Joint Optimization of Idle and Cooling Power in Data Centers While Maintaining Response Time

Faraz Ahmad and T. N. Vijaykumar
School of Electrical and Computer Engineering
Purdue University
{fahmad,vijay}@ecn.purdue.edu

Abstract

Server power and cooling power amount to a significant fraction of modern data centers' recurring costs. While data centers provision enough servers to guarantee response times under the maximum loading, data centers operate under much less loading most of the times (e.g., 30-70% of the maximum loading). Previous server-power proposals exploit this under-utilization to reduce the server idle power by keeping active only as many servers as necessary and putting the rest into low-power standby modes. However, these proposals incur higher cooling power due to hot spots created by concentrating the data center loading on fewer active servers, or degrade response times due to standby-to-active transition delays, or both. Other proposals optimize the cooling power but incur considerable idle power. To address the first issue of power, we propose *PowerTrade*, which trades-off idle power and cooling power for each other, thereby reducing the total power. To address the second issue of response time, we propose *SurgeGuard* to over-provision the number of active servers beyond that needed by the current loading so as to absorb future increases in the loading. SurgeGuard is a two-tier scheme which uses well-known over-provisioning at coarse time granularities (e.g., one hour) to absorb the common, smooth increases in the loading, and a novel fine-grain replenishment of the over-provisioned reserves at fine time granularities (e.g., five minutes) to handle the uncommon, abrupt loading surges. Using real-world traces, we show that combining PowerTrade and SurgeGuard reduces total power by 30% compared to previous low-power schemes while maintaining response times within 1.7%.

Categories and Subject Descriptors: C.5.5 [**Computer System Implementation**] Servers

General Terms: Design; Measurement; Performance

Keywords: data center; power management; idle power; cooling power; response time

1 Introduction

Data centers have emerged as compute engines for many market segments offering Web-based services (e.g., database and Web services) and for commercially-important large-scale computations in the Internet- and Web-computing domains such as machine learning and data mining. By sharing hardware, software, and infrastructure resources among many service providers, data centers reduce the total cost for the service providers. However, electric power consumption is a major component of a data center's total cost of ownership which includes the cost of the building, servers, air conditioners, power distribution equipment, and electric power. While some of these components are related to peak power (e.g., cost of air conditioners and power distribution equipment), the actual electric power drawn accounts for about 25% of the total cost [15]. For instances, by 2011, US data centers are expected to spend $7.4 billion per year on electric power [13].

A data center consumes electric power for operating both its servers and its cooling system which removes the heat dissipated by the servers; the corresponding power components are called *server power* and *cooling power*, respectively. These components are related via *power usage effectiveness* (PUE), which is defined as total electric power / server power, where the total electric power = server power + cooling power + power distribution losses. Modern data centers' PUEs are close to 2, and server and cooling power contribute significantly to the total power [4].

While data centers provision for the number of servers to guarantee response times specified in service-level agreements under the maximum loading, data centers operate under much less loading most of the times (e.g., 30-70% of the maximum loading) [3, 5]. Thus, data center servers idle a significant amount of time. Consequently, the server power includes not only the active power consumed while processing requests but also the idle power consumed while waiting for requests. Server idle power adds not only to the server power but also to the cooling power needed to remove the heat due to the idle power. The pioneering work in [7, 26] proposes to reduce server idle power by concentrating the data center loading on a subset of the servers and powering-off the rest of the servers (or putting them in standby mode). While this approach, which we call as *spatial subsetting*, significantly reduces idle power, the approach raises two concerns: increased cooling power due to higher temperature and response time degradation.

Spatial subsetting concentrates the data center loading on fewer servers which run at higher utilization than they would if the loading were distributed over all the servers. Consequently, the active servers reach higher temperatures creating hot spots, even if the servers are chosen to be physically distributed across the data center. Because cooling effort increases with higher temperature even if the amount of heat removed is the same (i.e., cooling requires more input electric power), spatial subsetting increases cooling power. While this effect is not considered by the spatial subsetting papers, other papers observe that because cooling invariably is

ASPLOS'10 March 13-17, 2010, Pittsburgh, Pennsylvania, USA.

uneven, even identical servers under equal loading reach different temperatures [21, 31]. For instance, air flow through the middle of the aisles between machine racks is better than that near the ends of the aisles, making the middle cooler than the ends. Therefore, uniformly distributing the loading over all the servers results in hot spots. Instead, the papers reduce cooling power by distributing the loading over all the servers as the inverse of the server temperature, and achieving uniform temperature. While this simple and effective strategy, which we call as *inverse-temperature assignment*, reduces the temperature and hence the cooling power, this strategy spreads the loading over *all* the servers and increases the server idle power under the usual less-than-maximum loading. Thus, spatial subsetting may increase cooling power more than it reduces idle power (vice versa for inverse-temperature assignment).

Prior work has explored an alternative to spatial subsetting: Instead of keeping only a subset powered on as in spatial subsetting, all the servers are kept powered on and the power at less-than-maximum loading is reduced by dynamic voltage and frequency scaling (DVFS) [8]. We call this approach as *temporal subsetting*. Because temporal subsetting does not concentrate the loading on fewer servers, temporal subsetting may not incur the temperature problems of spatial subsetting. However, DVFS transition times are too long and degrade response times [20], which is the second issue in reducing idle power (relevant to both spatial subsetting and temporal subsetting). In addition, DVFS is applicable only to the CPU, which contributes only about one-thirds of the total system power [20]. Furthermore, the supply voltage is already close to about 1 V below which CMOS circuits may not work reliably, leaving little room for DVFS in future technologies.

To address DVFS's problems, another recent temporal subsetting proposal, called PowerNap [20], puts idle servers in well-known standby modes, generally found in laptops, instead of applying DVFS (e.g., the ACPI S3 state for the CPU and the self-refresh state for the DRAM). Unlike DVFS, standby mode is applicable to the whole system and is not limited by supply-voltage scaling. While the PowerNap paper provides a detailed queuing-theoretic analysis of response times, the authors presume standby-to-active transition times of 1-100 ms which are inconsistent with real-world systems. For instance, Linux documentation [11] shows transition times of 1-2 seconds for ACPI S1 state and 3-5 seconds for ACPI S3 state. Even with transition times as short as 100 ms, PowerNap reports 1.5-3.5x response time degradations at low loading where many servers are in standby mode. Like temporal subsetting schemes, spatial subsetting also degrades response times. While the spatial subsetting scheme [7] assumes infrequent loading changes (e.g., 2-3 times a day), real-world loading changes continually, which would require frequent transitions and would degrade response times. While inverse-temperature assignment avoids response time degradation by keeping all the servers activated, the scheme incurs high idle power, as discussed above.

To tackle the first issue of cooling power in spatial subsetting, we make the key observation that there is a trade-off between server idle power and cooling power in spatial subsetting and hence, the two should be optimized *together* to reduce the total data center power. Accordingly, we propose a novel joint optimization, called *PowerTrade*, which reduces the sum of idle power and cooling power, and hence, the total power. PowerTrade reduces the sum by activating more servers than spatial subsetting to reduce the load on each server, and hence its temperature, but fewer active servers than inverse-temperature assignment to reduce the idle power. We describe two implementation variants of PowerTrade.

To tackle the second issue of response time degradation in spatial subsetting, we make the key observation that the data center loading changes mostly smoothly over coarse granularities of time (e.g., 30-60 minutes). Accordingly, we employ *SurgeGuard* to over-provision the number of active servers beyond that needed by spatial subsetting so as to absorb increases in the loading. SurgeGuard is a two-tier scheme which operates at two time granularities: For the common, smooth increases in the loading, we employ over-provisioning [8, 34] to handle a maximum increase in the loading in a pair of consecutive coarse-grain time intervals (e.g., one hour). However, uncommon, abrupt loading surges may deplete the over-provisioned reserves (e.g., CNN.com meltdown after the 9/11 attacks). Though uncommon, such surges often test service providers' ability to retain customers by ensuring good response times [18]. Previous over-provisioning schemes do not address the surges which degrade response times. In contrast, we employ a novel scheme to replenish the reserves at a fine granularity (e.g., five minutes). Because the over-provisioning required is relatively small for the increases in the loading in practice, we can maintain response times while incurring only a small increase in server idle power (e.g., in real-world traces, the average hourly loading increase is 3%).

To reduce both total power and response time degradation, we combine PowerTrade and SurgeGuard by simply applying SurgeGuard's over-provisioning to the number of servers arrived at by PowerTrade. We summarize our key contributions as follows:

- We propose a novel joint cooling-power-idle-power optimization, called PowerTrade, to reduce the sum of the cooling power and the idle power;
- We propose a novel two-tier over-provisioning scheme, called SurgeGuard, to alleviate response time degradation;
- Using real-world traces, we show that combining PowerTrade and SurgeGuard (1) reduces total power by 30% over the *better* of spatial subsetting which targets only idle power and inverse-temperature assignment which targets only cooling power; and (2) maintains response times within 1.7%, as compared to spatial subsetting which degrades response time by a factor of 2.8.

The remainder of the paper is organized as follows: We describe PowerTrade in Section 2 and SurgeGuard in Section 3. In Section 4, we explain our experimental methodology. Section 5 presents our results. We discuss related work in Section 6 and conclude in Section 7.

2 Optimizing both idle & cooling power

As mentioned in Section 1, spatial subsetting reduces server idle power but increases cooling power whereas inverse-temperature assignment reduces cooling cost but incurs high idle power. We propose a joint optimization, called PowerTrade, to reduce idle power and cooling power together so that the total power is reduced. Because PowerTrade builds on spatial subsetting and inverse-temperature assignment, we describe these schemes first.

