E-cubed: End-to-End Energy Management of Virtual Desktop Infrastructure with Guaranteed Performance

Xing Fu, Tariq Magdon-Ismail, Vikram Makhija, Rishi Bidarkar, Anne Holler
VMWare

Jing Zhang
University of Southern California

Abstract

Virtual desktop infrastructure (VDI) deployments are a rapidly growing segment in the Mobile/Cloud Era. Compared to traditional enterprise desktop deployments, VDI can reduce the total cost of ownership by as much as 50%. However, the cost of powering a VDI deployment is still a significant IT expense. Typically these deployments consist of a complex system of interconnected server, storage and networking components. Thus, it is challenging to minimize energy consumption of the entire system while at the same time satisfying performance requirements. In this work, we first derive an accurate VDI performance model and then propose a hierarchical heuristic to minimize energy consumption without violating performance constraints. We also demonstrate that such an approach reduces algorithmic complexity significantly. Results from hardware experiments show that in scenarios with low consolidation ratios energy savings range from 3% to 6%, while for high consolidation ratios they range from 11% to 25%. In all cases, the measured end-to-end performance penalty is minimal.

1 Introduction

Virtual infrastructure allows data center operators to reduce IT costs, including electricity. Virtual machine consolidation increases the utilization of physical infrastructure, making the data center more efficient and reducing its carbon footprint. Closely following on the heels of server consolidation, enterprises are fast adopting virtual desktop infrastructure (VDI) to replace and consolidate existing physical desktops as well. In a VDI environment, a user's operating system instance and applications are run on a virtual machine hosted in the enterprise data center. Users remotely control the virtual machines using thin clients such as stateless hardware terminals, smartphones or tablet PCs.

Many have studied aspects of energy management for pieces of a VDI system, such as networking [10][5], embedded and mobile devices [8][12][17] and data centers [9][7] etc. However, all existing work tackled these separately, as isolated components. An integrated solution for VDI energy management does not exist in industry today. A challenge in extending existing work and applying it to VDI energy management is that they enforce performance (such as CPU utilization [8] or throughput) at the granularity of a server [10][7]. In contrast, a VDI deployment requires integrated management of end-to-end energy and performance.

In this work, we minimize energy consumption by manipulating various knobs such as CPU DVFS levels and consolidating virtual machines. A key challenge is to guarantee that performance will not be adversely affected, leading to undesired violations of service level agreements. To address this challenge, we establish a performance model which predicts end-to-end performance of VDI workloads, given CPU DVFS levels and consolidation ratios etc. We select a collection of typical applications used by VDI users, and define a relevant end-to-end performance metric. We do this instead of adopting well-known CPU utilization or throughput metrics because they don't sufficiently reflect a user's experience with interactive applications (which is of prime importance in a VDI deployment). We derive an accurate performance model using a black-box modeling approach.

Based on the model, we formulate an optimization problem to minimize the energy consumption while guaranteeing performance, and transform it into a canonical form which can be solved by standard optimization solvers. However, no polynomial time solvers exist to obtain the optimal solution. To scale the proposed solution in VDI deployments, which can have thousands of seats (VMs), a two-step heuristic algorithm is designed to reduce the algorithmic complexity significantly.

We prototype the proposed solution and analyze the overhead of each component of the implementation to ensure that the overall solution will not introduce significant performance degradation or increased energy consumption. Experimental results from a hardware test bed demonstrate the efficacy of the proposed solution in

terms of energy and performance. It significantly outperforms the state-of-the-art baseline widely adopted in industry today.

The remainder of the paper is organized as follows. In Section 2 we describe end-to-end energy management with performance guarantees. In Section 3, we present details of the system implementations and experimental results. Finally, Section 4 concludes the paper.

2 Energy Management with Performance Guarantee

In this section, we first present the system architecture of E-cubed (End-to-End Energy Management). We then present a model of end-to-end performance for a VDI workload. Finally, we present a formulated optimization algorithm and heuristic for large-scale deployments.

