GreenGear: Leveraging and Managing Server Heterogeneity for Improving Energy Efficiency in Green Data Centers

Xu Zhou[†*], Haoran Cai[*], Qiang Cao[†*§], Hong Jiang[‡], Lei Tian[§] and Changsheng Xie[†*]
[†]School of Computer, Huazhong University of Science and Technology, Wuhan, China
[*]Wuhan National Laboratory for Optoelectronics, Wuhan, China
[‡]Dept. of Computer Science and Engineering, University of Texas at Arlington, Arlington, Texas, USA
[§]Tintri, California, USA
[$]Corresponding Author: caoqiang@hust.edu.cn

ABSTRACT

In this paper, we propose *GreenGear*, the first heterogeneous strategy that incorporates wimpy servers into existing green data centers to dynamically deal with power mismatches. Our techniques exploit intelligent green power scheduling policies to provide efficiency-aware power management. We evaluate the *GreenGear* design on a prototype installed in a test-bed. Compared with a homogeneous server system, *GreenGear* is able to significantly increase the effective use of the renewable and battery power sources without the supplement of grid power, extending their runtime by 57%, lengthening the UPS lifetime by 2.04X, and improving renewable energy utilization by 51%.

Categories and Subject Descriptors

C.0 [**Computer Systems Organization**]: General

Keywords

Server Heterogeneity; Power Management; Battery; Renewable Energy

1. INTRODUCTION

The power consumption in data centers is experiencing an alarming increase. It is projected that the total power consumption of cloud computing infrastructures will reach 1700TWh in 2030, which is more than double such consumption in 2011 [1]. As a result, the enormous power demands not only significantly raise the total cost of ownership (TCO) of data centers but also impose profound burden on the environment. For example, the annual carbon emissions of data centers will approach 1.54 metric billion tons, which will cause the IT industry to be the largest greenhouse gas emitter by 2020 [2, 3]. Consequently, both industry and academia have been exploring ways to leverage renewable energy to deal with the considerable electricity cost and environmental issues.

However, efficiently matching power demand with power supply over time is one of the biggest challenges for building green data centers with the clean energy that is naturally fluctuating and intermittent. At any time when the power provisioning in a green data center cannot appropriately match power demands under varied workloads, a power mismatch occurs. The consequence of power mismatches is that either extra energy supplies from green power, batteries, or backup power grid have to be introduced to overprovision the power supply, or green power is severely underutilized or wasted. But the power demand from servers in data centers are actually overestimated due to the fact that typical servers in data centers, referred to in this paper as *beefy servers* since they are generally powerful but energy-inefficient computing machines, are known to exhibit power inefficiency, e.g., even when they are idle they draw up to 60% their peak power [4]. Unfortunately, data centers generally spend considerable portions of their time operating at low to moderate load levels. For example, the average server utilization is usually between 10% and 50% according to a prior study [5]. As shown in Figure 1, even when the actual load varies widely during the span of a day, the power demand of the beefy servers remains relatively stable with much less variance than the actual load. As a result, even during low-load periods these servers still draw a large amount of power. In green data centers, such high power demands during long periods of low to moderate loads result in significantly excessive demand of renewable energy or other power sources. It is a serious and challenging problem facing today's green data centers.

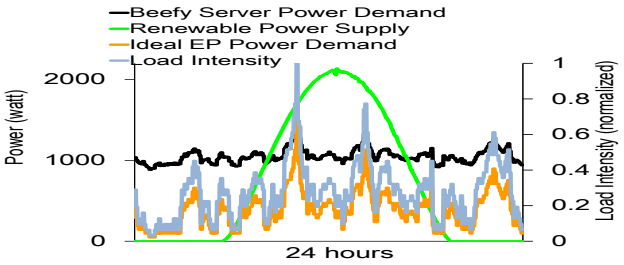


Figure 1: Ideal EP power demand represents the power demand of an idealized full system energy-proportional server. Beefy server power demand is the measured power from our test-bed (detailed in Section 6). The diurnal workload pattern comes from a study of Google data centers [6]. The difference in power requirement between Ideal EP and Beefy server, particularly during low to moderate loads, leads to significant renewable energy wastage.

ICS '16, June 01-03, 2016, Istanbul, Turkey

DOI: http://dx.doi.org/10.1145/2925426.2926272

Existing techniques addressing the power mismatch problem mainly focus on adapting the workload power demand to tracking the renewable power supply, which can be broadly divided into two categories: (1) power tuning by leveraging the active processor power states [7, 3, 8], and (2) load power demand shaping by consolidation [9, 10, 11, 12]. The first category adapts the load power demand to the time-varying green power through changing processor power demand in the servers via techniques such as DVFS. However, since no single component in the server dominates the power demand, e.g., processors merely account for 25% power consumption in modern servers [4] while other server components incur fixed amounts of power consumption when active, the effect on reducing power mismatch by this approach is rather limited. The second category tracks the renewable power supply to consolidate the load onto a fraction of servers, turning the rest off. Unfortunately, this approach is only feasible for throughput-oriented batch workloads. Some interactive services, such as web-search, social networking, and software as a service (SaaS), are not suitable, due to the data availability concerns stemming from the distributed datasets that can be rendered unavailable or incur expensive data migration costs by consolidation [13, 14]. These interactive services also require strict service level objectives (SLOs), that are in the range of a few milliseconds or even hundreds of microseconds [13], which may not be readily achievable since the application state cannot fit in a small number of consolidated servers.

Recently, energy storage devices (e.g., batteries) have been widely used to offer stable power supply to smooth the power mismatches [6, 15, 16, 17, 18] for performance considerations. More importantly, without effective power management at low to moderate loads, severe power mismatches can lead to several problems: (1) frequent and excessive battery discharging activities, which will compromise the lifetime of these expensive devices, (2) high discharge current at the low to moderate load levels reduces the available energy capacity owing to the Peukert's law effect [17], (3) the ill-matched power demand at the low to moderate load levels will quickly deplete the stored energy that is critical for power demands for peak loads. Therefore, *can we find an alternative solution to the power mismatch problem while meeting the SLO target and avoiding the constraints on batteries?*

To this end, we propose a fundamentally different renewable energy-aware power management scheme *GreenGear*, that fully leverages the nature of server heterogeneity in data centers, and then exploits the synergy between a heterogeneous computing platform and renewable energy system in data centers. Heterogeneity in data centers emerges in part because servers are gradually deployed and upgraded over the typical 12 year lifetime of a data center [19, 20, 21], and in part because both industry and academia have recently started to build servers with less powerful but more energy-efficient processors, referred to in this paper as *wimpy processors*, that were originally designed for mobile and embedded platforms [22, 23].

More specifically, we employ the *wimpy* computing node with low-end CPUs and low-power components integrated with the conventional *beefy* computing node to provide an energy-efficient complement to address the problem of unexpected power mismatch by judiciously adapting the right power source to the right type of computing nodes. The advantages of this heterogeneity-based strategy are threefold. First, wimpy servers are shown to have high-energy efficiency, in particular at low power demand with low and fixed power overheads for the server subsystems. Second, it helps extend the battery discharging duration. And third, it enables the storing of surplus renewable energy into the batteries during the periods of low to moderate loads when green power generation is abundant. The key to the design and implementation of this strategy is an intelligent power management scheme that achieves high efficiency of the renewable energy. For a given power mismatch scenario, there exists an optimal scheduling of battery charging and discharging activities to sufficiently store the surplus renewable energy into batteries during the periods of green power abundance to allow for the longest possible discharging duration during periods of insufficient green power.

In this paper, we make the following contributions: (1) We propose several metrics to quantify power mismatch and renewable energy utilization and to identify root cause to inefficient use of renewable energy during the low to moderate load periods. By comparing the energy efficiency and request latency, we also demonstrate the feasibility of the renewable energy driven design that incorporates wimpy servers as an integral part into a heterogeneous system in data centers. (2) We present *GreenGear*, a heterogeneity-based renewable energy driven architecture that enables data centers to optimize the overall power match and renewable energy utilization and to increase the battery lifetime. As a cross-layer power management scheme, *GreenGear* resides between the front-end renewable energy generation system and the back-end heterogeneous computing system to provide a coordinated management between the green power supply and the workload power demands. The architecture of *GreenGear* is based on a flexible provision scheme for the renewable power and battery power systems that is convenient to scale and configure. (3) We propose a customized renewable energy scheduler that can dynamically and automatically optimize the green power distribution and tune the workload power demand subject to the SLO requirement in the *GreenGear* architecture. (4) We evaluate *GreenGear* in a scaled-down experimental platform, which allows us to gain important insights into the relationships among a range of workload traffic, workload shift points, and SLO constraints. Driven by representative data center benchmarks, our prototype evaluation shows that *GreenGear* can achieve 100% renewable energy utilization without the backup of utility grid power while prolonging the effective runtime of renewable and battery systems by 57%, improving renewable energy utilization by 51% and extending the battery lifetime by 2.04X on average.

2. ANALYSIS OF POWER MISMATCH

To analyze the impact of power mismatch, we evaluate the power demand scenarios of a cluster with the workload traffic from a Google cluster [6] in a 24-hour span, as mentioned in Figure 1. To understand the challenges of handling power mismatch, we collect measurement full system power data from a cluster running the SPECjbb benchmark [24]. We model a 12V 6-cell value-regulated lead-acid battery to investigate the battery discharging/charging activities and the battery energy efficiency as reported in [3].

We use the power mismatch index f_t proposed in [25] to describe the synergy between load power demand and renewable power supply for each time slot t.

$$f_t = \frac{GreenPower_t + BattGreen_t}{LoadPower_t} \quad (1)$$

where $GreenPower$ denotes the renewable power generation, $BattGreen$ denotes the green power supply from battery and $LoadPower$ is the load power demand. Obviously, power mismatch is minimum (zero) when $f_t = 1$. In other words, ideal power match for green data centers is achieved when the renewable power and/or battery can exclusively power the workloads, i.e., $GreenPower + BattGreen = LoadPower$. However, f_t only accounts for power usage and cannot directly reflect the power mismatch variations across different load power demand and green power gener-

ation levels. Thus, we define the power match ratio (PMR):

$$PMR = \frac{\sum t(f_t = 1)}{\sum T} \quad (2)$$

Where $\sum t(f_t = 1)$ is the total amount of time during which power match is achieved by the green energy from renewables and/or batteries. $\sum T$ is the total workload execution time. Clearly, the maximum value of PMR is 1. The higher the PMR value is, the longer time the workloads are powered entirely by the renewable and battery power without the supplement of the utility power.