2.1 Background: Spatial Subsetting and Inverse-Temperature

Recall that spatial subsetting reduces server idle power by concentrating the data center loading on as few servers as possible. In spa-

tial subsetting, the resource allocator chooses a subset of servers sufficient to satisfy the demand based on the current loading, and assigns incoming requests to the subset. The number of active servers can be determined using simple queueing theory estimates for a given loading characterized by the request rate and size. The rest of the servers are powered off (or put in standby mode) to reduce idle power. If the loading increases, more servers are brought into the active pool; and as the loading decreases, the excess servers are powered off. While the spatial subsetting papers do not discuss temperature issues, the active pool of servers can be chosen to be *physically* distributed over the data center to help cooling. Nevertheless, the high temperature of the active servers leads to hot spots, which cause an increase in the cooling costs (e.g., a 1120-server data center exhibits a temperature variation of 22^oC - 37.12^oC in the room at 60% loading). As mentioned in Section 1, activating the servers (from power off or standby states) takes time (e.g., 1-2 seconds for modern standby modes). We address the resultant response time degradations in Section 5.

Recall that the inverse-temperature schemes [21, 31], on the other hand, reduce the temperature to optimize the cooling cost by distributing the load among all the servers as the inverse of the server temperature, and achieving uniform temperature. We describe only two variants here which represent the rest [21]. The first variant, called *CoolestOutlets* [31], exploits temperature sensors to sense the air temperature at the outlet of all the servers. The resource allocator assigns work inversely proportional to the server temperature. The allocator changes the load distribution, as the loading of the data center and/or the temperature of the servers vary, by assigning less work to hotter servers and more work to cooler ones. Accordingly, for the same loading, the temperature is lower for CoolestOutlets than that for spatial subsetting (e.g., under CoolestOutlets, a 1120-server data center exhibits a temperature variation of 26^oC - 32^oC in the room at 60% loading). The second variant, called *MinHR* [21], reduces cooling costs by minimizing the heat that recirculates within a data center. Such heat recirculation may result from obstructions in the server air flow, server fans drawing air from the (hot) neighboring servers, and blockage in the return vents of the air conditioners. MinHR distributes the load proportional to the ratio of heat produced to heat recirculated, so that the servers that recirculate more heat than others are assigned less work**.** This ratio is pre-computed for every server via calibration runs. In either of the variants, because all the servers remain powered on, there is significant idle power in the common case of less-than-maximum data center loading. While PowerTrade can be built on top of either of the variants, our implementation builds on CoolestOutlets due to its simplicity.

2.2 PowerTrade

Recall that (1) the total power = *server power* + *cooling power* + power distribution losses, and PUE = total power / server power*; (2)* the server power has two components; *compute power* and *idle power*; and (3) PowerTrade reduces both idle power and cooling power. The cooling power required to remove the heat is given by the relation *server power/COP* where *COP* is the coefficient of performance, an efficiency metric, for the computer room air conditioning (CRAC) units in the data center. The *COP* is a decreasing function of the average temperature of the hottest servers in the room (e.g., the hottest 5% of servers). A higher value of the *COP* means more efficient cooling (typical value for the *COP* is in the range of 1-2 [25, 29]). Because we target server + cooling power and not power distribution losses, we define *net power* to be server power + cooling power. Thus,

$$\text{NetPower} = \text{ServerPower} \times \left(1 + \frac{1}{\text{COP}}\right)$$

$$\text{ServerPower} = \text{computepower} + \text{idlepower}$$

Spatial subsetting reduces the server power by concentrating the loading on fewer servers and reducing idle power. Although reducing the idle power decreases the amount of heat to be removed, concentrating the load results in hot spots in the room due to increased per-server utilization. The hot spots, in turn, lower the COP and increase the cooling power. On the other hand, inverse-temperature reduces the temperature and improves the COP by distributing the loading among all the servers to avoid hot spots. While the improved COP reduces the cooling power, running all the servers increases the idle power (and the accompanying cooling power) especially in low-loading periods.

To reduce the sum of the cooling power and the idle power, PowerTrade should activate more servers than needed by spatial subsetting to reduce the load on each server, and hence its temperature, but fewer active servers than inverse-temperature assignment to reduce the idle power. To this end, we would, ideally, like to compare the gain in cooling power by keeping a server active against the gain in idle power by putting a server to standby and redistributing the server's load among the remaining active servers. This comparison would allow us to reduce the sum of idle power and cooling power. However, performing this comparison for every server, either at the time of every incoming request or even periodically over time, would be hard to implement given the large number of servers. Hence, we resort to a coarser granularity by exploiting the fact that the data center can be divided into a few, nearly-iso-temperature *zones*, which are formed naturally due to air flow patterns in the data center. Because hot air rises from bottom to top and the air flow is poor at the ends of the aisles, the top shelf servers in each rack and the side racks of each row are hotter than the rest and the servers at the very ends of the rows are the hottest (see Figure 1), The zone division leads us to break up the problem of distributing the overall loading to reduce the net power into two questions - one *across* the zones and the other *within* a zone:

- how is the overall loading distributed across the zones (i.e., what is the load distribution ratio across zones)? and
- how is a zone's share of the load distributed among its servers?

While the first question deals with zones which are only a few in number as compared to the total number of servers, the second question deals with servers within each zone which are many in number bringing back the original issue of assigning the loading for each of the numerous servers. To avoid this issue, we use a single policy for an entire zone and keep the problem tractable.

In the rest of the paper, we assume three zones — *cool*, *warm*, and *hot* — in the data center (see Figure 1). However, our schemes are applicable to more zones if needed, depending upon the desired level of accuracy, temperature variations, and the size of the data center.

We propose three implementations, two static and one dynamic, where the static (dynamic) schemes employ static

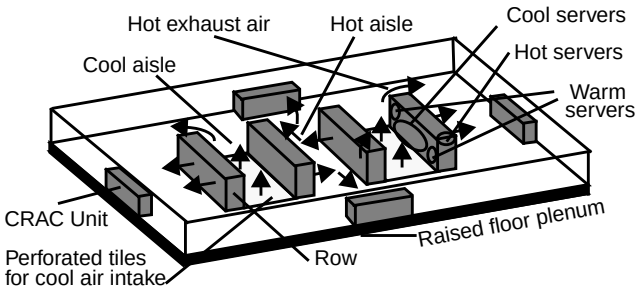

Figure 1: Data Center Model

(dynamic) methods to answer the above two questions. In addition, the static schemes reduce both the idle power and the cooling power, targeting each separately without *directly* comparing the idle power reduction against the cooling power savings so that the net power is reduced, as is done by the dynamic scheme.

2.2.1 PowerTrade-s

The first static scheme, called PowerTrade-s, primarily targets the cooling power in across-zone load distribution (the first question above) and the idle power in within-zone load distribution (the second question above). The within-zone load distribution also secondarily targets cooling power within each zone. We assume that the data center loading is monitored and quantified so that PowerTrade-s can perform its across-zone and within-zone load distributions. We describe PowerTrade-s assuming the data center loading does not change and discuss such changes later.

For the across-zone load distribution, PowerTrade-s determines the ratio of load distribution *across* the zones, called the *iso-temp ratio*, by using a simple calibration run. In this run, we load the *cool* zone to a particular reference loading (say 50%) and measure the zone's temperature. Then, we load every other zone, one by one in isolation, until the same temperature is reached. To simplify the calibration process, we uniformly distribute the load within each zone across the zone's servers. In line with its simple nature, PowerTrade-s assumes that the iso-temp ratio remains the same even when the loading is different from the reference loading, as shown in Figure 2**.** In this figure, a 50% loading of the cool zone and a 25% loading of the warm zone reach the same temperature (22.5 ^{o}C), giving a cool-zone-to-warm-zone iso-temp ratio of 2:1. Because load distribution as per the iso-temp ratio ensures that the other zones do not exceed the cool's zone temperature, PowerTrade-s prevents hot spots in the warm and the hot zones.

Targeting idle power primarily and cooling power secondarily in within-zone load distribution**,** we exploit the idle-power saving feature of spatial subsetting and the cooling-power saving feature of inverse-temperature. Because the cool zone is the least prone to heating up, reducing the idle power is more important than avoiding hot spots. Accordingly, PowerTrade-s employs spatial subsetting within the cool zone to distribute the cool zone's share of the load, as per the iso-temp ratio (i.e., concentrate the load on fewer servers). In contrast, because avoiding hot spots is important in the warm zone, PowerTrade-s distributes the warm zone's share of the load among all the warm-zone servers (equal distribution is equivalent to inverse-temperature as all the active servers in a zone have nearly the same temperature). Thus each zone uses a single policy for load distribution within a zone. Because of their high propensity to heating up, the hot-zone servers are kept in standby mode unless the loading in the data center cannot be satisfied by the other two zones. Whenever needed, the hot-zone servers are equally loaded in the zone to prevent hot spots.

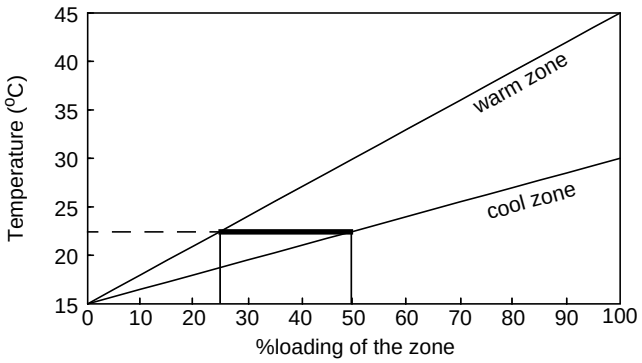

Figure 2: Iso-temp ratio for PowerTrade-s

When a production-run loading exceeds the reference loading, it is possible that distributing the load as per the iso-temp ratio exceeds the cool or warm zone's capacity (the capacity of the hot zone, which is used only as a last resort, is never exceeded). In such cases, the iso-temp ratio is disregarded and the cool, warm, and hot zones are filled to capacity in that order.

