SpinSmart: Exploring Optimal Server Fan Speeds to Improve Overall System Energy Consumption

Min Tian
mtian39@gwu.edu
Department of Electrical and Computer Engineering
George Washington University
Washington, DC

Arun Vishwanath
arvishwa@au1.ibm.com
IBM Research Australia

Guru Venkataramani
Suresh Subramaniam
{guruv, suresh}@gwu.edu
Department of Electrical and Computer Engineering
George Washington University
Washington, DC

ABSTRACT

Cost of data centers has risen sharply in the past few years. Today, it represents about 3% of total US energy consumption with projections to increase further in the coming years. In this paper, we focus on the server infrastructure and observe that workload consolidation techniques, which maximize power efficiency of server systems, do not automatically optimize the overall system power efficiency especially when compute engines and the corresponding on-board cooling systems are considered holistically. We design SpinSmart, a framework that explores optimal server fan speeds to minimize the overall system energy consumption. We explore core capping strategies that estimate the desired number of CPU cores to be used at any given time to minimize combined CPU+fan power. Our experimental results show that we are able to achieve 1) energy savings of up to 10% of total energy and 80% of cooling energy when compared to workload consolidation without core capping strategy; 2) cooling energy savings up to 42% when compared to the strategy that randomly assigns jobs to all the servers and cores.

CCS CONCEPTS

• **Hardware → Temperature control**; **Temperature optimization**.

KEYWORDS

Data centers, Fan speed control, Energy optimization

ACM Reference Format:
Min Tian, Arun Vishwanath, Guru Venkataramani, and Suresh Subramaniam. 2020. SpinSmart: Exploring Optimal Server Fan Speeds to Improve Overall System Energy Consumption. In *The Eleventh ACM International Conference on Future Energy Systems (e-Energy'20), June 22–26, 2020, Virtual Event, Australia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3396851.3402655

e-Energy'20, June 22–26, 2020, Virtual Event, Australia

ACM ISBN 978-1-4503-8009-6/20/06…$15.00
https://doi.org/10.1145/3396851.3402655

1 INTRODUCTION

As the number and size of active components (servers and network equipment) continue to dominate the energy landscape, we note that the cooling elements, which actively work to maintain the appropriate temperature inside data centers, account for almost 50% of total energy [1, 15]. At server level, fans also consume a significant part of the total energy, sometimes up to 51% according to the configuration of the servers [6]. Right-sizing the energy budgets of compute and cooling, while maintaining the Quality of Service (QoS) requirements in data centers, is a challenging problem [20]. Prior works, that optimize system energy under QoS constraints of workloads, typically consolidate the workload onto the fewest number of servers so that other servers can remain in sleep (inactive) states [7, 9, 10, 19]. While such a strategy is effective to reduce CPU power, it can inadvertently raise the thermal profile of active servers, resulting in increased power consumption of its fans. Consequently, the overall system power efficiency may be lower. This may also lead to increased QoS violations especially when fans are no longer able to handle increased workload, leading to frequent operational interruptions (to bring down system temperature).

In this paper we propose SpinSmart, a framework that explores server fan speed control through predicting CPU temperature with a thermal resistance-based model [18] . We design an optimal fan speed control algorithm with lookahead capability that determines system cooling needs based on past power profile. We also introduce additional optimization, such as core capping, that estimates the desired number of active cores among the available ones, such that we can minimize the overall system energy (that includes CPU and server fan components). We perform a detailed evaluation of SpinSmart using a combination of analytical modelling and real system experiments using Intel Xeon-based tower servers.

The contributions of our work are as follows:

- We observe that server consolidation techniques, which pack workloads onto the fewest number of servers possible, can lead to increased power consumption of server fans. We posit that the overall system power efficiency can be affected when the corresponding cooling systems are not considered for system energy optimization.
- We propose SpinSmart, a framework that explores how to determine optimal server fan speeds based on CPU power, thermal profile and ambient temperature (along with lookahead capability for adjusting fan speeds), such that the overall system power is minimized.

Table 1: States for server and CPU.

Symbol	Description
$C0_{(a)}S0$	There are tasks to process, both the system and CPU are active
$C0_{(i)}S0$	There are no tasks to process, both the system and CPU are active
$C1S0$	The CPU enters halt state (shallow package sleep state), while the rest of the system remains active
$C6S0$	The CPU enters deep sleep state, while the rest of the system remains active
$C6S3$	The system (including CPU) enters deep sleep sate

- We present a core capping strategy, where we determine the desired number of active cores at any given time, such that the combined CPU and fan power can be optimized. Our experimental results show that SpinSmart is able to 1) achieve energy savings of up to 10% of total energy and 80% of cooling energy when compared to workload consolidation that greedily assigns jobs to maximize sleep periods of inactive servers; 2) achieve up to 42% savings in cooling energy compared to a framework without workload consolidation.

2 BACKGROUND

In this section, we provide some background on server workload consolidation techniques and their effect on system power.

To reduce energy consumption of servers in data centers, prior works have proposed the use of low power states [7, 10]. However, higher job latency can be caused by placing servers into low power states, resulting in the failure to meet the Quality of Service (QoS) requirements [19]. Job service latency can be high due to (1) queuing effects - for example, this will happen if there are not enough servers to handle a burst of arriving jobs; (2) having to wake up from low power states. As shown in Table 1, for system and CPU, there are different sleep states that can save energy by power gating different components [7]. C_x state is the low-power state for processor, while S_x is the low-power state for the system. There is latency associated with waking up for both the system and CPU, and latencies are higher if CPU is in deeper sleep states, as observed in Table 2 for a typical server. These latencies are reported by Linux cpuidle driver [14]. Also, such higher wakeup latencies can impact job completion times, which may in turn lead to extra system energy consumption.