Figure 2 illustrates three power mismatch regions based on the power mismatch metric f_t, namely, the surplus renewable power generation region, the insufficient renewable power generation region and the intermittent renewable power outage region.

Mismatch Region I: Surplus Generation of Green Power. During the period when there is an abundance of green power output, power mismatch occurs since green power supply is greater than the load power demand. In this region, not only can the green power independently power the workloads, but it can charge the battery to store the surplus green power. In this case, $GreenPower_t > LoadPower_t$, $BattStore = \sum_t^{T_I}(GreenPower_t - LoadPower_t) \times \delta t$, where δt denotes the duration for each time-slot t and T_I denotes the duration for Region I. On the other hand, the power demand $LoadPower$ determines how much of the green energy can be stored in the battery.

Mismatch Region II: Insufficient Generation of Green Power. During the period when the green power generation is low, the renewable power can be complemented by the battery to handle the power mismatch: $LoadPower_t = GreenPower_t + BattGreen_t$. The power demand $LoadPower$ will determine the amounts of the direct usage of green power and the complementary power in the battery ($BattStore$). The available green energy from the battery will be constrained by the battery energy capacity and the stored clean energy in Region I. In case of a battery energy shortage, it will force the use of the utility grid to power the workloads while the $GreenPower$ can only be used to charge the battery.

Mismatch Region III: Outage of Green Power. During the period when the green power source experiences power outages, the load power demand can only be supplied by the battery system, where a power mismatch can only be handled when $LoadPower_t = BattGreen_t$. In this case, the power demand for the battery can be greater than that in Region II.

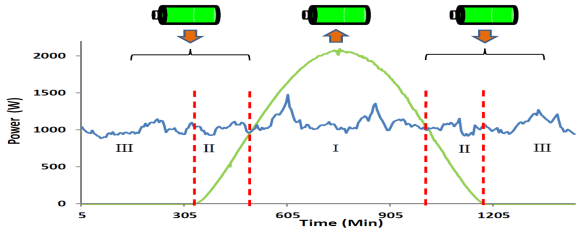


Figure 2: Power mismatch regions.

Figure 3 shows the diurnal load variation for SPECjbb on an example beefy server cluster in the event of green power supply. We also plot the power draw of two ideal energy efficient systems, *Ideal EP* and *Dynamic EP*. The *Dynamic EP* in Figure 3 is defined such that the power demand at 100% load (peak load) is 100% of peak power of the cluster, the power demand at 0% load is the idle power of the cluster, and all power demands have been linearly interpolated between the two points [13]. Note that, *Dynamic EP* represents the power demand of a cluster assuming that the idle power always exists and is often employed to model the server power consumption [26]. The *Ideal EP* system, as introduced earlier, is another perfect hypothetical system that has ideal energy proportionality with zero idle power. These two systems are used to describe the theoretical lower bounds of power consumption in this analysis. From this analysis we draw several interesting observations. First, the renewable energy exhibits the most efficient power consumption at high loads. This is because high loads lead to high server utilization, meaning that supplying renewable energy to peak power demands can result in the best energy efficiency. Second, the cluster exhibits energy inefficiency during the periods of low to moderate loads. Ideally, power supply from the renewable power system and battery should closely track workload power demand, especially at low loads. However, the actual beefy server power draw is disproportional large during low to moderate loads: at 20% load the cluster draws 70% of its peak power. This is because of the fixed power consumed by various components of the server system, such as fans, power supplies, chipset,

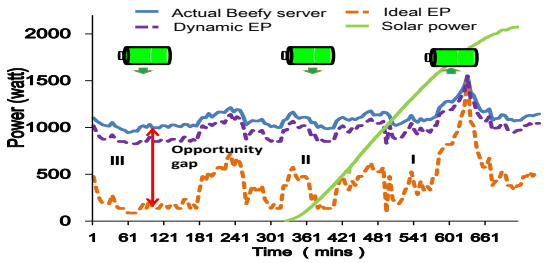


Figure 3: An example of power draw under the diurnal load for an actual beefy server cluster. For an ideal, hypothetical energy proportional cluster (Ideal EP), power draw would perfectly track load (RPS, requests per second). Dynamic EP indicates the hypothetical amount of power demand where the idle power is actually measured and assumed to be constant while the active power is ideal energy proportional.

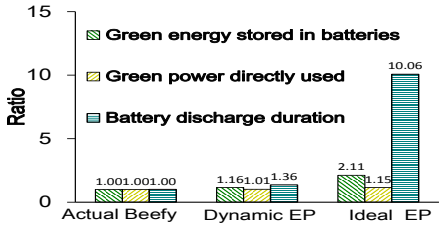


Figure 4: Comparison of the Actual Beefy, Ideal EP and Dynamic EP systems (normalized to Actual Beefy system). The Ideal EP system can improve the green energy state of charge to 2.11X , improve the green power direct usage to 1.15X and improve the battery discharging duration to 10X.

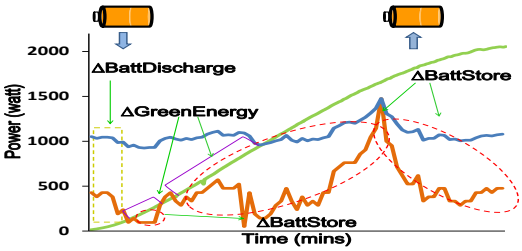


Figure 5: Explanation for high green energy efficiency achieved by an Ideal EP cluster.

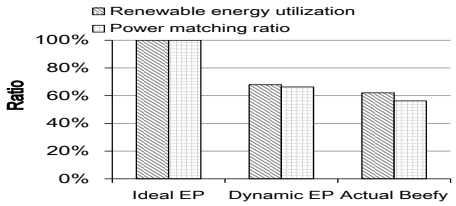
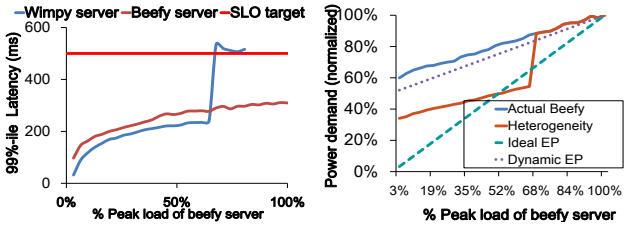


Figure 6: Comparison of the Actual Beefy, Ideal EP and Dynamic EP systems in terms of power match ratio (PMR) and renewable energy utilization (REU).



(a) Latency result for the wimpy server.

(b) Power demand for the wimpy-beefy pair.

Figure 7: Characterization of latency and power demand of SPECjbb for the wimpy server at various load intensities.

voltage converters, network components, etc. Consequently, even though the power mismatch can be handled by the green power, the energy inefficiency at the lower loads for the current beefy server still prevents the renewable energy from being fully utilized to service the current workload requests. To reduce the unnecessary power demand, we focus on addressing idle power draw during the periods of low to moderate loads. This is the opportunity gap that we believe should be explored to improve the effective usage of the green energy.

In fact, besides the energy inefficiency from the power demand side, the high power demands at low to moderate loads can cause the renewable energy to be inefficiently used in the event of varied renewable power supply. In Region I of power mismatch with surplus renewable energy, the high power demands at low to moderate loads also limit the amount of green energy that can be actually stored in the battery (e.g., state of charge). In Region II with insufficient renewable energy, there exists an efficiency threshold for the green power supply, that is, the renewable power can only be utilized when the renewable power supply is equal to or larger than the load power demand ($GreenPower_t + BattGreen_t \leq LoadPower_t$). In this case, high power demands can result in wastage of part of the renewable energy without directly powering the workloads - servers cannot work when the current green power is less than the workload power demand. Even if the insufficient green power can be stored in the battery, the battery still incurs energy-inefficient due to the round-trip energy loss (15%-25%) [9]. In addition, Peukert's law implies that the heavy power burden during the periods of low to moderate loads will cause the battery energy to sink heavily since large discharging current will cause the usable capacity of a battery to decrease exponentially. In Region III with intermittent power outages, the servers entirely depend on the battery provisioning to power the workload so that they need a large energy capacity to handle the power mismatch. These observations clearly indicate that the excessive power demand at low to moderate loads is a major cause for the power mismatch problem.

Figure 4 compares the green energy efficiency of the Actual Beefy, *Dynamic EP* and *Ideal EP* clusters in different renewable power usage and battery activity scenarios. We explain the reason why *Ideal EP* can significantly improve the green energy efficiency when the green energy is utilized as shown in Figure 5. Comparing with the *Ideal EP* server system, we find that (1) more renewable power (denoted by $\Delta(GreenEnergy)$) can be directly used by the efficient server system than the actual beefy cluster while avoiding charging the green power into battery, thereby reducing the energy loss, (2) the *Ideal EP* cluster can help save more green energy in battery, denoted by $\Delta(BattStore)$, during the surplus Region I, which is crucial for Regions II and III, (3) the *Ideal EP* system can lessen the power burden on battery in Regions II and III, which can prolong the battery discharging duration (denoted as $\Delta(BattDischarge)$). As shown in Figure 6, when comparing to *Ideal EP* and *Dynamic EP*, we observe that after decreasing the power demands for low to moderate loads, one can effectively handle the power mismatch issue. Therefore, high energy efficiency under low to moderate loads will lessen dependency on utility power grid and large energy backup storage (e.g., batteries) to fix the demand-supply power mismatch problem.

Based on the analysis above, we introduce a novel metric to measure the renewable energy utilization (REU) to determine how much renewable energy can be utilized by the load requests. We define in the REU metric as:

$$REU = \frac{\sum_t^T Load(f_t = 1)}{\sum_t^T Load(t)} \quad (3)$$

where $\sum Load(f_t = 1)$ is the renewable energy that can be directly used to serve the workloads (a.k.a., green energy translated to throughput), $\sum Load$ is the total load demands of the IT system (a.k.a., total amount of requests or queries). When the renewable energy can exclusively power the workloads ($\sum Load(f_t = 1) = \sum Load$), $REU = 1$. A higher REU indicates higher renewable energy efficiency, which means that more green energy can be effectively used to power the workloads. Note that, the REU definition introduced in this study can directly reflect the effective renewable energy usage from both the power demand and supply sides. Figure 6 shows the renewable energy utilization given various power demands for low to moderate loads based on the Google workload patterns we mentioned earlier. The results show that the high power demands at low to moderate loads significantly decrease the renewable energy utilization. Hence, in order to solve the power mismatch problem, we cannot solely rely on exploiting new technologies/devices for renewable power and energy storage systems to provide sufficient power supply [17, 16], we must also improve how the renewable energy is used at the power demand side due to the dynamic nature of workloads and the fact of power inefficiency at low to moderate loads [5].