So far, we have described PowerTrade-s for a fixed data center loading. When the loading changes, PowerTrade-s readjusts the across-zone and within-zone load distributions. We discuss SurgeGuard to handle response-time issues with loading changes in Section 3. We also discuss how PowerTrade-s can be combined with SurgeGuard in Section 3.2.

Being a simple scheme, PowerTrade-s has three limitations: First, PowerTrade-s's simplifying assumption does not account for the fact that the iso-temp ratio for different reference loadings would be different due to non-linear relationship among server loading, air flow, and temperature. Second, PowerTrade-s uses the fixed policy of spatial subsetting in the cool zone to target idle power while ignoring hot spots and the fixed policy of inverse-temperature in the warm zone to target cooling power while ignoring idle power. However, at high loading of cool zone and low loading of warm zone, the cool zone may have hot spots and the warm zone may have many servers idling, making this strategy sub-optimal. And finally, PowerTrade-s ignores the exchange of heat across the zones due to air flow. The calibration run loads each zone in isolation until the desired temperature is reached but in production runs the zones are loaded together and the zones exchange heat.

To address the first limitation in PowerTrade-s, we propose a variant, called PowerTrade-s++, which determines the iso-temp ratios at multiple reference loadings. PowerTrade-s++ uses these ratios to produce a piece-wise linear approximation of the iso-temp ratio for all other loadings, as shown in Figure 3. In this figure, a 30% loading of the cool zone and a 7% loading of warm zone reach the same temperature (20 ^{o}C) and the iso-temp ratio changes with the loading. In our experiments we found that seven reference ratios result in a good approximation. The other two limitations listed above are addressed by our dynamic scheme but not by our static schemes.

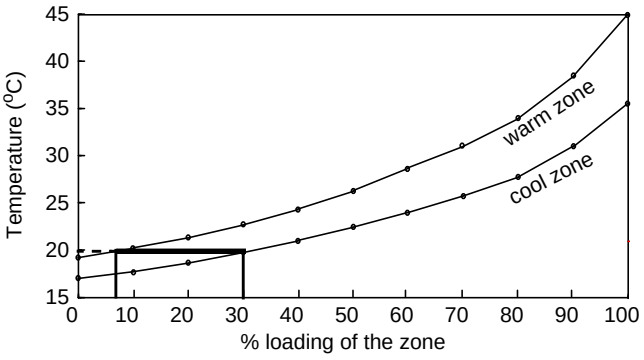

Figure 3: Iso-temp ratio for PowerTrade-s++

2.2.2 PowerTrade-d

The last two limitations above indicate that the two questions of across-zone and within-zone load distributions, and reductions in idle power and cooling power are inter-dependent and cannot be answered in isolation. However, minimizing the net power using an offline analysis would require considering numerous variables such as per-server temperature profile and loading, air flow, and heat exchange for the current load distribution. Therefore, such an analysis would be hard. Consequently, PowerTrade-d uses an online approach to dynamically adjust the load distribution both across and within zones based on the observed temperatures and hence the cooling power, and the observed idle power dissipation. Thus, instead of offline computation, PowerTrade relies on online measurement to guide the optimization. PowerTrade directly compares the potential idle power reduction achieved by activating fewer servers against the potential cooling power savings achieved by activating more servers. PowerTrade chooses the larger of the two to reduce the net power.

As mentioned before in Section 2.2.1, performing this comparison for all the servers together in one shot would be hard. On the other hand, performing this comparison one server at a time would be slow because temperature takes some time to reach a steady state after each step (e.g, 3-5 minutes in our simulated data center). Instead, PowerTrade-d uses a coarser granularity of *groups* of servers to perform this comparison. Groups are much smaller than zones but large enough to impact the temperature to a measurable extent (e.g., 10 servers). PowerTrade successively reduces the net power in steps of one group at a time until converging to *a* minimum (i.e., where the net power does not reduce any more).

Three key issues with any successive refinement scheme are (1) whether the scheme converges to the global minimum; (2) whether the convergence is fast; and (3) the choice of server groups for data centers providing multiple services. For the first issue, we observe that because the idle power and cooling power, respectively, are monotonically increasing and decreasing functions of the number of active servers, the sum of these two components has a single global minimum, as illustrated in Figure 4. Consequently, our refinement process will converge towards this global minimum without getting stuck in some local minimum. However, the convergence point may not be the optimum due to discretization errors from our groups, measurement errors, and other errors due to the empirical nature of our approach.

The second issue is important because during the time of the convergence the scheme operates sub-optimally. We observe that because loading changes only slowly over large time intervals, our groups are large enough to capture this change in only a few steps of refinement (e.g., 3 steps per hour on the average). Consequently, the refinement's convergence delay is short. While increasing the group size achieves faster convergence, doing so results in less power reduction due to the coarseness of the refinement steps (and vice versa for decreasing the group size). We empirically found that a group size of 10 works well.

The third issue is that for data centers providing multiple services, each server typically handles a specific service. Because servers in a server groups are (de)activated together, server groups that span servers for different services may either activate servers and exceed the demand for their particular service or deactivate servers and not meet the demand. To avoid this problem, we constrain server groups to comprise servers for only one service, thus making our groups service-aware. Because server groups are much smaller than the set of servers for a given service, this constraint is acceptable (e.g., 10 in a group versus 500 for a service).

Our successive refinement is triggered only when the data center loading changes; otherwise the current data center configuration of active servers can satisfy the demand. Because a loading change of less than 1% on our simulated data center did not affect the temperature, and hence the cooling power, our refinement algorithm is not triggered below this threshold (real data centers would also have a similar threshold). There are two cases of loading change: an increase and a decrease. In the first case, we initially continue with the current configuration and distribute the load among the currently active servers; if the loading cannot be satisfied with the current number of active servers then we activate new servers (i.e., transition them from the standby mode to the active mode). We measure the cooling power for this initial distribution which gives us one of the two configurations to be compared in each of our refinement steps. To obtain the other configuration, we observe that inverse-temperature, and not spatial subsetting, minimizes net power when the loading increases to the point that all the servers need to be activated. At this point, while the idle power cannot be reduced any further, the cooling power can be reduced by using the inverse-temperature assignment. Because our current loading has increased, we move the data center configuration *towards* the inverse-temperature strategy of activating *all* the servers. Accordingly, we activate a new group of servers previously in standby mode and redistribute the load among the active servers as per inverse-temperature. We measure the resultant cooling power once the temperature settles after the redistribution. Because the new configuration has more active servers, the cooling power of the new configuration is likely to be better than that of the initial configuration. However, the new configuration incurs higher idle power due to the newly-activated group of servers. We compute the extra idle power based on the servers' power ratings.

If the cooling power reduction from the initial configuration to the new configuration is larger than the extra idle power then we keep the new configuration and proceed to the next refinement step by activating another group of servers. Otherwise (i.e., the extra idle power is larger than the cooling power reduction), we revert back to the initial configuration and our refinement process ends for the current increase in the loading.

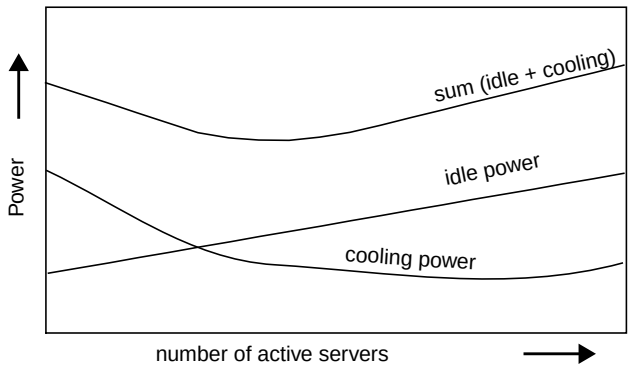

Figure 4: Idle power versus cooling power

In the second case of decreasing load, we start by measuring the cooling power needed for the current configuration which is the initial configuration in this case. To obtain the other configuration for our comparison, we observe that spatial subsetting, and not inverse-temperature, minimizes net power when the loading decreases to the point that only a few servers need to be active. At this point, the load is already distributed (keeping the temperature low), however, the idle power can be reduced by using spatial subsetting. Because our current loading has decreased, we move the data center configuration towards the spatial subsetting strategy of keeping as few servers activated as needed to meet the loading demand. Accordingly, we deactivate a group of servers (i.e., send them to the standby mode) and redistribute the load among the rest of the active servers as per inverse-temperature. We measure the cooling power of the new configuration after the temperature settles. We compute the idle power saved in the new configuration due to the deactivated servers. If the idle power savings exceeds the cooling power increase going from the previous configuration to the new configuration then we keep the new configuration and proceed to the next refinement step by de-activating another group of servers. Otherwise, we revert back to the previous configuration, ending our refinement process for the current loading decrease.

The above discussion addresses whether server groups, but not which server groups, are activated or de-activated. When moving towards spatial subsetting, it is a straightforward choice to deactivate the hottest server group. However, when moving towards inverse-temperature, the choice of which inactive server group to activate is harder. We choose the server group with the lowest idle power rating, assuming the data center is heterogeneous. If all the inactive server groups have the same idle power rating, then we choose from the cool, warm, and hot zones (Section 2.2), in that order.

The above discussion also implicitly assumes that the loading does not change during the convergence process. If it does, then servers are activated or deactivated appropriately depending on the change in the load and the desired response time, as described in Section 3.2. The resource allocator triggers these (de)activations independently of PowerTrade-d's refinement process. In such cases, the refinement process abandons its current convergence and starts afresh with the new configuration as the initial configuration.

PowerTrade-d addresses all the three limitations of the static schemes discussed in Section 2.2.1. In contrast to the static schemes, the dynamic scheme does not use any pre-computed across-zone load distribution ratios (first limitation); does not apply a fixed loading policy (spatial subsetting or inverse-temperature) to each zone (second limitation); and takes into account the actual loading of the data center, and the air flows (heat interactions) among the zones and servers (third limitation).