Frameworks that use low-power states [8, 19] to minimize system energy typically decrease the number of active servers/CPUs by concentrating jobs onto a few servers and allowing the rest of the CPUs to enter a deeper sleep state. This is broadly referred to as workload consolidation (WC). WC tends to concentrate the jobs onto as few servers as possible and let the remaining servers enter the package sleep state to save energy. On the other hand, when there is no WC, jobs tend to be assigned to servers randomly making it difficult for servers to go to sleep.

One of the advantages of WC is that by scheduling servers into active and sleep states based on workload requirements, they can minimize the energy cost while meeting the QoS requirements. To study the impact of the WC strategy on overall system power, we use one of the recently proposed state of the art frameworks,

Table 2: Wake-up latencies for different C-states and S-states [19].

State	Wake-up Latency
Core Sleep C1	10 μs
Core Sleep C6	82 μs
Package Sleep C6	1 ms
System Sleep	5 s

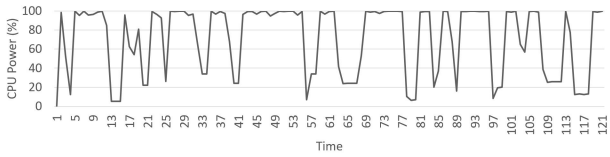

Figure 1: A CPU power trace of Workload-Adaptive Server Provisioning (WASP).

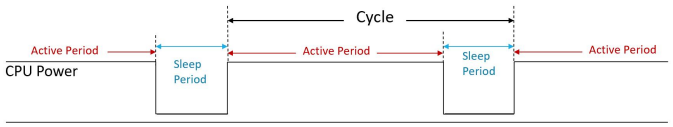

Figure 2: CPU power fluctuations in active and inactive (sleep) periods.

Workload-Adaptive Server Provisioning (WASP) [19], that orchestrates servers into different low power states.

Figure 1 shows the CPU power of a single server under 30% utilization on a 10-server configuration running websearch workload with 4 ms job size for WASP [19]. The utilization can be controlled by setting different job inter-arrival intervals according to job sizes. By subdividing the CPU power trace into active and package sleep periods, we observe a cyclic pattern in power consumption (Figure 2). This is because, the workload consolidation strategy packs all of its jobs to maximize server utilization (high CPU power), and puts the inactive servers into sleep mode (low CPU power). Such a strategy can save basic system energy (not including CPU part) through sleep states, however it would also lead to a high CPU utilization when the servers are active. We note that without judicious use of server workload consolidation, the cooling system could be under peak utilization as well. In this paper, we use CPU fan to understand server cooling requirements.

3 UNDERSTANDING SYSTEM POWER AND THERMALS

In this section, we study the relationship between CPU and fan. We utilize the thermal model from Wang et al. [18] and model a single CPU tower server with the necessary parameters in that model.

The thermal model for a tower server under steady-state conditions is given by (1), where the ambient temperature and the heat generation have stabilized. We note that stability is reached when the cooling system is capable of removing the heat generated and maintains a nearly constant temperature.

$$T_{CPU} = Q\left(\frac{C_3}{(n_f \cdot FS)^{n_R}} + C_4\right) + T_{amb} \quad (1)$$

where, T_{CPU} is the temperature of the CPU, T_{amb} is the temperature of the ambient environment, Q represents the heat generated by the CPU, n_f is the number of fans, and FS is the fan speed. Note that in the server we use for our experiments (PowerEdge T620), the speeds of all the fans are the same and therefore consider their thermal resistances R as a whole. C_3, C_4 and n_R are the parameters that need to be identified through experiments.

A steady state thermal model helps to predict CPU temperature under given ambient temperature, CPU power, and fan speed. It reveals the balance between the heat generated by CPU (CPU power), the heat transferred from CPU to the air and the CPU temperature; thus this model can also be used to generate desired fan speed under certain temperature and power requirement. According to the interface and tools we use in later sections, it takes about 1 sec to return the measurement of fan speed and CPU power, and around 3 sec to adjust fan speed. A steady-state based model would fail to predict temperature if the granularity of CPU power changes is much smaller than the time it takes for the system to reach its steady state after a CPU power change, that is, around 2 to 3 seconds. Therefore, we need to consider transient heat to predict CPU temperature accurately. To do so, we consider a time sloted system; here we use 1 sec as the prediction granularity.