3. OPPORTUNITIES AND CHALLENGES OF EXPLOITING SERVER HETEROGENEITY

3.1 Performance and Energy Characterization of Wimpy Server

We built a test-bed that consists of several pairs of wimpy server (Intel Core-i5 based server) and beefy server (Xeon E5-2620 based server). Our experimental test-bed allows us to characterize possible benefits of employing wimpy servers powered by renewable energy and battery systems.

Latency and Power Demand Analysis: Before we discuss energy efficiency of the wimpy server, we examine

the capability limit of a wimpy server at a given load to find the point that satisfies the SLO target. We define the capability of a wimpy server as the fraction of throughput that the wimpy server can provide compared to a beefy server constrained by the latency requirement.

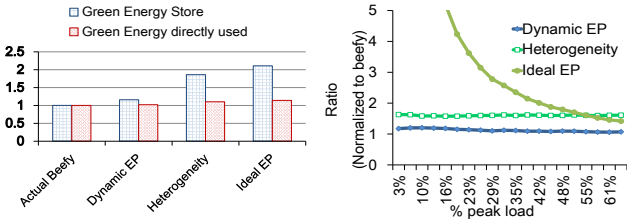
In this study we specify the target SLO for SPECjbb as 500ms (99%-ile tail latency) as prescribed in [24]. Note that the SLO target for requests can be defined at a fixed value as detailed in [13]. Figure 7(a) presents the 99%-ile tail latency achieved by the wimpy server and beefy server at different load levels for SPECjbb respectively. As shown in this figure, with low or moderate loads, the performance difference in 99%-ile tail latency between the wimpy server and beefy server is negligible. As the workload intensity increases, the latency on the wimpy server increases rapidly, and the latency gap between the wimpy and beefy servers becomes much wider. Considering the 99%-ile latency at 500ms as the SLO target, the wimpy server can sustain up to 64.5% of the peak load of the beefy server. Even for higher loads (60% or higher of the peak load), a tight latency headroom can be translated into power reduction. The wimpy server can help reduce more than 43% power as shown in Figure 7(b). Another interesting observation from Figure 7(b) is that the wimpy server can save a significant amount of power even when the strict tail latency (99%-ile) is enforced on. Comparing to the ideal energy-proportional server (*Ideal EP*), the wimpy server leads to lower power demands at loads between 52% - 64.5% of the peak load.

The experimental results show that at low loads the latency delivered by the wimpy server is not only satisfactory but, more importantly, at a much lower power cost, making the wimpy server a desirable alternative to the beefy server at low to moderate loads. Motivated by these observations and the combined need for high performance at high load and energy efficiency at low to moderate loads, we exploit server heterogeneity in our *GreenGear* design.

Efficiency Analysis of Green Energy: Figure 8(a) illustrates how much different server types use and store green energy in the event of power mismatches as shown in Figure 3. Figure 8(b) compares battery discharge with a heterogeneous wimpy-beefy server pair, the beefy-only server and the *Dynamic EP* server, which reflect different power demands at low to moderate loads respectively. Given the low power demand, the wimpy server can significantly increase the renewable energy efficiency. Compared to the beefy server, the heterogeneous server can help save more energy in batteries (e.g., 86% improvement) during the surplus Region I while directly utilizing more green energy (e.g., 10% improvement) (recall Section 2). In addition, even during the outages of green power generation, the wimpy server can lessen the burden on batteries while delivering a longer discharge duration than the beefy servers (e.g., around 60% improvement) as shown in Figure 8(b). Heterogeneity design always outperforms the *Dynamic EP* server. Moreover, as shown in Figure 8(b) when load intensity is higher than 52% of peak load, the wimpy server also results in a longer battery discharge duration than *Ideal EP* server. Furthermore, deploying the wimpy server can also lead to higher utilization of green energy due to its lower power demand than the *Ideal EP* server because the battery discharge activity is dominated by the idle power in this case.

3.2 Challenges of Adopting Heterogeneity

Based on the performance and energy efficiency characterization analyzed above, we believe that it will be greatly beneficial to leverage the wimpy servers in green data centers to efficiently utilize the renewable energy and battery systems. However, challenges arise when directly adopting such heterogeneous architectures in data centers without an effective power scheduling of renewable energy and batteries. Figure 9 shows the power match ratio (PMR, recall Section 2) results for different green energy assignment scenarios from renewables and batteries. In this experiment, we vary the green energy assignment to wimpy servers and beefy servers to measure the maximum power match ratio (PMR) with a constant workload power demand and a fixed renewable power generation. The WimpyFirst or BeefyFirst mechanism always chooses to assign the green energy from renewables and batteries to the wimpy server or the beefy server from a server pair first respectively, and then powers the other one if the green energy is sufficient. As shown in the figure, there exists an optimal green energy assignment that can provide the highest PMR and decrease utility energy requirement. For example, by heavily assigning green energy to wimpy servers (e.g., WimpyFirst), the power match ratio can be decreased by 14% on average from the optimal. If we ignore the potential of wimpy servers (e.g., BeefyFirst), the power match ratio will be reduced by 18%. The challenge facing the power source scheduling is that there is no changeless optimal green energy distributed point. This is because the desirable power match point not only depends on the current power generation of the renewable energy system and the current energy capacity of batteries but also relies on the fluctuated workload dynamics and load levels, as well as the power demand and processing capability of the wimpy server. Therefore, we have to dynamically find the optimal point at runtime to schedule appropriate green power from the renewable energy system and/or batteries based on the heterogeneous computing architecture upon a demand-supply mismatch scenario.



(a) Green energy store and direct use comparison. (b) Battery discharge duration at various load intensities.

Figure 8: Efficiency comparison with actual beefy, Dynamic EP, heterogeneity and Ideal EP server.

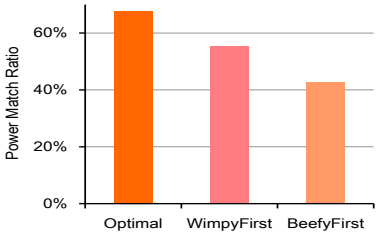


Figure 9: Power match ratio (PMR) under different scheduling schemes of power sources for SPECjbb based on 40% depth of discharge (DoD) of batteries.

4. GREENGEAR ARCHITECTURE

4.1 Composite Server System

As shown in Figure 10, we design a *composite server* that logically consists of a wimpy server and a beefy server. The wimpy or beefy server has its own independent memory, CPU and motherboard, and can be powered on and off independently. The beefy-wimpy combination in each com-

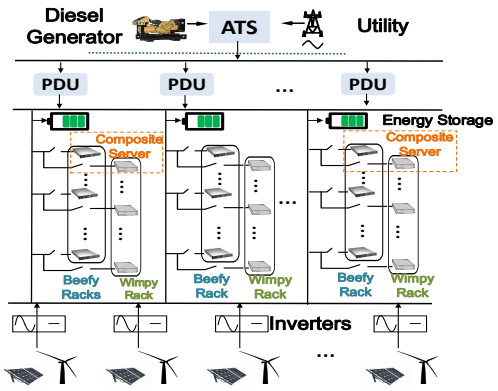


Figure 10: GreenGear System Architecture

posite server enables two *power gears* (a.k.a., wimpy gear and beefy gear) at the sever level. This design results in two energy efficient operating regions for renewable energy. Each server within a composite server, wimpy or beefy, has its individual IP address to send messages to wake up the counterpart. There is always only one server within a composite server being kept on-duty and this active server is capable of remotely waking up the other one, by using the Wake-on-LAN technique. Whenever the wimpy gear is estimated to be incapable of meeting the SLO target, the beefy gear will take over the loads immediately. Reversely, as the load level drops to a certain threshold also determined by the SLO, the beefy gear will be switched to the wimpy gear. In fact, the inactive power gear can potentially be used to service other types of workload such batch or scientific computation. However, the consideration of mixed workloads in the composite server is beyond the scope of this work. Note that the server-level heterogeneity can help reduce the switching overhead (a.k.a., duration for waking up) due to the server on/off operation. When we employ a set of wimpy servers composed of a set of beefy servers to create a heterogeneous cluster-level composite, the switching operation will be more frequent and in turn impact the performance. This may be considered a performance cost traded for the creation of more energy efficient regions for the low to moderate loads.

A crucial part of our composite server design is the choice of the wimpy server. Our choice is based on two considerations: (1) power demand and (2) service capability. A reasonable wimpy server capability is to be able to tolerate severe load level fluctuations to avoid highly frequent inter-server switching that may offset the benefits of the wimpy server. Moreover, latency-critical workloads often require large memory capacity to buffer the working set to reduce the disk I/Os. Hence, such wimpy server candidates as Atom-based or ARM-based systems with low memory capacity [14] would not be suitable for *GreenGear*. In the meanwhile, while heterogeneous cores based systems, such as the Intel QuickIA platform [27], may be a reasonable choice, they will not be able to achieve the same energy efficiency as the server-level heterogeneous composite can because of the sharing of the memory and subsystems among the heterogeneous cores of the former. On the contrary, the beefy gear and wimpy gear in our composite server are designed to share nothing in order to originate separate efficient regions.

4.2 Distributed Energy System

Renewable Power System Provision: As shown in Figure 10, *GreenGear* connects renewable power system at power distribution unit (PDU) level to act as distributed power systems. The PDU-level renewable power integration allows us to flexibly scale the green power capacity on a per-rack basis. *GreenGear* does not synchronize the green power with the utility power since the renewable power often challenges the stability of the utility power. Since the composite server system is able to lower the power demand and the energy requirement for the renewable power system, *GreenGear* explores a "pay-as-you-use" model - provisioning the renewable power capacity by tracking the energy demand of the composite servers. Moreover, since data centers routinely face load imbalance – the hotness of the index shard for latency-critical workloads will likely result in severe fluctuation in load intensity in the cluster [13]. In case of load imbalance, deploying the composite system may not necessarily maintain power match all the time, due to the limited capability of wimpy server. Hence, an intuitive way to achieve a desirable renewable energy utilization is increasing the renewable power capacity for the racks with hot nodes. However, the existing uniform centralized power capacity provision [10, 28, 11, 29] mechanism forces all servers to obtain the same or similar renewable power capacity with or without hotness index shard, significantly impacting the realized power match. The renewable energy utilization of the entire data center will be constrained by the hot racks that cannot obtain a suitable level of green power supply. When the centralized power capacity integration mechanism is deployed for the entire data center level energy demand, the PDU will become the power delivery bottleneck. In addition, today's data centers typically under-provision the power capacity such as the PDU [30, 15], thereby further exacerbating the power delivery bottleneck. Simply increasing renewable power capacity at data center level does not necessarily ensure that the required increase in green power budget for the server racks since their related PDUs may have already reached their capacity limit. Therefore, we are convinced that the distributed power integration that supports fine-grained green power provision is the right power provision mechanism for *GreenGear*.

