Temperature Aware Power Allocation: An Optimization Framework and Case Studies

Li, Shen; Abdelzaher, Tarek; Wang, Shiguang; Kihl, Maria; Robertsson, Anders

Published in:
Sustainable Computing: Informatics and Systems

2012

Sustainable Computing: Informatics and Systems 2 (2012) 117–127

Contents lists available at SciVerse ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

Temperature aware power allocation: An optimization framework and case studies

Shen Li[a,*], Shiguang Wang[a], Tarek Abdelzaher[a,b], Maria Kihl[b], Anders Robertsson[b]

[a] University of Illinois at Urbana Champaign, USA
[b] Lund University, Sweden

ARTICLE INFO

Article history:
Received 30 October 2011
Accepted 30 April 2012

Keywords:
Data center
Map-reduce
Web-server
Thermal-aware optimization
Energy management

ABSTRACT

In this paper, we propose a general power model along with a versatile optimization methodology that can be applied to different applications for minimizing service delay while satisfying power budget in data centers. We test our methodology on two totally different applications from both cloud computing and enterprise scenarios. Our solution is novel in that it takes into account the dependence of power consumption on temperature, attributed to temperature-induced changes in leakage current and fan speed. While this dependence is well-known, we are the first to consider it in the context of minimizing service delay. Accordingly, we implement our optimization strategies with Hadoop, Tomcat, and MySQL on a 40-node cluster. The experimental results show that our solution cannot only limit the power consumption to the power budget but also achieves smaller delay against static solutions and temperature oblivious DVFS solutions.

1. Introduction

In this paper, we propose a general power model and temperature aware power allocation (TAPA) optimization framework that help to minimize services' delay with a given power budget in data centers. We implement and empirically evaluate our framework on a Map-Reduce cluster and multi-tier web servers respectively. The research challenge is to optimize the trade-off between service delay and power consumed. We do so by developing algorithms that minimize delay for any given power budget. Various implementations [7–9] of Map-Reduce have been developed in recent work. We modify Hadoop in our implementation to embody our optimization solutions. For the web server case, we use Tomcat on the second tier servers, and MySQL on the third tier servers.

Our algorithm is novel in that it accounts for thermally induced variations in machine power consumption. Prior work on energy-efficiency of Map-Reduce and web server workloads considered power consumption models that are a function of only the energy state and machine load. In reality, both leakage current and fan speed depend on machine temperature. Hence, a higher power consumption is incurred at higher temperatures, even at the same energy state (*e.g.*, DVFS setting) and at the same load. Modifications are thus needed to existing energy management algorithms to account for the aforementioned temperature-dependency of power consumption. While the effect of temperature on leakage current (and, hence, energy) is itself well-known [11–15], we are the first to consider its implications on performance (namely, latency) and energy trade-offs in data centers.

Our algorithm requires that a power budget be expressed. This can be given by the power supplier. Some data centers do have a fixed peak power budget [17,30] to avoid excessive charges for peak power consumption. Even in scenarios without a global power bound, a local power budget is often used as a proxy for controlling heat distribution since most power consumed by servers turns into heat. Some researchers [23,22] proposed to divide the whole data center into small zones and apply elaborately calculated power budgets to each zone to counter-balance temperature differences and remove hot spots. Finally, even in systems where no power budget exists at any level, it is useful to optimize the power-performance trade-off. This optimization problem can be generically cast as one of maximizing performance for any given input power. Hence, our algorithm maximizes the computational capacity of the cluster, without exceeding a stated power budget.

Improving the power-performance trade-off may result in significant monetary savings for data center operators. Hamilton [1] estimated that, in 2008, money spent on power consumption of servers and cooling units had exceeded 40 percent of total cost for data centers, which reached more than 1 million per month for a single data center. The cost is still increasing as both prices of energy and scale of data centers are on the rise [2]. The U.S. Department of Energy states that, by 2012, the cost of powering up

* Corresponding author.
E-mail addresses: shenli3@illinois.edu (S. Li), zaher@illinois.edu (T. Abdelzaher).

2210-5379/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.suscom.2012.04.002

a data center will surpass the original IT investment [3]. Our experiments show that the temperature-dependent component of power contributes a considerable part to the total power consumption of machines. Given the large variations in temperature among different machines (depending on their placement in the cluster) [16,18], accounting for temperature dependencies is essential to achieving a more efficient cluster energy usage and hence a non-trivial improvement in data center operation costs.

The remainder of this paper is organized as follows. In Section 2, we summarize the related work of this paper. Section 3 demonstrates the general power model, optimization framework and two case studies. Implementation details are described in Section 4. Finally, our experimental results are shown in Section 5.

2. Related work

Current cluster energy saving policies cover several different basic techniques, such as dynamic voltage and frequency scaling (DVFS) [10,19–21,31,32], workload distribution [24,25,29], as well as turning machines on and off [26,28,27]. While much work considered energy minimization, other literature [17,30] focused on maximizing service capacity under a power constraint. Few papers consider temperature as a parameter when designing energy saving policies. Given that most power consumed by servers turns into heat, some algorithms, such as OnePassAnalog [22], use the amount of power as a proxy of workload, and assign an amount of power to each server inversely proportional to the server's inlet temperature. Moore et al. designed the Zone-Based Discretization (ZBD) algorithm [23] to improve the granularity of load and power distribution. When one server consumes more power than it is assigned, this server will "borrow" power from its neighbor servers, *i.e.*, neighbor servers have to use less power than assigned. Banerjee et al. [29] proposed Highest Thermostat Setting (HTS) algorithm to smooth out hot spots and increase the set point in CRAC. In their algorithm, each server s_i has its own thermostat setting requirement (TSR), which indicates the highest possible set point of CRAC to satisfy s_i's cooling demand. They rank servers by TSR in descending order and place workload according to this ranked list. The CRAC are set to lowest TSR of all servers.

Dean et al. [4] first proposed Map-Reduce as a programming paradigm for distributed computing. They divide machines into one master node and several worker nodes. The master node chops data and passes the chunks to mapper nodes that generate intermediate results that are then aggregated on the reducers into the final result. Their original paper did not address energy concerns. Since then, much literature addressed energy efficiency in Map-Reduce. For example, Chen et al. [5,6] discuss statistics of Map-Reduce workloads and suggest possible strategies to improve their energy behavior. Their experiments show that batching and staggered launching of batched jobs can help save energy. To the best of our knowledge, this paper is the first in considering the effect of the dependence of power consumption on temperature in the context of investigating energy-delay trade-offs of data-center workloads.

3. System design

In this paper, we propose a general optimization framework for different applications, that helps to minimize service delay without violating a known power budget.

3.1. General problem description

In order to provide a general framework for optimization, we need to be aware of the controllable knobs in data centers. In this paper, we focus on computing and I/O resources. Since the power consumption of each component only depends on its own states, other resources, such as network and memory, can be measured separately and enclosed into the general model in a similar way. Let l denote the number of tiers in data centers. We assume that the ratio r_i between computational workload W_i^C and overall workload W_i in tier i is known for specific applications. This ratio is characteristic to the tier's workload and hence is on average the same for all machines in the tier. Since load balancers are often available in data-centers, the amount of workload w_{ij} assigned to jth machine in tier i is one obvious knob. The workload w_{ij} can be further divided into computational workload w_{ij}^C and I/O workload w_{ij}^{IO}. Technologies like DVFS and Wake on Lan (WOL) help us enable two other widely used knobs: CPU frequency f_{ij} of jth machine in tier i and the number of powered-on machines n_i in tier i.

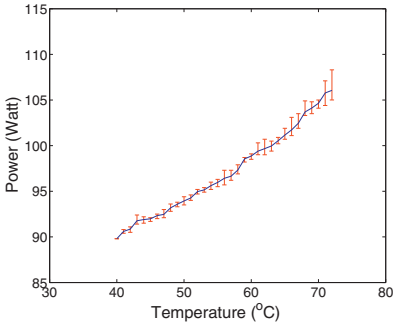


Fig. 1. Maximum performance.

