output and predicted output. $\phi(k)$ is a vector composed of the model parameters. $X(k) = [\xi(k)^T, r(k)]$ is a vector containing the current and previous outputs and inputs of the control system. The parameter vector $\phi(k)$ is estimated so that the error value in Eq. (22) is minimized.

$$Error = \sum_{k-1}^{k} (e(k)^2 + 0.5e(k-1)^2). \tag{22}$$

We apply both the current error $e(k)$ and the previous error $e(k-1)$ to estimate the parameter vector. The parameters of fuzzy model are updated according to the RLS method as follows:

$$\begin{aligned} \phi(k) =& \phi(k-1) + Q(k)X(k-1)[E(k) - X(k-1)\phi(k-1)] \\ Q(k) =& \frac{1}{2}[Q(k-1) \\ & - \frac{Q(k-1)X(k-1)X^T(k-1)Q(k-1)}{\frac{1}{2} + X^T(k-1)Q(k-1)X(k-1)}] \end{aligned} \tag{23}$$

Here $Q(k)$ is the updating matrix. The initial value of $\phi(0)$ is the value obtained in an offline identification. The initial value of $Q(0)$ is equal to $(X(0)^T X(0))^{-1}$.

6 System Implementation

6.1 The Testbed

We built a testbed in a university prototype datacenter, which consists of five Dell PowerEdge R610 servers and two Dell PowerEdge R810 servers. Totally, they have 10 Intel 6-core Xeon X5650 CPUs, 8 Intel 6-core E7540 CPUs, and 704 GB memory. The servers are connected with 10 Gbps Ethernet. VMware vSphere 5.0 is used for server virtualization. VMware vSphere module controls the CPU usage limits in MHz allocated to the virtual machines (VMs). It also provides an API to support the remote management of VMs. The version of Hadoop used in the experiment is 1.0.03. The Hadoop cluster is configured with 11 VMs. Each VM is allocated 1 VCPU and 1 GB memory. One VM runs the *JobTracker* and hosts the *NameNode*. Each of the remaining 10 VMs hosts a *DataNode* and a *TaskTracker*. As in the work [23], each Hadoop DataNode is congured with a single map and reduce slot. The transactional workloads are hosted in a VM with 4 VCPU and 4 GB memory. All VMs use Ubuntu Server 10.04 with Linux 2.6.32.

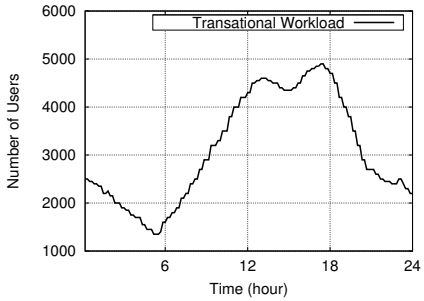


Fig. 6. A transactional workload.

ePower uses dynamic power supply to infer the amount of available CPU resource in GHz and optimizes CPU resource allocation under the resource constraint. ePower's SA algorithm determines the CPU usage limit of individual vCPU and enforces it via the vSphere management interface. The usage limit for each vCPU is in the range of 0 to 2.8 GHz. For example, when there is low power supply, the virtual CPU allocation is reduced to save the power consumption. How many cores should be switched off or turned to low power state is controlled by the hypervisor (VMware) in our testbed.

6.2 Workloads

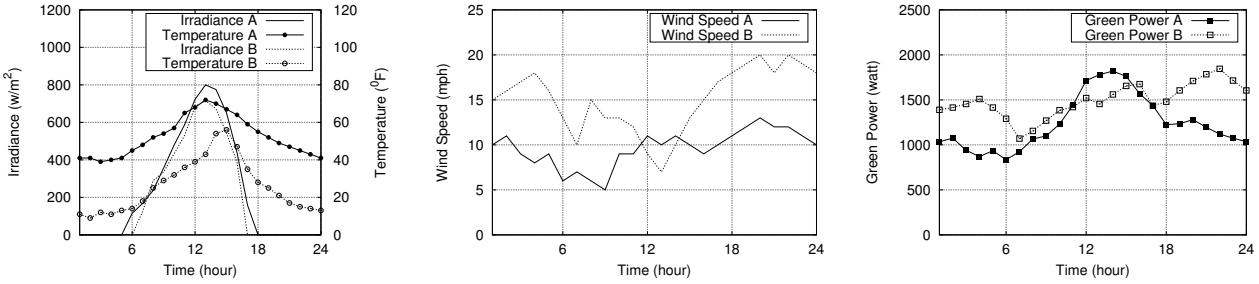
6.2.1 Transactional Workload

As others [14], [10], [24], we use open-source RUBiS as the transactional benchmark application in the experiments. RUBiS provides a web auction application that is modeled in a similar way to *ebay.com*. It characterizes the workload into three categories, browsing, bidding, and selling. The RUBiS workload generator emulates user requests at different concurrent levels. We use a real Internet trace from Wikipedia.org [11] to mimic the daily dynamics of workload volume, which represents the users' behavior in visiting the Wikipedia website. Figure 6 shows the transactional workload that we use for the experiments. The number of concurrent users dynamically changes from 1300 to 5000 in 24 hours. Similar to the work in [14], our experiments set the soft response time bound to be 1000 ms and the hard response time bound to be 1500 ms.