2.2.3 Other issues

Both PowerTrade-s and PowerTrade-d employ some form of spatial subsetting — PowerTrade-s applies spatial subsetting to the cool zone and PowerTrade-d approaches spatial subsetting when the loading decreases. Server wear-and-tear due to thermal cycling resulting from repeated activations and deactivations is a concern for spatial subsetting [8] (this problem is less severe in inverse-temperature which keeps all the servers activated). This problem can be mitigated by simply using a round-robin order among the cool-zone servers when choosing the servers to be activated.

Another concern with spatial subsetting is the power dissipation during the activation of servers [8]. However, because data center loading changes are smooth and slow over coarse time granularities, this overhead adds little to the net power.

Finally, data centers typically employ stateless frontend servers for handling requests and stateful backend servers for holding data. Blindly putting stateful servers into standby mode would make the servers' data unavailable for requests. This problem is incurred by any scheme that employs spatial subsetting or temporal subsetting (including PowerTrade). Addressing this issue may involve state-aware grouping of servers while defining PowerTrade's server groups. While this issue is beyond the scope of this paper, we emphasize that any solution would benefit from PowerTrade by jointly optimizing cooling and idle power.

3 SurgeGuard: Maintaining Response Time

Recall from Section 1 that response time degradation is a key issue for data center power-reduction techniques. The inverse-temperature assignment does not degrade the response time as it keeps all the servers activated (i.e., in active mode). However, the spatial subsetting assignment activates only a subset of servers to reduce the idle power. Thus, any increase in the loading requires more servers to be activated by transitioning from standby mode to active mode which takes time (e.g., transitioning from the ACPI S1 state, which consumes about 5 W in standby mode, takes about 1-2 s). As mentioned in Section 1 and later in Section 5, the data center loading changes almost continually, though smoothly, requiring frequent server activations which degrade response times. Because both PowerTrade-s and PowerTrade-d use spatial subsetting, this degradation is a concern.

3.1 SurgeGuard

Recall from Section 1 that we employ *SurgeGuard* to address this degradation based on the observation that the data center loading changes mostly smoothly over coarse granularities of time (e.g., over 30 to 60 minutes) as we show in Section 5. SurgeGuard uses well-known over-provisioning of the number of active servers beyond that needed by spatial subsetting so as to absorb increases in loading [8, 34]. SurgeGuard is a two-tier scheme which operates at two time granularities: To handle the common, smooth increases in loading, we over-provision by an amount called *server_reserve* to handle a maximum increase in the loading in a pair of consecutive coarse-grain intervals (e.g., one hour). However, uncommon,

abrupt loading surges may deplete the over-provisioned reserves, as discussed in Section 1. To address such surges, we replenish the reserves at a fine granularity (e.g., five minutes). Previous over-provisioning schemes do not address such surges which result in degraded response times. Because the over-provisioning required is relatively small for the increases in the loading in practice, we can maintain response times while incurring only a small increase in server idle power (e.g., real-world traces show an average and maximum loading increase from one hour to the next of 3% and 10%, respectively).

Server_reserve can be chosen based on the worst-case increase in the loading in the past from one interval to the next or on the amount of loading change that can be tolerated for a desired power-performance goal. Different server_reserve values can be used for busy periods of the day or busy days of the week, if at all needed.

Because SurgeGuard provides an over-provisioning over spatial subsetting, we first describe how to determine the number of servers needed for a given loading in spatial subsetting while ignoring transition delays. SurgeGuard simply pads this number by some amount to account for the delays. We compute the number using queuing theory. A data center can be modeled as a GI/G/m queue — i.e., an m-server queuing system serving requests with generalized arrival and request-size distributions where each server has a generalized service time distributions.

Using the well-known Allen-Cunneen approximation [2, 6] for the GI/G/m model, the response time and the number of servers needed to satisfy a given demand are related as follows:

$$\overline{W} = \frac{1}{\mu} + \frac{P_m}{\mu(1-\rho)}\left(\frac{C_A^2 + C_B^2}{2m}\right)$$

where

$\overline{W}$ is the mean response time,

$1/\mu$ represents the mean service time of a server,

λ is the mean request arrival rate,

$\rho = \frac{\lambda\varphi}{mf}$ represents the average utilization of a server,

φ is the mean request size,

$f = \mu\varphi$ is the service rate of a server,

m is the number of servers available to serve the request,

$P_m = \rho^{\frac{(m+1)}{2}}$ for $\rho \le 0.7$ and $P_m = \frac{\rho^m + \rho}{2}$ for $\rho > 0.7$,

and $C_A{}^2$ and $C_B{}^2$ represent the squared coefficient of variation of request inter-arrival times and request sizes, respectively.

The first term is the service time for a request and the second term represents the average waiting time in the queue. Spatial subsetting can monitor the average request arrival rate and request sizes to determine number of servers required to achieve a desired response time.

To take response times into account, SurgeGuard exploits the fact that the loading change in one interval to the next usually does not exceed server_reserve. Accordingly, SurgeGuard simply adds server_reserve to the number of active servers needed for the average loading of a coarse-grain interval and uses the sum as the predicted number of active servers for the next interval. We obtain the average loading in an interval by monitoring the loading, and the number of active servers needed for a given average loading and desired response time by using the above expression. If the average loading decreases during an interval compared to the previous interval then the predicted number for the next interval decreases accordingly and the next interval begins by deactivating the excess active servers. However, in the uncommon case, the loading may increase during an interval to the point that server_reserve is completely depleted (occurs in 2.90% of intervals in our traces). To handle this case, we replenish server_reserve by activating more servers at the finer granularity of *mini-intervals* (e.g., 5 minutes). The mini-intervals should be short enough so that server_reserve is not exceeded in all but extreme cases and long enough that unnecessary replenishments due to near-instantaneous loading changes are avoided. We empirically found that five-minute mini-intervals work well. Server_reserve was exceeded only in about 0.15% of all the five-minute mini-intervals in our traces meaning that SurgeGuard was able to maintain response times for 99.85% of all the mini-intervals.

Repeatedly activating and deactivating the servers at the mini-interval granularity would waste power and increase server wear-and-tear [8]. To avoid these problems, we only activate the servers to maintain server_reserve but do not deactivate servers if the loading decreases at the mini-interval granularity. SurgeGuard deactivates any excess active servers at the end of the coarse-grain interval. Because the depletion of server_reserve is rare and because the excess servers stay activated only till the end of the current interval, not deactivating the excess servers as soon as the mini-interval ends, adds only a modest amount of idle power. Furthermore, to avoid large current surges due to simultaneously deactivating a large number of servers at the end of each interval, we stagger the deactivations over a few minutes. Server activations occur over multiple mini-intervals and are automatically staggered; each mini-interval activates at most server_reserve number of servers which is small enough so as not to create large current-surge problems.

Finally, while spatial subsetting can be augmented with over-provisioning as described above, augmenting temporal subsetting may be hard: While it is relatively straightforward to determine *how many* servers should be active to match the expected increase in the loading, deciding *when* to activate the servers may require predicting the arrival of individual requests, which seems harder than predicting overall loading changes.

3.2 Combining SurgeGuard with PowerTrade

SurgeGuard addresses spatial subsetting's response-time degradation. As such, SurgeGuard can be applied to any scheme that employs spatial subsetting as is the case with PowerTrade-s. SurgeGuard provides the predicted number of required servers (including server_reserve) at every interval. PowerTrade-s uses the predicted number to determine its across-zone and within-zone load distributions (Section 2.2.1). PowerTrade-d uses the number to determine whether the load has increased or decreased in order to move the server configuration towards inverse-temperature or spatial subsetting. respectively (Section 2.2.2). Further, each refinement step in PowerTrade-d takes about the same time as a mini-interval in SurgeGuard. Consequently, at the end of each mini-interval, (1) SurgeGuard replenishes its server_reserve, if

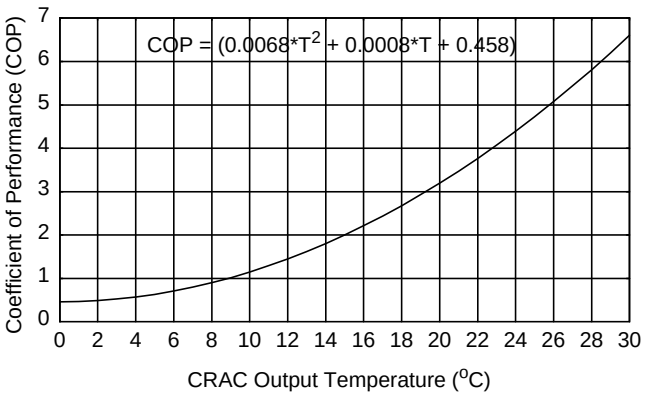

Figure 5: COP Curve

need be, and (2) PowerTrade-d chooses the better of its previous and new configurations, as described in Section 2.2.2, and determines whether its refinement has converged or should be continued.

4 Experimental Methodology

We employ simulators and real-world traces to evaluate our techniques. We use AirPAK [1], a computational fluid dynamics (CFD) simulator by Ansys Inc. (formerly Fluent Inc.) to model cooling in a data center.

4.1 Data Center Model

The simulated data center's parameters are consistent with those used in previous studies [21, 22, 31]. The data center (40' x 12' x 30') consists of four rows of 1120 server blades with each row containing seven 40U racks. For ease of configuration, we group four servers into a 4U server block. As shown in Figure 1, the data center employs the standard configuration of alternating hot and cold aisles to facilitate air flow and to avoid mixing of hot air with cold air [25, 33]. Our servers are modeled after the state-of-the-art servers (e.g, IBM Systems x3650 M2 or HP Proliant DL3xx series [10]) with power consumption of 100 W when idle, 300 W at 100% utilization, and 5 W in standby mode. Because server power varies almost linearly with server utilization [30], a 40%-utilized server consumes 100 + (300-100) x 0.4 = 180 W. Each server has a volumetric flow rate of 0.068 m^3/s. Because modeling variable-speed fans is hard, we conservatively assume that the server fans run at the highest speed for good airflow but consume low power (~2W) in *all* schemes. There are four CRAC units in the center, each of which pushes chilled air at 15^{o}C into a raised floor plenum (1.5 ft. high) at a rate of 9000 ft^3/min. The cool air enters the cold aisles (inlets of servers) and hot air exits the hot aisles (outlets of servers) to the CRAC exhaust vents.