2.1 System Architecture

Figure 1 shows the E-cubed system architecture which works as follows. (1) Each user remotely controls a virtual machine called a desktop VM via a thin client (such as a smartphone or a tablet PC). In the data center, physical servers host all the desktop VMs. The desktop VMs are responsible for workload execution while the thin clients only render the display and transmit user input (from keyboards and mice) using a virtual desktop communication protocol. A monitor periodically collects the utilization of the desktop VMs and detects input events (such as key presses) to determine the number of active desktop VMs. The number of active desktop VMs changes over time due to idle periods during nights and holidays and long idle interval in office hours [15]. The number of active desktop VMs and their associated physical servers are sent to the E-cubed. (2) The core of E-cubed is a constraint nonlinear optimizer that minimizes the end-to-end energy consumption by various knobs. They include throttling CPU DVFS levels of the thin clients, throttling link rates and shutting down any idle ports on the network switches, and decreasing hard disk rotational speeds of the shared storage. Since energy consumption of servers accounts for a large portion of all energy consumption, in this work, we focus on manipulating server DVFS levels and consolidating desktop VMs. The work can be extended to integrate all knobs in the future. A key challenge is to define the performance metrics of a VDI deployment and model the relationship between the performance metrics, CPU DVFS levels and VMs consolidate ratios (a.k.a the number of VMs hosted on a single host). A performance model enforces a user-specified requirement. The detailed derivation of the performance model is in Section 2.2. Moreover, the optimizer takes into account hardware constraints such as making sure the CPU frequencies are adjusted according to hardware specific ranges. (3) The optimizer throttles the CPU DVFS levels of physical servers and consolidates desktop VMs by live VM migration according to the output of the optimizer.

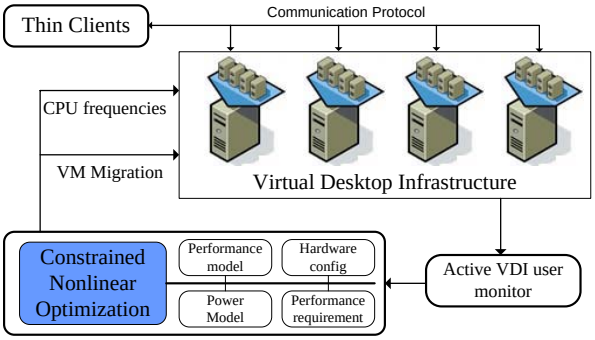


Figure 1: Integrated management architecture for virtual desktop infrastructure.

2.2 Performance Model

We first introduce notations. Q_i is the performance, f_i is chip-wide CPU frequency[1], and R_i is the number of desktop VMs hosted on the ith server. N_{host} is the total number of physical servers in a VDI deployment. N_{VM} is the total number of active desktop VMs. S_i is the status of the ith physical server. If the ith server hosts dekstop VMs, $S_i = 1$. If the server is idle and powered off, $S_i = 0$. QoS is a user-specified performance requirement, which is a constant. $f_{i,\min}, f_{i,\max}$ are minimum and maximum frequencies of the ith physical server.

2.2.1 End-To-End Performance

View Planner is used to generate realistic VDI workloads by emulating user operations. It performs a series of random operations of all applications of the collection. Between operations, a random sleep interval emulates user think time. Different operations emulate different users. We select a collection of applications which represent typical workloads for most VDI deployments in a production environment. The collection includes Adobe Reader, Word, Excel, Outlook, PowerPoint, Video, Internet Explorer, and 7-ZIP. For each application, users may perform various operations. For example, open a Word document, browse, edit, finally save the document.

[1]Recent CPU models from both Intel and AMD support per-core DVFS throttling and per-tile DVFS throttling. More fine grained DVFS throttling can be utilized in future work.

Because VDI users expect their VMs to be "responsive", the first priority performance metric is response time of an operation. The end-to-end performance is defined as a high percentile response time of all operations performed by a VDI user. Usually, it is defined as the 95th percentile response times or the 98th percentile response times. Since the response time measurement has variation, we take the average to reduce its variation. Detailed analysis of the VDI performance definition is out of scope of this paper.

2.2.2 Black-box Approach

In order to have an effective E-cubed design, it is necessary to model the performance of a VDI deployment, specifically, the closed-form mathematical relationship between the 95th percentile latency of all VDI user operations and the actuators described in Section 2.1. Since a virtual desktop infrastructure deployment is a complicated computer system, a well-established physical equation based on a queuing system is not available. To address the challenge, we select a black-box approach, namely *surface fitting*.