6.2.2 Batch Workload

We use a synthetic workload, called "Facebook-Derived (FaceD)", which models Facebook's production workload [12]. Appendix B shows the details of FaceD workload characteristics. FaceD is a scaled-down version of the workload studied in [19], [12] due to the scale limitation of the testbed cluster. As the work in [19], we do not run the Facebook code itself. Rather, we mimic the characteristics of the jobs using "loadgen". Loadgen is a configurable MapReduce job from the Gridmix2 benchmark included in the Hadoop distribution.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2428695, IEEE Transactions on Computers



(a) Irradiance and temperature. (b) Wind speed. (c) Green power supply.

Fig. 5. Green power generation by solar and wind in the self-sustainable datacenter in two one-day scenarios.

6.3 Green Power Supply

We use the prediction methods proposed in work [25] to estimate the amount of the green power supply based on the natural weather condition in our local city. We utilize the data to simulate the real green power generation and supply in the experiments. As many others, we consider two main renewable sources, solar and wind. We assume that the sustainable datacenter has 7 solar panels and one micro-turbine, each with the capability of producing 1.8 KW power. We choose a power prediction interval as long as the typical weather condition dynamic length that is in the order of 10 minutes. We also set the control interval of the resource provisioning to be 10 minutes in the experiments. In our testbed, the proposed ePower algorithm takes approximately 30$\sim$40 ms to execute. This overhead is negligible compared to the 10-minute control interval.

To emulate the intermittent availability of renewable energy, we use meteorological data from the Measurement and Instrumentation Data Center (MIDC) of the National Renewable Energy Laboratory [13]. A variety of meteorological data, including irradiance, temperature, and wind speed, is covered in those records from the MIDC. Moreover, prior studies [5], [25] have shown that the data from the MIDC is sufficiently accurate.

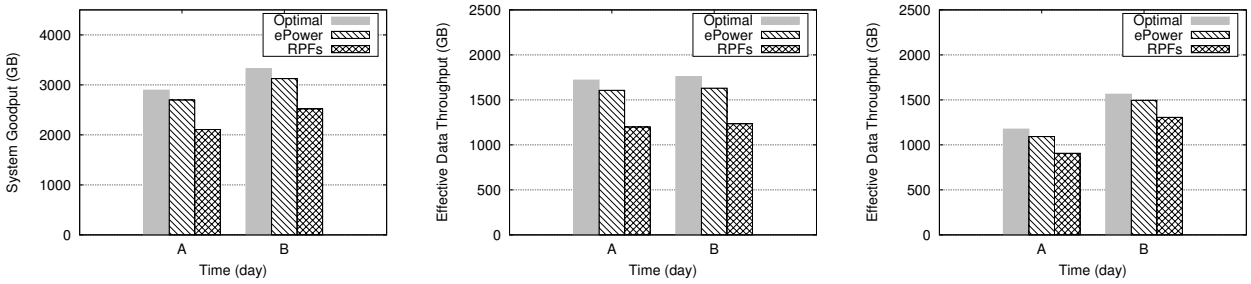
Based on the prediction methods and the weather condition records obtained from MIDC, we obtain the green power supply for the self-sustainable datacenter. Figure 5(a) and Figure 5(b) show the weather records of irradiance, temperature, and wind speed of our local city in two typical days, a summer day A and a winter day B. Figure 5(c) shows the time-varying green power supply in the datacenter in two one-day scenarios.

6.4 ePower Components

We implement ePower components on a separate VM, which issues commands to the virtualized server cluster via VMware vSphere API 5.0. ePower uses dynamic power supply to infer the amount of available CPU resources (e.g., GHz) and optimizes vCPU resource allocation under the constraint. For I/O-bound and memory-bound workloads, power consumption is mainly connected with the number of memory activities and disk spinning rate rather than the disk/memory sizes. Although models can be used to derive the upper bound of such activities and spinning rate given a power supply, there is a lack of management interface in existing virtualization platforms to control the allocation of these resources. Based on these practical issues, in the experiments ePower focuses on allocating CPU resources, which are commonly believed to be the major power consumer in datacenters [10], [19], [26]. Its SA algorithm determines the CPU usage limit and the limit is then applied to the vSphere management interface. The usage limit for each vCPU is in the range between 0 to 2.8 GHz.

