vPnP: Automated Coordination of Power and Performance in Virtualized Datacenters

Jiayu Gong and Cheng-Zhong Xu
Department of Electrical and Computer Engineering
Wayne State University
Detroit, MI 48202
{jygong, czxu}@wayne.edu

Abstract

Both power and performance are important issues in today's datacenters. It is hard to achieve optimization in both aspects on shared infrastructures due to system dynamics. Previous work mostly emphasized on either aspect or relied on models that were trained off-line for specific workload. In this paper, we present vPnP, a feedback control-based coordination system that provides guarantees on a service level agreement with respect to performance and a power budget in virtualized environments. This system can adapt gracefully to workload change. It consists of two self-tuning model predictors and a utility function optimizer. The predictors correlate system resource allocation to power and performance, respectively. The optimizer finds the optimal solution for a tradeoff between power and performance. Experimental results using TPC-W benchmark show vPnP can achieve different levels of tradeoff in a more flexible way than an existing two-layer feedback control approach. More importantly, vPnP shows its robustness over a variety of workloads. It reduces performance relative deviation by 17% compared with the two-layer feedback controller.

1 Introduction

Virtualization technology has brought in a number of benefits to datacenters, such as performance isolation, server consolidation, and system manageability. A variety of applications, from parallel computing to cloud services, can be hosted on virtualized datacenters. The power consumption due to the increasing requirement of application performance and the increasing popularity of high-density servers poses a key challenge in the design and management of the datacenters. To meet the challenge, researchers have developed a wide array of technologies on power management, including dynamic voltage and frequency scaling (DVFS) and low-power sleep states on processors, low-power DRAM states on memory. In addition, server virtualization facilitates server consolidation, leading to more power savings.

From datacenter administrators' perspective, a primary concern is the service-level agreement (SLA) in performance. The impact on system performance due to power management varies greatly depending on the characteristics of time-varying workloads. It becomes even more complicated due to resource sharing in virtualized environments.

In analogy to SLA in performance, power budget sets a power consumption cap of the system. Enforcing power budgets can be physical or contractual [4]. Physically, limiting power consumption in a datacenter can help address thermal conditions or temporary reductions in cooling or power delivery capacity [11]. One possible approach to power budgeting is to use the power management capabilities of a processor based on hardware controllers [9, 16]. However, this class of approaches may not be applicable in virtualized environments since the platform is shared by all resident virtual machines.

There are a number of work devoted to power management in virtualized servers. Most of them aim to energy consumption limits or requirement of provisioning power budget in an SLA-aware manner but without SLA guarantees [18, 13, 15, 12]. In these work, power (or energy) is the first-class control target in adjustment of hardware power states. There are other performance-oriented work that aims to meet performance SLAs and meanwhile reduce power consumption in a best-effort manner [21]. The work in [3, 20] deal with both power and performance in a coordinated way, focusing on virtual machine placement [3] or relying on off-line system identification [20].

Control approaches have been successfully applied to power management in virtualized environments. However, they require off-line system identification in controller design [21, 20], or need the expertise of control

parameter tuning [12]. There are also approaches designed for specific workloads [8]. The controllers designed in such manners may not be able to adapt gracefully to situations with abrupt workload change though they can achieve control accuracy and system stability within a range theoretically. As a result, existing solutions hardly provide guarantees on an SLA in performance and a power budget without off-line modeling for a variety of applications in virtualized environments.

In this paper, we propose vPnP, a feedback control-based coordination system that can achieve the goal with respect to both application-level performance and underlying physical host power consumption in a virtualized environment. This framework consists of two monitors and a coordinator. The monitors provide real-time measurement of power and performance, respectively. The coordinator, including two model predictors and a utility function optimizer, produces policies for execution. The power prediction model should be the first that can predict power consumption accurately without need for off-line training or peeking into the system for additional performance metrics. The framework employs user-specified parameterized utility functions representing different levels of tradeoff between power and performance. The optimal solution to the utility function, achieved by the optimizer, directs the resource allocation amongst virtual machines. We have implemented vPnP in a Xen-based infrastructure with different coordination policies. The experimental results using TPC-W benchmark demonstrate the adaptivity of the performance and power predictors. The results also show that vPnP can achieve a user-specified performance and power guarantee in a flexible way. We further show vPnP is more robust than the two-layer feedback controller in [20] over a variety of workloads. It can reduce up to 17% performance relative deviation.

The reminder of this paper is organized as follows. Related work is discussed in Section 2. In Section 3, we present the system design of vPnP. Section 4 describes the implementation of vPnP. Evaluation results are presented in Section 5. Finally, we conclude the paper in Section 6.

2 Related Work

Towards managing power and performance for enterprise servers, control theory is widely applied in recent years. According to the number of control inputs and outputs, the work using control theory can divided into two major categories: Single-Input-Single-Output (SISO) control and Multi-Input-Multi-Output (MIMO) control.

Lefurgy et al. [9] designed a Proportional (P) controller to cap the peak power of a server. Wu et al. [23] managed power by controlling the synchronizing queues in multi-clock-domain processors. Zhang et al. [24] adjusted the resource demands of virtual machines based on resource availability. These work concerned themselves with a single control output employing control over a single resource. SISO controller is mostly used in this kind of work.