We establish the performance model for a single physical server off-line by running a typical VDI workload and varying CPU DVFS levels and consolidation ratios and measuring the end-to-end performance. Based on the collected data, an accurate model is derived by surface fitting the datapoints. View Planner can be used to generate realistic VDI workloads by emulating user operations. It performs a series of random operations of all applications of the collection. Between operations, a random sleep interval emulates user think time. Different operations emulate different users. CPU frequency is varied from the highest frequency to the lowest frequency, and the consolidation ratio is increased until the performance is much lower than a specified threshold. For clusters consisting of homogeneous servers, which is the most common in production, the performance model for a single physical server holds for the other identical servers. For clusters of heterogeneous servers, each type of server will need its own model.

Performance measurement shows a strong nonlinear relationship between the end-to-end performance and manipulated variables (CPU DVFS levels and consolidation ratio). Beyond a certain threshold, performance degrades significantly as DVFS level decreases and consolidation ratio increases. Some black-box techniques such as system identification [13] and linear regression [11] cannot be applied because of their assumption of linearity. Machine learning is a powerful approach [14] for deriving a complex model. However, [14] adopts an artificial neural network to represent a derived model, and an explicit mathematical formula is unavailable. The performance model for the ith server can be expressed as follows.

$$Q_i\left(f_i, R_i\right) = K_i\left(\frac{a_1}{f_i} + a_2\right)\left(b_1 {R_i}^2 + b_2 R_i + b_3\right) \tag{1}$$

Constant K_i denotes response time when only one desktop virtual machine runs on the ith server and its CPU runs at the maximum frequency. The term $\left(\frac{a_1}{f_i} + a_2\right)$ represents response time inflation due to CPU frequency throttling, and equals to 1 when $f_i = f_{\max}$. The term $\left(b_1 {R_i}^2 + b_2 R_i + b_3\right)$ represents response time inflation due to VM consolidation, and equals to 1 when $R_i = 1$. The coefficient of determination $\mathrm{R}^2 = 0.85$.

Figure 2 shows the difference of predicted performance based on the model and actual measurement. Configurations 1-8 are combinations of randomly-selected CPU DVFS levels and randomly-selected consolidation ratios. Those configurations are different from configurations used to establish the performance model as described before. Small difference shown in Figure 2 shows the end-to-end performance model (1) predicts accurately.

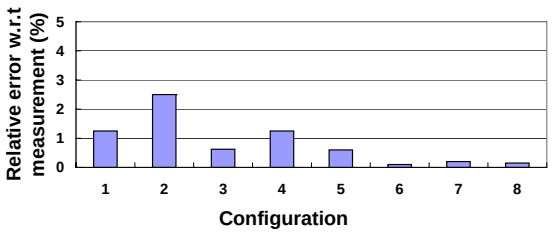


Figure 2: Validation of the performance model against measurement.

It is possible that the performance model derived off-line does not hold in a production environment in some scenarios. For example, hardware upgrade, physical server replacement, users running applications which are not listed in the profiled collection, or computer security compromises and resource exhaustion due to a Denial-of-service attack. However, those scenarios can be detected by comparing the predicted value based on the model and real-time performance measurement. If the difference is abnormally large for a long interval, a change of a VDI deployment has happened. For permanent changes such as hardware upgrade, recalibration of the model is necessary during maintenance time. If a new application is introduced, the model can be extended.

2.3 Optimization

The following optimization will be invoked when the total number of active desktop VMs N_{VM} is changed.

2.3.1 Optimal Algorithm

Cost function:

$$\mathrm{P}_{VDI}=\sum_{i=1}^{N_{host}} S_i\left(\alpha_i f_i^{\beta_i}+\mathrm{M}_i R_i+\mathrm{L}_i\right) \tag{2}$$

subject to:

$$\begin{gathered} Q_i\left(f_i, R_i\right) \leq QoS \\ f_{i,\min} \leq f_i \leq f_{i,\max} \\ \sum_{i=1}^{N_{host}} R_i = N_{VM} \end{gathered} \tag{3}$$

where $\alpha_i, \beta_i, \mathrm{M}_i, \mathrm{L}_i$ are server-specific parameters. Those parameters can be estimated using high dimensional fitting. The detail of power modeling is similar to the performance modeling and is omitted.

In a VDI deployment, minimization of energy consumption is equivalent to minimization of power consumption. Thus, the cost function to be minimized is the power consumption. Since servers consume a majority part of the total power consumption, we only minimize the power consumption of servers and the power consumption of other components is constant.