Distributed Energy Storage System: At present, Google and Facebook have employed distributed energy storage to avoid energy loss due to double conversion in conventional centralized Uninterrupted Power Supply (UPS) systems [16]. This decentralized design can also avoid single-point of failure and increase overall data center power availability. Moreover, the distributed battery topology has also been further used for peak power shaving to reduce grid power capacity requirement [15, 17, 6, 30]. The shortcoming of centralized UPS system is that, it cannot offer a fraction amount of the energy - the data center load power demand has to be shifted between the power supply (e.g., green or grid) and UPS in its entirely. Hence, the batteries have to undergo much larger discharge current that significantly decreases battery lifetime. Moreover the centralized UPS battery system lies on the critical power path between the Automatic Transfer Switch (ATS) and the PDU. When UPS is required to manage the power mismatch, it supports power transferring for the entire data center but it cannot deal with the power mismatch in a fine-grained way. Therefore, we leverage distributed battery for regulating the power mismatch and improving the renewable energy utilization.

Because both the beefy and wimpy gears can selectively utilize the renewable, battery and grid sources, as shown in Figure 10, the *GreenGear* design has three unique capabilities. First, it ensures power stability while avoiding brownout due to the intermittent nature of renewable power. Second, it allows both gears to operate with direct green power supply while reducing the dependency on batteries. And third, it enables the beefy and wimpy gears to share the battery energy capacity and renewable power capacity, thereby reducing the battery energy capacity and renewable power capacity originally dedicated to the wimpy racks.

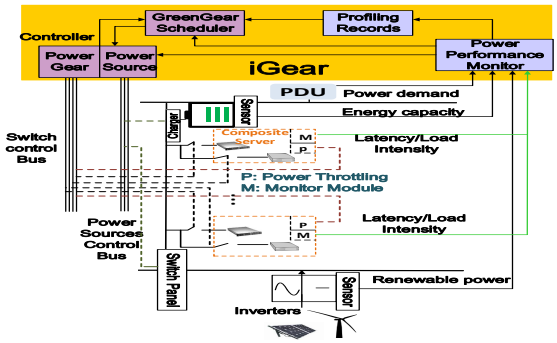


Figure 11: A Rack-level View of the GreenGear Control Architecture.

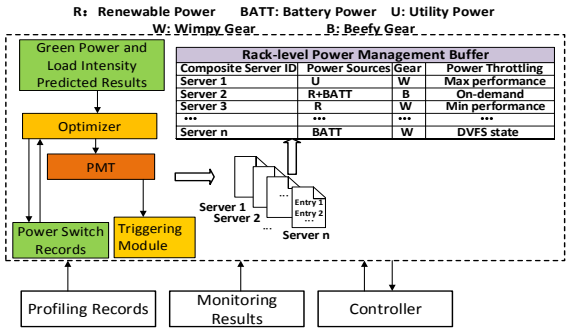


Figure 12: GreenGear scheduler.

4.3 Control Architecture

To handle the power mismatch, *GreenGear* dynamically schedules the renewable energy and batteries to power the composite server farm while judiciously allocating workloads in the composite servers. The supply/demand variability makes the control challenging for two reasons. First, from the supply side, the scheduling of power sources must take the renewable power variations into consideration. Second, from the demand side, the wimpy gear should be on duty for as long as possible while avoiding any SLO violation induced by workload fluctuation.

Figure 11 shows a rack-level view of *GreenGear* control architecture *iGear*. In most scenarios, servers can be powered by three power sources: renewable, utility grid and battery. The *GreenGear* controller (*iGear*) is a key decision-making module that determines when to switch among these power sources and switch between the wimpy and beefy gears. The power source controller makes the switching decision based on the discrepancy between the load power demands and green power supply. The power gear switch is triggered based on the SLO requirement and also driven by the power supply of the renewables and battery. The measurement of discharge current of the battery and the available power of renewable power system gathered from the sensors is transmitted to the *GreenGear* scheduler (Section 5). The rack-level power controller gathers the switching outcomes (i.e., performance and power results) of each composite server load and further feeds this information to the *GreenGear* scheduler for optimizations. With the feedback on server power demands and available green power, *GreenGear* schedules each power source to distribute a suitable amount of power to the appropriate power gear through a control bus. The renewable power can charge the battery when the load power demands are predicted to be lower than the renewable power generation. In addition, the grid power can also charge the battery whenever the renewable power cannot charge sufficiently.

As shown in Figure 11, the performance monitor module monitors workload latency as well as the load intensity and reports the results to the *iGear*. With the performance state feedback, *iGear* sends the signals to the power gear controller to trigger a power gear switch via a switch control bus. During the periods with sufficient latency headroom, *iGear* will automatically select the wimpy gear until the latency headroom is below a predefined threshold. Conversely, when the measured user latency is close to the SLO target, *iGear* will launch the beefy server to meet the SLO requirement. To ensure stability, we develop a run-after mechanism that wakes up the successor gear in advance and then re-dispatches the jobs after the successor gear resumes the service (detailed in Section 5.2). Moreover, *iGear* can also collaborate with the server-level power throttling module to further control power demand based on the SLO requirement. This will help further handle the power mismatch.

There are three main reasons for employing a rack-level *iGear* controller. First, it allows us to deploy the rack-level energy storage system, avoiding unnecessary DC/AC conversion. However, the disadvantage of this approach is that the renewable power and energy storage systems for each rack of composite servers are independent and cannot share their capacities. Second, the distributed rack-level power controller can provide a fine-grained way to track load variability and perform a gear switch to guarantee the SLO target since rack-level provisioning will face more load variations than the cluster-level strategy. And third, in the case of load imbalance, the distributed controller can allow the hot nodes to be at a much higher power mode to meet the overall SLO while the vast majority of the nodes will run at a lower power mode. It has a much higher potential of saving power than the uniform cluster-level deployment.

5. GREENGEAR SCHEDULER

This section introduces the supply/demand cooperative power management scheme of *iGear*. As shown in Figure 12, a profiling records module continuously profiles the latency, power demand and allocated renewable energy and battery energy at each time-slot for each server. An optimizer uses the predicted workload and renewable power availability information along with profiling data to optimize the green power supply and the load power demand. We use weather report and time-series analysis methods mentioned in [28] to initialize the prediction module. The *GreenGear* scheduler maintains a power management table (PMT) for the composite server farm. This table is required to control power demand for each composite server and distribute the green power. Each entry of the PMT contains the renewable power supply, the battery energy capacity, the power demand for each power gear. The PMT also records the green energy assignment ratio η which balances the energy usage between the wimpy and beefy gear. All the states of composite servers within each rack are indexed in a power management buffer (PMB). The PMB provides the power supply status (e.g., green and grid power) and power demand status (e.g., gear state and throttling power state) of each composite server. A gear switch records table is used to store the gear switching frequency for each composite server for each workload. A trigger module is used to initialize the switching process. The controller invokes power management activities in response to fluctuations in renewable power generation and load intensity. Once required, the scheduler sends the signals to the controller for execution.

Techniques	Time to take effect	Power after activation
Sleep	Tens of seconds	2-4W per DIMM
Hibernation	A few minutes	0W

Table 1: Impact of techniques on the save state of *GreenGear*.

5.1 Managing Switching Activities

In this study, we adopt a renewable energy-driven mechanism to avoid frequent gear switches in the composite server farm. The *GreenGear* scheduler only harvests at the relatively stable load intensity levels. At each fine-grained time interval, before a switch is triggered by the controller, a predicted load assignment will be transferred to the scheduler to calculate the benefits for power gear switching (e.g., predicted power match ratio minus the prior operation) via the triggering module. To prevent the gear-switching from being triggered too frequently and thus from becoming counterproductive, the scheduler will only instruct the power gear controller for switching if the calculation suggests an increase in the number of wimpy servers in the composite server farm (e.g., due to the currently low to moderate load levels) that is larger than a threshold (e.g., 10% of the primary switch). Further, within each rack of composite servers, *GreenGear* also collects the gear switch records to indicate and balance the power gear switch in the composite server farm, because, frequently tuning a small set of composite servers may incur more latency impact and high communication traffic.

5.2 Managing Server-level Power Demand

GreenGear controls the power demand for each composite server based on two approaches: (1) workload shift between wimpy and beefy gears, (2) gear-level power demand control via power state throttling (e.g., DVFS). Each composite server $j = 1, ..., N$ can operate in a particular power gear mode D^g, where $g \in [W, B]$, W and B denotes the wimpy gear and beefy gear, respectively. The wimpy/beefy gear can also support a power throttling mode $D^g(i)$ (sleep/hibernation, DVFS states), ordered as $D^g(0)$ where the power gear enters in save-state (sleep/hibernation), to $D^g(d)$ which is the highest power demand state and best in performance. We denote the load intensity of a workload as L, also sorted from the lowest and highest as $L_1, ..., L_l$. Hence, the power demand for each composite server j during each time-slot t can be denoted as $LoadPower_t = P_t(D_t^W(i)) + P_t(D_t^B(i))$. Note that, *GreenGear* can calculate the power demand based on the profiling records module. Hence we can phrase our objective function of minimizing the power demand for each composite server j to initialize the power demand result in the PMT.

$$minimize\ LoadPower_t = P_t(D_t^W(i)) + P_t(D_t^B(i)) \quad (4)$$

Managing gear wakeup schedule: When we trigger the transition between the power gears, the switch threshold λ_s is impacted by the save-state of the inactive power gear. As shown in Table 1, the *Sleep* state requires the server to enter S3 (suspend to RAM) state where the volatile memory (DRAM) is still required to operate in self-refresh mode and all other server subsystems are turned off. The resume time of *Sleep* is fast after the gear switch is required. The *Hibernation* state needs to flush the application state to local persistent storage and, unlike the *Sleep* state, allows the complete power down of the power gear but with a longer wakeup duration [18].