MIMO control can be employed when there are multiple control outputs in power management for enterprise servers. Wang et al. [19] developed a MIMO control algorithm for cluster-level power control in a non-virtualized environment. However, this work requires pre-defined power models and provides no performance guarantee. Kusic et al. [8] presented a power and performance management strategy using lookahead control. This work was designed in a workload-specific way at cluster-level requiring off-line models without explicit power and performance guarantee. In addition, Kephart et al. [7] have proposed a coordinated management scheme to achieve tradeoffs between power and performance by optimizing a utility function for a non-virtualized server. Our work belongs to this class of approaches. In contrast to the above work, our work coordinates power and performance for a virtualized server in a non-workload-specific manner.

Multilayer control can be applied to deal with the similar problem which needs MIMO control. Representative work includes [21, 12, 20], which employ two-layer feedback controllers. In [21], one control loop controls the CPU resource to each virtual machine to guarantee performance while the other one controls CPU frequency for power efficiency. In [12], one loop limits the power consumption and the other one bids resource based on shadow price for each virtual machine. However, the work in [21, 12] are either performance-oriented or power-oriented without explicit coordination of power and performance. The work in [20] shares a similar objective to ours. The power consumption is constrained by scaling CPU frequency and the performance guarantee is achieved by allocating CPU resources among virtual machines. The design of the controllers elaborates off-line system identification. The controllers designed based on a static model for one workload may not adapt gracefully to workload change though the accuracy and stability can be assured within a range. In contrast, our work can adapt to a variety of workloads by online modeling with negligible runtime overhead.

Adaptive control can cope with the dynamic of a system by modifying the control law. A self-tuning admission controller was designed for a 3-tier web sites in [6]. In [22], an adaptive fuzzy controller was proposed to guarantee client-perceived end-to-end QoS. To manage resource in virtualized environments, Padala et al. [14] proposed an adaptive estimator to capture the relationship between allocated system resource and performance. In [5], the Kalman filter was integrated into feedback controllers to dynamically allocate CPU re-

sources to virtual machines. Rao et al. [17] proposed a reinforcement learning based approach for virtual machine configuration which is adaptive to heterogeneous virtual machines. In the area of power management for virtualized servers, the power model is complicated by multiple hosted heterogeneous virtual machines consuming different amount of resources. Our proposed power prediction model should be the first adaptive one, to our knowledge.

3 System Architecture

In this section, we first discuss the mechanism to regulate the performance and power of a server. Then we present the design of our vPnP system with a detailed description of the key components.

3.1 Control Power and Performance with VCPU Caps

Modern server processors often support multiple classes of execution states for the purpose of power management. These states include the frequency and voltage operating points (P-states), throttling states (T-states) in active mode, and sleeping states (C-states) in idle time. P-states are well documented and can significantly affect active power consumption. But it has only very limited speed stages. In multi-core processors, it is not flexible to manipulate the P-states due to the dependencies of the cores residing on the same die. Things become even more challenging in virtualized environments. Since multiple virtual machines may share a single core, tuning P-states of a core would threaten desired performance isolation properties. T-states can further throttle down a CPU by inserting stop clock signals and thus omitting duty cycles. However, T-states are not always well documented and may need to modify the clock modulation register. C-states can be utilized when the CPU is idle. But it incurs relatively large switch overhead and might not be effective when the system is in a low utilization. In this paper, we regulate the power consumption by re-allocating CPU resources to virtual machines (VMs).

We assume a hypervisor scheduler to limit processing time to guest VMs. Non-work-conserving scheduling is employed. Xen provides the capability to cap the CPU time of VCPUs allocated to a VM. By capping VCPUs, the utilization of underlying physical processors can be constrained thereby regulate power consumption. In this paper, we use VCPU capping as the actuator for power management.

3.2 Design of vPnP

Figure 1 shows the architecture of the vPnP power and performance coordination system. A physical server

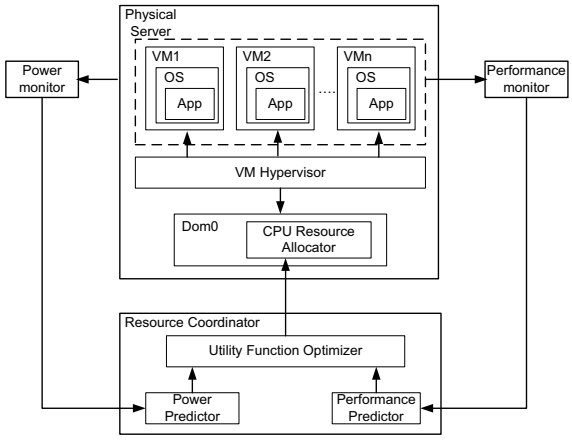


Figure 1. System architecture of vPnP.

can host multiple VMs on which different applications are running. A power monitor is employed to measure the power consumption of the whole physical server. The real-time performance of each VM can be collected by a separate performance monitor. A number of performance data, such as mean response time, CPU utilization and throughput, can be collected by this performance monitor. In our system, we currently only measure performance in terms of throughput. All the power and performance data are reported to the resource coordinator synchronously. The power and performance predictors can predict the future power consumption and performance, respectively. A multi-criteria utility function is defined to meet the power and performance requirements. The ultimate goal is to find a utility-optimizing policy which will be conveyed to the hypervisor to regulate the VCPU cap for each VM.