The above optimization formulation cannot be solved using existing solvers directly. We present high-level steps of tranforming the formulation to MINLP (Mixed-Integer Nonlinear Programming). For the cost function (2), the relationship between S_i and R_i can be established using a signum function. $S_i = sgn\left(R_i\right)$. MINLP requires the cost function to be a continuous function while a signum function is not continuous at 0. To meet the requirement of MINLP, the signum function can be approximated using a special continuous function such as tanh. A MINLP solver will determine R_i and f_i to minimize the cost function (2). Several constraints on R_i and f_i exist. The CPU frequency cannot be adjusted arbitrarily and has to be within a range. VDI administrators may assign a VM to a physical server in a static way. A VM must be assigned to only one physical server and all VMs must be assigned. The details of the mathematical transformation is not presented here.

2.3.2 Scalable Algorithm

A key observation is that VM consolidation and idle physical server shutdown can lead to significantly more energy savings than throttling DVFS levels. Idle power consumption of a physical server accounts for more than 70% of total power consumption [4]. Thus, we design a two-step heuristic to obtain a near optimal solution to the optimization problem formulated in 2.3.1. In the first step, we minimize idle server energy by consolidating VMs and run each active server at the highest CPU DVFS level. In the second step, we further throttle DVFS levels.

Algorithm 1 Heuristic for large-scale VDI deployments

begin

1: The 1st step:
2: Calculate the number of desktop VMs per server according to the performance model as Eqn (1), and denote the number by R_0.
3: Turn on $\left\lfloor \frac{N_{VM}}{R_0} \right\rfloor$ physical servers which runs at the highest DVFS level;
4: **if** $Q\left(f_{\max}, R_0\right) < QoS$ **then**
5: The 2nd step: Invoke the optimization algorithm (4)
6: **end if**
7: Actuators: shut-down idle physical servers, enforce f_i.

end

The detailed huristic for a large-scale VDI deployment consisting of homogeneous servers is shown in Algorithm 1. The 2nd step of Algorithm 1 solves the following optimization problem. The optimization is a constrained nonlinear multivariable which can be solved directly using fmincon in Matlab. Although no polynomial solver exists for fmincon, the optimization is conducted on a single physical server basis rather than on a cluster basis . Thus, the the huristic reduces the algorithm complexity significantly and obtains a near optimal solution as the MINLP solver in Section 2.3.1.

$$\mathrm{P}_i = \alpha_i f_i^{\beta_i}+\mathrm{M}_i R_i+\mathrm{L}_i \tag{4}$$

subject to:

$$\begin{gathered} f_{i,\min} \leq f_i \leq f_{i,\max} \\ Q_i\left(f_i, R_i\right) \leq QoS \end{gathered} \tag{5}$$

Where R_i is determined in the first step of Algorithm 1.

3 Evaluation Results

3.1 Implementation

In this section, we introduce our physical test bed, as well as the implementation details of each component.

Our test bed consists of two physical servers. Host1 has an AMD Opteron 6128 12-core CPU 1.9GHz with

Table 1: System Configuration

Server Cluster:	2 AMD servers	vSphere
Fabric:	1 Gb Ethernet	vCenter Enterprise
Shared Storage:	Openfiler	View Planner

16Gb main memory. Host2 has two AMD Opteron 254 dual-core CPU 2.8GHz with 4Gb main memory. It is connected to a shared storage Openfiler by Ethernet. The detailed hardware configuration is presented in Table 1. The host1 processor supports five DVFS levels: 1.9GHz, 1.5GHz, 1.3GHz, 1GHz, and 800MHz. The virtual machine operating system is Windows 7. We run View Planner V2.0 [3] to emulate VDI user operations, which run all the typical applications in Section 2.2. The length of each run is approximately 4 hours.

CPU Frequency Control : we use Intel's Enhanced Intel SpeedStep Technology to enforce the new frequency. To change the CPU frequency, VMWare ESXi contains a command line tool to determine the current frequency levels, frequency islands information, and modify frequencies within allowable ranges. Some advanced servers come with power management policy built in to the BIOS to manipulate CPU frequencies. To avoid conflicts, BIOS power management policies needs to be disabled. The average overhead (i.e., transition latency) for frequency change is approximately 100 μ. The CPU frequency calculated by the optimization algorithms is continuous and physical CPUs only support discrete number of frequency levels. A delta sigma converter is implemented to approximate a continuous frequency using discrete CPU frequencies.