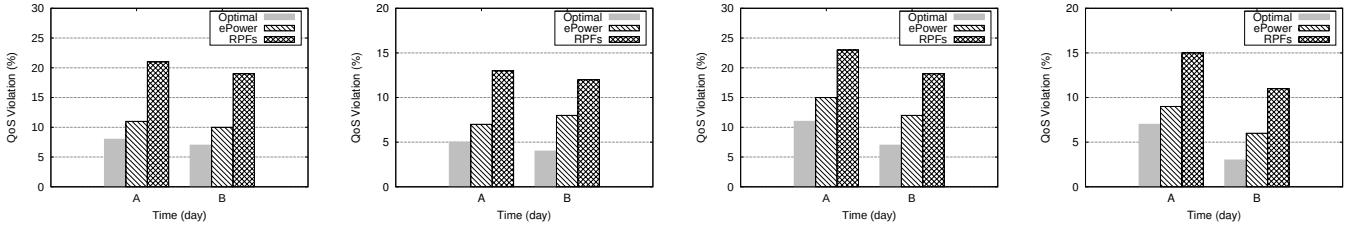
1) **Power Monitor:** The real-time power consumption of the virtualized cluster is measured at the resource pool level. The power monitor gathers the measurement data through VMware ESX 5.0 Intelligent Power Management Interface sensors. It implements a threshold-based power capping mechanism [27].
2) **Performance Monitor:** For transactional applications, it uses a sensor program provided by RUBiS client for performance monitoring in terms of request response time and the data size of each request. For batch jobs, it measures each job completion progress and job size by *JobTracker* on the *NameNode* periodically. Then the monitor calculates each batch job's virtual performance.
3) **ePower Controller:** It applies the SA algorithm for system resource provisioning optimization, which is implemented as a standalone daemon on the management machine. The batch job deadline proximity threshold σ is set to 0.7. We empirically determined that the threshold prevents most batch job violations while giving sufficient flexibility to ePower for prioritizing transaction workloads. For each interval, the annealing algorithm uses a $h = 3$ step prediction of power supply and searches the solution space for $N = 500$ iterations. Two self-learning fuzzy modules run as daemons on the RUBiS server and the Hadoop's *NameNode*, respectively.
4) **Resource Allocator:** It uses vSphere API to impose CPU usage limits on the VMs. The vSphere module provides an interface to execute a method `ReconfigVM` to modify a VM's CPU usage limit. The granularity of CPU allocation is 0.01 GHz.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2428695, IEEE Transactions on Computers



(a) Combined workloads. (b) RUBiS workload. (c) Hadoop workload.

Fig. 7. The system goodput achieved by optimal, ePower and RPFs in two one-day scenarios.



(a) RUBiS soft violations. (b) RUBiS hard violations. (c) Hadoop soft violations. (d) Hadoop hard violations.

Fig. 8. Soft and hard QoS requirement violations of RUBiS and Hadoop workloads in two one-day scenarios.

7 Performance Evaluation

We first demonstrate that ePower performs closely to an offline determined optimal approach and achieves better system goodput and less SLO violations than a representative resource provisioning scheme. We then show its adaptability under dynamic green power supply and how it prioritizes transactional workloads and compensates batch jobs. Finally, we show the accuracy of the self-learning performance modeling.

To obtain the optimal system goodput, we derived the optimal resource allocation by applying the proposed SA algorithm offline using the workload trace in Figure 6 and the power availability trace in Figure 5(c). For comparison, we also implemented Relative Performance Functions (RPFs) [6], a representative online approach for resource provisioning of heterogeneous workloads in a datacenter. RPFs was proposed for fair trade-offs between the different workloads in terms of the relative distance from their performance goals. We modified RPFs so that it also supports power prediction and power driven resource provisioning with respect to dynamic green power supply. However, RPFs does not prioritize the power budget of heterogeneous workloads with respect to various green power supply.

7.1 Optimizing System Goodput

Goodput improvement. Figure 7 compares the system goodput achieved by the optimal approach, ePower and RPFs on two days with different power supply conditions. In the experiments, ePower does not count any throughput towards goodput if a batch job missed its hard deadline. Figure 7 shows that the overall system goodput of ePower outperformed RPFs by 28% and 24% on day A and day B, respectively. ePower also achieved near-optimal performance with goodput that is 93% and 94% to the offline optimal solution on day A and day B. Figures 7(b) and 7(c) show the performance comparison of RUBiS and Hadoop workloads by the different approaches. For the RUBiS workload, ePower outperforms RPFs by 34% and 32% while achieving 93% and 92% of the optimal solution on day A and day B. For the Hadoop workload, ePower outperforms RPFs by 20% and 15% while achieving 93% and 96% of the optimal solution on day A and day B, respectively.