3.2.1 Power and Performance Predictor

Since the system behavior could be non-linear, time-varying and workload-dependent, it is not necessarily accurate to represent the relationship between power (or performance) and VCPU cap by a static linear function. One feasible approach to capturing the system behavior is to use a linear model to approximate locally on the neighborhoods (operating range) of an operating point. In order to let this linear model adapt to different operating ranges, the model parameters must be updated dynamically on the fly.

In [14], the authors experimented with such a dynamic model, Autoregressive-moving-average (ARMA), to approximate the quantitative relationship between allocated resource and normalized performance. ARMA model is a combination of an autoregressive (AR) part and a moving average (MA) part, referred to as the ARMA(l, m) model, where l and m are the orders, indicating the numbers of previous values considered, of the AR and MA part, respectively. In our work, we

extend this model to predict power consumption of a physical server. Like the model in [14], we use an ARMA(2,2) model to characterize the relationship between the power consumption of the physical server and the VCPU caps of all hosted VMs in a control interval.

Let k denote the k^{th} control interval, $p(k)$ be the average power consumption of the physical server, $C(k)$ be the vector of VCPU caps of all hosted VMs. Assume that $g_i(k)$ and vector $H_i^T(k)$ capture the correlation between power and VCPU caps, respectively. It follows that:

$$p(k) = \sum_{i=1}^{2} g_i(k)p(k-i) + \sum_{i=0}^{1} H_i^T(k)C(k-i). \quad (1)$$

Let $r(k)$ and $c(k)$ represent the throughput and VCPU cap of a VM, respectively, $a_i(k)$ and $b_i(k)$ be time-varying model parameters to capture the relationship between throughput and VCPU cap. A self-adaptive ARMA(2,2) model which is similar to that in [14] represents the relationship between throughput and VCPU cap for a VM:

$$r(k) = \sum_{i=1}^{2} a_i(k)r(k-i) + \sum_{i=0}^{1} b_i(k)c(k-i). \quad (2)$$

Notice the model parameters $g_i(k)$, $H_i^T(k)$ in (1), and $a_i(k)$, $b_i(k)$ in (2) vary with time. These parameters are updated every control period using the data collected in a given range of past intervals, say M intervals, similar to the sliding window size. These data include the measured performance, power, and CPU resource allocated to each VM in the past M intervals. Least-square regression can be used to obtain these model parameters. The assumption to use ARMA model is that significant workload disturbance, which may lead to tremendous change in the model parameters estimation, seldom occurs. In this case, the convergence of the model parameters can be achieved. However, the dynamics of workloads retard this process. If the workload changes abruptly, the model parameters may not be estimated accurately during a period. It may take as long as M intervals to get the accurate model parameters estimation for the system with new workload since all the data collected for the past M intervals can be replaced. It implies a smaller M is apt to adapt to the change in the system responsively but it can easily be affected by the infrequent disturbance.

3.2.2 Utility Function Optimization

Suppose there are n VMs hosted in a physical machine. Let $\hat{r_i}$ $(1 \leq i \leq n)$ denote the SLA of performance for the i^{th} VM. Let p_s denote the power budget for the physical server. We define two step functions to quantify the SLA of performance and the power consumption, respectively, in the following:

$$\Theta_i(r_i(k)) = \begin{cases} 1 & r_i(k) \geq \hat{r_i}; \\ r_i(k)/\hat{r_i} & \text{otherwise.} \end{cases} \quad (3)$$

$$\Gamma(p(k)) = \begin{cases} 1 & p(k) \leq p_s; \\ p(k)/p_s & \text{otherwise.} \end{cases} \quad (4)$$

In general, a utility function should be defined in such a manner that optimizing the utility function is to determine the resource allocation to each VM to meet performance SLAs and power budget. Consider the violations of SLA in performance and a power budget as penalties, we have:

$$U_1 = \alpha \sum_{i=1}^{n} (1 - \Theta_i(r_i(k)))^2 + (1-\alpha)(\Gamma(p(k)) - 1)^2, \quad (5)$$

where the parameter α represents the weight of SLAs of power and performance for different levels of tradeoff. The goal is to minimize the utility function (5) by finding a column vector $C(k)$. Intuitively, this utility function can achieve its optimal value, 0, by meeting both performance SLA and power budget. In case this minimum value cannot be achieved, there is no solution to guarantee both power and performance. Instead, minimizing this utility function represents the tradeoff of power and performance.

Consider the SLA in performance as a gain and power consumption as a cost, we have:

$$U_2 = \alpha \sum_{i=1}^{n} (\Theta_i(r_i(k)))^2 - (1-\alpha)(\Gamma(p(k)))^2. \quad (6)$$

Its goal is to maximize the value U_2. Besides similar observation to function (5), we can see these two utility functions achieve the optimization goal in different manners when there is no solution to guarantee both power and performance while the power factor is dominant. The utility function (5) would be minimized by balancing the performance SLA for each VM in this case, while (6) would unbalance the performance SLAs. This can easily be proved by using Cauchy-Schwarz inequality.