Besides these well studied knobs, we argue that CPU temperature also exerts considerable influence on power consumption. Figs. 1 and 2 show the relationship between temperature and power in experiments where we run a CPU-intensive program to keep CPU utilization of the machine at 100%. The ambient temperature is then changed by controlling the set point of the CRAC unit. The CPU fan in our server (Dell PowerEdge R210) does not have a constant speed option. Instead, we measure the relationship between temperature and power in two cases; one where the fan policy was set to maximum performance (Fig. 1) and one where it is set to minimum power (Fig. 2). We can clearly see that, in both cases, power consumption increases as temperature rises, even though the load remains the same. In commercial data centers, heat is not uniformly distributed. The temperature differences can reach about 10 °C in the same aisle [16]. The thermal gap can be even larger between CPUs in different server boxes. Fig. 3 illustrates measured temperature variability in our testbed when the Hadoop cluster is running one CPU-intensive job. Hence, it is possible to improve energy savings by better exploiting temperature differences. In this paper, we summarize variables relevant to the optimization problem by the following four vectors of parameters: $\mathbf{N} = \{n_1, n_2, \ldots, n_l\}$, $\mathbf{W} = \{w_{ij} \mid i \le l, j \le n_i\}$, $\mathbf{F} = \{f_{ij} \mid i \le l, j \le n_i\}$,

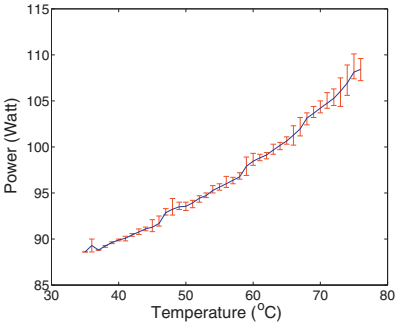


Fig. 2. Minimum power.

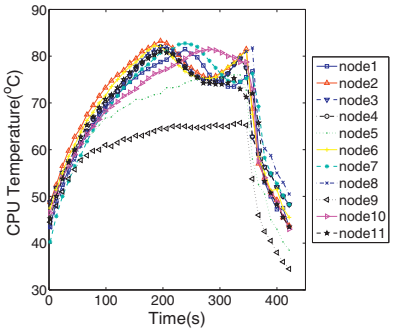


Fig. 3. Temperature differences.

and $\mathbf{T} = \{T_{ij} \mid i \le l, j \le n_i\}$. They denote the number of machines per tier, the load distribution across machines, the frequency settings of machines, and the machine temperature values, respectively. Table 1 describes the meaning of related parameters used in this section.

The power consumption of one machine consists of two components: static and dynamic. The static component is incurred even when the machine is idle. The dynamic component depends on several parameters, such as CPU utilization, disk utilization, and CPU frequency. Our goal is to provide a general power model that applies to different applications. One server is a collection of several different components, such as, CPU, disk, memory, and NIC. The total power consumption of a server is the sum of components' power draw. This nature of computer systems helps us to decouple the big problem into smaller ones. We can model the power consumption of different components respectively and then, piece them together. Let P_{ij} denote the total power consumption of machine j in tier i, P_{ij}^C and P_{ij}^{IO} denote the power spent on CPU and disk respectively, and P^{idle} denote the power consumption when the machine is idle. Therefore, for CPU and I/O workload, we can write:

$$P_{ij} = P_{ij}^C + P_{ij}^{IO} + P^{idle}. \tag{1}$$

As mentioned in [27], when u_{ij} and T_{ij} are fixed, the power P_{ij}^C can be modeled as a kth-order polynomial of f_{ij}. According to our experiments described in Section 4, $k = 2$ fits our hardware best. Literature [12] states that the leakage current is proportional to the square of temperature. Since, in our server, the DVFS component changes CPU frequency by tuning voltage which also affects the leakage power, P_{ij}^C should be proportional to the product of f_{ij} term and T_{ij} term.

Table 1
Parameters description.

Symbol	Description	Symbol	Description
P^b	Power budget	P_{ij}^{IO}	Dynamic part of disk power consumption
W_i	Total workload in tier i	u_{ij}	CPU utilization of jth server in tier i
P^{idle}	Power consumption of idle server	f_{ij}	CPU frequency of jth server in tier i
D_{ij}	Service delay on jth server in tier i	n_i	Number of powered-on machines in tier i
$\overline{D}$	Average Service delay	P_{ij}^C	Dynamic part of CPU power consumption
w_{ij}	Workload on jth server in tier i	P_{ij}	Power consumption of jth server in tier i
r_i	CPU workload ratio in tier i	T_{ij}	CPU temperature of jth server in tier i

CPU utilization u_{ij} is the ratio of CPU busy time over total time in each given time interval. Since, the non-busy (idle) power is already covered in P^{idle}, we should consider only busy time. Hence, P_{ij}^C is proportional to the product of the aforementioned three terms. Eq. (2) shows the relationship between the dynamic part of CPU power consumption and these three parameters when computationally intensive workloads are running.

$$P_{ij}^C = u_{ij}(a_1 f_{ij}^2 + a_2 f_{ij} + a_3)(T_{ij}^2 + a_4 T_{ij} + a_5) \tag{2}$$

We show empirically later that the above equation happens to fit our measurements very well, and hence will be used for our computational power model. Note also that, CPU utilization represents busy CPU cycles over total CPU cycles. Given appropriate units, it can be represented as amount of computational workload over CPU frequency:

$$u_{ij} = \frac{r_i w_{ij}}{f_{ij}}. \tag{3}$$

In [34], Heath et al. proposed using a linear model of disk utilization to estimate disk power consumption. Inspired by this idea, to compute the I/O power consumption, P_{ij}^{IO}, we conducted experiments to measure the relationship between power consumption and disk Blocks-read-Per-Second (BPS). As we will discuss in Section 4.2, our experiment result confirms validity of the linear model. Thus, we use Eq. (4) to model the dynamic part of disk power consumption.

$$P_{ij}^{IO} = a_6(1 - r_i)w_{ij} \tag{4}$$

Since the two power models above contain only hardware status, they can be applied to different types of applications. Finally, P^{idle} is just a machine constant. Combining Eqs. (2) and (4) with Eq. (1) produces the general power model.

The purpose of our optimization is to calculate right states for controllable knobs such that the delay is minimized. Among the four parameters, CPU temperature is not directly tunable. Since the thermal state changes much slower than computing state, we propose to run optimizations periodically with the latest reported CPU temperature. Therefore, by substituting Eqs. (2)–(4) in (1), we have:

$$\begin{aligned} P_{ij} &= w_{ij}\left(\alpha_{ij} f_{ij} + \beta_{ij} + \gamma_{ij}\frac{1}{f_{ij}}\right) + P^{idle} \\ \text{where}\quad \alpha_{ij} &= r_i a_1 \widetilde{T}_{ij} \\ \beta_{ij} &= r_i a_2 \widetilde{T}_{ij} + a_6(1 - r_i) \\ \gamma_{ij} &= r_i a_3 \widetilde{T}_{ij} \\ \widetilde{T}_{ij} &= T_{ij}^2 + a_4 T_{ij} + a_5 \end{aligned} \tag{5}$$

The power model applies to different kinds of applications. In contrast, the service delay highly depends on the nature of application itself. For example, in a web service system, if real-time request arrivals are Poisson and service time is exponentially distributed, the delay can be approximated by an M/M/1 model [33]. However, if we use Map-Reduce to process a large volume of data, jobs run in batch and the delay is simply modeled as the total amount of work over the resources the job occupies [35]. Given the diversity of applications, it is very hard to provide a general delay model. However, given a general delay function that is monotonically increasing with load and is a function of machines used and their frequencies, different applications may share the same framework for delay minimization subject to power constraints. We let

$D = G(\boldsymbol{W}, \boldsymbol{F}, \boldsymbol{N})$ denote the delay model. The general optimization problem becomes:

$$\begin{aligned} \text{minimize} \quad & D = G(\boldsymbol{W}, \boldsymbol{F}, \boldsymbol{N}) \\ \text{s.t.} \quad & \sum_{i=1}^{l}\sum_{j=1}^{n_i} P_{ij} \le P^b \\ \text{where} \quad & P_{ij} = w_{ij}\left(\alpha_{ij} f_{ij} + \beta_{ij} + \gamma_{ij}\frac{1}{f_{ij}}\right) + P^{idle} \end{aligned} \tag{6}$$

With the general formulation above, different applications may follow the same set of steps to calculate the global optimal:

1 Build a delay model, $G(\boldsymbol{W}, \boldsymbol{F}, \boldsymbol{N})$, according to application characteristics.
2 Simplify the general power expression for the specific application. (*e.g.*, r_i equals 1 for workloads consuming only CPU resources, f_{ij} is a constant if the control policy does not tune CPU frequencies.)
3 If a convex problem is achieved in step 2, periodically conduct convex optimization to calculate and maintain optimal states.