We define a system parameter called accommodated load decrease/increase rate (LR, denoted as δ) to be the rate of increase/decrease of the load intensity when the load demand approaches the capability of the current gear. For simplicity, we measure the LR in terms of load intensity (e.g., requests/second) via the historic profiling results. We can minimize the latency impact through determining a feasible wakeup point of the successor gear such that if the offered load changes with rate δ. The wakeup overhead of the power gear D^g is denoted as $\omega(D^g)$. Let t_k be the time when load demand begins to exceed the capability of wimpy gear with highest throttling state. Then, we can obtain the power gear switch threshold λ_s:

$$\lambda_s = L_{t_k} \pm \delta(\omega(D^g)) \quad (5)$$

As shown in the equation, the successor gear should be waken up in advance to cope with the wakeup overhead that is crucial for the SLO target. Hence, in this paper, we employ a run-after gear switch mechanism. That is, immediately after a switching time point, the successor gear will wake up while keeping the source gear powered to service the workload until the former power gear can take over the workloads. Then the source power gear flushes its dirty data to the persistent storage to ensure the data consistence and then enters to the save state. In this paper, we assume the persistent state of the workload can be achieved by a shared storage server similar in [18, 30].

Guaranteeing SLO Target: *GreenGear* allows the datacenter operator to specify an SLO violation limit (not a hard limit) denoted as SLO_{vio} similar to Deadzone algorithm proposed in [31, 27], the SLO_{vio} is the maximum percentage of SLO violation expected in the system (SLO_{vio} is defined as the maximum percentage of the SLO violations for the beefy gear with the highest throttling state).

At each time-slot, the measured latency $R_t(L_t, D_t^g(i))$ offered by composite server j depends on the load intensity L_t and its power mode $D_t^g(i)$, where $L_t \in [L_1, ..., L_l]$. We specify the switch trigger condition from beefy gear to wimpy can be defined : $R(L, D^B) < SLO \cdot \alpha_{Down}$, when the measured latency drops below a lower bound $SLO \cdot \alpha_{Down}$. Similarly, the switch from wimpy to beefy is triggered when the measured latency exceeds an upper bound: $R(L, D^W) > SLO \cdot \alpha_{Up}$. Note that, $\alpha_{Down}(< 1)$ and $\alpha_{Up}(< 1)$ are the SLO controlled variable as the percentile of the monitored request latency. The rational for the gear switch threshold is that, for any two consecutive time slots k and $k+1$, the following SLO guarantee conditional probability must be satisfied: $Pro(R_{k+1} > SLO | R_k < \alpha_{Up} \cdot SLO) <= SLO_{vio}$. The initial value of α_{Up} and α_{Down} can be obtained through empirically measured data from the historic load intensity distribution via the profiling records module. *GreenGear* can monitor the measured latency result to dynamically change the load intensity threshold $\lambda_s(Up, Down)$ to guarantee the SLO target. The parameter $(Up, Down)$ denotes the thresholds for wimpy-beefy switch and beefy-wimpy switch, respectively. At the end of each fine-grained time-slot, the switch threshold will be updated following the rule: If $R > \alpha_{Up} \cdot SLO$, the *GreenGear* gradually reduces the gear switch threshold by $\Delta\lambda_s$ (default 5%) to ensure the SLO guarantee. If $R < \alpha_{Down} \cdot SLO$, the *GreenGear* gradually increases the threshold by $\Delta\lambda_s$ to improve the energy efficiency. These dynamic optimizations can not only minimize the latency oscillation but also allow to deliver the energy efficiency.

Similarly, we denote two latency controlled variable $\alpha_{Down}^L(< 1)$ and $\alpha_{Up}^L(< 1)$ to indicate the throttling states adjustment through using the conditional probability calculation. The variable $\alpha_{Down}^L \cdot SLO$ denotes the threshold to determine when a high power state will be transitioned to a lower state. And $\alpha_{Down}^L \cdot SLO$ is the threshold to determine when a low power state will be adjusted to a higher state. We can calculate the controlled variables a prior through the profiling records. The following two optimization methods are then applied: (a) If $R > \alpha_{Up}^L \cdot SLO$ (overload), the current power gear steps up its power throttling states in

a round-robin fashion if the additional SLO violation still exists. (2) If $R < \alpha^L_{Down} \cdot SLO$ (underutilized), the active power gear steps down its throttling power states to the next lower discrete setting.

5.3 Managing Green Energy Assignment

Given a fixed power mismatch scenario of a scheduling horizon T (e.g., 24 hours in our design) and a power source scheduling policy, the total amount of green energy from the renewables and batteries can be denoted as $GreenEnergy$. The green energy for the wimpy gear is $\eta \cdot GreenEnergy$, where η is the ratio of green energy assigned to the wimpy gear. Likewise, the energy distributed to the beefy gear is $(1-\eta) \cdot GreenEnergy$. At the beginning of each power supply interval T_G (default 10 minutes), the *GreenGear* scheduler obtains the current available capacity of batteries ($BattEenergy$) based on the collected information from the sensors. It also obtains the predicted renewable power generation $GreenPower$ and the predicted load power demand $LoadPower$ based on the predicted load intensity L. To handle the power mismatch, *GreenGear* dynamically schedules the green power supply based on these four variables and calculates the power match ratio (PMR, recall Section 2) at the end of the scheduling horizon T. Figure 13 shows the power supply management framework.

Thus, the scheduling goal of *GreenGear* is:

$$maximize\ PMR(BattEnergy, GreenPower, LoadPower, \eta) \quad (6)$$

we also can adjust the scheduling goal to minimize the utility tariff and/or maximize peak power shaving to save the operation expenditure and capital expenditure as detailed in [6, 28, 17, 30]. This, however, is beyond the scope of this paper and considered as our future work.

5.4 Optimizing Demand/Supply Activity

Note that, the PMT table cannot always ensure the optimal power demand schedule and the optimal green energy assignment since (1) the limited indexed data provided in PMT are based on a prior run and cannot fully cover all power mismatch events in data centers, and (2) the inaccuracy of predicted results will likely decrease the effectiveness of power match. Therefore, the PMT table needs to be gradually optimized.

Server-Level Power Demand Optimization: In Figure 14 we show the pseudo code of power demand optimization. The *GreenGear* scheduler dynamically monitors the load power demand every time-slot (e.g., every second) and adjusts the power states supported in the composite server based on the discrepancy between the initial green power supply budget (e.g., power derived from the renewable and/or batteries) profiled in the PMT table and the actual power demand. Assume that the actual load power demand for composite server is $LoadPower_t$ at time-slot t; we lower the load performance by scaling down the power throttling states for each gear whenever $LoadPower_t$ exceeds the expected green power supply during each power schedule interval T_G in the PMT table. *GreenGear* uses the SLO_{vio}-aware load power demand control mechanism to achieve a better trade-off between green power budget and workload performance. At the end of each time-slot for a certain power gear, the *GreenGear* scheduler checks the current load intensity and latency results for exploiting power saving opportunities. It estimates the chances of SLO guarantee based on the historical profiling results to predict whether the workload can meet the SLO in the future with a lower power throttling state. It updates the power throttling states to indicate a new power demand result for the future scheduling. If at a certain time point $LoadPower_t$ is lower than the expected power supply, *GreenGear* will increase its load performance through scaling up the throttling states, and thereby revert to its expected power demand recorded in the PMT table. This can utilize the bonus green power to improve the workload performance.

Green Energy Assignment Optimization: Figure 15 shows the pseudo code for the green energy allocation optimization between the wimpy and beefy gears that aims to balance their energy supply to maximize the power match ratio. When updating the PMT table, the *GreenGear* scheduler checks the remaining capacity in batteries, $BattEnergy$, and the used renewable energy, $\sum GreenPower$, at the end of each green power scheduling interval (T_G, default 10 minutes). If the actual battery usage rate is larger than the expected result profiled in PMT, the *GreenGear* scheduler decreases the assignment ratio by $\Delta\eta$=1% (default) in the PMT table to decrease the green energy used for the active gear while reserving green energy for the future energy allocation. If the actual green energy usage rate is slower than the profiling result, *GreenGear* reduces the green energy ratio to increase the energy usage of the current on-duty power gear. In practice, we can adjust the ratio η of battery usage to indicate the green energy assignment operation, since battery discharge activity can directly reflect the behavior of the green energy as detailed in Section 2. This optimization operation is to balance the energy supply of wimpy gear and beefy gear for maximally handling the power mismatch.

6. EXPERIMENTAL EVALUATION

In this section, we conduct experimental studies to validate and evaluate our proposed *GreenGear*, based on a scale-down prototype test-bed of a solar-based data center.

Experimental Setup: Our test-bed consists of 20 servers that are organized into 10 wimpy-beefy composite servers. Each beefy server is configured with an Xeon E5-2620 Intel processor (12 cores per server with 2.0 GHz), 64 GB DRAM, one 500 GB hard drive, and each server consumes around 80 watt in the idle state. Each wimpy server has two candidate configurations: (1) an Intel dual core i5-4460 processor, with 48GB DRAM and one 500 GB hard disk, and (2) an Intel dual core i3-2120 processor with 36 GB DRAM and one 500 GB disk capacity. Each server runs Ubuntu Linux with all power saving features provided (DVFS, clock throttling, etc.). Workload generation and data collection of power and performance measurement are handled by a dedicated controller node. The power consumption of each server is measured using distributed power meters [32] and power data is recorded and transferred to the node. To ensure the data consistency, the wimpy-beefy server pair share data through the network file system (NFS), with the controller node acting as the NFS server.

Energy Provision: To simulate the renewable power generation, we use the renewable energy traces in [33], including irradiance every minute, and adjust the chosen trace to fit the peak power demand (the maximum peak power our workloads can hit) on our test-bed. We use 40 12V 24Ah lead-acid batteries for the *GreenGear* racks, which can store the surplus renewable energy. The choice for energy capacity of the battery considers both the renewable energy production and energy consumption of the composite servers. The energy efficiency of the battery is set at 80% as detailed in [3]. To better understand the benefits of our design, we use batteries only for charging renewable energy and draining out the green energy to evaluate the results.