According to transient heat transfer theory [18], the rate of change in a device's temperature, the rate of heat generated from the device and the rate of heat transferred from a device to air are relevant to predicting the temperature in the next time slot [18]. As shown in (2), Q is the heat generated from the device, $C_1 \frac{dT_{CPU}}{dt}$ represents the rate of CPU temperature change and $\frac{C_2}{R}(T_{amb} - T_{CPU})$ is the rate of heat transferred from CPU to the air. By including the parameters of transient heat transfer into our thermal model, we can reduce the error in temperature prediction. The thermal model with transient heat transfer is shown below:

$$C_1 \frac{dT_{cpu}}{dt} = \frac{C_2}{R}(T_{amb} - T_{cpu}) + Q \tag{2}$$

$$\Delta T_{cpu}(k+1) = (1 - \frac{\Delta t C_2}{C_1 R})\Delta T_{cpu}(k) - \frac{\Delta t}{C_1} Q \tag{3}$$

$$R = \frac{C_3}{(n_f \cdot FS)^{n_R}} + C_4 \tag{4}$$

$\Delta T_{CPU}(k)$ here is the difference between T_{CPU} and T_{amb} at slot $time = k$ ($\Delta T_{CPU}(k) = T_{CPU}(k) - T_{amb}(k)$). C_1, C_2 are the coefficients of the rate of change in CPU temperature and the rate of the heat transferred from CPU to air respectively; they can be identified through experiments. Q represents the heat generated by server while R is the thermal resistance. Δt is the granularity of time slot. R only depends on FS, the relationship between R and FS is shown in (4), where C_3, C_4 and n_R are the same as (1). The details of calculating the value of parameters (C_1 to C_4) are given in Section 5.2.

It is well-known that the relationship between fan power and fan speed (rpm) is approximately a cubic function [3]. We use equation (5) to represent this cubic function, where P_{FS} is the fan power, FS is the fan speed, and a_i are constants to be determined.

$$P_{FS} = a_0 \cdot FS^3 + a_1 \cdot FS^2 + a_2 \cdot FS + a_3 \tag{5}$$

Note that given the cubic law relationship in (5), WC strategies, which concentrate jobs in as few servers as possible, can lead to significant increase in cooling demand, in turn increasing the CPU power of these servers as well.

4 OPTIMAL FAN SPEED CONTROL

Under low-power state based framework, the server utilization varies, thus the fan speed control should be considered independently from other servers and it depends on the utilization of the server itself. In this section, we develop a framework to minimize the sum of power of all fans in a server subject to maintaining the CPU temperature under a desired limit.

4.1 Fan Speed Control Algorithm

Using equations (3), (4) and (5), the CPU temperature in time slot $k+1$ can be estimated for given CPU power, ambient and current CPU temperature and fan speed. For each time slot, we use (8) to predict temperature, and limit the total fan power under a maximum value as in (7). Server fan speed settings are typically discrete, thus we limit the optimal fan speed to be one of the available speed settings. For the sake of completeness, we also simulate the continuous speed case. These are shown in (9).

$$\min n_f \cdot P_{FS} \tag{6}$$

$$s.t \quad T_{cpu}(k+1) \leq T_{maα} \tag{7}$$

$$\Delta T_{cpu}(k+1) = (1 - \frac{\Delta t C_2}{C_1 R})\Delta T_{cpu}(k) - \frac{\Delta t}{C_1} Q \tag{8}$$

$$\begin{cases} FS_{min} \leq FS \leq FS_{max}, & if\ FS\ is\ continuous \\ FS \in FS_{available}, & if\ FS\ is\ discrete \end{cases} \tag{9}$$

Algorithm 1: Basic Fan Speed Control (BFC) Algorithm

```
Get current T_amb, P_cpu(k), T_cpu(k);
Use thermal model (3) (4) and FS_min to predict the
  T_cpu(k+1) of next slot;
if T_cpu(k+1) > T_max then
  | Find FS(k+1) such that T_cpu = T_max;
else
  | FS(k+1) = FS_min
end
Return FS(k+1);
```

Here n_f represents the number of fans. Note that the thermal resistance and the speed are the same for all the fans. The constraint is that the predicted temperature in the next slot does not exceed the desired maximum CPU temperature, and the fan speed does not exceed its maximum allowable speed. The Δt we use here is 1 sec granularity (based on IPMItool temperature measurement specifications [5]). For given CPU power and ambient temperature, we want to find an appropriate fan speed that 1) ensures that the temperature will never rise over the maximum no matter how long the current CPU state lasts, and 2) is as small as possible. Algorithm 1 is the basic fan speed control algorithm (BFC), which always gives

the minimum fan speed and fan power to contain the temperature within desired settings.

4.2 Fan Speed With Look-Ahead

Recall that in Figure 2 we showed system power profile with WC. If we consider the fan speed value obtained from BFC, the fan speed will remain consistently low for a number of time slots, until when a prediction is made in slot k that the CPU will overheat in slot $k+1$, at which point the fan speed will rise rapidly and remain high. This phenomenon is shown Figure 3.

Given the cubic relationship, the sudden increase in fan speed may not be the most power-efficient way to operate the cooling system, and could result in increased overall energy consumption of the cooling equipment (since the fan runs longer to bring down the CPU temperature). To alleviate this problem, we develop a look-ahead based optimization framework to minimize the overall fan power, using which we derive the optimal fan speeds. We formulate the problem as a linear program, and solve it using AMPL [16], a sophisticated modeling tool that supports diverse optimization.