In clusters, the controllers usually do not need to tune $\boldsymbol{W}$, $\boldsymbol{F}$, and $\boldsymbol{N}$ with the same time scale. The CPU frequencies are stateless, and the changes apply instantaneously. Given the states and durations of jobs, the workload assignments vary in a more gentle way. Turning on and off servers should be performed infrequently, since the action itself costs extra energy. With these properties, we are often able to manipulate one knob at a time at any given time-scale. Therefore, by virtue of this time-scale separation, convex problem approximations can often be achieved without too much difficulty.

In the following, we show the detail of how we apply our framework to two totally different applications and achieve global optimal.

3.2. Cloud computing workload

Hadoop is a widely used implementation of Map-Reduce in many cloud computing services. The performance objective of the target system in this case study is to minimize delay of serving one computationally intensive job. According to [35], the delay is inversely proportional to the amount of resources one job occupies. Since there is only one tier of worker nodes, l equals to 1 in this case study. Below, we consider CPU intensive jobs. Hence, the overall delay of one job can be modeled as:

$$D \propto \frac{W_1}{\sum_{j=1}^{n_1} f_{1j}}, \tag{7}$$

It is obvious to see that minimizing delay is the equivalent to maximizing the summation of CPU frequencies for all time units.

In Hadoop, the master node splits one job into many smaller tasks, and whenever one worker has any available resource slots, it will get one task to run with. This property means that the bottleneck resource on worker nodes will always be fully utilized. In our case, the CPU utilization is approximately 1. Since we are considering computational workload, the ratio r_1 is also 1. Hence, with appropriate units, the workload assigned to each machine can be represented as:

$$w_{1j} = f_{1j} \tag{8}$$

The purpose of our optimization is to find the right frequency for each machine such that the sum is maximized. The algorithm re-computes the solution periodically at fixed intervals. Note that, when a machine keeps running at high utilization, the CPU temperature does not change very quickly. Thus, each time when calculating the frequency for the current time period, we use the most recent reported temperature of each machine from the previous period as a constant to estimate the power consumption P_{1j}. In this way, we take the temperature dynamics into consideration in the equation, and, at the same time, we calculate the optimal frequency assignment taking temperature into account. Intuitively, we are using CPU frequency to compensate for the power contribution made by CPU temperature dynamics. Thus, substituting Eq. (8) in (6), we get a new estimated power equation:

$$P_{1j} = \alpha_{1j} f_{1j}^2 + \beta_{1j} f_{1j} + \gamma_{1j} + P^{idle}. \tag{9}$$

In this equation, the value of α_{1j}, β_{1j}, and γ_{1j} may vary for different machines due to their different individual CPU temperature. Thus, the optimization problem can be formulated as shown below.

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{n_1} f_{1j} \\ \text{s.t.} \quad & \sum_{j=1}^{n_1} P_{1j} < P^b \\ \text{where} \quad & P_{1j} = \alpha_{1j} f_{1j}^2 + \beta_{1j} f_{1j} + \gamma_{1j} + P^{idle}. \end{aligned} \tag{10}$$

The beauty of this problem is convexity. The objective function is linear. Since $\alpha_{1i} > 0$, each quadratic P_{1i} in the only constraint is positive definite and thus convex, so is their sum. Obviously, the definition domain is also convex. As a result, the problem is convex. Furthermore, the *Slater's* condition is also satisfied, because we can easily find at least one interior point in the feasible set after checking the constraint. Hence, the KKT condition is sufficient for the optimality of the primal problem. In other words, we can exactly obtain the optimum by solving the KKT conditions because the duality gap is zero.

Introducing Lagrange multiplier μ for the constraint, define:

$$L(\mathbf{F}, \mu) = \sum_{j=1}^{n_1} f_{1j} + \mu\left(\sum_{j=1}^{n_1}(\alpha_{1j} f_{1j}^2 + \beta_{1j} f_{1j} + \gamma_{1j} + P^{idle}) - P^b\right). \tag{11}$$

Note that, the original problem described in Eq. (10) has an optimal point at $\mathbf{F}^*$, if and only if there exists the optimal point $(\mathbf{F}^*, \mu)$ that maximizes Eq. (11) with respect to $\mathbf{F}$. We then check the stationarity of the KKT conditions:

$$\frac{\partial L(\mathbf{F}, \mu)}{\partial \mu} = \sum_{j=1}^{n_1}(\alpha_{1j} f_{1j}^2 + \beta_{1j} f_{1j} + \gamma_{1j} + P^{idle}) - P^b = 0,$$

$$\frac{\partial L(\mathbf{F}, \mu)}{\partial f_{1j}} = 1 + \mu(2\alpha_{1j} f_{1j} + \beta_{1j}) = 0.$$

Finally, from the 2 equations above, we get the optimal solution which also indicates the new frequency assignment for each machine.

$$\mu = \left[\frac{\sum_{j=1}^{n_1}\frac{1}{4\alpha_{1j}}}{P^b - n_1 P^{idle} - \sum_{j=1}^{n_1}(\gamma_{1j} - (\beta_{1j}^2)/(4\alpha_{1j}))}\right]^{1/2} \tag{12}$$

$$f_{1j} = \frac{-1 - \mu\beta_{1j}}{2\alpha_{1j}\mu} \tag{13}$$

To calculate the optimal frequencies, we need to know the temperature of all machines in the cluster. Hence, there should be

a global node that is responsible for the calculation. Fortunately, Map-Reduce itself provides the master node that collects information from all nodes through heartbeat messages. We add a temperature field in the heartbeat message and a frequency field in the return value. The size of heartbeat message may range from tens of bytes to hundreds of bytes depending on different MapReduce cluster setup. Appending a four-byte float number will not lead to much network overhead. We discuss the implementation details in Section 4. The formulation above also indicates that the computational complexity is linear for both μ, and $\boldsymbol{F}$. Thus, even though the optimization is done by a single centralized node, it will not consume too much computational resources. The performance evaluation of the algorithm is presented later in Section 5.1.

3.3. Enterprise workload

The 3-tier web server farms are a widely used architecture serving many enterprise web e-commerce systems. The first tier offers the static contents in web pages and typically contributes the least to latency. The second-tier servers usually take care of web applications, while the third tier consists of a collection of database servers, which handle disk-intensive workload. In this section, we discuss an optimization that minimizes the delay of user queries while satisfying a given power budget. The incoming requests are uniformly distributed to all available second tier servers. The hosting application utilizes the Round-Robin balancer of JDBC to access third tier MySQL servers. Tuning both $\boldsymbol{F}$ and $\boldsymbol{N}$ makes the problem too complex to be solved by convex optimization. Therefore, we consider the more powerful knob $\boldsymbol{N}$. Let n_2 and n_3 denote the number of machines in 2nd-tier and 3rd-tier respectively. Therefore, we have:

$$\begin{aligned} w_{ij} &= \frac{W_i}{n_i} \\ \text{where} \quad & i \in \{2,3\}, j \le n_i. \end{aligned} \tag{14}$$

According to our experiments shown in Fig. 7(b) and (c), M/M/1 model captures the service delay trends of both second and third tier servers. Thus, we use Eq. (15) to estimate delay where b_{22} and b_{32} correspond to the service rate for second tier servers and third tier servers respectively.

$$\overline{D} = \sum_{i=2}^{3} \left(\frac{n_i b_{i1}}{n_i b_{i2} - W_i} + b_{i3} \right). \tag{15}$$

The definition domains of n_i is $((W_i)/(b_{i2}), \infty)$, which guarantees the system stability according to M/M/1 queueing model.