Workload Configurations: We consider three latency-critical workloads, namely, SPECjbb [24], Memcached and Web-Search applications obtained from Cloudsuite [34] as shown in Table 16(b). We specify the target latency for Web-Search as 500ms (90%-ile tail latency) as indicated in Cloudsuite. For Memcached, we use 10 ms (95%-ile tail latency) as the latency target. We tune the workload generators to issue operation/query requests from the SPECjbb

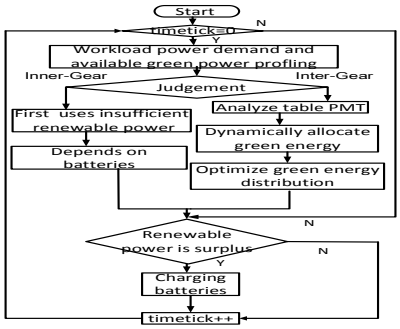


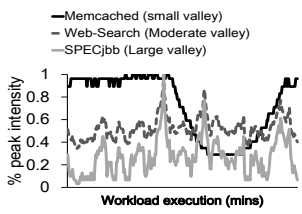
Figure 13: Green power scheduling Framework.

```
// At the end of each fine-grained time-slot t
Definition:
T_G: Green power schedule interval
GreenPower_index :Obtain Power Supply budget from PMT
R_t : Obtain the latency result
1. for Each power demand tuning timestamp t in T_G
2.  Actual load power measurement  LoadPower_t
3.     if load power demand changes
4.     ΔLoadPower← |GreenPower_index−LoadPower_t|
5.       if (R_t ≤ SLO) // judge the SLO guarantee
6.       Adjust power demand based on ΔLoadPower
7.       else  Adjust load power demand based on SLO
8.     end if
9.    end if
10.  end for
11. Update LoadPower_t, GreenPower_t in PMT
12. end
```

Figure 14: Algorithm for load demand optimization.

```
// At the end of each power schedule interval T_G
Calculate initial GreenEnergy_index used in the PMT
Obtain initial battery capacity: BattEnergy_index and
initial renewable energy used ∑_t^{T_G} GreenPower_index in PMT
1. for Each power schedule interval T_G
2.   Actual measurement:
     GreenEnergy_{T_G} = BattEnergy_{T_G} + ∑_t^{T_G} GreenPower_t
3.   if (GreenEnergy_TG > GreenEnergy_index)
4.                η= η −Δη
5.          else if (GreenEnergy_TG <GreenEnergy_index)
6.                η= η +Δη
7.          end if
8.   end if
9.  end for
10.  Update BattEnergy_TG , η in PMT
11. end
```

Figure 15: Algorithm for green energy assignment optimization.



(a) Load Demand.

Workload	Metric	SLO Target	Memory Usage	Load Pattern
SPECjbb	99%-ile latency	500 ms	6 GB	Large valley
WebSearch	90%-ile latency	500 ms	10 GB	Moderate valley
Memcached	95%-ile latency	10 ms	20 GB	Small valley

(b) Workload Description.

Figure 16: Workload configurations.

Schemes	Architecture	Description
BeefyOnly	Beefy servers Only	Only use power scheduling to maximize the PMR
BeefyDVFS	Beefy servers Only	Draw support from power throttling technique to maximize the PMR
WimpyFirst	Heterogeneous	Supplying the wimpy gear first, then the beefy gear if the green energy is sufficient
BeefyFirst	Heterogeneous	Supplying the beefy gear first, then the wimpy gear if the green energy is sufficient
Gear-S	Heterogeneous	Green-aware scheme based on statistics and limited profiling information
GreenGear	Heterogeneous	Green-aware scheme based on our dynamic and optimized scheduling

Table 2: The evaluated power management schemes.

and Web-Search workloads based on a time-varying load trace similar to the Google traffic pattern in [6]. And we use the traffic pattern in [13] to generate small load valleys for Memcached. We use the default latency sampling mechanism provided in the workload benchmarks to generate the latency results. To comprehensively evaluate *GreenGear*, we apply the three workloads whose power demand curves are shown in Figure 16(a) by adjusting the load intensity (valley duration) in a way similar to [35]. The maximum-capacity load controlled by the dedicated node is adjusted based on the capability of our beefy servers so that the latency critical workloads running on our beefy servers can meet the specified latency target.

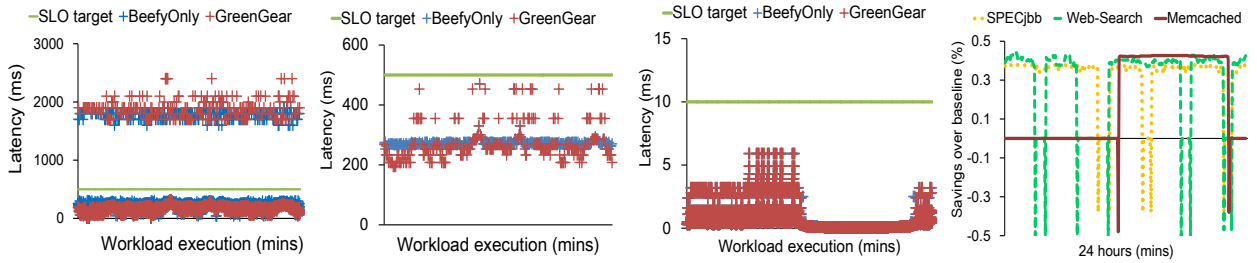
To be more specific, we compare *GreenGear* to five different power management schemes summarized in Table 2. To fairly evaluate the homogeneous beefy-based mechanisms and heterogeneity based mechanisms, the same energy capacity of battery and power capacity of renewable systems are applied in all cases.

6.1 Latency and Power Results

We compare the *GreenGear* scheme to one with the best performance (*BeefyOnly*). Note that for the SPECjbb workload both *BeefyOnly* and *GreenGear* have similar amount of SLO violations, as shown in Figure 17(a). These SLO violations are unavoidable because they are caused by the requests exceeding the provisioned process capacity of the beefy servers. However, as Figure 17(d) shows, during the low to moderate load levels, *GreenGear* can save more than 38% power compared to the BeefyOnly scheme. The power savings are obtained without latency penalty, since these savings do not introduce additional SLO violations. Nevertheless, the switching process causes extra power consumption for the *GreenGear* scheme as indicated by the occasional short periods of negative values in the figure. The negative results show the switching overhead for *GreenGear*, since the two gears will run simultaneously for a short while due to the run-after operation (detailed in Section 5.2). In this paper, we directly put the server in the *Hibernation* state after the gear switch is finished. The *Hibernation* techniques for beefy gear and wimpy gear needs 247 seconds and 52 seconds to save the workload state to the local storage, respectively and our beefy server requires 255 seconds to resume the service and wimpy server needs 78 seconds. Thus the switching energy penalty is negligible compared to the long duration of 24 hours. The *Sleep* state can reduce the wakeup duration (e.g., 8 seconds and 12 seconds for the wimpy gear and beefy gear, respectively) as well as the application save time (6 seconds and 8 seconds for the wimpy gear and beefy gear, respectively) for the *GreenGear* mechanism. However, keeping the workload state in memory may not be worthwhile considering the fact the inactive duration often far exceeds the time to recover the workload state from the persistent storage and the *Sleep* state requires much higher energy from the renewable and battery systems than the *Hibernation* state. Over a 24-hour experiment, the *GreenGear* scheduling scheme can save 31% total energy for SPECjbb, 27% for Web-Search and 18% for Memcached compared to the BeefyOnly mechanism.

6.2 Power Mismatch Handling

Figure 18(a) shows the overall power match ratio (PMR) results (recall Section 2). *GreenGear*'s clear superiority in PMR implies that it can prolong the runtime of the renewable and battery systems. Unlike the conventional homogeneous *BeefyOnly* design, the heterogeneous designs can achieve 100% power match for SPECjbb regardless of the power scheduling policy. For SPECjbb and Web-Search, the heterogeneity-aware schemes significantly outperform the homogeneous schemes in PMR. The reason why the *BeefyFirst* policy performs the worst among the heterogeneous schemes is because it always results in large battery discharge, which reduces the opportunity for utilizing battery for the low power demand wimpy servers. On the other hand, when first supplying green power to wimpy servers, more renewable power can be directly used to obtain more



(a) SPECjbb latency results. (b) Web-Search latency results. (c) Memcached latency results. (d) Power savings gained by *GreenGear* over the baseline scheme BeefyOnly under the three workloads.

Figure 17: Latency and power saving comparison.

power matches while preserving the green energy in the battery for beefy servers. Nevertheless, due to the limited capability of wimpy servers, batteries always have to handle power mismatch when the beefy gear is on duty, the large power demands for beefy gears still leads *WimpyFirst* to underperform the *Gear-S* scheme. However, for small valley duration of Memcached, *BeefyFirst* with brute green power scheduling policy shows an even worse PMR than *BeefyOnly* scheme, the cause is that *BeefyFirst* requires the beefy gear to process for a large portion of loads while does not effectively utilize the insufficient green power (as detailed in Section 2) due to the large power demand and it also makes batteries deplete quickly. *Gear-S* consistently underperforms *GreenGear* because of the limited information in the profiling table and the prediction errors.

6.3 Renewable Energy Utilization

We use the renewable energy utilization (REU) metric to evaluate how much renewable energy is utilized to effectively service the workload (Recall Section 2). Results from Figure 18(b) show that schemes incorporating the wimpy servers to efficiently deal with low to moderate loads consistently outperform *BeefyOnly* and *BeefyDVFS* for workloads with moderate to large valley duration in their load profiles (Figure 16(a)). In fact, these schemes, *BeefyFirst*, *WimpyFirst*, *Gear-S* and *GreenGear*, achieve 100% REU for SPECjbb. For the Memcached workload with small valley duration in its load profile (Figure 16(a)), the *BeefyFirst* scheme underperforms because it missed the opportunity to supply green energy to the energy-efficient wimpy gear. Moreover, *BeefyDVFS* shows a better result than *BeefyFirst* since (a) *BeefyDVFS* can bring a lower power demand than *BeefyFirst* for the beefy gear at the high loads and (b) *BeefyDVFS* uses a smart green power scheduling scheme that can help delivery the green energy to the most energy-efficient region.

