

Feedback Driven QoS-Aware Power Budgeting for Virtualized Servers

Ripal Nathuji, Paul England, Parag Sharma, and Abhishek Singh
Microsoft

Redmond, WA, USA

{ripal.nathuji, paul.english, parag.sharma, absingh}@microsoft.com

ABSTRACT

Technological advances including multicore processors and small form factor blade servers are enabling continued improvements in computing densities for modern datacenters. Large scale facilities leverage these densities to deploy web applications and cloud based services, but must simultaneously address the ensuing power and cooling management issues that impact datacenter sustainability and costs. Virtualization has emerged as an enabling underlying technology for meeting datacenter management needs, accelerating its deployment onto enterprise servers. In this paper, we consider how a key power management capability, server power budgeting, can be provided through the virtualization management layer. Our approach utilizes feedback controllers that are designed to address two goals. The first goal is to honor platform power budgets. This is achieved using a feedback controller that monitors server power consumption and determines a platform-level CPU allocation that conforms to a power budget. The second goal is to enable QoS-aware management of virtual machines under imposed power constraints. To achieve this, additional feedback controllers are distributed across guest virtual machines (VMs). These controllers react to “shadow prices” provided by the system resource manager with resource bids based upon the QoS requirements of the VM. Finally, the resource manager combines the information from the power budget controller and guest QoS controllers to determine VM allocations. An evaluation of our system using the Hyper-V virtualization platform highlights the feasibility and benefits of feedback based control for QoS-aware power budgeting.

1. INTRODUCTION

Modern datacenters are being provisioned with significant computational power based upon high performance multicore processors and dense server configurations. These environments lend themselves to a variety of applications, from supporting IT services to providing backend resources for cloud computing infrastructures [1, 13]. Unfortunately, the power and cooling overheads in these systems are becoming increasingly large. Therefore, it is clear that the cost effectiveness of large scale datacenters depends upon the continued development of management systems to address these fundamental facility issues [3, 5, 14].

Hardware vendors have begun to help address power management needs by improving platform support for manageability [8]. For example, processors today support a variety of power management states, including dynamic voltage and frequency scaling (DVFS), as well as low power sleep states. In addition, server processors routinely provide hardware extensions [17] to support virtualization software [2, 24, 26] for power saving consolidation of workloads. Virtualization provides additional benefits including fault isolation and improved manageability through dynamic

resource provisioning and live migration of virtual machines [4]. Therefore, it is beneficial to extend management capabilities that integrate with system virtualization stacks [15, 23].

The ability to specify power budgets on server platforms is becoming increasingly useful in various datacenter scenarios. For example, datacenters can be provisioned more aggressively if there are guarantees that the aggregate power consumption of deployed servers cannot exceed the maximum delivered to racks [5]. Similarly, the ability to limit power consumption in physical areas of a datacenter can help address thermal conditions or temporary reductions in cooling or power delivery capacity [14]. One approach for power budgeting includes integrating hardware controllers that make use of underlying processor power management capabilities transparently to system software [11, 22]. In the context of virtualized servers, a key drawback to hardware implementations is the fact that all virtual machines are treated equally. As previous work has shown, it can be beneficial to treat virtual machines (VMs) in a differential manner to improve overall system utility [16]. In this paper we build on this concept by proposing a software management architecture for quality-of-service (QoS) aware power budgeting of virtualized servers based upon feedback control methods.

Our power budgeting framework utilizes a control theoretic approach to driving system parameters under power constraints. In particular, we employ a proportional-integral-derivative (PID) [12] controller to dictate a system-level CPU usage constraint based upon monitored power consumption and a specified power limit. The output of this controller, along with feedback from VMs, is used by a resource manager to disperse CPU allocations amongst guests. The VM feedback is constructed using a congestion pricing based scheme [18], allowing guests to affect system-level decisions in a distributed manner based upon their respective QoS tradeoffs. Our evaluation of the system illustrates that power budgets can effectively be enforced by virtualization software while providing QoS benefits for performance sensitive VMs.