A utility function optimizer is designed to determine the VCPU caps to optimize the defined utility function. Taking (5) for example, since $r_i(k)$ $(1 \leq i \leq n)$ and $p(k)$ can all be represented by linear functions using $c_i(k)$ $(1 \leq i \leq n)$, the utility function (5) can ultimately be represented by a quadratic function $Q(c_1(k), c_2(k), \ldots, c_n(k))$. Thus the VCPU caps can be found by solving the following problem:

$$\text{minimize } Q(c_1(k), c_2(k), \ldots, c_n(k)) \quad (7)$$

$$\text{subject to } c_{low} \leq c_i(k) \leq c_{up}, 1 \leq i \leq n. \quad (8)$$

where c_{low} and c_{up} represent the minimal and maximal values of VCPU caps, respectively. Constraint (8) ensures the allocated VCPU caps will not be out of the

range. In practice, we set the c_{up} to 100% and c_{low} to be 10% to avoid starvation. The objective function is quadratic and convex. An ellipsoid method can solve this problem in polynomial time. In practice, we can use an off-the-shelf quadratic programming solver to compute the solution. Similar approach can be applied to the utility function (6). In this case, the objective function is quadratic but not convex. Thus this optimization problem is NP-hard.

Notice here we treat all VMs with the same priority. In order to differentiate the performance priority, we can add weights to the performance gained in each VM.

4 Implementation Issues

Implementation details of each component in vPnP are presented as follows:

Power monitor: We measured the power consumption of a physical server by the use of a WattsUp Pro [2] power meter. This power meter has an accuracy of $\pm 1.5\%$ of the measured RMS power with a sampling rate of 1Hz. The measured power are sent to the resource coordinator continuously.

Performance monitor: vPnP runs a small daemon program associated with each application to record the time stamps of incoming requests. When a request is finished, another time stamp can be obtained. The difference of time stamps is response time. The number of requests finished during a unit interval is throughput. In our work, we only measure performance in terms of throughput. The throughput is reported to the resource coordinator every control interval which is set to be 30 seconds for a tradeoff of system response and possible transient noise.

Resource coordinator: The resource coordinator consists of three parts: performance predictor, power predictor and utility function optimizer. The performance and power predictors update the control parameters at the end of every control period using least-square regression. The data in the past 20 intervals would be used in regression. The overhead for both regression was 15ms in average in a Xeon5450 quadcore server. The utility function optimizer employs a quadratic programming solver to calculate the VCPU caps for the next interval. The cost of this calculation overhead was 10ms. The solution will be sent to the CPU resource allocator.

CPU resource allocator: Xen uses a Credit Scheduler [1], Xen's proportional share scheduler, to allocate CPU resource. We use the VCPU cap to limit the CPU time that can be allocate to a VM. Though in Credit Scheduler, the weight can also specify the CPU resource allocation to each VM, we here only use VCPU cap to implement this allocation by setting same weights to all VMs. A VCPU cap of 0 means there is no upper cap.

5 Evaluation

In this section, we present the experimental results of vPnP. We compare vPnP with an existing two-layer feedback controller, Co-Con [20], from the perspective of flexibility and robustness of coordinated control of power and performance.

5.1 Experimental Methodology

vPnP was evaluated in an experimental server cluster consisting of five physical machines connected by a gigabit network. Two physical servers were used to host VMs. A third storage server was used for Network File System (NFS) to host all VM images so that live migration might be integrated to the system in the future. Each server was configured with two quadcore processors, Intel Xeon5450, with a total of eight cores, and 8GB RAM. All servers ran CentOS Linux 5.0 with kernel 2.6.18 and Xen 3.4.

We selected TPC-W [10] as the host application. TPC-W is an E-Commerce benchmark that models after an online book store. It provides workloads with different mixes. There are three predefined workload mixes: browsing (B), shopping (S) and ordering (O), which require different amount of resource. TPC-W consists of two tiers, application (APP) tier in the front and database (DB) in the back-end. Each tier was hosted in a VM with 2 VCPUs and 2GB memory. We ran two TPC-W applications and consolidated the two DB VMs on a `DB_HOST` and the two APP VMs on an `APPS_HOST`. We only evaluated vPnP on the `DB_HOST` since the DB tier tend to be a bottleneck of this application.

We ran the resource coordinator in dom0 of `APPS_HOST` in order to alleviate the computational stress at `DB_HOST` with the consideration of consolidation as well. The CPU resource allocator was deployed in dom0 of `DB_HOST`.

We used the other two servers as the client-side workload generators. The default concurrency level was set to 400, 400, and 1000, for browsing, shopping, and ordering, respectively. The default throughput targets for these workloads with default concurrency levels were 1200, 1500, and 3000 per interval, respectively. The client machines should be powerful enough to create resource stress on any of the VMs.

We conducted three experiments on the platform.