QoS violation improvement. Figure 8 compares the three approaches in terms of QoS violations. We draw the percent of hard/soft QoS violations, which is the ratio of requests/jobs that missed the hard/soft deadline and the total number of jobs, under different approaches for RUBiS and Hadoop workloads, respectively. Note that the power supply was not always sufficient for hosting the heterogeneous workloads during the two day periods, even the optimal approach incurred on average 6% and 4% violations to RUBiS and Hadoop, respectively. In day A (summer time), when the power supply varies more significantly than day B (winter time), both power supply and RUBiS closely followed the diurnal pattern. As a result, there was less flexibility to migrate resources between RUBiS and Hadoop because power was also inadequate when RUBiS traffic dropped. This explains why the optimal approach incurred a lower Hadoop QoS violation rate on day B. According to Figue 8, ePower performed closely to the optimal approach with on average 42% and 40% less hard or soft violations for RUBiS and Hadoop workloads than RPFs did.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2428695, IEEE Transactions on Computers

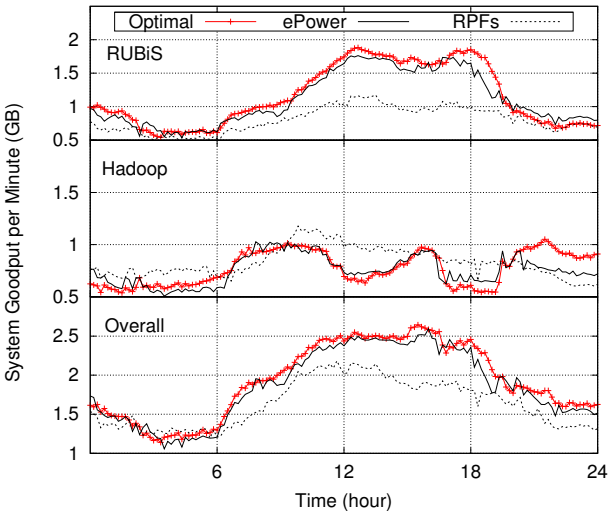


Fig. 9. Effective data throughput traces on day A.

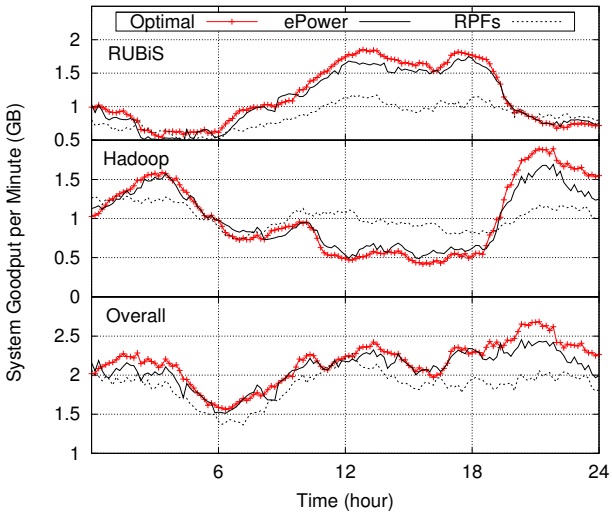


Fig. 10. Effective data throughput traces on day B.

Trace analysis. To better understand the performance of different approaches when power supply fluctuates, Figures 9 and 10 plot the microscopic view of the achieved goodput in the two day periods. The figures show the goodput of RUBiS workload, Hadoop workload, and the combined workloads due to ePower, optimal and RPFs, respectively. In the first six-hour period of day A, both RUBiS workload and the power supply were relative low so that there was little room for optimization. All three approaches performed similarly. In the same period of day B, when more power supply was available, the optimal approach and ePower allocated more resources to Hadoop without compromising the performance of RUBiS. Similar compensation to Hadoop can also be observed between the 18_{th} and 24_{th} hours on day B, which leads to overall goodput improvement.

When the RUBiS workload and the power supply were high on both days (i.e., between the 6_{th} and 18_{th} hours), a resource allocation scheme needs to make trade-offs between RUBiS and Hadoop. ePower and the optimal approach chose to prioritize RUBiS as it generates a higher margin in system goodput (as shown in Figure 3). As such, although RPFs achieved better goodput for Hadoop on both days, ePower and the optimal approach delivered much better goodput for RUBiS resulting in optimized system goodput. Note that ePower occasionally outperformed the optimal solution at some control intervals. ePower relies on the ARIMA model to predict the availability of power supply and may aggressively compensates Hadoop, which leads to a higher goodput, given that power supply is predicted to be too low to finish Hadoop jobs in time. Because the optimal solution works with the actual power trace and thus delivered better goodput over the 24-hour periods.