$$\min \sum_{k=0}^{T_c} n_f \cdot P_{FS}(k) \tag{10}$$

$$s.t \quad \text{for } k \text{ in } [0, \lfloor \alpha T_c \rfloor] : \tag{11}$$

$$\Delta T_{cpu}(k+1) = \Delta T_{cpu}(k) - \frac{\Delta t C_2}{C_1 R(k)} \Delta T_{cpu}(k) - \frac{\Delta t}{C_1} Q$$

$$\text{for } k \text{ in } [\lceil \alpha T_c \rceil, T_c - 1] : \tag{12}$$

$$\Delta T_{cpu}(k+1) = \Delta T_{cpu}(k) - \frac{\Delta t C_2}{C_1 R(k)} \Delta T_{cpu}(k)$$

$$\text{for } k \text{ in } [0, T_c - 1] : \tag{13}$$

$$FS_{min} \le FS(k) \le FS_{max}$$

$$\text{for } k \text{ in } [0, T_c - 1] : \tag{14}$$

$$P_{FS}(k) = a_0 \cdot FS^3(k) + a_1 \cdot FS^2(k) + a_2 \cdot FS(k) + a_3$$

$$\text{for } k \text{ in } [0, T_c - 1] : \tag{15}$$

$$T_{cpu}(k) \le T_{max}$$

$$T_{cpu}(0) = T_{initial} \tag{16}$$

In this optimization, T_c is the total time of a cycle as shown in Figure 2, while α represents the utilization of this cycle. The constraints (11) and (15) require that every prediction of the CPU temperature during the active period ($[0, (1-\alpha)T_c]$) does not exceed its maximum temperature. Similarly, constraint (12) and (15) give the requirement during package sleep period. $T_{initial}$ is the CPU temperature at the beginning and also the end of each cycle.

Compared to (6), this optimization gives the optimal fan speed for each slot based on a given cycle rather than an instantaneous optimal value. This is based on a profiled CPU power trace as shown in Figure 2, which reveals that active and sleep periods alternate, determined by profiling the history of CPU power patterns.

By analyzing the CPU power trace generated by any workload aware server provisioning strategies such as WASP, we can determine, for each cycle in Figure 2, that the active period is at least $T_{a_{min}}$ and the duration of the package sleep state must be at least $T_{i_{min}}$. We can use this insight in the optimization for a given cycle by setting $T_c = T_{a_{min}} + T_{i_{min}}$ and $\alpha = \frac{T_{a_{min}}}{T_c}$.

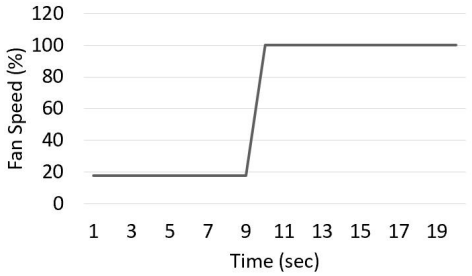

Figure 3: Optimal fan speed during active period in a basic fan speed control (BFC) cycle.

From the results of this linear program, we have observed that once the CPU is active, the optimized fan speed increases linearly (with time) after the temperature reaches a certain point; we set this point as the temperature increase point ($T_{increase}$) and set the fan speed at this point as the initial fan speed ($F_{initial}$). $F_{initial}$ and $T_{increase}$ and the linear function only depend on T_{max}.

Algorithm 2: Lookahead-based Fan Speed Control (LFC) Algorithm

```
Get current T_amb, P_cpu(k), T_cpu(k);
Get current T_increase and F_initial according to T_max,
  generate linear function for fan speed;
Use transient thermal model and FS_min to predict the
  T_cpu(k+1) of next slot;
if T_cpu(k+1) > T_max then
    Find FS(k+1) that can stablize T_cpu at T_max;
else
    if P_cpu(k) > P_active then
        if T_cpu(k+1) > T_increase then
            Use linear function to generate FS(k+1)
        else
            FS(k+1) = FS_min
        end
    else
        FS(k+1) = FS_min
    end
end
Return FS(k+1);
```

Our lookahead-based fan speed control algorithm (LFC) is given in Algorithm 2. For lower complexity, instead of solving the linear program (10) online during the execution of the LFC algorithm, we solve it offline by using the power profile in an active period, and obtain the linear function for fan speed. This function is then used along with T_{max}, $T_{increase}$, and $F_{initial}$ to determine the optimal fan speed. Compared to BFC, LFC considers the history of CPU power pattern, and sets the fan speed accordingly.

5 EVALUATING SERVER FAN CONTROL ALGORITHMS

In this section, we compare the energy for the fan speed control algorithms outlined in Section 4.

5.1 Platform

We use Remote Access Controller Admin (RACADM) and IPMItool to measure and gather data from our servers (PowerEdge T620). RACADM command-line utility provides a scriptable interface that allows users to configure Remote Access Controllers (RAC), which are responsible for system profile settings and remote management [2]. Through this interface we can manage and change fan speeds using the 'system.thermalsettings' command. Other information like temperature and power can also be measured and logged by this utility, however it takes a few seconds for RACADM to give feedback on the data it gathers.

We use IPMItool to measure CPU temperature and power [5]. IPMItool is a utility to configure the Intelligent Platform Management Interface (IPMI) - through this interface users can systematically monitor and manage the health and other features of the system. We use subcommands such as sdr and sensor [5] to read the data from the sensors inside RACADM. The processing time is much faster compared to the RACADM command line. In practice, the measurement granularity for IPMItool is 1 sec, and for RACADM is 5 sec. We validate the readings from IPMItool by using power meters that report the power consumption of the whole server.