Recall that cooling power = server power/COP and that the COP is a decreasing function of the temperature in the room. The higher the temperature of the room, the lower the temperature of the air output by the CRAC units for quick removal of the heat, the lower the COP. Consequently, the COP is an *increasing* function of the CRAC output temperature (i.e., the lower the output temperature, the lower the COP). We used the function in Figure 5, which models the chilled-water CRAC units at the HP labs Utility Data Center [21]. To determine the CRAC output temperature at any point in time of data center operation, we use the following methodology from [21]. We assume a safe server outlet temperature, T_{safe}, of 25 ^{o}C. From the simulator, we obtain the average outlet temperatures T_{server} of 80 hottest servers for a given combination of loading and choice of schemes for idle power and cooling power reduction. Now $T_{adj} = T_{safe}$-T_{server} is the amount by which the CRAC output temperature, T_{out}, needs to be adjusted to cool the servers to the safe temperature. The adjusted new CRAC output temperature is $T_{new} = T_{out} + T_{adj}$. If T_{adj} is negative, the output temperature will need to be reduced to provide more cooling and vice versa. Based on the COP function shown in Figure 5, T_{new} determines the actual COP which, in turn, determines the cooling power.

We validated our data center model by matching the temperature profile reported in [21, 31] under uniform distribution of a fixed loading across all the servers.

4.2 Implementation of various schemes

For inverse-temperature, we simulate the implementation described in [21] where each server's temperature is sensed for the purpose of load assignment. To implement spatial subsetting, we use the *GI/G/m* model in Section 3.1 to compute the number of servers required to achieve a desired response time for a given loading demand — i.e., request arrival rate (λ), request size (φ), and their coefficients of variation (C_A and C_B). These loading parameters come from real-world traces described in Section 4.4. We assume a response time ($\overline{W}$) of 6 ms and a service rate (f) of 2.6 Gbytes/sec, consistent with [8]. We verified that this response time was achievable for each trace provided an appropriate number of servers were activated. To supplement spatial subsetting with SurgeGuard, we add server_reserve to the number computed above to obtain the total number of servers to be activated. We describe how we compute server_reserve in Section 4.3. To alleviate hot spots as much as possible, we choose the active servers such that they are physically distributed throughout the data center.

To implement PowerTrade, we divide the data center into three zones based on a calibration run of loading the data center at 50% and uniformly distributing the load across all the servers. We assign servers to zones based on three suitably-separated ranges of server outlet temperatures. Out of 280 server blocks, 192 are in the cool zone, 62 in the warm zone and 26 in the hot zone. For PowerTrade-s, we use the same calibration run to determine the cool-zone-to-warm-zone iso-temp ratio to be 2:1 (Section 2.2.1). The cool-zone-to-hot-zone iso-temp ratio is 10:1, which is high implying that the hot zone is loaded only as a last resort when the demand exceeds the other zones' capacity. For PowerTrade-s++, we use seven calibration runs to obtain cool-zone-to-warm-zone iso-temp ratios of 4.3:1, 2.7:1, 2:1, 1.7:1, 1.6:1, 1.5:1 and 1.4:1 respectively, at seven different loadings of 30%, 40%, 50%, 60%, 70%, 80% and 90%. The cool-zone-to-hot-zone iso-temp ratios remained close to 10:1 across all the loadings. For PowerTrade-d, we simulate the refinement process to determine how many and which servers are activated, and the corresponding cooling power. To observe the time taken to reach steady-state temperature after a refinement step, we simulate the transient temperature of our data center. We found that starting from zero loading. the data center reached the steady-state temperature at 50% loading in about 3-4 minutes. Because most changes in the loading are smaller, each refinement step takes even less time.

4.3 Response Time Analysis

For SurgeGuard, we find the largest increase in the loading in a pair of consecutive one-hour intervals in each of our traces (Section 4.4). We then apply the GI/G/m model in Section 3.1 to determine server_reserve to be the difference in the number of servers needed for the pair. We found that a value of 112 (10% of the total number of servers) for server_reserve typically works well.

In our results, we compare SurgeGuard against the original spatial subsetting (without SurgeGuard) and temporal subsetting such as PowerNap (Section 1) in terms of response time. The queuing model in Section 3.1 determines the number of servers needed for a given loading and desired response time. However, the model does not account for the latency of the transition from standby mode to active mode. This latency is seen in the original spatial subsetting and temporal subsetting, and also in SurgeGuard in the uncommon case of full depletion of server_reserve within a mini-interval (Section 3.1). Though we use the *GI/G/m* model to compute the number of active servers needed in spatial subsetting (Section 3.1 and Section 4.2), we could not find a *GI/G/m* that captures this latency. Therefore, we compute the response time while accounting for the transition latency using the *GI/G/1* model *with exceptional first service* given by Welsh [35]:

$$\overline{W} = \frac{1}{\mu} + \frac{\rho}{\mu(1-\rho)}\left(\frac{C_A^2 + C_B^2}{2}\right) + \frac{2E[I] + \lambda E[I^2]}{2(1+\lambda E[I])}$$

where

$\overline{W}$ is the mean response time,

$1/\mu$ represents the mean service time of a server,

λ is the mean request arrival rate,

$\rho = \frac{\lambda}{\mu}$ represents the average utilization of a server,

C_A^2 and C_B^2 represent the squared coefficient of variation of request inter-arrival times and request sizes, respectively, and

E[I] represents the mean initial set-up time for the exceptional first service.

This model assumes that every request arriving while the server is idle experiences the exceptional service. However, in both the original spatial subsetting and SurgeGuard, a newly-transitioned server remains active even while idling for the rest of the interval. Thus, the server does not impose the exceptional service on all but those requests that arrive before or during the single transition. Therefore, this model provides an upper bound on the response times for the original spatial subsetting and SurgeGuard.

Using T_t to denote the transition time in our case, $E[I] = T_t$, and $E[I^2] = T_t^2$ for servers in standby mode, and $E[I] = E[I^2] = 0$ for servers in active mode. We assume T_t to be 1 s based on Linux's ACPI S1 transition times [11]. We assign values for μ, λ, C_A, and C_B, as described before for the *GI/G/m* model in Section 4.2.

4.4 Traces

We use the following traces provided by the Internet Traffic Archive [17]: (1) Fifa WorldCup 1998 — 14-day trace shown in Figure 6, (2) University of Berkeley's home IP server — 14-day trace shown in Figure 7, (3) ClarkNet WWW server — 14-day trace shown in Figure 8, (4) NASA Kennedy Space Center Web

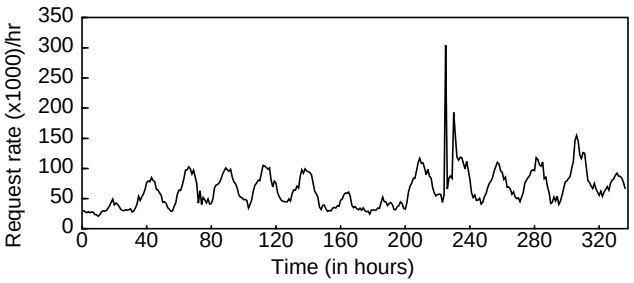

Figure 6: WorldCup 98 Trace

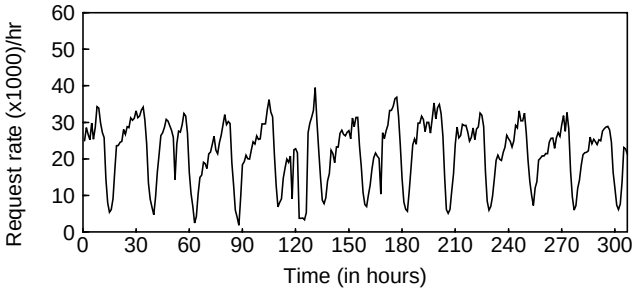

Figure 7: UC Berkeley IP Trace

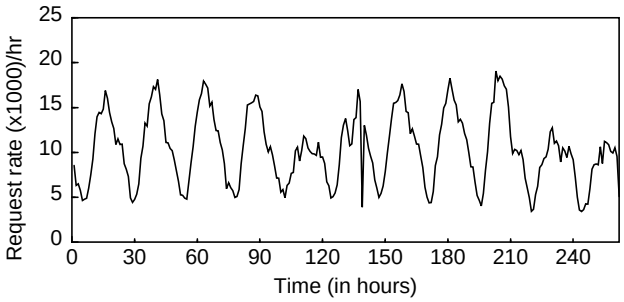

Figure 8: Clarknet-HTTP Trace

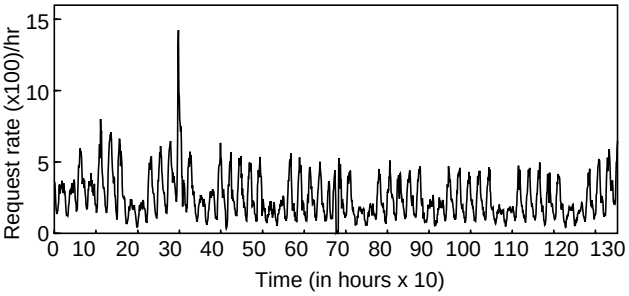

Figure 9: NASA-HTTP Trace

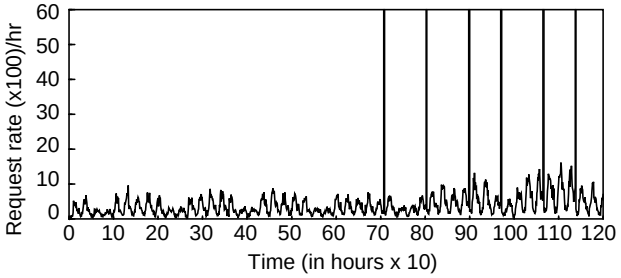

Figure 10: Saskatchewan-HTTP Trace

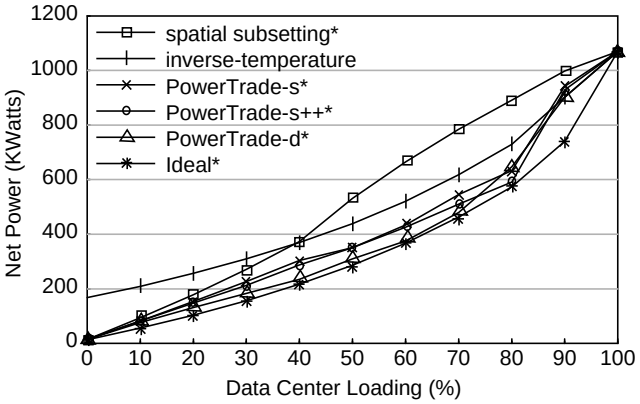

Figure 11: Net Power (* includes SurgeGuard)