Validating performance merits. For more comparisons, we also implemented another Cooperative Resource Provisioning Solution (CRPs) [7] for heterogeneous workload provisioning in a datacenter. The CRPs approach takes advantage of a group of heterogeneous workloads to save the peak resource consumption. Figure 11 compares the combined system goodput, RUBiS goodput, and Hadoop goodput achieved by ePower and CRPs respectively. It shows that ePower outperforms CRPs by 28% in terms of the overall system goodput. This is due to the fact that CRPs does not support resource migration between these two heterogeneous workloads while ePower does that by exploiting the dynamic characteristics of the RUBiS workload.

7.2 Adapting to Dynamic Power Supply

In this subsection, we show that ePower adaptively caps the total power consumption based on the availability of green power supply and dynamically allocates resources to heterogeneous workloads.

7.2.1 Adaptive Power Capping

ePower does not assume that the green power supply is always sufficient to power the whole datacenter. Instead, it controls the total power consumption by capping the available resources that can be used by hosted workloads. Similar to [18], ePower builds a model translating the power budget into an upper limit of resources. Figure 12(a) shows the system power consumption in the presence of dynamic power supply on day A and day B. We can observe that ePower was able to control the power consumption by adaptively controlling the overall available resources to heterogeneous workloads. This is due to its multi-interval power supply prediction based on the ARMIA model and its threshold-based power capping mechanism that ensures that the real power consumption is below the actual power supply. Nevertheless, there could exist overlaps between the power supply and consumption, depending on the threshold value and the real workload dynamics. In this particular experiment, note that we did not observe the violation.

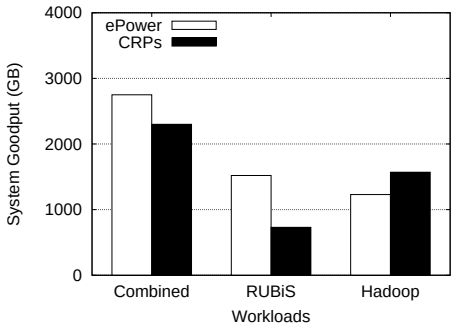


Fig. 11. Comparison between ePower and CRPs.

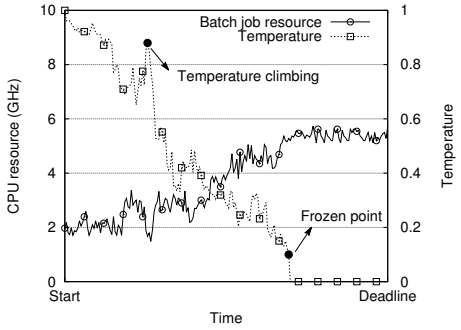


Fig. 13. Job temperature and resource.

7.2.2 Dynamic Resource Provisioning

While ePower controls the total power consumption, it also dynamically partitions the resources between transactional and batch workloads so as to optimize system goodput. Figures 12(b) and 12(c) show the dynamic resource allocations between RUBiS and Hadoop on day A and B, respectively. The Y-axis CPU allocation is that the sum total value of all the frequencies allocated to the two workloads. Note that the two different resource allocations for Hadoop and RUBiS workloads are not overlapped in Figures 12(b) and 12(c). The slash area represents the resource allocated to Hadoop workload and the grid area represents the resource allocated to RUBiS workload. Between the 6_{th} and 18_{th} hours on both days, when the RUBiS workload and the power supply increased, ePower allocated more resources to RUBiS to satisfy its demand and boosted overall system gooput. When RUBiS workload decreased but the power supply was still relative high, e.g., the first six-hour period and the last six-hour period on day B, ePower compensated Hadoop by allocating more resources. When power supply was scarce and RUBiS workload was relatively high, e.g., the last six-hour period on day A, ePower allowed the competition between RUBiS and Hadoop that led to the optimal system goodput.

7.3 QoS-aware Resource Migration

By design, ePower prioritizes transactional workloads as such workloads contribute more to system goodput than batch jobs contribute (as shown in Figure 3). However, batch jobs should be compensated when power supply is sufficient or their deadlines are approaching. The temperature in ePower's SA algorithm determines the resource migration between transactional and batch workloads. Figure 13 plots the resource allocation of a Hadoop job (the job running at the 18th hour in Figure 12(c)), against the cooling temperature on day A. We can see that ePower initially assigned a minimal amount of resource to the Hadoop job when the temperature was high. As Hadoop job's deadline was approaching, the temperature dropped quadratically and resources were migrated to the Hadoop job accordingly. When the power supply increased, the temperature climbed for a short period resulting in more resources being allocated to the transactional workload. After that the approaching Hadoop deadline became the dominant factor in setting the cooling temperature so that resources were migrated to Hadoop to avoid QoS violations. In ePower, the resource migration between two heterogeneous workloads is done by changing their vCPU capacities, rather than migrating processes across their VMs.