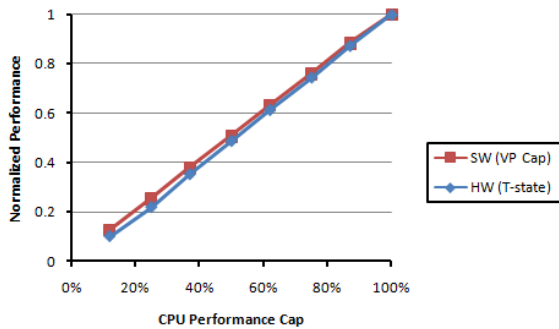
2. SYSTEM ARCHITECTURE

Enforcing a platform power budget requires the system to reduce the amount, or performance, of hardware resources available when the active power consumption exceeds the power budget. The QoS impact of reduced performance capacity on guest applications can vary, however, depending upon their respective requirements. Our goal is to build a set of distributed controllers that allow virtualized servers to observe power limits while providing differential services to VMs based upon their application specific tradeoffs. In this section, we describe the various controllers utilized by our power budgeting framework and how they interact to cooperatively manage the system.

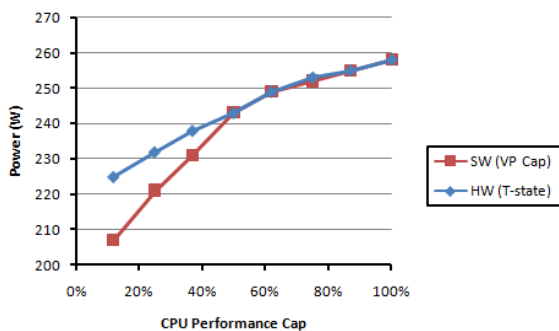
2.1 Power Budget Control with VP Caps

Server processors typically support multiple classes of execution state that can be used by software for power management. During active periods, the frequency and voltage (P-state) of a part can be reduced, impacting dynamic power consumption. Moreover, microarchitectures are also incorporating sleep states (C-states) that reduce power consumption when parts are idle. A third state, the throttle state (T-state), has been utilized for implementing power caps in hardware controllers [11]. Though P-states can significantly impact active power consumption, the ability to leverage them is becoming increasingly marginal due to, for example, dependencies amongst cores on a multicore package. C-states, on the other hand, can be utilized relatively independently amongst cores, though deeper sleep states still retain some dependencies due to shared caches, voltage planes, etc. Therefore, we have chosen to implement software power budgeting by leveraging idle power management features in multicore processors.

Hypervisor schedulers can be used to limit processing time to guest VMs. For example, previous work has demonstrated the use of the SEDF scheduler in the Xen [2] hypervisor to implement “soft scaling” [15]. The idea here is to utilize non-work-conserving scheduling, thereby allowing hardware to enter low power idle states. Hyper-V [26] provides the ability to cap the utilization of virtual processors (VPs) that are associated with a VM. By capping VPs, the utilization incurred on physical processors decreases, introducing idle time and thereby reducing power consumption. In this manner, VP capping can be used as a management actuator to implement power budgets.



(a) Workload Performance



(b) System Power

Figure 1: VP Capping Compared to Throttle States

Figure 1 validates the use of VP capping as a mechanism for power budgets. Figure 1(a) provides measured performance re-

sults using a simple compute bound microbenchmark running in a VM. For comparison, the figure includes data from software level VP capping, as well as hardware capping using T-states. As expected, we see that performance scales linearly with both mechanisms. These results support the intuition that from a performance impact perspective, software level capping has a similar effect to hardware capping mechanisms used in published literature [11]. Figure 1(b) provides power impacts of CPU throttling, again using both VP capping and T-states. We observe from the figure that for higher capping values, both software and hardware mechanisms have similar power profiles. As the cap is reduced further, however, the VP capping mechanism is able to achieve lower power due to the ability of the system to utilize deeper C-states during inactive phases. That is, as VP capping is applied, the duration of idle periods gradually become long enough to allow the system to place processors into deeper idle states, reducing power beyond just a halt state. For example, we observe processor times spent in lower power C-states of 69% and 85% for VP caps of 25% and 12% respectively. These results highlight a possible benefit of capping at the software layer, the ability to perform more work under the same power constraint by enabling power efficient states during idle periods. In summary, VP capping can serve as an effective control actuator for power budgeting.