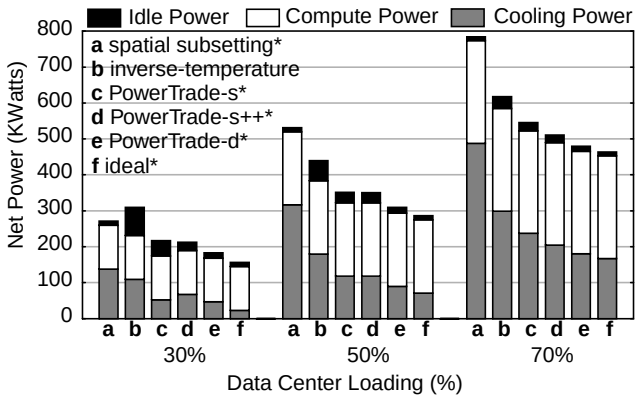

Figure 12: Net Power Breakdown (* includes SurgeGuard)

server — 60-day trace shown in Figure 9, and (5) University of Saskatchewan's WWW server — 60-day trace shown in Figure 10. We analyze the traces in terms of power and response time in Section 5.

5 Results

We start by quantifying PowerTrade's power reduction over spatial subsetting and inverse-temperature at various fixed loadings. To understand the power savings, we show both a breakdown of the net power and the temperature of the zones for the different schemes. We also show the sensitivity of our power reduction to the servers' idle power rating. Next, we present our total energy savings under real loadings seen in the traces. Finally, we present response times with and without SurgeGuard.

5.1 Net Power Savings at fixed loadings

We compare PowerTrade against spatial subsetting and inverse-temperature to show PowerTrade's power savings. Figure 11 shows the various schemes' net power consumption along the Y-axis at different loadings of the data center along the X-axis. The net power includes the cooling power, server power (= compute power + idle power) of active servers and standby power of inactive servers (for spatial subsetting and PowerTrade). We apply SurgeGuard to PowerTrade and spatial subsetting to alleviate response time degradation. Inverse-temperature naturally incurs no response time degradation by keeping all the servers active. Though PowerTrade and spatial subsetting incur extra idle power due to SurgeGuard, alleviating response time degradation is important for a fair comparison (recall that SurgeGuard's server_reserve is only 10% of all the servers present in the data center). The graph also includes an *ideal* curve which takes the server power + standby power of spatial subsetting combined with SurgeGuard and distributes this power among all the servers using inverse-temperature while ignoring the idle power of the excess servers (all servers - active servers of spatial subsetting). Thus, the ideal curve has the minimum server power (from spatial subsetting) and the minimum cooling power (from inverse-temperature). From Figure 11, we see that spatial subsetting performs better than inverse-temperature for lower loadings where hot spots are rarer and thus saving idle power is more important than saving cooling power. And the converse is true for higher loadings. The static PowerTrade schemes perform better than both spatial subsetting and inverse-temperature for most of the loading span (inverse-temperature performs slightly better at 87% loading and above). PowerTrade-d is better than both for the entire span while staying close to our ideal in the 40-70% loading range.

In Table 1, we show the percent reduction in net power achieved by PowerTrade and the ideal over the *better* of spatial subsetting and inverse-temperature at various loadings. We see that for the range of 30-70% loadings where data centers operate most of the time [3, 5] (confirmed in Section 5.2), PowerTrade-d achieves substantial power savings of 22-36%. Recall from Section 1 that because power is a major component in the recurring cost of a data center, such improvements are significant. At 80% loading, the static PowerTrade schemes perform slightly better than PowerTrade-d. This exception is due to the fact that at higher loadings, the temperature is sensitive to even a slight increase in server utilization and the group size for PowerTrade-d is too coarse to capture this sensitivity.

5.1.1 Net Power Breakdown

We explain the above power savings by breaking down the net power into its components — idle, compute, and cooling. Figure 12 shows these components (Y-axis) for the low-power schemes at three typical data center loadings (X-axis). We choose only three loadings to avoid overcrowding the graph with too many bars. Ideal's idle power comes from SurgeGuard's extra servers added for iso-response-time comparison, as discussed before, and standby power of inactive servers.

Spatial subsetting incurs only a little idle power, which is contributed by SurgeGuard's extra servers and by the servers in

Table 1: Net Power Savings (* includes SurgeGuard)

Loading	PowerTrade -s*	Power-Trade-s++*	PoweTrade-d*	Ideal*
10%	12%	12%	18%	39%
20%	14%	17%	26%	42%
30%	16%	22%	33%	42%
40%	19%	22%	36%	41%
50%	20%	20%	30%	35%
60%	17%	18%	28%	29%
70%	12%	17%	22%	25%
80%	13%	19%	11%	21%
90%	-5%	-3%	0%	18%

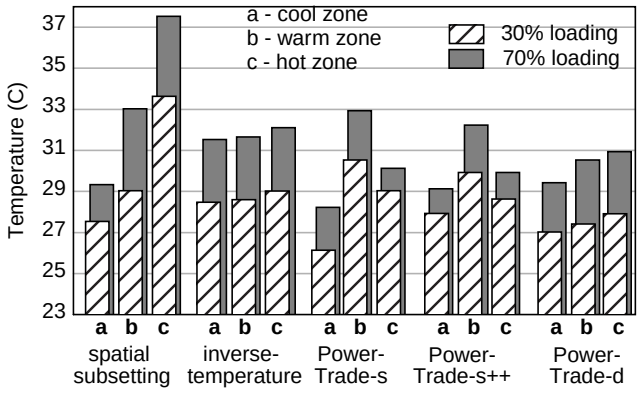

Figure 13: Temperature variation across zones

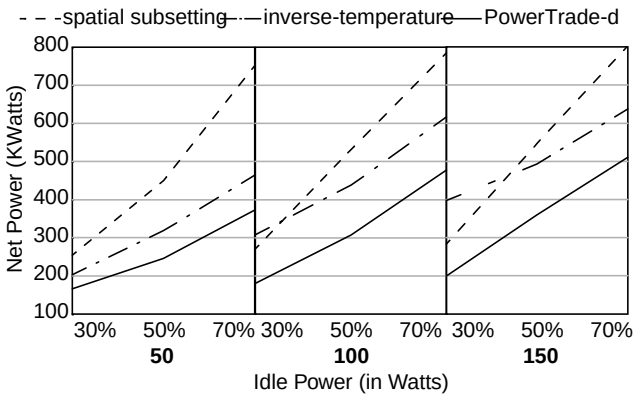

Figure 14: Idle power sensitivity

standby mode, but significantly high cooling power due to hot spots created by concentrating the loading on a subset of servers. Conversely, inverse-temperature incurs high idle power due to keeping all the servers activated but low cooling power by avoiding hot spots. PowerTrade incurs both low idle power and low cooling power by avoiding hot spots while keeping as few servers activated as needed to alleviate response time degradation. PowerTrade's cooling power is lower than that of inverse-temperature because cooling power depends not only on the total amount of server heat (recall that Cooling Power = ServerPower/COP*)* but also on the temperature because the COP is a *non-linear* decreasing function of temperature (i.e., lower temperature implies better COP). By keeping fewer servers active to reduce idle power, PowerTrade also reduces the cooling power. Compute power is not affected by the choice of the scheme and hence remains the same for all the schemes at a particular loading. At 50% loading, PowerTrade-s and PowerTrade-s++ have identical power components because their iso-temp ratios are equal at this loading.

5.1.2 Temperature Profile

To explain the cooling power component in the above breakdown, we show the temperatures of the zones for the schemes. Figure 13 shows the zone temperatures along the Y-axis at 30% and 70% loading of the data center for spatial subsetting, inverse-temperature, and PowerTrade along the X-axis. Spatial subsetting has the highest temperature variation across the zones whereas the temperature is even for inverse-temperature. The static PowerTrade schemes perform better than spatial subsetting but worse than inverse-temperature. This worse behavior stems from PowerTrade-s's limitations — fixed policy per zone and ignoring heat exchange across zones (Section 2.2.1). For PowerTrade-d, not only the across-zone temperature variation is small (close to that of inverse-temperature) but also the absolute temperatures of the zones are lower than those in inverse-temperature due to lower idle power.

5.1.3 Sensitivity to Idle Power

The idle power rating of the servers is an important factor in the server's net power and the amount of savings achieved by our schemes. Further, this rating has been improving, if slowly, over server generations. To capture the effect of these improvements, we show our power savings as the idle power rating is lowered while holding the server power at 100% utilization to be a constant 300 W as before. In Figure 14, the Y-axis shows the net power consumption for various schemes at the three fixed loadings of 30%, 50%, and 70%, and the X-axis shows the idle power varied as 50 W, 100 W (default), and 150 W. We note these idle-power values cover both the state-of-the-art and near-future server offerings. We show only PowerTrade-d and not PowerTrade-s to avoid crowding the graph.