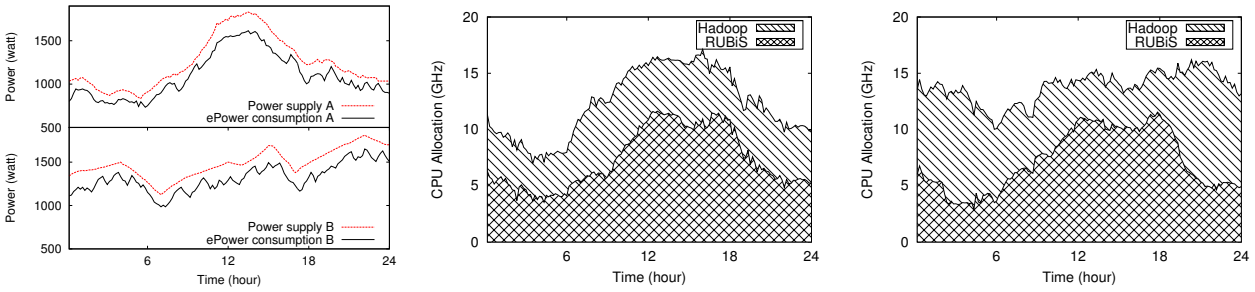
7.4 Overhead and Scalability

We conduct overhead and scalability analysis of the ePower approach. The overhead mainly comes from two sources: (1) The time required to perform the optimization algorithm in each control interval; (2) The time required to activate control adjustment (i.e., dynamic resource allocations) according to the algorithm. Our results show that the overhead of the algorithm is between 30 ms to 40 ms, which is relative small compared with the 10-minute control interval. Then we measure the activation overhead in each control interval. ePower took on average of 2.4 seconds to activate resource allocation adjustments for the VMs in the resource pool. The overhead is also negligible compared to the 10-minute control interval. The overhead of the optimization algorithm is quite stable (i.e., 30-40 ms) when the testbed grows, since the algorithm itself is independent of the testbed size. The overhead of the resource allocation is increased slightly by 4% when the overall resource pool size is increased by 10 times. Thus, ePower approach is scalable and applicable to larger scale systems. Appendix C shows the accuracy of the self-learning fuzzy performance modeling used in ePower.

8 Related Work

Power management in datacenters is an important and challenging research area. It is a research trend that power efficiency and application performance control are jointly tackled in virtualized datacenter servers [9], [10], [18], [28]. Those previous studies consider the power-stable situation when the traditional electrical grid is used. They did not consider the situation when the power supply is not stable.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2428695, IEEE Transactions on Computers



(a) ePower dynamic power control. (b) ePower resource allocations on day A. (c) ePower resource allocations on day B.

Fig. 12. Self-adaptiveness of ePower control in power consumption and resource allocation in two one-day scenarios.

Recently, a number of studies aim to achieve sustainable operation by renewable energy supply in green datacenters [29], [27], [30]. Stewart *et al.*, proposed renewable energy management approaches to maximize the use of off-grid renewable energy in datacenters [29]. Wang *et al.*, presented a theoretical framework for capturing important characteristics of different energy storage technologies, the trade-offs of placing them at different levels of the power hierarchy, and quantifying the resulting cost-benefit trade-offs as a function of workload properties [30]. Those studies focused on the renewable energy supply side of datacenters.

Some other studies focus on the power demand of green datacenters by workload management and resource provisioning of applications [3], [5], [19], [31]. Zhang *et al.*, proposed GreenWare, a novel middleware that dynamically dispatch requests to maximize the percentage of renewable energy used to power a network of distributed data centers, subject to the desired cost budget of the Internet service operator [25]. The work only considers Internet transactional workloads.

Goiri *et al.*, proposed GreenHadoop [19], an interesting MapReduce framework for a datacenter powered by a solar array and the backup electrical grid. They investigated how to manage a datacenter's computational workload to match the green energy supply. GreenHadoop predicts the amount of solar energy that would be available in the near future and schedules the MapReduce jobs to maximize the green energy consumption within the jobs' completion time bounds. It uses the electrical grid as backup to avoid time bound violations. The work only considers MapReduce workloads.