6.4 Battery Lifetime

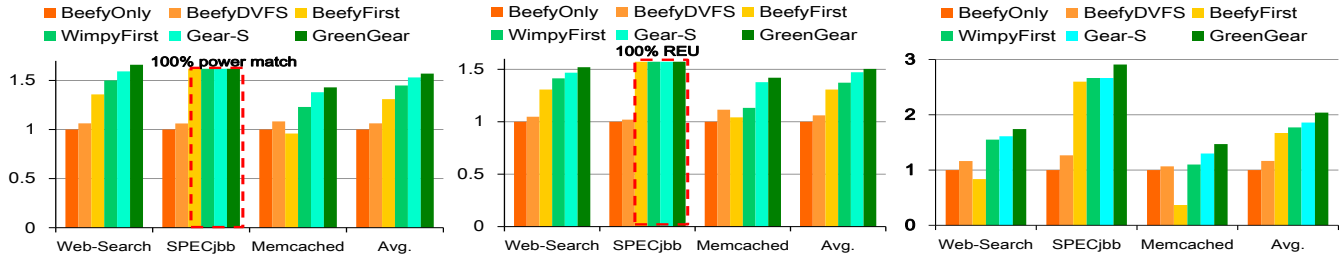
One of the primary goals of leveraging wimpy servers in a heterogeneous system is to keep batteries from large current discharging and extend their lifetime. We use the Ah-Throughput Battery lifetime Model in [36] to evaluate the battery lifespan based on our proposed schemes. As shown in Figure 18(c), the heterogeneous designs leveraging the wimpy servers help prolong the battery lifetime because wimpy servers greatly reduce power demand for low to moderate load levels (e.g., SPECjbb). This is particularly true for workloads with long valley duration in their load profiles, as beefy servers are only used to process a small portion of the workload and batteries protected from large current discharging. However, the *BeefyFirst* scheme underperforms the *BeefyOnly* scheme for workloads with small valley duration in their load profile (Figure 16(a)) (e.g., Memcached) since it requires frequent supplement power from the battery, leading to large current discharging of batteries, and the batteries will be drained out before switching to the wimpy gear. This implies that simply reducing power demand without sufficiently utilizing the available green power does not necessarily result in improvement on battery lifetime.

6.5 Sensitivity Study

Switching Threshold: We evaluate the impact of the switching threshold between the wimpy server and beefy server, on PMR, REU and battery lifetime. All the results, shown in Figure 19(a), are normalized to that of the stable threshold ($\lambda(s)(Up, Down)$) for the dynamic *GreenGear* scheme. We change the threshold value (30%, 40%, 50%) to emulate the wimpy gear capability decrease. Recall from Section 5 that, for a target threshold $\lambda(s)(Up, Down)$ of 50%, the wimpy server will be switched on to process the loads when the load intensity is lower than 50% of peak load. Figure 19(a) shows that a higher threshold consistently results in better power mismatch handling, renewable energy utilization and battery lifetime.

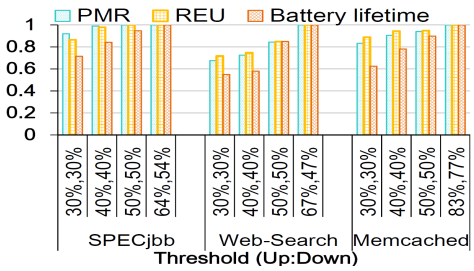
Green Energy Assignment Ratio: As shown in Figure 19(b), we evaluate the PMR impact of the green energy assignment ratio between wimpy servers and beefy servers based on the *GreenGear* scheme for Memcached workload. In the figure, η:(1-η) means the assignment ratio between the wimpy gear and the beefy gear and all the results are normalized to that of the best ratio (6.5:3.5). The results show that when the ratio is less than (6.5:3.5), the more green energy is distributed to the wimpy servers, the higher the PMR value is. When we assign green energy to the wimpy servers at a ratio higher than (6.5:3.5), the PMR value will decrease because the energy of batteries has been heavily used for the wimpy gear and the beefy gear cannot obtain enough power supplement from the batteries, thus causing PMR to decrease.

Wimpy Server Configuration: Figure 19(c) shows the difference between two wimpy systems, that is, one with core-i3 based wimpy servers and the other with core-i5 based wimpy servers, and how they stack up against the beefy server based homogeneous systems *BeefyOnly* and *BeefyDVFS*. As the wimpy server capability decreases from core-i5 to core-i3, the PMR, REU and battery-lifetime performances of *GreenGear* also decrease because it now must stay in the beefy gear longer. Moreover, we observe that the core-i3 based *GreenGear* shows a negligible improvement over the *BeefyDVFS* scheme since beefy servers serve a large fraction of the loads while DVFS can help reduce the power demands for the load levels that the wimpy server cannot reach. Thus, for certain workloads with high load intensity, low capability wimpy server may not be an ideal solution.

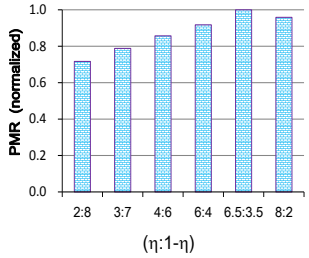


(a) PMR (Power Match Ratio, GreenGear improves by 1.57X on average). (b) REU (Renewable Energy Utilization, GreenGear improves by 1.51X on average). (c) Battery Lifetime (GreenGear improves by 2.04X on average).

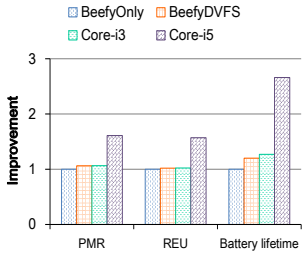
Figure 18: The comparison of *GreenGear* against five other power management schemes (all results are normalized to that of the BeefyOnly policy).



(a) The impact of switching threshold.

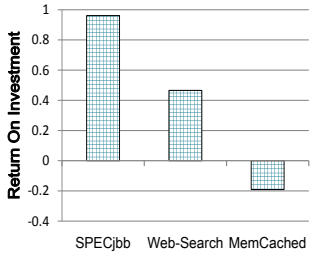


(b) The PMR impact of green energy distribution ratio for the Memcached workload.

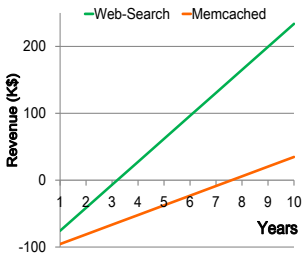


(c) The SPECjbb results for core-i3 based wimpy server.

Figure 19: Sensitivity Study.



(a) ROI in different workload traffics.



(b) The cost benefits of grid peak power and energy reduction by *GreenGear*.

Figure 20: TCO analysis.

7. TCO ANALYSIS

We consider two aspects of the TCO analysis when employing heterogeneous servers in green data centers: (1) the potential investment reduction for renewables and batteries offered by *GreenGear* compared to the conventional design (e.g., BeefyOnly scheme) based on the totally green design and (2) the saved operational expenditure (OP-EX) (e.g., utility peak power and energy cost) gained by *GreenGear* compared to the conventional design for data centers with partly green design.

Return On Investment (ROI): Because *GreenGear* is capable of improving the renewable energy utilization, we save a large amount of battery capacities and renewable power capacities to achieve the goal of totally green (e.g., PMR=100%). Otherwise, we have to provision a large-scale renewable power and battery systems to supply enough green power to match the workload power demand, which is not cost-effective. We further evaluate in view of the investment of green power infrastructure (a.k.a, renewables and battery systems), whether it is worthwhile to provide server-level heterogeneity to reduce the expenditure of renewable power and battery capacities for workloads with different diurnal patterns upon the totally green design. Note that, even deploying an additional wimpy server to create a server-level heterogeneity will not impact the rack density as illustrated in [14]. Since the wimpy server will be merely powered on at low to moderate load levels, the cooling capacity in data center will be not impacted by the server-level heterogeneity design. But we consider the maintenance cost for wimpy servers as reported in [16] in our cost analysis. We define the cost of procuring additional wimpy server as C_{wimpy} ($/server) and the server maintenance cost is C_m ($/server/year). Comparing with the homogeneous design, we denote the reduced investment of photovoltaic (PV) panel through *GreenGear* design is C_p ($/Watt) and the saved battery cost is C_{bat} ($/KWh). The return on investment for heterogeneous servers can be calculated as: $(C_p + C_{bat})/(C_{wimpy} + C_m) - 1$. We assume the battery cost is 300$/KWh and the price of PV panel is 4.74$/Watt, as reported in [17], [35], [37]. The wimpy server deployed in our prototype is 550$/server and the maintenance cost for wimpy server is $18.72/server/year as detailed in [16]. We calculate the ROI under different diurnal workload patterns as shown in Figure 20(a). Note that the cost is amortized during the lifetime (e.g., 4 years for server, 12 years for PV panel equivalent to the lifetime of data center and 4 years for battery). We observe a positive ROI across the diurnal load patterns with long service duration for wimpy server. This suggests that deploying heterogeneous servers is worthwhile for totally green design. However, leveraging heterogeneous servers for workload traffic with high load intensity may be

less profitable than utilizing homogeneous servers (e.g., the negative ROI result for Memcached).

Gain from Operational Expenditure (OP-EX) Reduction: One of primary goals of employing renewable power to data centers is save the OP-EX (e.g., utility peak power and energy cost). Since our *GreenGear* scheme can improve power match and renewable energy utilization by 57% and 51% respectively, which will result in more OP-EX reduction than the homogeneous design (e.g., BeefyOnly scheme). We consider the cost-benefit of investing in additional wimpy server in light of the saved OP-EX offered by *GreenGear* compared to the homogeneous design for data centers partially powered by green energy. We assume a 100kW green data center deployed with 200 wimpy-beefy pairs. We assume the peak tariff is 12$/kW and the utility energy cost is 0.047$/KWh [35]. We calculate break-even point (in year) - the time it takes to recover the procurement cost of wimpy server and start to profit - for workloads with different load intensity curves that are managed by *GreenGear*. Figure 20(b) shows the revenues due to the saved OP-EX offered by *GreenGear* compared to the homogeneous design within 10 years. Even though the *GreenGear* design has expensive initial wimpy server cost (including the maintenance cost) than beefy server only scheme, the break-even point (in year) for Web-Search workload is 3.2 years, which occurs before the wimpy servers have to be replaced due to their lifetimes being reached. However, from the result of Memcached workload (7.6 years), we observe that additional server investments will not result in benefits for workloads with short service duration of wimpy servers. In fact, the break-even results are conservative in our analysis since *GreenGear* design can help achieve a large grid power peak shaving, which will result in larger capital expenditure (CAP-EX) reduction of grid power infrastructure than the homogeneity design. The CAP-EX benefits for *GreenGear* will further quicken the occurrence of break-even point.