Spatial subsetting performs worse than inverse-temperature as we decrease the idle-power rating because spatial subsetting's reduction of idle power becomes less critical than inverse-temperature's reduction of cooling power. On the contrary, as the idle-power rating is increased, reducing the idle power becomes more important giving spatial subsetting the upper hand at higher loadings. PowerTrade-d performs the best over the whole range of idle-power ratings, achieving significant power savings even at 50-W rating (18-23% improvement over inverse-temperature).

5.2 Total Energy Savings under real loadings

While the previous section shows net power, here we present total energy savings of PowerTrade over inverse-temperature and spatial subsetting summed over all five of our traces (recall that total power = net power + distribution losses). We show energy instead of power to capture the time duration of the traces. All the schemes except inverse-temperature are combined with SurgeGuard for iso-response-time comparison, as discussed before. To compute the energy for each trace under each scheme, we determine the loading in each one-hour interval of each trace and look up the power for that loading under each scheme in Figure 11. Because the absolute loadings across the traces vary, we normalize the loadings by setting the maximum loading in each trace to be 100% and scale the rest of the trace accordingly. To simplify the energy calculation, we ignore the fact that the COP, and hence the cooling power, changes with the temperature during the time the temperature settles after loading changes. Instead, we assume that the temperature settles instantaneously. Because loading changes and hence the temperature changes are not large, our approximation is reasonable. However, for PowerTrade-d, we assume that a 1% loading change triggers the refinement process. During each step of the process (5 minutes) we assume the power of the previous (sub-optimal) configuration to be conservative. Because the traces cover unequal time spans, we consider only the first 14 days from each trace to compute the sum over the traces.

Before we discuss PowerTrade's energy savings, we make a few observations about our traces shown in Figure 6 through Figure 10. The loadings in the traces are often well below the maximum, providing opportunity for idle power reduction (e.g., 42%, 67%, and 88% of the traces are, respectively, under 30%, 50%, and 70% loading). Further, loadings vary smoothly and slowly over hours-long periods of time (the X-axis spans many days). The average and maximum increase in the loading from one interval to the next for the traces in Figure 6 through Figure 10, is 3% and 10%, respectively. Consequently, SurgeGuard's server_reserve is small — 112 or 10% of all servers. Because of the variations being slow, PowerTrade-d's refinement process took only 1.2 steps to converge for each loading change in the traces. Table 2 shows the total energy breakdown and energy savings for each scheme. Total energy is the sum of the server energy, cooling energy, and power distribution losses (Section 2.2). We assume that the distribution losses are 10% of the server power as per [14]. In all the three PowerTrade schemes, power savings at fixed loadings translate well into significant energy savings under real loadings.

5.3 Response Time

Recall that PowerTrade achieves power reduction by keeping only a subset of servers active and putting the rest into standby mode. Also recall that SurgeGuard alleviates response time degradation by padding PowerTrade's active server count at the first-tier interval granularity with server_reserve. Further, SurgeGuard replenishes server_reserve at the second-tier mini-interval granularity if any loading surge depletes the reserve. However, loading surges that exceed the reserve result in response time degradation as new servers are transitioned from standby. To evaluate SurgeGuard's effectiveness, we show in Table 3 the response time degradation of PowerTrade-d with three configurations for SurgeGuard — no SurgeGuard,. one-tier SurgeGuard, and full SurgeGuard — for our traces (Figure 6 through Figure 10). In one-tier SurgeGuard, server_reserve is not replenished at mini-intervals and continues to get depleted until either the interval ends or the loading exceeds the capacity of the active servers incurring response time degradation as new servers are activated. We show only PowerTrade-d because it has better power savings than the static PowerTrade schemes. We compute the response times for the *current* interval as a mean of two parts: (1) response time of servers that either are active from the *previous* interval or are part of server_reserve (if SurgeGuard is present) and (2) response time of servers that are newly activated because the servers in the first part are insufficient. We determine the number of servers in the second part as the difference between (1) the *actual* number of servers needed for the current interval using the *GI/G/m* model in Section 3.1 and (2) the number of servers in the first part. To compute the response time of each part, we use the *GI/G/1 with exceptional first service* model in Section 4.3 with transition times being 0 for the first part and 1 s for the second part (like ACPI S1 [11]). The overall response time is the mean over the *actual* number of servers. We assume 1-hour intervals, 5-minute mini-intervals, and server_reserve to be 112 (10% of all servers). The column "Distribution (% success)" shows the percent of all mini-intervals where the loading falls within the capacity of the active servers (with or without SurgeGuard). The column "Average relative RT" shows the response time normalized against the desired response time of 6 ms.

Table 2: Total Energy Savings

Scheme (* includes SurgeGuard)	**Server Energy (MWh)**	**Cooling Energy (MWh)**	**Distribution Losses (MWh)**	**Total Energy (MWh)**	**% Energy Savings**
Inverse-temperature	169	113	17	298	---
Spatial subsetting*	140	146	14	299	---
PowerTrade-s*	113	104	11	228	23%
PowerTrade-s++*	113	97	11	220	26%
PowerTrade-d*	107	89	11	207	30%

Table 3: SurgeGuard's Response Time (RT): Distribution & Average

Trace	**Distribution (% success)**			**Average relative RT**		
	No Surge-Guard	1-tier Surge-Guard	Full Surge-Guard	No Surge-Guard	1-tier Surge-Guard	Full Surge-Guard
worldcup98	84.7	97.85	99.93	3.05	1.05	1.014
NASA-http	79.2	96.72	99.72	3.24	1.07	1.018
UC-Berkeley	83.3	97.30	99.91	3.07	1.06	1.014
ClarkNet-http	84.1	97.65	99.93	3.07	1.04	1.014
Saskatchewan	80.2	96.86	99.65	3.18	1.08	1.018
average	81.6	97.10	99.85	3.12	1.07	1.017

Before we discuss PowerTrade's response time, we make a few more observations about our traces shown in Figure 6 through Figure 10. The loading varies continually confirming that putting all un-utilized servers in standby mode to save power (e.g., original spatial subsetting) would degrade response time. However, because the variations are smooth and slow, SurgeGuard is likely to perform well. In Table 3, PowerTrade-d without SurgeGuard cannot meet the loading demand in more than 15% of all the mini-intervals (see "Distribution (% success)"), degrading response time by more than a 2x factor. PowerTrade-d with one-tier SurgeGuard improves on these numbers and covers about 97% of all the mini-intervals maintaining average response time within 7%. In contrast, PowerTrade-d with full SurgeGuard meets the loading demand in more than 99.85% of all the mini-intervals and maintains average response time within an almost-imperceptible 1.7%. The difference between one-tier SurgeGuard and full SurgeGuard shows the value of SurgeGuard's second tier, which is our novelty, in handling uncommon and abrupt, but important, loading surges (Section 1).

Next, we compare PowerTrade-d against spatial subsetting and PowerNap, a temporal subsetting scheme which puts servers into standby mode upon service completion if no further requests are present [20]. We perform this comparison by varying the interval length as 30, 60 (default), and 180 minutes while holding the server_reserve and mini-interval length constant at 112 and 5 minutes, respectively. In Figure 15, the X-axis shows the various interval lengths and the Y-axis shows the normalized response times averaged over all our traces for PowerNap, spatial subsetting, PowerTrade-d without SurgeGuard, and PowerTrade-d with one-tier SurgeGuard. We do not show PowerTrade-d with full SurgeGuard

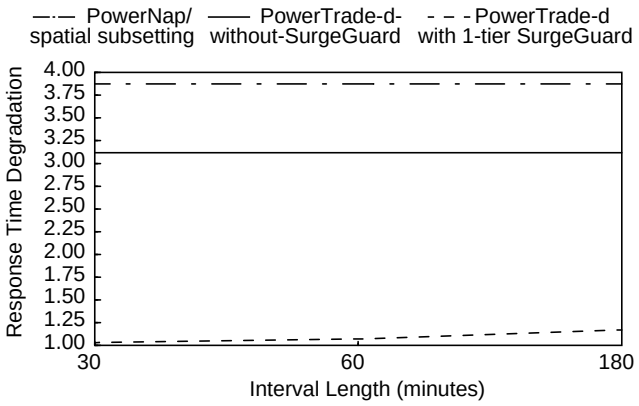

Figure 15: Interval length sensitivity

because Table 3 shows the degradation to be 1.7% which would not be visible in the graph. For PowerNap, we used *M/G/1 with exceptional first service* model they adopted. We assume that spatial subsetting activates new servers as the loading increases, incurring the same delays as PowerNap (original spatial subsetting assumes that loading changes are infrequent and so does not protect against increases in the loading). Therefore, we show PowerNap and spatial subsetting in the same curve.

PowerNap, spatial subsetting, and PowerTrade-d-without-SurgeGuard considerably degrade response time — by more than a factor of 2X — due to the standby-to-active transition delay. Because these schemes do not use intervals, their response times do not change with interval length. Because PowerTrade-d-without-SurgeGuard keeps more servers active than PowerNap for avoiding hot spots, PowerTrade-d-without-SurgeGuard performs a little better than PowerNap. We note that the PowerNap paper reports much less degradation (around 12% degradation at 40% loading) because the paper assumes standby-to-active transition times of 10 ms whereas we assume 1-s transition times consistent with Linux documentation [11]. Recall from Section 1 that it is not easy to apply SurgeGuard to PowerNap for reducing this degradation because using SurgeGuard for temporal subsetting, as in PowerNap, amounts to predicting the arrival times of individual requests. In contrast to PowerNap and spatial subsetting. PowerTrade-d with one-tier SurgeGuard maintains response time within 3%, 7% and 17% at interval lengths of 30 minutes, 1 hour and 3 hours, respectively, and PowerTrade-d with full SurgeGuard (not shown) degrades response time only by 1.7% (at 60-minute intervals).