5.2 Validation of Thermal Models

We use the sensors embedded in the PowerEdge T620 server to obtain the CPU temperature, the fan speed, and its corresponding fan power. The first step is to obtain the power for each CPU core and different C-states. We modify the available C-states in the Linux kernel to make sure that the core will not enter certain low-power states while it is idle. Then under these different C-state scenarios, we use the `dd Linux` command to control CPU utilization and measure the power using the IPMItool and power meter (HOBO onset UX120-018). The results from IPMItool are verified using multimeter wall power readings. We use the measurement of the CPU power under zero utilization as the baseline power. Since CPU power increases linearly with number of active cores, we use a linear function to represent the dependence between the number of active cores and CPU power. We also assume that the power during wake-up period is half of the sum of peak power and the power of current low-power state; this was validated by prior studies as well [19]. The results for a 10-core CPU are shown in Table 3, where the constant values represent the base power for CPU chip and the platform power (off-chip components like DRAM, power supply unit etc.). The dynamic power consumption for each additional active core is obtained via regression through running CPU-bound kernels. While the server we have in the lab is a 10-core CPU, we extrapolate the model for a 28-core CPU (PowerEdge T620) and simulate the performance of our algorithms on such a server. The power model of the (theoretical) 28-core CPU obtained from the 10-core CPU is shown in Table 4.

We use RACADM to set different fan speeds (720 RPM, 2160 RPM and 4120 RPM) and measure the power using both the IPMItool and power meter. By fitting the power measurement P_{FS} and corresponding fan speed FS to (5), we can obtain the parameters from a_0 to a_3 and build this power model.

Then we measure the corresponding CPU temperature under different fan speeds given different CPU powers. These measurements

Table 3: Power parameters for server with one 10-core CPU under different states.

State	Power(Watts)
Core Sleep C1	41.5+4.089 · (n_a-1) +81.5
Core Sleep C6	39.5+4.31 · (n_a-1) +81.5
Package Sleep C6	7.0+74.5
System Sleep	13.8

n_a is the number of cores in active state.

Table 4: Power parameters for server with one 28-core CPU under different states.

State	Power(Watts)
Core Sleep C1	68.87+4.85 · (n_a-1) +81.5
Core Sleep C6	61.79+5.11 · (n_a-1) +81.5
Package Sleep C6	104.75
System Sleep	13.8

n_a is the number of cores in active state.

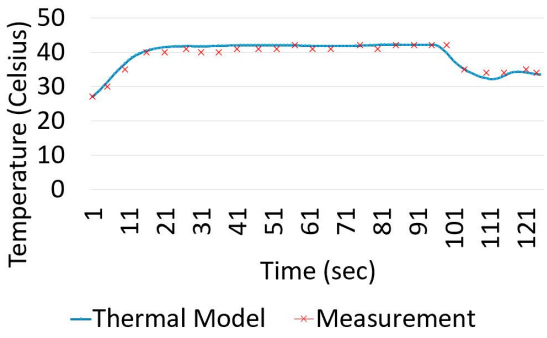

Figure 4: Temperature profiles measured on CPU versus estimation by our thermal model.

are used to generate the stable thermal model. Stable model here is used both in BFC and LFC to find a minimum fan speed to ensure the CPU temperature will never be higher than the desired maximum under current CPU temperature and ambient temperature. The CPU power represents Q in (1). For given Q, T_{amb} and T_{CPU}, we can obtain the thermal resistance R for different fan speeds, and use that to define the value of C_3 and C_4 according to (4).

Next, we change the CPU workload using `dd Linux` command, and use IPMItool to measure the CPU temperature and CPU power during this transient period. We also change the fan speed and measure the temperature trace under the same CPU power trace. This temperature trace under different changes can be used to identify the parameters C_1 and C_2 to build the transient model.

Since the thermal resistance R is only relative to the fan speed [18], by applying (4) to (3), we can predict the CPU temperature in the next slot ($T_{CPU}(k+1)$) according to current CPU temperature ($T_{CPU}(k)$), CPU power (represented by Q) and the fan speed (represented by R). Figure 4 shows the comparison between the measured and predicted CPU temperature. We note that our estimated temperature fairly tracks the measured temperature, validating the efficacy of the model to predict CPU temperature accurately.

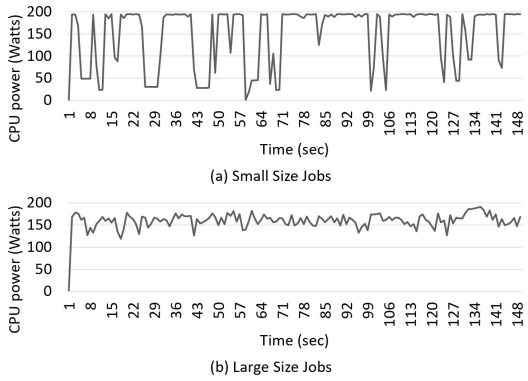

Figure 5: The CPU power trace with workload consolidation under different sized jobs and utilization around 30%.

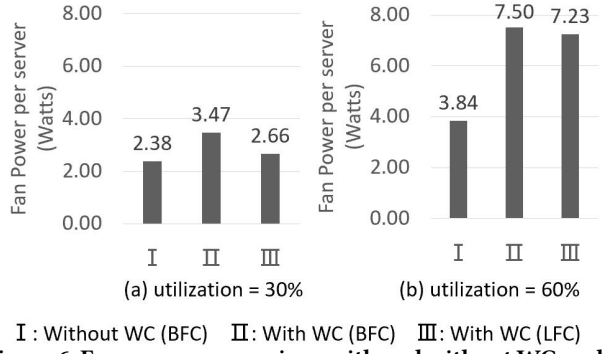

Figure 6: Fan power comparison with and without WC under different algorithms.

5.3 Fan Speed Control with and without look-ahead

Here we assume a CPU with 28 cores as shown in Table 4, allowing us to stress test the model on the cooling system.