8. RELATED WORK

Low Power Design: Wimpy nodes [23, 38, 39, 40], that is, low-power energy-efficient nodes are introduced to data centers and applied to workloads that can tolerate higher latency. However this approach may degrade QoS during traffic spikes, thus requiring over-provisioning. PowerNap [4] manipulates workload's idle periods to save energy by speedily entering a low-power state. Knightshift [14] leverage high energy efficiency wimpy node under low server utilization by using Atom-based servers to achieve energy proportionality. However, without an intelligent power supply-aware management, this approach can only gain very limited energy efficiency return.

Power Mismatch Handling: The renewable energy driven computing system design is to manage power demand/supply variability and their associated power/energy costs. To avoid brownout, recent studies postpone batch workloads [11] [10] to match power demand to the green power supply. Both approaches are unfeasible for data centers that have strict latency requirement specified in the SLOs. Therefore, to meet the performance requirement, energy storage elements [6, 15] and distributed UPS system [3, 16, 37] are also used to handle power mismatch. In contrast to existing work, our study explores the benefits of server heterogeneity and then designs a new architecture to make wimpy servers more efficient in green data centers while employing a supply/load cooperative optimization to handle power mismatch in data centers.

Emerging Green Computing Design: Blink [41] modulates the motherboards' duty cycles to track green power variability. Researchers from Rutgers University have built a solar-powered micro data center called Parasol [28] whose workloads and energy sources are managed by a scheduler called *GreenSwitch*. Parasol schedules non-critical and interactive workloads to minimize the use of grid energy. Li et al. propose a solar-powered prototype called *Oasis* [37] to ensure power capacity scale out economically and sustainably as data centers reach the power budget of grid utility. Oasis motivates us to consider distributed renewable power system in data center design. Liu et al. design *HEB* [17] to leverage hybrid energy buffers to improve energy efficiency from the green power supply side to handle power mismatch. In contrast to this work, we mainly focus on handling ineffective power demand while exploiting the potentials of heterogeneous servers for better energy efficiency.

The most similar prior study to *GreenGear* is *GreenMR* [42], a workload and power source co-scheduling scheme on a MapReduce cluster. *GreenMR* schedules the renewables and/or batteries based on the energy efficiency of MapReduce workloads. *GreenGear* is also proposed to solve the problem of the varying green power and the low energy efficiency at low to moderate load intensity levels. However, *GreenGear* differs itself from *GreenMR* in the below four aspects. (1) *GreenMR* focuses on the MapReduce workloads (e.g., batch workloads), which can be postponed according to the predefined slack to match the power supply from renewables and/or batteries to improve energy efficiency. *GreenGear* focuses on latency critical workloads. The user-facing workloads cannot be deferred due to the strict quality of service guarantee on tail latency. (2) *GreenMR* merely depends on the power scheduling scheme and uses the energy efficiency characteristic of the MapReduce jobs to guide the power sources scheduling. In contrast, *GreenGear* emphasizes the advantage of the heterogeneity design to reduce the ineffective power demand. Moreover, *GreenGear* dynamically adjust the renewable energy allocation ratio based on the effective usage of the renewables and/or batteries. (3) *GreenMR* focuses on using batteries to lower the usage of grid energy and intends to leverage the batteries to shift green energy to the MapReduce jobs with high energy efficiency. In contrast, *GreenGear* pays more attention to the renewable energy utilization. The advantage of *GreenGear* design is that, it can allow direct use of more green power to reduce the energy loss caused by batteries while the low power demand for wimpy server will help improve the battery lifetime. (4) *GreenMR* relies on a centralized green power integration mechanism while *GreenGear* focuses on distributed renewable power provision at PDU level. The advantage of such design is that, it provides a fine-grained way to scale the renewable power capacity following the energy demand of the workload.

9. CONCLUSION

In this study, we propose *GreenGear*, a new heterogeneity-based green power provisioning architecture that enables data centers to flexibly deploy the heterogeneous servers. In order to handle power mismatch, we tailor a power management framework to intelligently and dynamically schedule green power and control power demands between wimpy servers and beefy servers for achieving higher energy efficiency and meeting SLO target. Compared with different power management schemes, the experimental results show that *GreenGear* could improve power matches by 57%, extend battery lifetime by 2.04X, and improve the renewable energy utilization by 51% on average.

10. ACKNOWLEDGMENTS

We would like to thank anonymous reviewers for their insightful comments. This research is sponsored by the National Basic Research 973 Program of China under Grant No. 2011CB302303, the National Natural Science Foundation of China under Grant No. 61300046, and the Fundamental Research Funds for Central Universities, HUST, (Grant No. 2013KXYQ003).

11. REFERENCES

[1] M.Mills, "The cloud begins with coal: Big data, big networks, big infrastructure, and big power," in *National Mining Association and American Coalition for clean Coal Electricity*, 2013.

[2] G. Boccaletti, "How it can cut carbon emissions," in *McKensey Quarterly*, 2008.

[3] C. Li, R. Zhou, and T. Li, "Enabling distributed generation powered sustainable high performance data center," in *HPCA*, 2013.

[4] D. Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: Eliminating server idle power," in *ASPLOS*, 2009.

[5] L. A. Barroso and U. Hölzle., "The case for energy-proportional computing.," *Computer*, vol. 40, no. 12, pp. 22–40, 2007.

[6] D. Wang, C. Ren, A. Sivasubramaniam, B. Urgaonkar, and H. Fathy, "Energy Storage in Datacenters: What, Where, and How much?," in *SIGMETRICS*, 2012.

[7] C. Li, W. Zhang, C.-B. Cho, and T. Li, "Solarcore: Solar energy driven multi-core architecture power management," in *HPCA*, 2011.

[8] C. Lefurgy, X. Wang, and M. Ware, "Server-level power control," in *ICAC*, 2007.

[9] C. Li, A. Qouneh, and T. Li, "iSwitch: Coordinating and Optimizing Renewable Energy Powered Server Clusters," in *ISCA*, 2012.

[10] I. 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 *SC*, 2011.

[11] I. Goiri, K. Le, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "GreenHadoop:Leveraging Green Energy in Data Processing Frameworks," in *EuroSys*, 2012.

[12] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in *NSDI*, 2008.

[13] D. Lo, L. Cheng, R. Govindaraju, L. A. Barroso, and C. Kozyrakis, "Towards energy proportionality for large-scale latency-critical workloads," in *ISCA*, 2014.

[14] D. Wong and M. Annavaram, "Knightshift: Scaling the energy proportionality wall through server-level heterogeneity," in *Micro*, 2012.

[15] S. Govindan, D. Wang, A. Sivasubramaniam, and B. Urgaonkar, "Leveraging Stored Energy for Handling Power Emergencies in Aggressively Provisioned Datacenters," in *ASPLOS*, 2012.

[16] 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 *ISCA*, 2012.

[17] L. Liu, C. Li, H. Sun, Y. Hu, J. Gu, T. Li, J. Xu, and N. Zheng., "HEB: Deploying and managing hybrid energy buffers for improving datacenter efficiency and economy," in *ISCA*, 2015.

[18] D. Wang, S. Govindan, A. Sivasubramaniam, A. Kansal, J. Liu, and B. Khessib, "Underprovisioning Backup Power Infrastructure for Datacenters," in *ASPLOS*, 2014.

[19] J. Mars, L. Tang, and R. Hundt, "Whare-map: Heterogeneity in "homogeneous" warehouse-scale computers," in *ISCA*, 2013.

[20] C. Delimitrou and C. Kozyrakis, "Paragon: Qos-aware scheduling for heterogeneous datacenters," in *ASPLOS*, 2013.

[21] L. A. Barroso and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Morgan and Claypool Publishers*, 2009.

[22] Seammicro, "Seamicro introduces the sm10000-64hd," 2011.

[23] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan, "Fawn: A fast array of wimpy nodes," in *SOSP*, 2009.

[24] "SPECJBB 2013:Java Business Benchmark.." `http://www.spec.org/jbb2013/`.

[25] I. Sartoria, A. Napolitano, and K. Voss, "Net zero energy buildings: A consistent definition framework," *Energy and Buildings*, vol. 48, pp. 220–232, 2012.

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

[27] V. Petrucci, M. A. Laurenzano, J. Doherty, Y. Zhang, D. Mossé, J. Mars, and L. Tang, "Octopus-man: Qos-driven task management for heterogeneous multicores in warehouse-scale computers," in *HPCA*, 2015.

[28] I. Goiri, W. Katsak, K. Ley, T. D. Nguyen, and R. Bianchini, "Parasol and Greenswitch: Managing Datacenters Powered by Renewable Energy," in *ASPLOS*, 2013.

[29] 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 *SIGMETRICS*, 2012.

[30] X. Zhou, Q. Cao, H. Jiang, L. Tian, and C. Xie, "Underprovisioning the grid power infrastructure for green datacenters," in *ICS*, 2015.

[31] T. Horvath, T. Abdelzaher, K. Skadron, and X. Liu., "Dynamic voltage scaling in multitier web servers with end-to-end delay control.," *IEEE Trans. Comput.*, vol. 56, no. 4, pp. 444–458, 2007.

[32] "Zh-101 portable electric power fault recorder and analyzer," 2009.

[33] "Measurement and instrumentation data center." `http://www.nrel.gov/midc/`.

[34] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the clouds: a study of emerging scale-out workloads on modern hardware," in *ASPLOS*, 2012.

[35] S. Govindan, A. Sivasubramaniam, and B. Urgaonkar, "Benefits and Limitations of Tapping into Stored Energy for Datacenters," in *ISCA*, 2011.

[36] H.Bindner, "Lifetime modelling of lead acid batteries," in *Risϕ National Laboratory*, 2005.

[37] C. Li, Y. Hu, R. Zhou, M. Liu, L. Liu, J. Yuan, and T. Li, "Enabling datacenter servers to scale out economically and sustainably," in *MICRO*, 2013.

[38] V. J. Reddi, B. Lee, T. Chilimbi, and K. Vaid, "Web search using mobile cores: Quantifying and mitigating the price of efficiency," in *ISCA*, 2010.

[39] U. Hölzle, "Brawny cores still beat wimpy cores, most of the time," in *IEEE Micro*, 2010.

[40] M. Guevara, B. Lubin, and B. C. Lee, "Navigating heterogeneous processors with market mechanisms," in *HPCA*, 2013.

[41] N. Sharma, S. Barker, D. Irwin, and P. Shenoy, "Blink: Managing server clusters on intermittent power," in *ASPLOS*, 2011.

[42] Z. Niu, B. He, and F. Liu, "Not all joules are equal: Towards energy-efficient and green-aware data processing," in *IC2E*, 2016.