6 Related Work

Prior work on data center power can be divided into two categories: server power optimizations and cooling power optimizations. Here we discuss prior work other than spatial subsetting [7, 26], inverse-temperature [21, 31] and PowerNap [20] which we have already discussed. In the first category, Ranganathan *et al.* [29], Fan *et al.* [14], Raghavendra *et al.* [27] propose optimizations for power and resource management at the enterprise level by combining resources from different enterprise components. Lim *et al.* [19] propose to use low-power embedded CPUs and flash-based disk caching in data center servers. However, these schemes do not address the implications on response times. Two other proposals propose over-provisioning to address response time degradation in two different contexts. Urgaonkar *et al.* [34] propose a single tier of over-provisioning in allocating servers of a shared, multi-service cluster to each service. This work considers only allocation and not any power optimizations. Chen *et al.* [8] consider response time impact in data centers while optimizing server power using DVFS. The authors use a queuing-theoretic predictive scheme to estimate the combination of number of active servers and clock frequency. This estimate includes some over-provisioning to alleviate response time degradation. However, the scheme does not account for depletion of the over-provisioned resources due to spikes in the data center loading, unlike SurgeGuard's second tier. Choi *et al.* [9], Sharma *et al.* [32] and Elnozahy *et al.* [12] also explore DVFS for server applications. However, DVFS has several limitations as discussed in Section 1. Heath *et al.* [16] and Rusu *et al.* [30] propose power management techniques for matching workloads to servers in heterogeneous server clusters. VirtualPower [24] integrates power management with virtualization in data centers by coordinating the power management policies of independent guest VMs.

In the second category of cooling optimizations, Patel *et al.* [25] and Sullivan [33] optimize layout of the data center to improve air flow. C-Oracle [28], Weatherman [22] propose software infrastructure to predict heat profiles in the data center for dynamic thermal management. Mukherjee *et al.* [23] propose a software infrastructure to monitor and control resource management to enable global thermal-aware scheduling decisions in a data center. These schemes are orthogonal to PowerTrade which optimizes the sum of cooling and server power.

While the above schemes optimize either server power or cooling power, PowerTrade is the first proposal to optimize the two together.

7 Conclusions

Server power and cooling power amount to a significant fraction of modern data centers' recurring costs. Previous server idle power optimizations concentrate the data center loading on as many servers as needed while putting the rest into low-power standby modes, but increase the cooling power by creating hot spots. Previous cooling optimizations, on the other hand, reduce the cooling power by distributing the loading among all the servers to achieve uniform temperature, but increase the idle power by keeping all the servers active. Further, server idle power optimizations also degrade response times due to standby-to-active transition delays,

We proposed a novel joint optimization of idle power and cooling power, called *PowerTrade*, which trades-off idle power and cooling power for each other, thereby reducing the total power. We addressed response time degradations via *SurgeGuard*, which over-provisions the number of active servers beyond the current need so as to absorb future loading increases. SurgeGuard is a two-tier scheme which uses well-known over-provisioning at coarse time granularities (e.g., one hour) to absorb the common, smooth increases in the loading, and a novel fine-grain replenishment of the over-provisioned reserves at fine time granularities (e.g., five minutes) to handle the uncommon, abrupt loading surges. We showed that PowerTrade and SurgeGuard reduce the total power for real-world traces by 30% compared to previous low-power schemes while maintaining response times within 1.7%.

Acknowledgments

We thank Peter Chen, our shepherd, and the anonymous reviewers for their helpful comments and suggestions. This work is supported, in part, by the National Science Foundation (Award Number: 0621457).

References

[1] AirPAK. Computational fluid dynamics (CFD) software by Ansys Inc. *http://www.ansys.com/products/airpak/default.asp*.

[2] O. Allen. Probability, statistics and queuing theory with computer science applications. 1990.

[3] L. A. Barroso and U. Hölzle. The case for energy-proportional computing. *IEEE Computer*, 40(12):33–37, 2007.

[4] C. Belady. Green grid data center power efficiency metrics, PUE and DCIE. *White paper: Metrics & Measurements. http://www.thegreengrid.org*, 2007.

[5] P. Bohrer, E. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony. The case for power management in web servers. *Power aware computing*, 2002.

[6] G. Bolch, S. Greiner, H. Meer, and K. S. Trivedi. Queuing networks and markov chains: Modeling and performance evaluation with computer science applications. 1998.

[7] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle. Managing energy and server resources in hosting centers. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 103–116, 2001.

[8] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam. Managing server energy and operational costs in hosting centers. In *SIGMETRICS '05: Proceedings of ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, vol. 33, pages 303–314, 2005.

[9] J. Choi, Y. Kim, A. Sivasubramaniam, J. Srebric, Q. Wang, and J. Lee. Modeling and managing thermal profiles of rack-mounted servers with thermostat. In *HPCA '07: High Performance Computer Architecture*, pages 205–215, 2007.

[10] Standard Performance Evaluation Corporation. http://www.spec.org/power_ssj2008/results/power_ssj2008.html.

[11] Linux Documentation. latest release 2.6, http://www.kernel.org/doc/documentation/power/states.txt.

[12] E. Elnozahy, M. Kistler, and R. Rajamony. Energy-efficient server clusters. In *Proceedings of the 2nd Workshop on Power-Aware Computing Systems*, pages 179–196, 2002.

[13] U.S. EPA. Report to congress on server and data center energy efficiency. In *U.S. Environmental Protection Agency, Tech Report*, 2007.

[14] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *34th International Symposium on Computer Architecture*, pages 13–23, 2007.

[15] J. Hamilton. Internet-scale service infrastructure efficiency. In *ISCA '09 keynote, http://perspectives.mvdirona.com/2008/11/28/CostOfPowerInLargeScaleDataCenters.aspx*, 2009.

[16] T. Heath, B. Diniz, E. V. Carrera, W. M. Jr., and R. Bianchini. Energy conservation in heterogeneous server clusters. In *PPoPP '05: 10th ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 186–195, 2005.

[17] Traces in the Internet Traffic Archive. http://ita.ee.lbl.gov/html/traces.html.

[18] W. LeFebvre. CNN.com: Facing a world crisis. *http://www.tcsa.org/lisa2001/cnn.txt*.

[19] K. Lim, P. Ranganathan, J. Chang, C. Patel, T. Mudge, and S. Reinhardt. Understanding and designing new server architectures for emerging warehouse-computing environments. In *ISCA '08: Proceedings of the 35th International Symposium on Computer Architecture*, pages 315–326, 2008.

[20] D. Meisner, B. T. Gold, and T. F. Wenisch. Powernap: Eliminating server idle power. In *ASPLOS '09: Proceeding of the 14th conference on Architectural support for programming languages and operating systems*, pages 205–216, 2009.

[21] J. Moore, J. Chase, P. Ranganathan, and R. Sharma. Making scheduling "cool": Temperature-aware workload placement in data centers. In *Proceedings of USENIX* , 2005.

[22] J. Moore, J. Chase, and P. Ranganathan. Weatherman: Automated, online and predictive thermal mapping and management for data centers. In *ICAC '06: Proceedings of IEEE Conference on Autonomic Computing*, pages 155–164, 2006.

[23] T. Mukherjee, Q. Tang, C. Ziesman, S. K. S. Gupta, and P. Cayton. Software architecture for dynamic thermal management in data centers. In *COMSWARE*, 2007.

[24] R. Nathuji and K. Schwan. Virtualpower: Coordinated power management in virtualized enterprise systems. In *SOSP '07: Proceedings of 21st ACM SIGOPS symposium on Operating systems principles*, pages 265–278, 2007.

[25] C. D. Patel, C. E. Bash, R. K. Sharma, and A. Beitelmal. Smart cooling of data centers. In *Proceedings of the Pacific RIM/ASME International Electronics Packaging Technical Conference and Exhibition, IPACK*, 2003.

[26] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath. Load balancing and unbalancing for power and performance in cluster-based systems. In *Workshop on Compilers and Operating Systems for Low Power*, 2001.

[27] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu. No "power" struggles: Coordinated multi-level power management for the data center. In *ASPLOS XIII: Proceedings of the 13th conference on Architectural support for programming languages and operating systems*, pages 48–59, 2008.

[28] L. Ramos and R. Bianchini. C-Oracle: Predictive thermal management for data centers. In *Proceedings of High-Performance Computer Architecture, HPCA*, pages 111–122, 2008.

[29] P. Ranganathan, P. Leech, D. Irwin, and J. Chase. Ensemble-level power management for dense blade servers. *ISCA '06: Proceedings of the 33rd annual international symposium on Computer Architecture*, 34(2):66–77, 2006.

[30] C. Rusu, A. Ferreira, C. Scordino, and A. Watson. Energy-efficient real-time heterogeneous server clusters. In *RTAS '06: Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 418–428, 2006.

[31] R. K. Sharma, C. Bash, C. D. Patel, R. Friedrich, and J. Chase. Balance of power: Dynamic thermal management for internet data centers. *IEEE Internet Computing*, 9(1):42–49, 2005.

[32] V. Sharma, A. Thomas, T. Abdelzaher, K. Skadron, and Z. Lu. Power-aware QoS management in web servers. In *RTSS '03: Proceedings of the 24th International IEEE Real-Time Systems Symposium*, pages 63–72, 2003.

[33] R. F. Sullivan. Alternating cold and hot aisles provides more reliable cooling for server farms. In *Uptime Institute*, 2000.

[34] B. Urgaonkar, P. Shenoy, and T. Roscoe. Resource overbooking and application profiling in shared hosting platforms. *SIGOPS Oper. Syst. Rev.*, 36(SI):239–254, 2002.

[35] P. Welch. On a generalized M/G/1 queuing process in which the first customer of each busy period receives exceptional service. In *Operations Research*, vol. 12, pages 736–752, 1964.