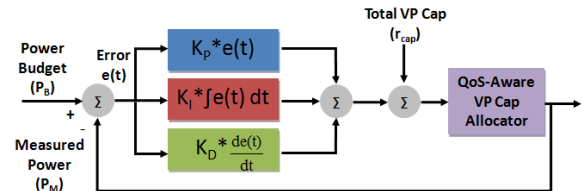


Figure 2: PID Based VP Cap Controller

There are various design options for implementing power budget controllers. Open loop controllers use a fixed setting for a given power budget. Since the power consumption based upon a set of VP cap values can vary across different workloads, an open loop approach must select a worst case configuration, resulting in reduced efficiency. On the other hand, closed loop feedback controllers have been shown to improve performance under constraints while meeting power budgets [11]. Therefore, to throttle power consumption using VP capping, we construct a proportional-integral-derivative (PID) feedback controller as illustrated Figure 2. The controller measures the error $e(t)$ between the monitored power consumption (P_M) and a power budget set point (P_B). The proportional component of the controller reacts to the current value of the error, the integral accounts for the recent history in error, and the differential accounts for the recent change in error. The weighted sum of these is used to update the actuator value, the sum of VP caps (r_{cap}) that can be assigned in the system. It should be noted that the controller can be changed into a modified PID controller (e.g. PI, PD, P, etc) by setting any of the weights, K_P , K_I , K_D , to zero.

In general, the power budget controller reduces the overall VP cap value when there is a sustained positive error and vice versa. The resource manager must then decide how to distribute the VP cap limit amongst guest VMs. One approach is to simply use a fair share strategy and set the same cap to all guests under the limit provided by the controller. Instead, we support a QoS-aware resource manager that uses guest input to determine when less demanding guests can be provisioned below their fair share to better accommodate more sensitive applications.

2.2 QoS Feedback with Congestion Pricing

In order to support differentiated services under power budgets, we must provide a way to account for VM QoS tradeoffs when assigning VP caps. One approach would be to have guests specify some type of utility function that describes the tradeoff between resource requirements and the overheads of obtaining them. However, this results in a centralized solution that must be aware of each of the different workloads in the system. A better solution is one that is decentralized and that removes the dependency of VMs having to provide a utility function. Previous work has used virtualized ACPI [8] states to obtain information regarding guest tradeoffs [15]. In this paper we employ a paravirtualized interface for guests that provides more meaningful information for feedback than what can be expressed using ACPI. In particular, based upon previous work, we employ a distributed framework that uses congestion pricing [9, 18, 19].

Congestion pricing allows a resource manager to signal application VMs when the system is resource constrained using *shadow prices*. The applications then react to these prices by altering the amount of resources they request. Here, the shadow price p_i is driven by the “costs” associated with provisioning a VM with some resource allocation r_i . Therefore, p_i is proportional to the resources allocated to the guest as well as the contribution to system congestion caused by the allocation. The guest provides feedback to the resource manager by changing its bid, B_i , on the amount of resources it desires. In particular, the resource bid by a guest is a function of the price p_i charged by the resource allocator, as well as the willingness of the guest to pay, w_i . The key to coordinating between the resource manager and the independent VM controllers, then, is how the shadow prices are calculated in the system.