The second step is applying application properties to achieve convexity. Since we tune only the number of powered on servers, the CPU frequencies are constants in this case. The 2nd-tier servers handle computational workload that does not touch disk. Therefore we ignore power dynamics induced by disk for these servers. Further, by substituting Eq. (14) in (6), we can reduce the power equation P_{2j}:

$$\begin{aligned} P_{2j} &= \theta_{2j} \frac{W_2}{n_2} + P^{idle}, \\ \text{where}\, \theta_{2j} &= \alpha_{2j} f_{2j} + \gamma_{2j} \frac{1}{f_{2j}} + r_2 a_2 \widetilde{T_{2j}}, \end{aligned} \tag{16}$$

where θ_{2j} is the temperature dependent parameter. As we will present in Section 4.2, the CPU utilizations for 3rd-tier servers are always very low (less than 1 percent), we ignore the power dynamics induced by CPU for these servers. Further, by substituting Eq. (14) in (6), we can reduce the power equation for P_{3j}:

$$\begin{aligned} P_{3j} &= \theta_3 \frac{W_3}{n_3} + P^{idle}, \\ \text{where}\, \theta_3 &= a_6(1 - r_3), \end{aligned} \tag{17}$$

For the purpose of optimization, we first regard n_i as a continuous value, and then round the optimal value to the nearest integer in implementation. Now, we formulate the optimization problem as follows.

$$\begin{aligned} \text{minimize} \quad & \overline{D} \\ \text{s.t.} \quad & \sum_{i=2}^{3} \sum_{j=1}^{n_i} P_{ij} < P^b \end{aligned} \tag{18}$$

We can evaluate that, in the definition domains of n_i, the Hessian matrixes of $\overline{D}$, and P_{ij} are positive definite. Therefore, the objective and constraint are all convex. Hence, the problem is convex. The Lagrangian, $L(\mathbf{N}, \mu)$, corresponding to our optimization problem can be expressed as follows:

$$\begin{aligned} L(\boldsymbol{N}, \mu) &= \overline{D} + \mu \left[P^b - \left(\sum_{i=2}^{3} \theta_i W_i + P^{idle} \sum_{i=2}^{3} \theta_i \right) \right] \\ \text{where}\, \theta_2 &= \frac{\sum_{i=1}^{n_2} \theta_{2j}}{n_2} \end{aligned} \tag{19}$$

At the optimal point, the partial derivation of the Lagrangian with respect to n_i, and μ are zero. Thus, we get:

$$\begin{aligned} \frac{\partial L(\mathbf{N}, \mu)}{\partial n_i} &= \frac{-b_{i1} W_i}{(n_i b_{i2} - W_i)^2} + \mu P^{idle} = 0, \\ \frac{\partial L(\mathbf{N}, \mu)}{\partial \mu} &= P^b - \left(\sum_{i=2}^{3} \theta_i W_i + P^{idle} \sum_{i=2}^{3} \theta_i \right) \end{aligned} \tag{20}$$

From Eq. (20), we get the optimal numbers of servers:

$$\begin{aligned} n_i &= \frac{Q W_i + K (b_{i1} W_i)^{1/2}}{Q b_{i2}}, \\ \mu &= \frac{Q^2}{K^2 P^{idle}}, \\ \text{where}\, K &= \left[\frac{P^b - \sum_{i=2}^{3} \theta_i W_i}{P^{idle}} - \sum_{i=2}^{3} \frac{W_i}{b_{i2}} \right] \prod_{i=2}^{3} b_{i2}, \\ Q &= b_{32} (b_{21} W_2)^{1/2} + b_{22} (b_{31} W_3)^{1/2}. \end{aligned} \tag{21}$$

Eq. (21) presents the closed form solution for n_i. The time computation complexities are linear for computing two knobs, since we only have one summation equation to calculate θ_i.

4. Implementation

In this section, we first specify the hardware used in our experiments. Then we demonstrate how we modify the Hadoop's heartbeat message. Finally, we show the experiments to derive power model and delay model respectively.

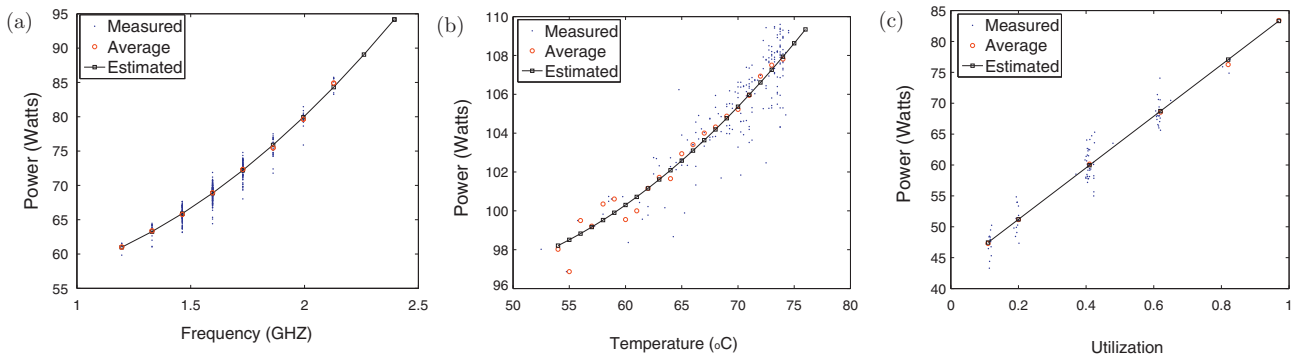


Fig. 4. Correlation between power consumption and different parameters.

4.1. System setup

Our evaluation involves 40 DELL PowerEdge R210 servers. Each server has a 4-core Intel Xeon X3430 2.4 GHz CPU. There are 10 available frequencies for each core: 1.197 GHz, 1.330 GHz, 1.463 GHz, 1.596 GHz, 1.729 GHz, 1.862 GHz, 1.995 GHz, 2.128 GHz, 2.261 GHz, and 2.394 GHz. CPU turbo boost is turned off to avoid unpredictable dynamics. The server itself is equipped with CPU temperature sensors, and we collect the reading via lm_sensor module. There are two options available for the fan speed of our servers: minimum power and maximum performance. Both of them employed a closed feedback loop to dynamically determine proper fan speeds. As shown in Fig. 3, the characteristic of temperature power relationship does not change much for two fan configuration options. There is only a constant difference between them. Therefore, we choose the minimum power option in all our experiments to minimize the power consumed by fans. For the cloud computing case, we use *Hadoop* – 0.20.2 as an implementation of Map-Reduce. One machine is nominated as a dedicated master node, *i.e.*, TaskTracker and other programs for executing tasks are not running on master node. For enterprise case, 18 dedicated 2nd-tier servers are installed with *Tomcat* – 6.0.32 and 15 dedicated 3rd-tier servers are installed with *MySQL* – 5.5.15. Besides, we have 3 dedicated Remote Browser Emulators (RBE), and one dedicated experiment controller which automates the experiment process and logs power consumption. We use one implementation of TPC-w benchmark designed by Wisconsin–Madison to populate our database [36]. Each MySQL server stores tables with 1×10^7 items (more than 5 GB) by *MyISAM* engine. The max_connection variable of MySQL is modified to tolerate up to 3000 connections. We use Avocent PM 3000 power distribution unit (PDU) to supply power for the rack, and directly collect readings from the PDU. The temperature of the server room is set to 18.3 °C (65 °F).