We plot in Figure 5 the CPU power trace of two different sizes of jobs under 30% CPU utilization. Since the CPU power for small jobs (Web service-like) stays at peak power once the CPU is in active period, it will cause a significant demand for cooling compared to large jobs (DNS service-like) .

This optimization begins to increase the fan speed gradually long before the time when the CPU temperature rises as in the optimization (9). This is because the optimization is based on a prediction that once the CPU wakes up, it will run at full utilization and last for at least $T_{a_{min}}$. However, this does not mean that once the CPU is awake, the fan will speed up immediately. Only if CPU is awake and the temperature grows larger than these increase points, the fan speed increases.

We now give energy comparison between with and without WC strategy under LFC. Since the workload assignment without WC is random and therefore the power profile is not predictable, we only use BFC for the case without WC. As shown in Figure 6, the energy saving efficiency of LFC is better for lower utilization; for 30% utilization it can almost eliminate the negative impact from WC.

In last section we mentioned that LFC works better when the initial temperature is lower. However, the initial temperature in a whole CPU power trace equals the temperature at the end of each package sleep state period. A lower initial temperature means a lower average temperature, which also means a lower CPU power. Though LFC can save more proportioned energy under low system utilization, the amount of the energy that is saved is small because the demand for cooling system is also low in this case.

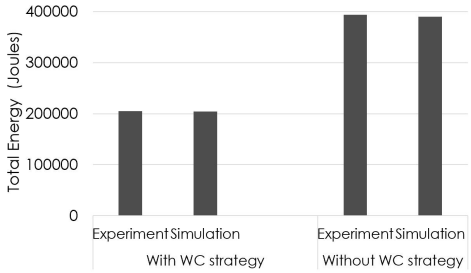

Figure 7: Energy comparison between simulation and experiment.

We also simulated a small server farm with 10 servers under 30% utilization and use the average of corresponding CPU power trace to generate `dd Linux` command line scripts. We ran these scripts and adjusted the fan speed to its optimal fan speed according to LFC, then used power meter to record total energy. Figure 7 shows the comparison between the simulation and the experimental results of the two frameworks with and without WC strategy, the results show that the error between simulation and experiment is very small (~4%).

6 CORE CAP FOR LOW-POWER STATE BASED OPTIMAL FRAMEWORK

Experimental results from the last section show that, WC frameworks such as WASP tend to concentrate jobs to as few servers as possible, so that other servers will have more opportunity to enter a deep sleep state to save energy. The implicit assumption in WC strategies is that all cores in an awake server are active. Note that the server power (and hence the fan power) can be decreased by limiting the number of active cores in the server. This strategy inevitably means that either more servers must be kept awake or the active servers will take longer to process the jobs. This section investigates the potential benefits in cooling offered by limiting the number of active cores in each server.

6.1 Core Cap

For a given server, as shown in Table 3, since the base power of the system is large, it seems to be more efficient to put the server into system sleep state. But, the wake-up energy from system sleep state of this server will also be high. The wake-up power is almost as much as the power of package sleep state, not to mention that a wake-up latency of 5 sec will also cause a potential QoS violation.

Compared to keeping all the servers active, or putting all of them into system sleep state from time to time, the most energy-saving arrangement is to find a subset of N_a servers, that will always be awake (never enter system sleep state), while the rest of the servers stay in system sleep state. As shown in (17), N represents the total number of servers, u represents the utilization of N servers, u_a is the average utilization of N_a servers that are chosen to keep awake.

$$N_a = \min \left\lceil \frac{N * u}{u_a} \right\rceil, \; u_a < 1 \tag{17}$$

In (17), the constraint of $u_a < 1$ makes sure that the utilization of each awake server will not reach or exceed 100 percent. As shown in Table 3, the server power can be seen as consisting of two parts: CPU power and basic system power. Reducing the number of awake servers and keeping them working at full load can save significant basic system energy, but may cause a severe latency issue. (17) uses N_a to limit the amount of active servers to minimize server energy. We suggest another constraint on the number of cores in active servers to minimize cooling energy. Capping the number of available cores will increase the server active time and the overall system energy not including cooling energy, but it will also decrease peak power of active servers and has the potential to lower the demand for cooling. We investigate the energy trade off between these two parts.

6.2 Constraint on Core Cap

The constraint on the core cap is that the utilization of available cores must not exceed 100% because the QoS rapidly degrades as the utilization approaches 100%. If N is the number of servers, U is the utilization, C_{max} is the number of total cores per server, and C_{cap} represents the core capping (i.e., number of available cores), $u_{C_{cap}}$ is the corresponding utilization for C_{cap} available cores:

$$u_{C_{cap}} = \frac{C_{max} N U}{C_{cap} N_a} \tag{18}$$

In (18), $C_{max} N U$ represents the necessary cores of the whole system, $C_{cap} N_a$ is the available servers and cores adjusted. To satisfy the workload demand, the utilization $u_{C_{cap}}$ will change along with C_{cap} and N_a. If the number of available servers and cores decreases, the utilization of cores will increase correspondingly. No matter how we change C_{cap} and N_a, $u_{C_{cap}}$ should be larger than 0 and smaller than 1.

6.3 Energy Comparison

The energy effects of core cap can be seen as two parts, the first part is the increase in the CPU energy, the second part is the energy saving from decreasing cooling demands.