First, we evaluated the accuracy of the self-adaptive power and performance predictors. Second, we evaluated the agility of vPnP to achieve different levels of tradeoff between performance and power. The tradeoff policy could be power-preferred, performance-preferred or other user-defined criterion. Third, we investigated the robustness of vPnP. vPnP is designed in a workload-independent manner. It can perform well for different applications or in the scenarios with dynamics.

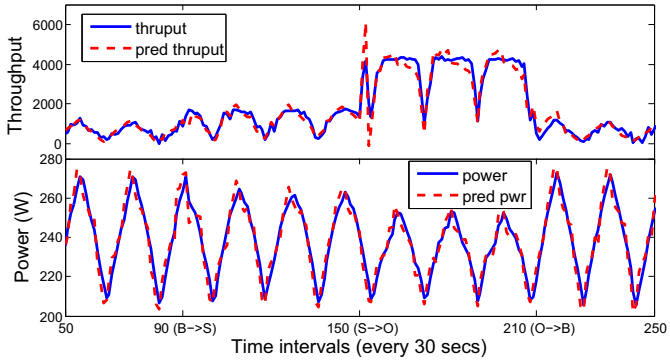


Figure 2. Performance and power prediction.

We adapted an existing two-layer controller from CoCon [20] as the baseline. The controllers were designed using the off-line data during a run of TPC-W with browsing mix following the methods in [20].

Each run of the workload lasted 200 intervals by default. Since vPnP used the data of the past 20 intervals for prediction, we started it at the 25^{th} interval. The results from the 30^{th} interval to the 100^{th} interval will be presented and the results thereafter will be omitted due to the similarity.

5.2 Experimental Results

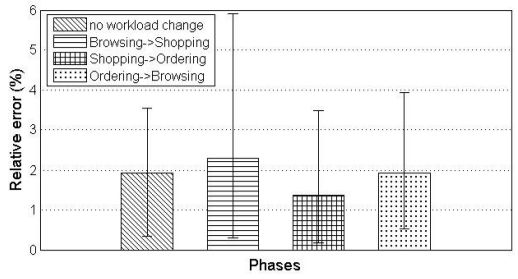
5.2.1 Prediction Accuracy

The online performance and power predictors can adapt the model parameters to the change of the system states. The accuracy of these two predictors affects the CPU resource allocation.

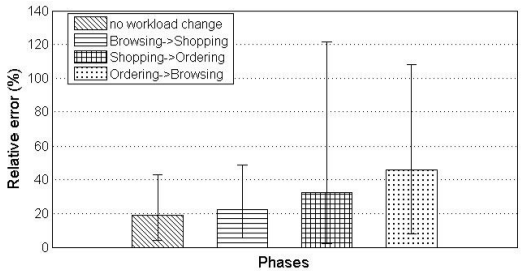
We conducted experiments to show the adaptivity of these two predictors by running a sequence of different workloads. We allocated the VCPU cap to the VM from 100% to 10% in a step length of 10% every interval back and forth. Starting from the 90^{th} interval, we changed the workload type every 60 intervals (30 minutes) in the order of: browsing→ shopping→ ordering→ browsing. Figure 2 plots the results from the 50^{th} interval to the 250^{th} interval when all workload changes finish.

To quantify the prediction accuracy of these predictors, we used relative error, defined as $|y(k)-\tilde{y}(k)|/\tilde{y}(k)$, where $y(k)$ and $\tilde{y}(k)$ denote predicted and measured power (or throughput) at k^{th} interval, respectively. We investigated the performance of predictors in different phases. To adapt to a new workload may take up to M intervals (data collected in past M intervals would be used for regression). So we investigated these M intervals as the adapting phase when workload changed. Figure 3 plots the means of relative errors as well as 95% confidence level confidence intervals for power and performance predictors.

The power predictor can predict the power consumption at all the time accurately, with mean relative error below 3%. The mean relative error for performance



(a) Power prediction accuracy



(b) Performance prediction accuracy

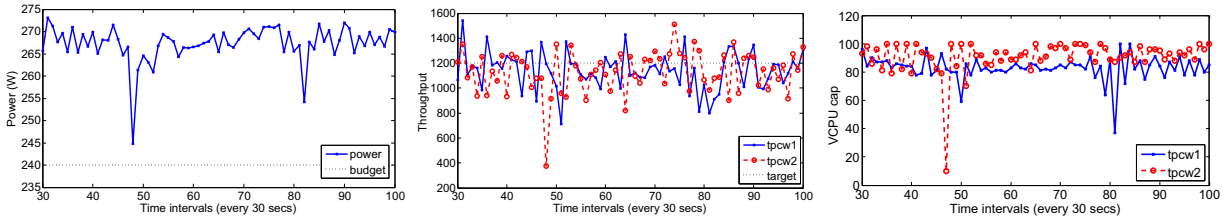
Figure 3. Prediction accuracy.

prediction may reach around 15% even if there was no workload change. This is because the power consumption only depends on current resource utilization and resource allocation. But the throughput will be affected by the process delay [22] and workload dynamics as well in addition to the control over resource allocation. During the adapting phase, the accuracy of performance prediction was degraded. If the workload change was not significant (for example, B→S), the performance predictor could adapt gracefully to workload change with narrow confidence intervals. In case of significant workload change, this predictor may not adapt well (for example, O→B).