4.2. Power equation

Because of temperature differences, the power consumption of one machine can be different even though it is running at the same frequency and same utilization. Understanding the relationship between CPU frequency, temperature and power consumption is crucial for conducting proper optimization. Fig. 4 shows how frequency and temperature influence power consumption.

Among, these three parameters, utilization and frequency can be well controlled by tuning workload and using DVFS. However, CPU temperature is not directly tunable. Actually, the CPU thermal states intricately relates to intake air temperature, air flow speed, location, CPU power draw, etc., which is very hard to model. To walk around this, we tune the CRAC set point to affect CPU temperature. Then, we conduct 350 experiments that covers all combination of 10 frequencies (1.2–2.4 GHz), 6 utilization states (0–100%), and 7 CRAC set points (50–80 °F). Finally, to plot the correlation of CPU power consumption with each individual parameter, we pick a subset of the data in which two parameters are roughly constant and the third one is changing. Fig. 4 shows the relationship between power consumption and three different parameters. The CPU utilization linearly relates to the power consumption while both CPU frequency and CPU temperature show a quadratic relationship. To profile disk power consumption, we submit *SELECT* request to MySQL data base with different frequency. Fig. 7(a) presents the linear relationship between disk block-read-per-second (br) relates and power consumption.

By piecing different parameters together, we get the following power model:

$$P = u \times (f^2 + 1.5253 \times f) \times (0.9937 \times T^2 - 1.3357 \times T + 3.2714) + 3.5321 \times br + 40.5743,$$

As we argued before, for CPU intensive Hadoop workload load, the CPU utilization is always high. Thus, we pick the data set whose CPU utilization is higher than 99, and rerun the regression algorithm with a simpler target equation. Thus, we get:

$$P = 0.0069 \times T^2 + 7.3865 \times f^2 + 0.0405 \times f \times T - 0.4351 \times T - 6.9793 \times f + 61.1205.$$

Regression result is shown in Fig. 5 which fits well with our observed data. The average error is about 0.106 W which is relatively small comparing to the total power consumption of one machine.

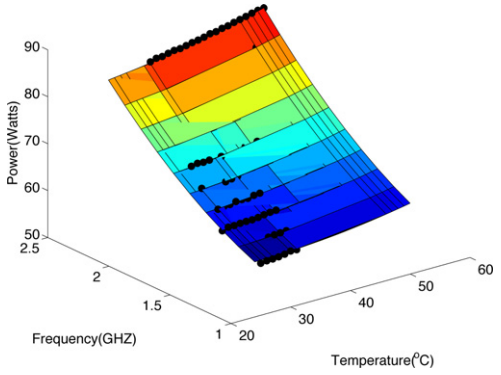


Fig. 5. Our power model and observed data: every dot shows the average power for one temperature–frequency pair according to our measured data.

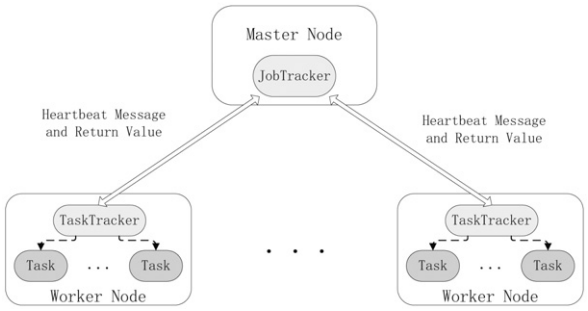


Fig. 6. Hadoop cluster: there is one master node and several worker nodes in a Hadoop cluster. The JobTracker running on master node is responsible for task assignment while the TaskTracker running on worker node will launch tasks and report available slots to JobTracker. JobTracker and TaskTracker communicate with each other via Heartbeat message and its return value.

In the enterprise experiment, by applying specific restrictions to power model, we have:

$$\begin{aligned} P_{2j} &= 0.0782 \times w_{2j} \times (0.9937 \times T_{2j}^2 - 1.3357 \times T_{2j} + 3.2714) + 40.5743, \\ P_{3j} &= 0.021 \times w_{3j} + 40.5743. \end{aligned} \tag{22}$$

One thing to mention is that, when the MySQL server is handling disk-intensive requests, we observed that the CPU utilization is always less than 1 percent. Thus, we ignore the power consumption induced by CPU for third tier servers.

4.3. Modify heartbeat message

In the cloud computing case study, a Map-Reduce cluster consists of a master node and many worker nodes. More specifically, for Hadoop, there is a JobTracker program running on master node and a TaskTracker program running on each worker node as shown in Fig. 6. The TaskTracker sends heartbeat messages to inform JobTracker that the worker node is alive. In the heartbeat message, the TaskTracker will also indicate whether it is ready to run a new task. JobTracker responses a return value to TaskTracker which includes response ID, the interval for next heartbeat and probably information about a new task if the TaskTracker is ready. This structure grants global information to master node which is naturally suitable to run some centralized optimization algorithms.

We add a CPU temperature field to the heartbeat message. TaskTrackers sense the CPU temperature of its hosting machine and update this information before sending out heartbeat. In this way, JobTracker knows temperatures of all machines. JobTracker periodically calculates optimal frequencies for the cluster and sends it back via the return value of heartbeat. The calculation interval can be a tunable parameter which indicates the trade-off between optimization overhead and agility to dynamics. As discussed in Section 3.2, the computational complexity of our optimization algorithm is linear. Thus, it will not be a high overhead even if the cluster contains thousands of machines. The JobTracker only returns the optimal frequency and the frequency selection is done by worker nodes in order to reduce computation overhead on master node.

4.4. Delay equation

In the enterprise workload case study, to capture the relationship between delay and load, we use one workload generator to produce *SELECT* queries, and record the average delay for every 100 request. Fig. 7(b) and (c) shows the experiment results of profiling delay-load relationship for second and third tier web servers. We use M/M/1 model to approximate the trends. Below show the detailed delay equation:

$$\begin{aligned} D_i^C &= \frac{16440}{62 - w_i^C} - 293, \\ D_i^{IO} &= \frac{10068}{300 - w_i^{IO}} - 330. \end{aligned} \tag{23}$$

5. Evaluation

In this section, we evaluate the power consumption, service delay and reactions to dynamics for TAPA with respect to both MapReduce and web enterprise web servers.

5.1. TAPA and cloud computing workload

Our evaluation includes five different settings, one with our temperature-aware DVFS algorithm enabled (TAPA), three static policies with CPU frequencies set to maximum, minimum and median respectively, and one temperature-oblivious DVFS policy proposed in [32]. All experiments involve 11 worker nodes, one master node, and one logging node which is responsible for log the real power readings from the power unit. We call them *TAPA*, *MAX*, *MIN*, *MED*, and *DVFS* below for short. For the DVFS one, we use their final control formulation shown in Eq. (24),

$$f_i(k) = \frac{f_i(k-1)\sum_{T_{jl} \in S_i} c_{jl} r_j}{(U_s - u(k))f_i(k-1) + \sum_{T_{jl} \in S_i} c_{jl} r_j}. \tag{24}$$

Descriptions of all symbols are shown in Table 2. Since this experiment focuses on computationally intensive jobs, we set the U_s to 0.9. In Map-Reduce framework, master node will assign new task to a worker node whenever there is empty slot on that node, which keeps every core busy. Therefore, the estimated execution time for each processor can be set to the length of one control period