The resource manager uses the set of bids received from guests to periodically update resource allocations and redefine shadow prices. As part of this, the manager calculates the “fair” VP cap, F , in the system based upon the r_{cap} provided by the power budget controller. A fundamental constraint of the resource manager is that it must assign a cap of at least F to a VM, unless $B_i < F$, in which case a cap as low as the VM bid is sufficient. If a set of VMs are provided caps below F , it frees up resources to over provision guests that are bidding above F . Therefore, at any given time, there are two types of costs in the system, an overage cost C_{over} based upon guests that obtain a cap less than F , and an opportunity cost C_{opp} based upon VMs that have bids B_i greater than the received cap r_i . These costs are summarized by Equations 1 and 2.

$$C_{over} = \sum F - r_i \quad (\forall r_i < F) \quad (1)$$

$$C_{opp} = \sum B_i - r_i \quad (\forall r_i < B_i) \quad (2)$$

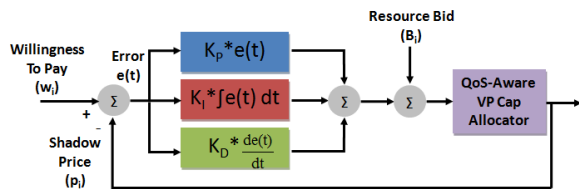


Figure 3: PID Based QoS-Aware VM Bid Controller

The resource manager uses an algorithm, described in Section 3, to update allocations based upon VM bids B_i , and then calculates current shadow prices using C_{over} and C_{opp} . Shadow prices for VMs with r_i greater than F are set to $\kappa_{over}C_{over}r_i$, and all others

are charged $\kappa_{opp}C_{opp}r_i$. Here, κ_{over} and κ_{opp} are configurable parameters in the system. Since the w_i of a guest reflects its current QoS tradeoffs, the goal of a VM agent during constrained periods is to place bids in a manner such that the shadow price p_i it is charged is approximately equal to its w_i . To achieve this, we again use a PID based controller in each guest that is driven by the error signal $(w_i - p_i)$ to adjust bids B_i as shown in Figure 3.

2.3 Architectural Summary

Having described the key components of our system, we provide the overall architecture in Figure 4. The figure illustrates a set of agents including a Root agent and multiple VM agents. The Root agent contains the power budget controller, as well as the QoS-aware resource manager. It takes as input the local power consumption, a power budget, and the set of VP cap bids communicated by VM agents over a VMBus interface. In addition, the Root agent outputs updates to VP caps to the hypervisor, as well as shadow prices that are communicated to VM agents. The VM agents themselves contain the PID controllers that drive guest bids. In general, the “willingness to pay” inputs to these agents are driven by a workload specific QoS manager. In this paper, we focus on evaluating the system using synthetic values for w_i , but plan to investigate the integration of a separate workload manager as future work.

3. IMPLEMENTATION AND EVALUATION

3.1 Methodology

Our experimental results are obtained with a representative enterprise server. The platform is a dual package configuration, with two quadcore processors for a total of eight cores. Additionally, the system is equipped with 6GB of physical memory. All guest VMs are provisioned with 1GB of memory for experiments. Online monitoring of server power is performed by measuring the wall AC power using the Extech 380801 analyzer which supports sampling rates of up to 2Hz.

Our experiments make use of a variety of workloads. First, we use a SPEC CPU2006 benchmark to represent computationally intensive transactions. In particular, we utilize the `gobmk` benchmark as it allows multiple threads to be executed within the VM memory constraints. Performance of the application is measured in terms of transaction processing rate. Another computationally intensive workload used in our evaluation is the Windows Advanced Rasterization Platform (WARP). WARP is a software rasterizer that allows DirectX 10 applications to run without physical graphics hardware. We thereby use it to perform graphics processing in VMs, with a performance metric of frames per second. Finally, we also implement a simple client-server application that models a backend image processing service. Here, an external client triggers a server running inside of a VM to perform a series of image manipulations (decode, encode, and compression). The response times of these requests are used as performance criteria.