Performance Measurement: latencies of operations on the data center side can be measured using CPU performance counters to obtain high-resolution timing information. By default, virtual machines cannot access performance counters. For newer releases of vSphere, virtual machines can have access by adding a flag to the vmx configuration files. We develop a watermarking technique to accurately measure the latency of the image transmission from the data center side to the client side. An encoded time stamp is placed in a fixed region of the image. When a thin client receives the frame, it reads and decodes the fixed region. The thin client and data center must be synchronized using NTP (network time protocol) with a common time source. More details can be found in [16].

VM migration: VMware vMotion can migrate running a virtual machine from one host to another. Even infrequent migrations of large VMs between hosts connected by a slow link may introduce significant overhead. By leveraging multiple NICs and efficient memory propagation, the overhead can be significantly reduced. Moreover, we can add host affinity rules to the optimization problem formulated in Section 2.3 to avoid costly VM migrations.

Power Measurement: The power consumption of the server cluster is measured with a WattsUp Pro power meter [6] by plugging servers into the power meter and then connecting it to a standard 120-volt AC wall outlet. The WattsUp power meter has an accuracy of 1.5% of the measured value and samples power data every second. Its internal memory and can store 18 hours of power data. The USB port of the power meter is connected to a desktop and a logger tool running on the desktop configures the meter and reads power data.

3.2 Baseline

Our baseline is state-of-the-art power management in a virtualized environment [1][2]. It controls the utilization of CPU and main memory within a fixed range by powering on and off hosts and migrating VMs but an end-to-end performance metric for VDI deployments is not taken into consideration. Although it works well for CPU and main memory intensive workloads, it could be conservative in term of energy saving for a VDI deployment in addition to not providing an end-to-end performance guarantee. The first reason is that a high utilization of vCPU and guest memory may not necessarily lead to low end-to-end performance. Another reason is the default maximum utilization is around 70% and a fixed safe margin of 30% exists. In contrast, the proposed solution eliminates such a margin by solving an optimization problem. At last, the proposed solution can further extend energy efficiency of the baseline by throttling CPU DVFS levels.

3.3 Experimental results

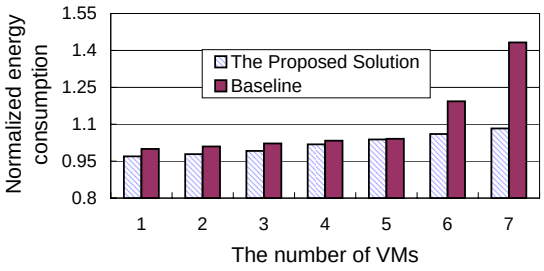
The purpose of this experiment is to compare the proposed solution against the baseline both in terms of energy efficiency and performance by varying the number of VMs (virtual machines) hosted by the hardware test bed in Section 3.1. This experiment demonstrates the main use case of the proposed solution in a production environment. The maximum number of VMs is determined by available free physical main memory within the cluster and represents no physical memory overcommitment scenario. During the experiment, we configure View Planner with three iterations which present a typical length of half a workday. Power consumption is measured at an interval of one second during the run and energy consumption is calculated as an integral of power with respect to time.

Figure 3(a) shows the measured energy consumption. When VDI workload is light and the number of virtual