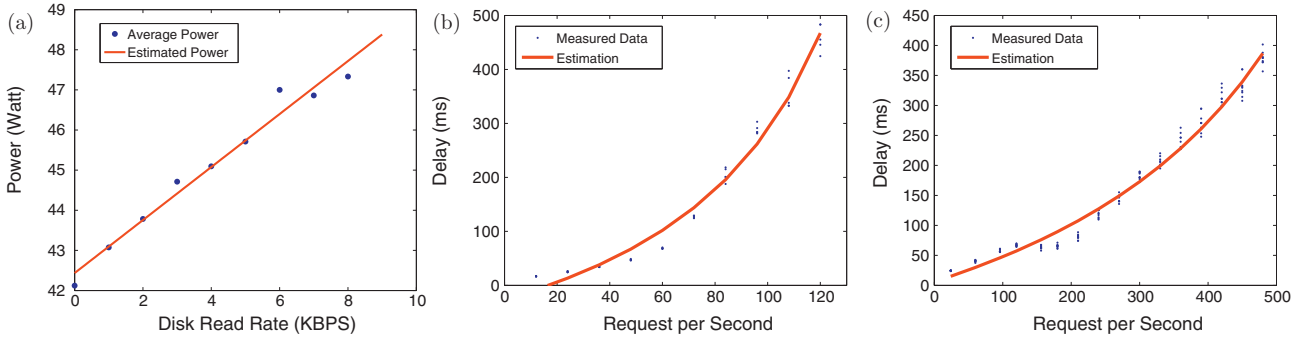


Fig. 7. (a) Relationship between power consumption and disk BPS for database workload. (b) and (c) Relationship between delay and request-per-second for second and third tier servers respectively.

Table 2
Symbol descriptions for temperature oblivious DVFS.

Symbol	Description	Symbol	Description
$f_i(k)$	CPU frequency in the *kth* control period	U_s	CPU utilization set point
$u(k)$	measured CPU utilization in period k	T_{jl}	subtask of Task T_j
S_i	set of subtasks located at processor P_i	r_j	calling rate of task T_j
c_{jl}	estimated execution time for subtask T_{jl}		

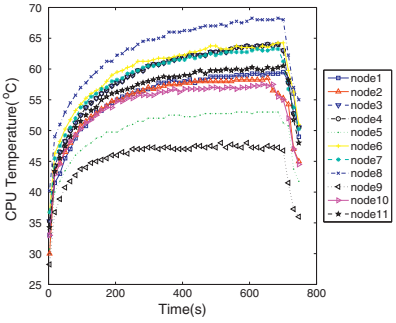


Fig. 8. Thermal gap in MED.

(*i.e.*, $\sum_{T_{jl} \in S_i} c_{jl} r_j$ is replaced by a constant value). We apply a 1000 W power budgets to TAPA with 50 W slack (*i.e.*, TAPA will try to keep the power consumption below 950 W).

All of the experiments with different settings run the same computing PI job with 6,000,000,000 sample points, an example job provided by the original Hadoop-0.20.2 release. We configure each machine to hold 4 mapping slots since each of them has 4 cores. The mapper class here generates random points in a unit square and then counts points inside and outside the inscribed circle of the square. The reduce class accumulates point counts results from mappers and use this value to estimate PI. For the mapping phase, the cluster is computation intensive since all worker nodes are busy calculating and the traffic in the network is mostly heartbeat messages and its return value. When this job reaches its reducing phase, the network starts to become busy because many data need to pass from mappers to reducers.

In the MIN experiment, the CPUs with minimum frequencies do not generate too much heat and our current cooling system is adequate to cool down servers with MIN setting. Hence, we do not see large thermal gaps in MIN experiments. As we have shown previously in Fig. 3, the thermal gap in MAX is about 20 °C. Here, we show the temperature differences for worker nodes of MED and TAPA experiment in Figs. 8 and 9 respectively. As the job transfers from mapping phase to reduce phase, the workload transfers from computationally intensive to network-intensive. Therefore,

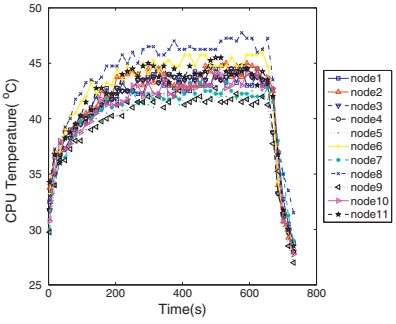


Fig. 9. Thermal gap in TAPA.

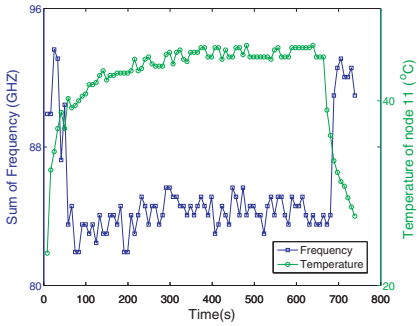


Fig. 10. Dynamic response.

there is a sudden drop in the end. Compared with Fig. 5, the thermal effect is even severe in this experiment. In MED experiment, the neighbors of each machine also generate a lot of heat. Therefore, the heat cannot be drained as easy as one machine cases. We can conclude at least two benefits from the comparison. Firstly, the maximum temperature difference between nodes in MED reaches 20 °C, while it is only about 5 °C in TAPA. It is obvious that TAPA achieves much smaller temperature deviation than MED. Secondly, the temperature of the hottest node is reduced from 68 °C to 48 °C. Although we do not consider eliminating hot spot as a direct objective, reducing energy cost by reducing temperature achieves this as a bonus effect. These two benefits are very meaningful in data centers. Smaller deviation means that the cooling part does not have to be over-provisioned. Operators can set the temperature target on CRAC to be just-enough for all machines. It can be a big saving from the cooling side. Lower temperature can help to increase system stability and lengthen device life time. Therefore, it is another save in the long run. For each machine, there is a temperature drop in the tail. We believe it is the time point when the map tasks have been finished and the whole cluster changes to the dedicated reduce phase. Therefore, at this time point, the job changes from computation intensive to communication intensive and the utilizations of CPUs decrease to low status.

In Fig. 10, we can see how our algorithm reacts to the temperature dynamics. This figure shows the relation between the frequency summation and the temperature trend. As shown in Fig. 9, different servers have a similar temperature trend. Thus, we only use temperature curve from one machine to show this trend without making the figure too complicated. At the beginning, the temperature increases very fast because the cluster just switched from idle to full utilization status. TAPA immediately decreases the computational capacity of the whole cluster to curb power consumption below the budget. As the system keeps running, the temperature changes which will lead to different power consumption. In order to compensate for temperature differences, our strategy decreases total CPU frequency when temperature increases and increases total CPU frequency when temperature decreases.

Finally, we show the comparison among TAPA, MAX, MIN, MED, and DVFS based on their power consumption and efficiency. Fig. 11 shows the experiment result. The power budget is set to be 1000 W with 50 W slack for TAPA. For other approaches, the power budget does not apply to them since they do not have mechanism to control power consumption. From the Fig. 11, we can see that TAPA makes full use of the power budget without violating it. In MAX experiment, the power consumption rises as time passing by. It is because the machines gets hotter and hotter when running this computational intensive job. It is amazing that the power gap caused by temperature reaches about 200 W in our small cluster. The power consumption does not increase remarkably in MIN because machines are set to use the lowest frequency, with which

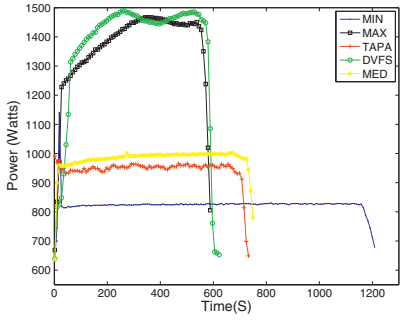


Fig. 11. Power consumption.

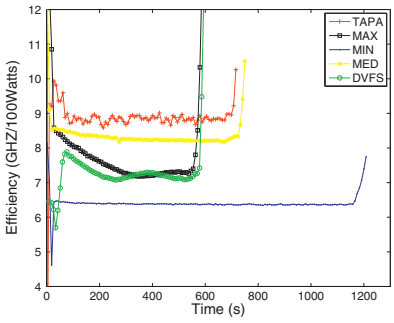


Fig. 12. Efficiency.