5.2.2 Tradeoff of Performance and Power

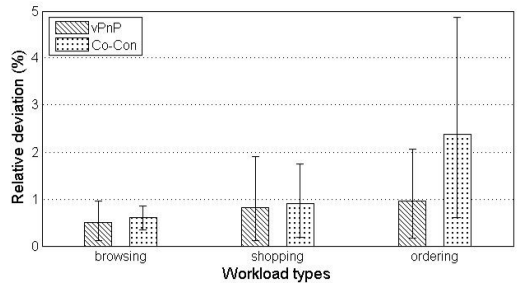
Using vPnP, different levels of tradeoff between power and performance can be achieved in a flexible way by tuning the weight in the utility function. Using the two utility functions defined in Section 3.2, in case both of power and performance SLAs cannot be met, the policy depends on the weights in the utility functions. A larger α represents the tendency to meet the performance SLA while a smaller α means satisfying power budget is more important. We defined two policies here for the purpose of evaluation: performance-preferred ($\alpha = 0.9$) and power-preferred ($\alpha = 0.1$). We conducted experiments using TPC-W browsing workload.

First, we applied the vPnP framework using the performance-preferred policy. Since power was not dominant, vPnP allocated large VCPU caps to meet performance SLA while power consumption was beyond budget. The oscillations of throughput were caused by both the disturbance of workload and the adjustment to the

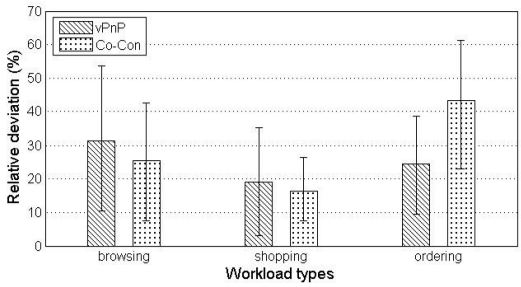


(a) Power consumption (b) Throughput (c) VCPU cap

Figure 4. Results of utility function U_1 with a performance-preferred policy ($\alpha = 0.9$).



(a) Power deviation



(b) Performance deviation

Figure 9. Performance of vPnP vs. Co-Con.

VCPU caps. The utility function U_2 performed in a similar way. For lack of space, we omitted this result.

Then the power-preferred policy was employed. Results are shown in Figure 5 and Figure 6. Both applications could not meet the performance SLA mostly but power consumption was close to the budget. These utility functions acted in different ways. The utility function U_2 would allocate enough VCPU cap to one VM while the utility function U_1 would allocate CPU resource in a relatively fair manner. For fairness, in the rest of experiments, we only used utility function U_1.

In contrast, the power control is always primary in Co-Con. It cannot make different levels of tradeoff between power and performance.

5.2.3 System Robustness

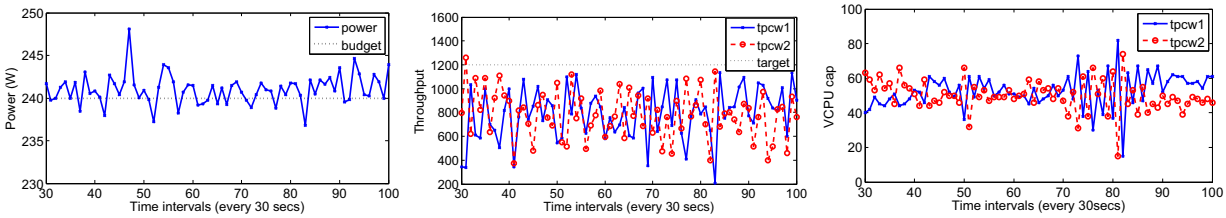
We investigated the robustness of vPnP by conducting experiments running different TPC-W workloads: browsing, shopping and ordering. Two identical applications were hosted for each run. Results from Co-Con were included as a baseline for comparison. For fairness in comparison, we used a strict power-preferred policy for vPnP by setting $\alpha = 0.01$.

Recall that Co-Con used a two-layer controller statically based on the collected data during a run of TPC-W browsing mix. When running shopping and ordering workloads, we found vPnP could limit the power consumption closer to the budget than Co-Con. The results are shown in Figure 7 and Figure 8. The static two-layer controller in Co-Con could not adapt its control parameters to the workload change thus it may not perform well when the workload changed. As vPnP doesn't rely on any off-line trained model, it can adapt to a large variety of workloads. Since workload disturbance and controller both affected the power and performance, the oscillations occurred.

To quantify the performance of Co-Con and vPnP, we defined relative deviation from a reference as the metric. The relative deviation for power is $|p(k)-p_s|/p_s$. Similarly, the relative deviation for throughput of one application is $|r(k) - r_s|/r_s$.