There are studies on the resource management of heterogeneous workloads (i.e., batch and transactional workloads) [6], [7], [21]. Carrera *et al.*, [6] proposed an important technique, Relative Performance Functions (RPFs), which allows integrated management of batch and transactional workloads. RPF for one application is a measure of the relative distance of the application's achieved performance from its goal. RPFs is used to make fair trade-offs between the different workloads. Zhan *et al.*, [7] proposed a Cooperative Resource Provisioning solution (CRPs) to decrease the peak resource consumption of workloads in a datacenter. It leverages the difference of heterogeneous workloads in terms of resource consumption characteristics and performance goals. Those works adopt QoS awareness but pay little attention to whether the power supply is stable.

Our recent work [32] focuses on the scenario of multiple self-sustainable datacenters and relies on the dynamic workload management. Instead, ePower focuses on elastic resource provisioning in a single datacenter.

9 Conclusions and Future Work

Resource provisioning of heterogeneous workloads is an important but challenging problem in datacenters. In this paper we focus on the problem in self-sustainable datacenters. We have proposed and developed an elastic power-aware resource provisioning approach (ePower) to improve the overall system goodput and control the system power consumption with respect to the dynamic green power supply. As demonstrated by modeling, optimization and experimental results based on the testbed implementation, its main contributions are near-to-optimal performance, resilience to dynamic power availability and improved system dependability. The main technical novelty of ePower lies in the developed power-aware simulated annealing based resource provisioning and self-learning fuzzy performance modeling techniques. ePower can significantly enhance the system performance of a self-sustainable datacenter by the elastic power-aware resource provisioning. Our future work will integrate weather prediction models with ePower and further explore sustainable cloud computing.

Acknowledgement

This research was supported in part by U.S. NSF CAREER award CNS-0844983, research grants CNS-1422119, CNS-1320122 and CNS-1217979, and NSF of China research grant 61328203. The authors are grateful to the anonymous reviewers for valuable suggestions.

References

[1] Various Green Datacenters. http://www.ecobusinesslinks.com/.
[2] I. Goiri, W. Katsak, K. Le, T. D. Nguyen, and R. Bianchini. Parasol and GreenSwitch: Managing Datacenters Powered by Renewable Energy*. In *Proc. ACM ASPLOS*, 2013.

[3] Z. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, Z. Wang, M. Marwah and C. Hyser. Renewable and cooling aware workload management for sustainable data centers. In *Proc. ACM SIGMETRICS*, 2012.

[4] R. Singh, D. Irwin, P. Shenoy, and K. Ramakrishnan. Yank: Enabling green data centers to pull the plug. In *Proc. USENIX NSDI*, 2013.

[5] B. Aksanli, J. Venkatesh, L. Zhang, and T. Rosing. Utilizing green energy prediction to schedule mixed batch and service jobs in data centers. In *Proc. USENIX HotPower*, 2011.

[6] D. Carrera, M. Steinder, I. Jordi Torres, and E. Ayguade. Autonomic placement of mixed batch and transactional workloads. *IEEE Trans. on Parallel and Distributed Systems*, 23(1), 2012.

[7] J. Zhan, L. Wang, X. Li, S. W., W. C., Z. W., and Z. X. Cost-aware cooperative resource provisioning for heterogeneous workloads in data centers. *IEEE Trans. on Computers*, 62(11), 2013.

[8] N. Mi, G. Casale, L. Cherkasova, and E. Smirni. Burstiness in multi-tier applications: Symptoms, causes, and new models. In *Proc. ACM/IFIP/USENIX Middleware*, 2008.

[9] R. Singh, U. Sharma, E. Cecchet, and P. Shenoy. Autonomic mixaware provisioning for non-stationary data center workloads. In *Proc. IEEE ICAC*, 2010.

[10] P. Lama and X. Zhou. Ninepin: Non-invasive and energy efficient performance isolation in virtualized servers. In *Proc. IEEE/IFIP DSN*, 2012.

[11] G. Urdaneta, G. Pierre, and M. van Steen. Wikipedia workload analysis for decentralized hosting. *Computer Networks*, 53(11), 2009.

[12] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica. Job scheduling for multi-user mapreduce clusters. *TR UCB/EECS-2009-55*, Berkeley, 2009.

[13] Measurement and instrumentation data center. http://www.nrel.gov/midc/.

[14] Y. Guo, P. Lama, and X. Zhou. Automated and agile server parameter tuning with learning and control. In *Proc. IEEE IPDPS*, 2012.

[15] V. Jalaparti, H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Bridging the Tenant-Provider Gap in Cloud Services. In *Proc. ACM SoCC*, 2012.

[16] J. Moses, R. Iyer, R. Illikkal, S. Srinivasan, and K. Aisopos. Shared resource monitoring and throughput optimization in cloud-computing datacenters. In *Proc. IEEE IPDPS*, 2011.

[17] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598) 1983.

[18] Q. Zhang, F. Mohamed, S. Zhang, Q. Zhu, B. Raouf, and L. Joseph. Dynamic energy-aware capacity provisioning for cloud computing environments. In *Proc. IEEE ICAC*, 2012.