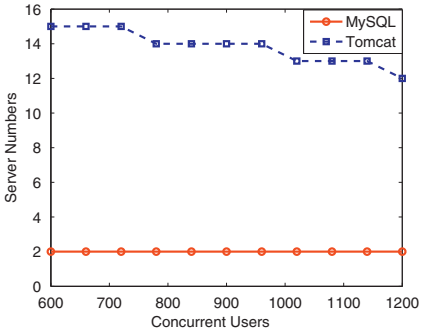


Fig. 14. Number of servers.

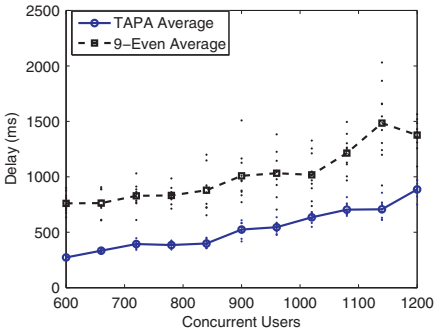


Fig. 15. Service delay.

the temperature can only reach about 34 °C as shown in Fig. 5. For the MED experiments, the average temperature is about 55 °C which leads to a 5 percent increase in total power. This also matches with our experiments result shown in Fig. 5. For the DVFS experiment, since all worker nodes stay in very high utilization status in map phase, the CPU frequency keeps increasing until it reaches the upper bound. Therefore, it shows a similar result as MAX. At the tail of each curve, there is a sudden drop of power. We believe it is because the cluster is transferring from mapping phase to reducing phase, and the CPUs no longer run at a high utilization.

Fig. 12 shows the frequency summation efficiency of five scenarios. The efficiency is defined as F_s/P, where F_s is the frequency summation of the whole cluster and P is the power consumption. It is clear that TAPA is more efficient to transform power into computational capacity when comparing with other solutions. As shown in Fig. 13, TAPA uses about 25% less energy comparing to the thermal-oblivious DVFS solution and about 5.6% less energy comparing to the most efficient static solution, MED. Given the huge expenditure spent on data center power supply, TAPA will help to achieve significant savings.

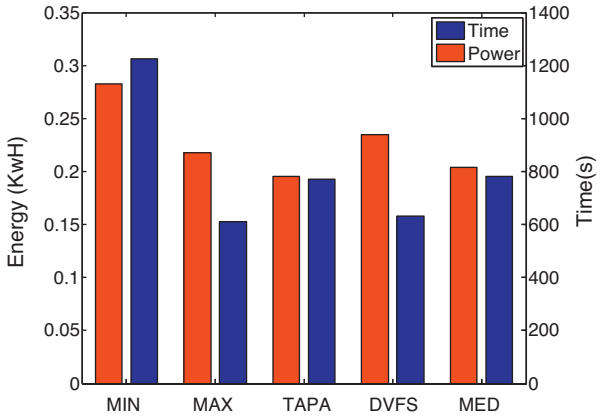


Fig. 13. Energy and duration.

5.2. TAPA and enterprise workload

In this section, we evaluate how our solution performs with web servers. The intuition behind TAPA is to allocate the right number of servers for second and third tiers respectively, which minimizes service delay while satisfying a global power constraint. The experiment is designed in the way that W^C and W^{IO} are the same, and when cluster utilization is very low, delays induced by Tomcat server and MySQL server are roughly the same. But, as the volume of load goes up, the delay on Tomcat servers increases much faster than MySQL servers. We compare our solution with a static allocation policy such that both second tier and third tier have 9 dedicated servers (9-Even). We set the RBE's average user think time to 1 second, and increase the number of users from 600 to 1200 with step 60.

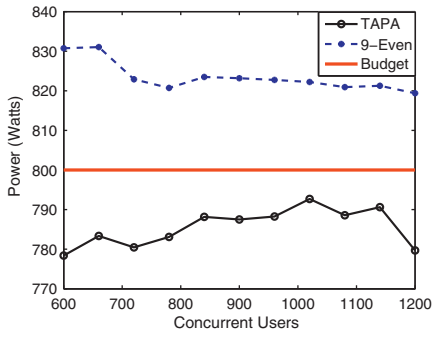


Fig. 16. Power consumption.

In Fig. 14, we show how TAPA reacts to as load increases. To curb the total power consumption, TAPA uses less machine to serve heavier load. Because the power induced by each machine rises with load, and we will have to reduce the amount of static power by turning off machines.

Figs. 15 and 16 together show the whole system power consumption and service delay for different numbers of users. We can clearly see that TAPA won in both aspects. In Fig. 15, even though 9-Even policy uses more servers, allocating even number of servers for each tier results in longer delay than TAPA. The reason is that, the bottleneck of the system is in 2nd-tier. Spending too much servers on 3rd-tier does not help to reduce delay. In Fig. 16, it is not surprising that 9-Even uses more power, since it is using more machines. One phenomenon we want to point out is that, the power consumption for 9-Even cases actually decreases as load goes up, we believe it is because when the arrival rate increases, the 2nd-tier spends more CPU time on accepting requests, which in turn reduces the number of requests it can serve in each second. The power consumption of tomcat servers will not change much, since the CPUs are already fully utilized. But, as the 2nd-tier passes less load to databases, the power consumption of the 3rd-tier will decrease.

6. Conclusion and future work

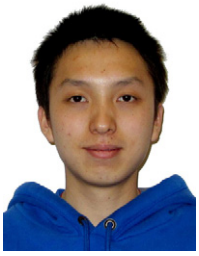
In this paper, we present a general optimization framework for minimizing delay with a given power budget. We also provide a versatile power model that takes both CPU and disk into consideration. Based on the power model, we apply our optimization framework to two very different types of applications: Hadoop, and enterprise web servers. For each application, we only need to plug its specific delay constraints into the framework, and derive a closed form optimization solution. The computational complexities for both cases are linear. The experiment results show that our solution can curb the power consumption very well in dynamic environments, simultaneously reducing both power consumption and delay. In our work, we consider only computational and I/O workloads. In order to handle all types of workload, we still need to conduct a more comprehensive study on other components, such as GPU, memory, and NIC, which is a subject of future work.

Acknowledgements

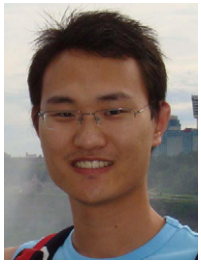
This work was supported in part byNSF grants CNS 09-16028, CNS 09-58314, CNS 10-35736, and CNS 1040380, as well as by the LCCC and eLLIIT centers at Lund University, Sweden.