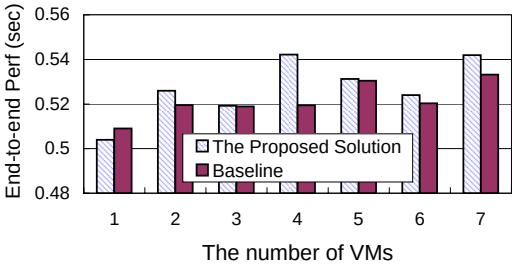
machines is from 1 to 5, the maximum energy saving is 3%. The modest saving is because when the number of VM is small, both the proposed solution and baseline utilize only one host and shut down the idle one. The proposed solution throttles DVFS levels aggressively to the lowest level and the baseline configures the highest frequency in a static way. Although the frequencies are very different, existing servers are non-power proportional and the power dynamic range of DVFS is limited. When the VDI workload is high and the number of virtual machines is larger than 5, the proposed solution outperforms the baseline by 11% and 24%. During a high workload, one host will accommodate 5 virtual machines and other virtual machines are placed on another host. The baseline consolidates based on utilization of vCPU and guest memory and will not consolidate all virtual machines until the utilization is below a certain threshold. When multiple virtual machines are powered on, utilization of vCPU and guest memory is very high due to the boot storm. After the boot storm, the utilization does not decrease quickly, and is calculated according to a weighting algorithm to avoid frequently shutting down and turning on hosts. Thus, the baseline will power on two hosts and consolidate VMs based on slowly-changing utilization. In contrast, based on the end-to-end performance metric, the proposed solution consolidates VMs into a single host and shuts down the idle host. Figure 3(b) shows the end-to-end performance. Currently the standard requirement for VDI QoS is 1.5s. Both the proposed solution and baseline satisfy the requirement. Compared to the baseline, the proposed solution achieves significant energy savings while introducing a minimum performance penalty.

4 Conclusions

Virtual desktop infrastructure is a promising virtualization technology to reduce enterprise IT expense. However, existing work cannot address the energy management issue in the context of VDI. In this paper, we first derived an explicit VDI performance model using surface fitting, then propose an optimization to aggressively reduce system energy consumption while guaranteeing performance by utilizing the performance model. Furthermore, a two-step heuristic is proposed to manage large-scale VDI deployments. Empirical results on a hardware test bed show that for high consolidation ratio scenarios the proposed solution achieves 11% - 24% better energy efficiency than a baseline widely adopted in production. It also ensures that user specified end-to-end performance requirements are met.



(a) Comparison of normalized energy consumption



(b) Comparison of end-to-end performance

Figure 3: Comparison of energy savings and performance of the proposed solution and the baseline.

References

[1] Dynamic Optimization and Power Optimization in System Center 2012 . technet.microsoft.com/en-us/library/gg675109.aspx.

[2] VMware Distributed Power Management Concepts and Use. www.vmware.com/files/pdf/Distributed-Power-Management-vSphere.pdf.

[3] VMware View Planner Installation and User Guide. communities.vmware.com/docs/DOC-15578.

[4] BARROSO, L. A., AND HOLZLE, U. The Case for Energy-Proportional Computing. *IEEE Computer* (2007).

[5] EBERLE, W., BOUGARD, B., POLLIN, S., AND CATTHOOR, F. From myth to methodology: cross-layer design for energy-efficient wireless communication. In *DAC* (2005).

[6] ELECTRONIC EDUCATIONAL DEVICES INC. Watts up pro power meter. http://www.wattsupmeters.com.

[7] FU, X., WANG, X., AND LEFURGY, C. How much power oversubscription is safe and allowed in data centers? In *ICAC* (2011).

[8] HUANG, H., QUAN, G., FAN, J., AND QIU, M. Throughput maximization for periodic real-time systems under the maximal temperature constraint. In *DAC* (2011).

[9] KANDLUR, D. D., AND KELLER, T. W. Green data centers and hot chips. In *DAC* (2009).

[10] KUANG, J., BHUYAN, L., AND KLEFSTAD, R. Traffic-aware power optimization for network applications on multicore servers. In *DAC* (2012).

[11] LIM, H., KANSAL, A., AND LIU, J. Power budgeting for virtualized data centers. In *USENIX* (2011).

[12] LIU, S., WU, Q., AND QIU, Q. An adaptive scheduling and voltage/frequency selection algorithm for real-time energy harvesting systems. In *DAC* (2009).

[13] MA, K., LI, X., CHEN, M., AND WANG, X. Scalable power control for many-core architectures running multi-threaded applications. In *ISCA* (2011).

[14] MARTINEZ, J., AND IPEK, E. Dynamic multicore resource management: A machine learning approach. In *Micro* (2009).

[15] REICH, J., GORACZKO, M., KANSAL, A., AND PADHYE, J. Sleepless in seattle no longer. In *USENIX* (2010).

[16] SPRACKLEN, L., AGRAWAL, B., BIDARKAR, R., AND SIVARAMAN, H. Comprehensive User Experience Monitoring. *VMware Technical Journal* (2012).

[17] WANG, X., FU, X., LIU, X., AND GU, Z. Power-aware CPU utilization control for distributed real-time systems. In *RTAS* (2009).