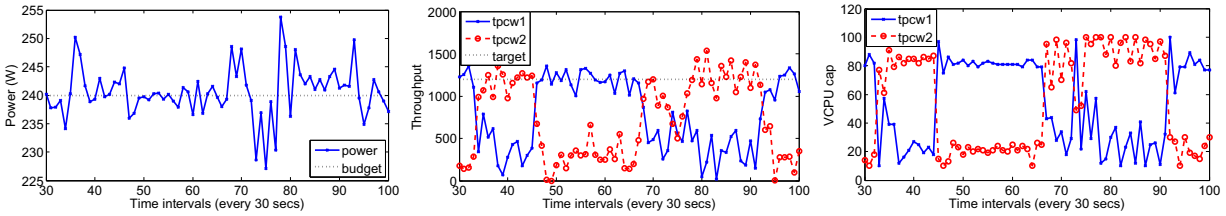
Figure 9 shows the means as well as the 95% confidence level confidence intervals for relative deviations for power and performance. Both vPnP and Co-Con could achieve very small relative deviation for power for all workloads. It implies they could provide power guarantee when the power budget was defined within a specific range (we will see later what will happen if power budget varies in a large range). For performance, Co-Con outperformed vPnP by around 5% when running browsing workload. The difference was marginal when running shopping workload. In this case, Co-Con slightly outperformed vPnP since the shopping workload is very similar to the browsing workload. Using ordering workload, we can see the performance relative deviation of vPnP was around 17% less than that of Co-Con. Overall, the performance relative deviation of vPnP could be limited to 32% with relatively small variation (less than 15%) over a variety of workloads. In contrast, the performance relative deviation of Co-Con may vary over a large range, from 16% to 43%. The results show good adaptivity of vPnP for a variety of workloads.

In practice, the power budget might change due to thermal condition or temporary reductions in cooling or power delivery capacity. We investigated how vPnP and Co-Con reacted to this power budget change. We assumed browsing workload in both with initial power



(a) Power consumption (b) Throughput (c) VCPU cap

Figure 5. Results of utility function U_1 with a power-preferred policy ($\alpha = 0.1$).



(a) Power consumption (b) Throughput (c) VCPU cap

Figure 6. Results of utility function U_2 with a power-preferred policy ($\alpha = 0.1$).

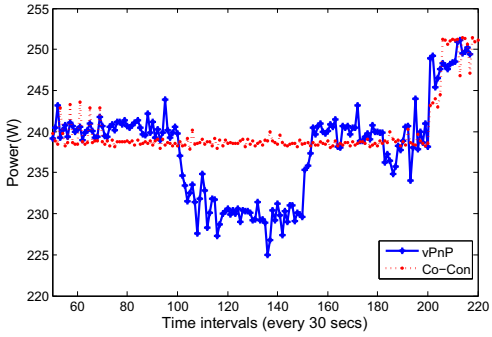


Figure 10. Reaction to power budget change.

budget 240W for `DB_HOST`. From the 100^{th} interval, the power budget changed every 50 intervals following this order: 240W→230W→240W→250W.

As shown in Figure 10, when power budget decreased to 230W, Co-Con failed to limit the power consumption to this budget. It is due to the limited stages of CPU frequency. Only 4 CPU frequencies are available for Intel Xeon5450 in our testbed. When the power budget was 240W, the frequency had already been set to the lowest. If the budget decreased more, Co-Con could not control power effectively since there was no lower frequency available. In contrast, vPnP could work in a large range of power budget using VCPU cap to regulate power, which provided a large difference in power consumption between the highest and the lowest cap.

To sum up, vPnP demonstrated a high flexibility than Co-Con. It outperformed Co-Con in terms of robustness over a variety of workloads.

6 Conclusion

In this paper, we present vPnP, a feedback control-based coordination system providing guarantees on an SLA in performance and a power budget in virtualized datacenters. The system consists of two online model predictors and a utility function optimizer. The predictors correlate CPU resource allocation to power and performance, respectively. The optimizer finds the solution that optimizes the utility function of power and performance.

We evaluated vPnP in a testbed using multi-tier benchmarks and compared it with the existing two-layer feedback controller for power and performance. Experimental results show the flexibility of vPnP to achieve different levels of tradeoff between power and performance, and the robustness over a variety of workloads in contrast to the two-layer feedback controller.

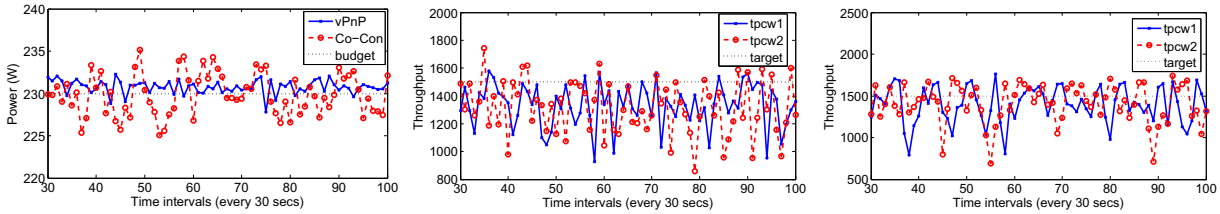
Process delay is expected to have a significant impact on performance prediction. This factor will be considered in the future.

Acknowledgement

We would like to thank the anonymous reviewers for their constructive comments and suggestions. This research was supported in part by U.S. NSF grants CNS-0702488, CNS-0708232, CNS-0914330.