References

[1] J. Hamilton, Cost of Power in Large-Scale Data Centers, http://perspectives.mvdirona.com/2010/09/18/OverallDataCenterCosts.aspx, November 2008.
[2] Gartner, Gartner Says Data Center Power, Cooling and Space Issues are Set to Increase Rapidly as a Result of New High-Density Infrastructure Deployments, http://www.gartner.com/it/page.jsp?id=1368614, May 2010.
[3] U.S. Department of Energy, Quick Start Guide to Increase Data Center Energy Efficiency, Retrieved June 2010.
[4] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, in: USENIX OSDI, December, 2004, pp. 137–150.
[5] Y. Chen, T. Wang, J.M. Hellserstein, R.H. Katz, Towards Energy Efficient MapReduce, http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-109.pdf, August 2009.
[6] Y. Chen, A. Ganapathi, A. Fox, R.H. Katz, D. Patterson, Statistical Workloads for Energy Efficient MapReduce, http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-6.html, January 2010.
[7] Hadoop, http://hadoop.apache.org, October 2011.
[8] Disco, http://discoproject.org, October 2011.
[9] Misco, http://www.cs.ucr.edu/~jdou/misco, October 2011.
[10] P. Macken, M. Degrauwe, M.V. Paemel, H. Oguey, A voltage reduction technique for digital systems., in: IEEE International Solid-State Circuits Conference, February, 1990.
[11] W. Liao, F. Li, L. He, Microarchitecture level power and thermal simulation considering temperature dependent leakage model, in: International Symposium on Low Power Electronics and Design, August, 2003, pp. 211–216.
[12] L. Yuan, S. Leventhal, G. Qu, Temperature-aware leakage minimization technique for real-time systems, in: ICCAD, Novoember, 2006, pp. 761–764.
[13] L. He, W. Liao, M.R. Stan, System level leakage reduction consider the interdependence of temperature and leakage, in: Design Automation Conference, July, 2004, pp. 12–17.
[14] W. Liao, L. He, Coupled power and thermal simulation and its application, in: Workshop on Power-Aware Computer Systems, December, 2003, pp. 148–163.
[15] Y. Liu, R.P. Dick, L. Shang, H. Yang, Accurate temperature-dependent integrated circuit leakage power estimation is easy, in: Conference on Design, Automation and Test in Europe, April, 2007, pp. 1–6.
[16] C.M. Liang, J. Liu, L. Luo, A. Terzis, F. Zhao, RACNet: a high-fidelity data center sensing network, in: ACM Sensys, November, 2009, pp. 15–28.
[17] A. Gandhi, M.H. Balter, R. Das, C. Lefurgy, Optimal power allocation in server farms, in: SIGMETRICS/Performance, June, 2009, pp. 157–168.
[18] C. Bash, G. Forman, Cool job allocation: measuring the power savings of placing jobs at cooling-efficient locations in the data center, in: USENIX Annual Technical Conference, June, 2007, pp. 363–368.
[19] T. Horvath, T. Abdelzaher, K. Skadron, X. Liu, Dynamic voltage scaling in multi-tier web servers with end-to-end delay control, IEEE Transactions on Computers 56 (April (4)) (2007) 444–458.
[20] V. Sharma, A. Thomas, T. Abdelzaher, K. Skadron, Z. Lu, Power-aware QoS management in web servers, in: IEEE RTSS, December, 2003, pp. 63–73.
[21] L. Wang, Y. Lu, Efficient power management of heterogeneous soft real-time clusters, in: IEEE RTSS, December, 2008, pp. 323–332.
[22] R.K. Sharma, C.L. Bash, C.D. Patel, R.J. Friedrich, J.S. Chase, Balance of power: dynamic thermal management for internet data centers, IEEE Internet Computing (January) (2005) 42–49.
[23] J. Moore, J. Chase, P. Ranganathan, R. Sharma, Making scheduling "cool": temperature-aware workload placement in data centers, in: USENIX Annual Technical Conference, April, 2005, pp. 61–75.
[24] J. Leverich, C. Kozyrakis, On the energy (in)efficiency of hadoop clusters, in: ACM SIGOPS Operating Systems Review, January, 2009, pp. 61–65.
[25] R. Kaushik, M. Bhandarkar, GreenHDFS: towards an energy-conserving, storage-efficient, hybrid hadoop compute cluster, HotPower (October) (2010) 1–9.
[26] W. Lang, J. Patel, Energy management for mapreduce clusters, VLDB, 2010, September, pp. 129–139.
[27] J. Heo, P. Jayachandran, I. Shin, D. Wang, T. Abdelzaher, X. Liu, OptiTuner: on performance composition and server farm energy minimization application, IEEE Transactions on Parallel and Distributed Systems (November) (2011) 1871–1878.
[28] Z. Liu, M. Lin, L.L.H. Andrew, Greening geographical load balancing, in: SIGMETRICS, June, 2011, pp. 193–204.
[29] A. Banerjee, T. Mukherjee, G. Varsamopoulos, S. Gupta, Cooling-aware and thermal-aware workload placement for green HPC data centers, in: IGCC, August, 2010, pp. 245–256.
[30] M. Etinski, J. Corbalan, J. Labarta, M. Valero, Optimizing job performance under a given power constraint in HPC centers, in: IGCC, August, 2010, pp. 257–267.
[31] S. Li, T. Abdelzaher, M. Yuan, TAPA: temperature aware power allocation in data center with Map-Reduce, in: IGCC, July, 2011, pp. 1–8.
[32] X. Wang, X. Fu, X. Liu, Z. Gu, Power-aware CPU utilization control for distributed real-time systems, in: Real-Time and Embedded Technology and Applications Symposium, April, 2009, pp. 233–242.
[33] M. Kihl, G. Cedersjö, A. Robertsson, B. Aspernäs, Performance measurements and modeling of database servers, in: Sixth International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks (FeBID 2011), June, 2011.
[34] T. Heath, B. Diniz, E.V. Carrera, W. Meira, R. Bianchini, Energy Conservation in heterogeneous server clusters, in: Proceedings of the Symposium on Principles and Practice of Parallel Programming (PPoPP), June, 2005, pp. 186–195.
[35] K. Morton, A. Friesen, M. Balazinska, D. Grossman, Estimating the progress of mapreduce pipelines, in: IEEE International Conference on Data Engineering, March, 2010, pp. 681–684.
[36] http://pharm.ece.wisc.edu/tpcw.shtml, October 2011.



Shen Li joined Computer Science Department in University of Illinois at Urbana-Champaign as a Ph.D. student in 2010. Shen Li received his B.S. degree at Computer Science Department of Nanjing University, P.R. China. His research interests fall in Cloud Computing, especially energy and resource management in Data Centers.



Shiguang Wang is a Computer Science Ph.D. student at University of Illinois at Urbana-Champaign. He received B.S. from Nanjing University, P.R. China, 2008 and MS from Illinois Institute of Technology, 2011. His research interests span algorithm and system design for wireless ad hoc and sensor networks.

Tarek Abdelzaher received his B.Sc. and M.Sc. degrees in Electrical and Computer Engineering from Ain Shams University, Cairo, Egypt, in 1990 and 1994, respectively. He received his Ph.D. from the University of Michigan in 1999 on Quality of Service Adaptation in Real-Time Systems. He has been an Assistant Professor at the University of Virginia, where he founded the Software Predictability Group until 2005. He is currently a Professor and Willett Faculty Scholar at the Department of Computer Science, the University of Illinois at Urbana Champaign. He has authored/coauthored more than 170 refereed publications in real-time computing, distributed systems, sensor networks, and control. He is Editor-in-Chief of the Journal of Real-Time Systems, and has served as Associate Editor of the IEEE Transactions on Mobile Computing, IEEE Transactions on Parallel and Distributed Systems, IEEE Embedded Systems Letters, the ACM Transaction on Sensor Networks, and the Ad Hoc Networks Journal. He was Program Chair of RTAS 2004, RTSS 2006, IPSN 2010, ICDCS 2010 and ICAC 2011, as well as General Chair of RTAS 2005, IPSN 2007, RTSS 2007, DCoSS 2008, and Sensys 2008. Abdelzaher's research interests lie broadly in understanding and controlling performance and temporal properties of networked embedded and software systems in the face of increasing complexity, distribution, and degree of embedding in an external physical environment. Tarek Abdelzaher is a member of IEEE and ACM.

Maria Kihl is an Associate professor at Lund University, where she belongs to the Broadband Communications Research group at the Department of Electrical and Information Technology. She is one of the principal investigators in the Linneaus Center LCCC, Lund Center for control of complex engineering systems. Also, she is involved in ELLIIT, the Excellence Center Linköping-Lund in Information Technology. During 2005–2006, she was a visiting researcher at NC State University. Her main research area is performance of distributed telecommunication applications. Currently, her projects are related to control of Internet server systems, Internet application monitoring and traffic measurements, and design of safety applications in vehicular ad-hoc networks.

Anders Robertsson received the M.Sc.E.E. degree and the Ph.D. degree in automatic control from LTH, Lund University, Lund, Sweden, in 1992 and 1999, respectively. He was appointed Docent in 2005. He is, since January 2012, Professor at the Department of Automatic Control, LTH, Lund University. His research interests are in nonlinear control theory, robotics, observer-based control, and feedback control of computing systems.