Here we give energy comparisons for different core caps and utilizations to find a balance point, where core caps can achieve a proper energy trade-off and minimize the total energy. We simulate frameworks with different core capping and record the energy with and without fan power under different utilizations (Low: 30%, Medium: 50% and High: 70%). The number of jobs is a constant, thus the total execution time will decrease while utilization increases, the system power without CPU or fan occupies more than half of the total power, and less execution time means less total energy.

Figure 8 gives the comparison of the energy with and without fans for different core caps under WC strategy. The energy without fan shows a slightly growing trend while core caps decrease, however the total energy shows a convex behavior, and reaches its minimum around the best core cap. The best core cap is the same for the three utilization levels. As shown in the figure, the core cap can decrease, or even eliminate the negative impact on cooling system

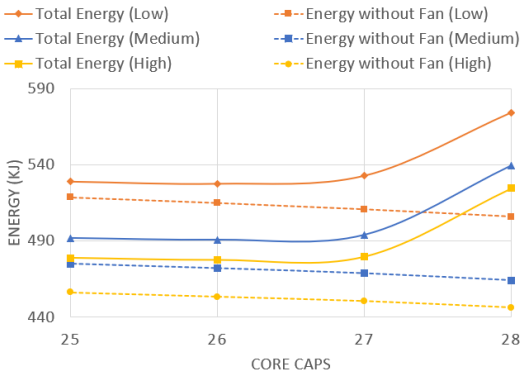

Figure 8: Energy with and without fan under different utilizations and core caps.

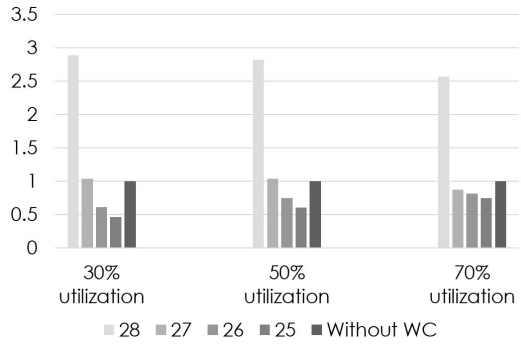

Figure 9: Normalized fan energy under different utilizations and core caps using 5-stage fan speeds.

caused by workload consolidation. Under a 26-core limitation, we can observe that such cap can save more than 80% cooling energy under low utilization, and the energy saving efficiency is still good (70%) under high utilization. The increase of the system energy is small, only around 1% under this best core limitation.

PowerEdge T620 has 3 controllable fan speeds: 720 rpm , 2140 rpm and 4120 rpm. Finally, to study the possible benefits with more fan speeds, we add two more values 1430 rpm and 3130 rpm to simulate under a 5-stage fan speed situation. Figure 9 shows the normalized energy comparison under this 5-stage fan speed model. We can observe that fan energy under WC (without core caps) is significantly higher than without WC, and core caps decrease the fan energy for WC strategy. Under 30% utilization, WC with 26 core cap can achieve 42% fan energy saving compared to without WC.

7 RELATED WORK

Several techniques have been suggested to optimize fan speed and minimize cooling energy for data centers. The workload assignment problem considering cooling system cost has been investigated before; for example, by Moore et al. [12]. They build a theoretic thermodynamic formulation based on steady-state hot spots and cold spots of data centers and develop workload assignment algorithms. Based on the results, they suggest an alternative approach which considers location in workload distribution. In later work, such as the research by Pakbaznia [13], the workload distribution

along with the ON-OFF state of the servers in data centers is investigated with the goal of minimizing the total energy cost for both servers and cooling equipment. They investigate High Performance Computing (HPC) scenarios where the jobs inside the data center run for hours or days; thus the granularity of the workload changes is assumed to be much longer than the time it takes for the temperature inside the data center to reach its steady state after a workload changes [11].

All of these above papers consider cooling energy in the workload assignment. However, for works such as [13], the granularity of workload assignment is large time scale, thus the utilization is more steady compared to the works which deal with finer time-scale granularity. Workload assignments suggested in [12, 13] consider the cooling efficiency of the servers based on their location inside the data center, as well as the different states of the server, such as ON and OFF. However, switching between different states might cause a critical QoS problem for small time-scale jobs, because of relatively large wakeup latencies, which are not considered.

Thermal model with optimal algorithm and Proportional-integral-derivative (PID) fan speed control techniques are normally used in the cooling energy optimization of servers. PID controller is also investigated to solve the stability concerns caused by the time lag and quantization of enterprise server sensors [4]. These techniques can make a good effort for any given CPU power or task schedule to save cooling energy; however failing to minimize CPU power and cooling power synchronously might reduce the effect. Studies such as [17] use Dynamic Voltage-Frequency Scaling (DVFS) with fan speed control to minimize the cooling power and CPU power together, while leakage power is also considered. Reducing cooling power can cause a high temperature and increase the leakage power, and it finds a balance between cooling and leakage power while minimizing the global power. However, the techniques applied single server might not optimize the overall system energy in a cluster of servers.

8 CONCLUSIONS

A typical approach for energy optimization is to consolidate the workload in as few servers as possible and let the remaining servers stay in sleep states. While this approach is effective in minimizing CPU energy, it is not effective when the energy of the fans are also taken into account. In this paper, we have developed an optimal fan speed control algorithm called SpinSmart when workload consolidation is employed, and workload scheduling strategies to minimize overall energy consumption in data centers are considered. Our LFC predicts the temperature of the CPU in the near future by considering the current temperature, the ambient temperature and the power profile trace of a typical server with WC, and then controls the fan speed to keep the CPU temperature within a prescribed safe limit, while minimizing overall energy.