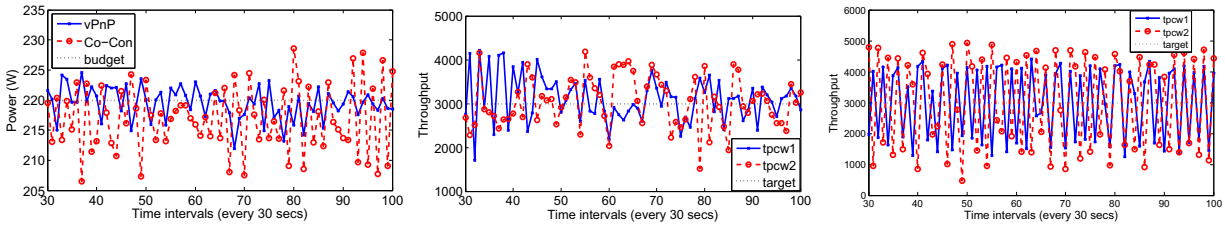
References

[1] "credit scheduler". `http://wiki.xensource.com/xenwiki/CreditScheduler`.



(a) Power consumption (b) Throughput under vPnP (c) Throughput under Co-Con

Figure 7. Comparison between vPnP and Co-Con running TCP-W shopping workload.



(a) Power consumption (b) Throughput under vPnP (c) Throughput under Co-Con

Figure 8. Comparison between vPnP and Co-Con running TCP-W ordering workload.

[2] Electronic educational devices inc., "watts up pro power meter". `http://www.wattsupmeters.com`.

[3] M. Cardosa, M. R. Korupolu, and A. Singh. Shares and utilities based power consolidation in virtualized server environments. In *IM'09*. IEEE Press, 2009.

[4] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *ISCA'07*. ACM, 2007.

[5] E. Kalyvianaki, T. Charalambous, and S. Hand. Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters. In *ICAC'09*. ACM, 2009.

[6] A. Kamra. Yaksha: A self-tuning controller for managing the performance of 3-tiered web sites. In *IWQoS'04*, 2004.

[7] J. O. Kephart, H. Chan, R. Das, D. W. Levine, G. Tesauro, F. Rawson, and C. Lefurgy. Coordinating multiple autonomic managers to achieve specified power-performance tradeoffs. In *ICAC'07*. IEEE Computer Society, 2007.

[8] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang. Power and performance management of virtualized computing environments via lookahead control. In *ICAC'08*. IEEE Computer Society, 2008.

[9] C. Lefurgy, X. Wang, and M. Ware. Server-level power control. In *ICAC'07*. IEEE Computer Society, 2007.

[10] D. A. Menasce. Tpc-w - a benchmark for e-commerce, 2002.

[11] J. Moore, J. Chase, P. Ranganathan, and R. Sharma. Making scheduling "cool": temperature-aware workload placement in data centers. In *ATC'05*. USENIX Association, 2005.

[12] R. Nathuji, P. England, P. Sharma, and A. Singh. Feedback driven qos-aware power budgeting for virtualized servers. In *FeBID'09*, 2009.

[13] R. Nathuji and K. Schwan. Vpm tokens: virtual machine-aware power budgeting in datacenters. In *HPDC'08*. ACM, 2008.

[14] P. Padala, K.-Y. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant. Automated control of multiple virtualized resources. In *EuroSys'09*. ACM, 2009.

[15] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu. No "power" struggles: coordinated multi-level power management for the data center. In *ASPLOS'08*. ACM, 2008.

[16] P. Ranganathan, P. Leech, D. Irwin, and J. Chase. Ensemble-level power management for dense blade servers. In *ISCA'06*. IEEE Computer Society, 2006.

[17] J. Rao, X. Bu, C.-Z. Xu, L. Wang, and G. Yin. Vconf: a reinforcement learning approach to virtual machines auto-configuration. In *ICAC'09*. ACM, 2009.

[18] J. Stoess, C. Lang, and F. Bellosa. Energy management for hypervisor-based virtual machines. In *ATC'07*. USENIX Association, 2007.

[19] X. Wang and M. Chen. Cluster-level feedback power control for performance optimization. In *HPCA'08*. IEEE Computer Society, 2008.

[20] X. Wang and Y. Wang. Co-con: Coordinated control of power and application performance for virtualized server clusters. In *IWQoS'09*, 2009.

[21] Y. Wang, X. Wang, M. Chen, and X. Zhu. Power-efficient response time guarantees for virtualized enterprise servers. In *RTSS'08*. IEEE Computer Society, 2008.

[22] J. Wei and C.-Z. Xu. eqos: Provisioning of client-perceived end-to-end qos guarantees in web servers. *IEEE Trans. Computers*, 2006.

[23] Q. Wu, P. Juang, M. Martonosi, L.-S. Peh, and D. W. Clark. Formal control techniques for power-performance management. *IEEE Micro*, 2005.

[24] Y. Zhang, A. Bestavros, M. Guirguis, I. Matta, and R. West. Friendly virtual machines - leveraging a feedback-control model for application adaptation. In *VEE'04*. ACM, 2004.