[19] I. Goiri, K. Le, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini. Greenhadoop: Leveraging green energy in data-processing frameworks. In *Proc. ACM EuroSys*, 2012.

[20] G. Box, G. Jenkins, and G. Reinsel,. Time Series Analysis, Forecasting, and Control. *Prentice-Hall*, third edition, 1994.

[21] Y. Chen, S. Alspaugh, D. Borthakur, and R. Katz. Energy efficiency for large-scale mapreduce workloads with significant interactive analysis In *Proc. ACM EuroSys*, 2012.

[22] Y. Chen, B. Yang, A. Abraham, and L. Peng. Automatic design of hierarchical Takagi-Sugeno type fuzzy systems using evolutionary algorithms. *IEEE Trans. on Fuzzy Systems*, 15(3) 2007.

[23] A. Verma, L. Cherkasova, and R. H. Campbell. Resource provisioning framework for MapReduce jobs with performance goals. In *Proc. ACM/IFIP/USENIX Middleware*, 2011.

[24] B. J. Watson, M. Marwah, D. Gmach, Y. Chen, M. Arlitt, and Z. Wang. Probabilistic performance modeling of virtualized resource allocation. In *Proc. IEEE ICAC*, 2010.

[25] Y. Zhang, Y. Wang, and X. Wang. Greenware: Greening cloudscale data centers to maximize the use of renewable energy. In *Proc. ACM/IFIP/USENIX Middleware*, 2011.

[26] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy. Optimal power allocation in server farms. In *Proc. ACM SIGMETRICS*, 2009.

[27] D. Gmach, J. Rolia, C. Bash, Y. Chen, T. Christian, A. Shah, R. Sharma, and Z. Wang. Capacity planning and power management to exploit sustainable energy. In *Proc. IEEE CNSM*, 2010.

[28] Y. Wang, X. Wang, M. Chen, and X. Zhu. Partic: Power-aware response time control for virtualized web servers. *IEEE Trans. on Parallel and Distributed Systems*, 21(4) 2010.

[29] C. Stewart and S. K. Some joules are more precious than others: Managing renewable energy in the datacenter*. In *Proc. USENIX HotPower*, 2009.

[30] D. Wang, C. Ren, A. Sivasubramaniam, B. Urgaonkar, and H. K. Fathy. Energy storage in datacenters: What, where, and how much?. In *Proc. ACM SIGMETRICS*, 2012.

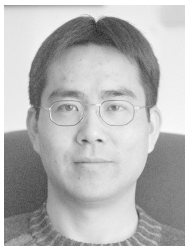
[31] C. Li, R. Zhou, and T. Li. Enabling distributed generation powered sustainable high-performance data center. In *Proc. IEEE HPCA*, 2013.

[32] D. Cheng, C. Jiang, and X. Zhou. Heterogeneity-aware workload placement and migration in distributed sustainable datacenters. In *Proc. IEEE IPDPS*, 2014.

Dazhao Cheng received his B.S. degree in Electronic Engineering from Hefei University of Technology, China, in 2006. He received his M.S. degree in Electronic Engineering from University of Science and Technology of China in 2009. Currently, he is working towards the Ph.D. degree in Computer Science at the University of Colorado, Colorado Springs. His research interests include sustainable cloud computing and autonomic resource management. He is a student member of the IEEE.

Jia Rao received his B.S. and M.S. degrees in Computer Science from Wuhan University in 2004 and 2006, respectively, and Ph.D. degree from Wayne State University in 2011. He is currently an Assistant Professor in the Department of Computer Science at the University of Colorado, Colorado Springs. His research interests include the areas of distributed systems, resource auto-configuration, machine learning and CPU scheduling on emerging multi-core systems. He is a member of the IEEE.

Changjun Jiang received the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 1995. Currently he is a Professor with the Department of Computer Science, Tongji University, Shanghai. He is also the Director of Professional Committee of Petri Net of China Computer Federation and the Vice Director of Professional Committee of Management Systems of China Automation Federation. His current areas of research are concurrent theory, Petri net, and formal verification of software, concurrency processing and intelligent transportation systems. He is a member of the IEEE.



Xiaobo Zhou obtained the BS, MS, and PhD degrees in Computer Science from Nanjing University, in 1994, 1997, and 2000, respectively. Currently he is a Professor and the Chair of the Department of Computer Science, University of Colorado, Colorado Springs. His research lies broadly in computer network systems, specifically, Cloud computing and datacenters, BigData parallel and distributed processing, autonomic and sustainable computing, scalable Internet services and architectures. He was a recipient of the NSF CAREER Award in 2009. He is a senior member of the IEEE.