We further improve our scheme by a strategy called core capping. Core capping limits the utilization of a CPU by capping the number of active cores inside a CPU. While this may lead to more CPUs being active, our counter-intuitive results suggest that this is an effective strategy for overall energy minimization because the cooling energy for a CPU with very high utilization may dominate the energy needed for keeping an additional CPU active.

ACKNOWLEDGMENTS

This material is based in part upon work supported by the National Science Foundation under Grant Number CNS178133.

REFERENCES

[1] Info-Tech Research Group et al. 2007. Top 10 Energy-Saving Tips for a Greener Data Center. *Info-Tech Research Group*. (2007).
[2] Dell Inc. [n.d.]. RACADM Command Line Reference Guide for iDRAC7 1.50.50 and CMC 4.5. http://nick.txtcc.com/nickfiles/dell-poweredge-drac7-1.50.50-command-line.pdf.
[3] R Jorgenson. 1961. Fan Engineering an Engineer's Handbook.
[4] Jungsoo Kim, Mohamed M Sabry, David Atienza, Kalyan Vaidyanathan, and Kenny Gross. 2014. Global fan speed control considering non-ideal temperature measurements in enterprise servers. In *Proceedings of the conference on Design, Automation & Test in Europe*. European Design and Automation Association, 276.
[5] Duncan Laurie. [n.d.]. Ipmitool(1) - Linux man page. https://linux.die.net/man/1/ipmitool.
[6] Charles Lefurgy, Karthick Rajamani, Freeman Rawson, Wes Felter, Michael Kistler, and Tom W Keller. 2003. Energy management for commercial servers. *Computer* 36, 12 (2003), 39–48.
[7] Yanpei Liu, Stark C Draper, and Nam Sung Kim. 2014. Sleepscale: Runtime joint speed scaling and sleep states management for power efficient data centers. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*. IEEE, 313–324.
[8] Bingqian Lu, Sai Santosh Dayapule, Fan Yao, Jingxin Wu, Guru Venkataramani, and Suresh Subramaniam. 2018. Popcorns: Power optimization using a cooperative network-server approach for data centers. In *2018 27th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 1–9.
[9] David Meisner, Brian T Gold, and Thomas F Wenisch. 2009. PowerNap: eliminating server idle power. *ACM SIGARCH Computer Architecture News* 37, 1 (2009), 205–216.
[10] David Meisner and Thomas F Wenisch. 2012. DreamWeaver: architectural support for deep sleep. In *ACM SIGPLAN Notices*, Vol. 47. ACM, 313–324.
[11] Justin Moore, Ratnesh Sharma, Rocky Shih, Jeff Chase, Chandrakant Patel, and Parthasarathy Ranganathan. 2004. Going beyond CPUs: The potential of temperature-aware data center architectures. In *First Workshop on Temperature-Aware Computer Systems*.
[12] Justin D Moore, Jeffrey S Chase, Parthasarathy Ranganathan, and Ratnesh K Sharma. 2005. Making Scheduling" Cool": Temperature-Aware Workload Placement in Data Centers.. In *USENIX annual technical conference, General Track*. 61–75.
[13] Ehsan Pakbaznia and Massoud Pedram. 2009. Minimizing data center cooling and server power costs. In *Proceedings of the 2009 ACM/IEEE international symposium on Low power electronics and design*. 145–150.
[14] Venkatesh Pallipadi, Shaohua Li, and Adam Belay. 2007. cpuidle: Do nothing, efficiently. In *Proceedings of the Linux Symposium*, Vol. 2. Citeseer, 119–125.
[15] Wojciech Piatek, Ariel Oleksiak, Micha vor dem Berge, Jens Hagemeyer, and Emmanuel Senechal. 2017. Intelligent thermal management in M2DC system. In *Proceedings of the Eighth International Conference on Future Energy Systems*. 309–315.
[16] Brian W. Kernighan Robert Fourer, David M. Gay. [n.d.]. AMPL A Modeling Language for Mathematical Programming. https://ampl.com/BOOK/CHAPTERS/01-title.pdf.
[17] Donghwa Shin, Jihun Kim, Naehyuck Chang, Jinhang Choi, Sung Woo Chung, and Eui-Young Chung. 2009. Energy-optimal dynamic thermal management for green computing. In *2009 IEEE/ACM International Conference on Computer-Aided Design-Digest of Technical Papers*. IEEE, 652–657.
[18] Zhikui Wang, Cullen Bash, Niraj Tolia, Manish Marwah, Xiaoyun Zhu, and Parthasarathy Ranganathan. 2009. Optimal fan speed control for thermal management of servers. In *ASME 2009 InterPACK Conference collocated with the ASME 2009 Summer Heat Transfer Conference and the ASME 2009 3rd International Conference on Energy Sustainability*. American Society of Mechanical Engineers, 709–719.
[19] Fan Yao, Jingxin Wu, Suresh Subramaniam, and Guru Venkataramani. 2017. WASP: Workload adaptive energy-latency optimization in server farms using server low-power states. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*. IEEE, 171–178.
[20] Fan Yao, Jingxin Wu, Guru Venkataramani, and Suresh Subramaniam. 2017. Tsbat: Leveraging temporal-spatial batching for data center energy optimization. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 1–6.