3.2 Implementation

We have evaluated our system on the Microsoft Hyper-V virtualization software deployed with the Windows Server 2008 operating system. Hyper-V provides a separate Root partition, similar to Domain Zero (Dom0) in Xen, that is able to perform privileged management operations on the platform. We deploy a user space implementation of the Root agent in this partition. The agent communicates with the underlying hypervisor to configure CPU caps for VMs. The hypervisor scheduler itself supports cap values on a VM basis, where all VPs associated with a VM are capped with

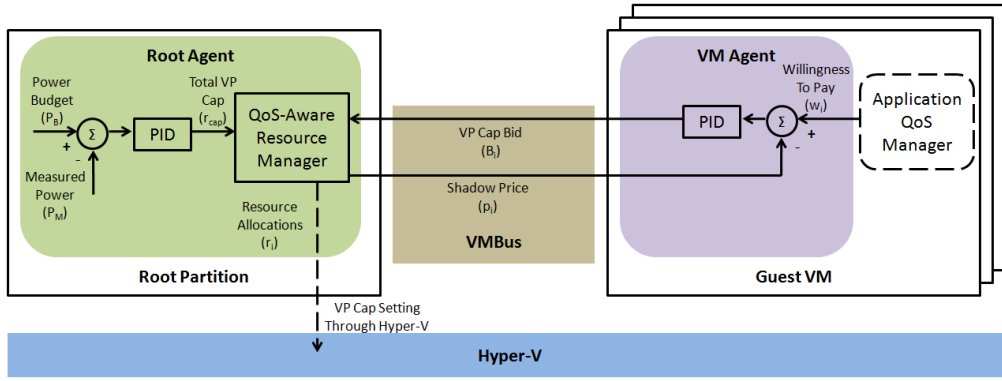


Figure 4: QoS-Aware Power Budgeting System Architecture

the same value. The cap value describes the maximum utilization that each VP can have, defined to be between zero and one hundred. Similar to the Root agent, VM agents are implemented as user space components.

The agents coordinate at every management interval, which for our implementation is one second. At each point, the Root agent samples the current power consumption and provides the error between the current power and the power budget set point to the PID power budget controller. The controller is designed to output a value between $10N$ and $100N$, where N is the number of VPs in the system. The output r_{cap} , divided by N , is the fair cap F for the interval. Thus, the fair cap is always between ten and one hundred (the lower limit of ten is the defined minimum level of service). The resource manager component of the Root agent then uses the set of bids B_i last received from guests to determine the new caps r_i . In particular, based upon F , the agent calculates the sum of all bids above F , B_{over} , and all other bids, B_{under} . These are then used to derive cap values as described in Algorithm 1.

Algorithm 1 QoS-Aware VP Cap Allocation.

```

1: if (( $B_{under} == 0$ ) OR ( $B_{over} == 0$ )) then
2:   Set cap to  $F \forall$  VMs
3: else
4:   if  $B_{under}$  allows  $B_{over}$  to be met within  $r_{cap}$  then
5:      $\forall$  VMs with  $B_i > F$ , set cap to bid
6:      $\forall$  VMs with  $B_i \leq F$ , set cap to bid. Distribute remaining
       cap under  $r_{cap}$  to these VMs in proportion to  $B_i$ 
7:   else
8:      $\forall$  VMs with  $B_i \leq F$ , set cap to bid
9:      $\forall$  VMs with  $B_i > F$ , set cap to  $F$ . Distribute remaining
       cap under  $r_{cap}$  to these VMs in proportion to  $B_i$ 
10:  end if
11: end if

```

The algorithm uses the fair cap if either B_{over} or B_{under} are zero. Otherwise, it distributes the system-level cap r_{cap} to over provision performance sensitive VMs that have bid over F up to their respective bids, while meeting at least the bid for all other VMs. Once determined, the Root agent uses the new VP caps r_i to calculate costs and new shadow prices p_i . The shadow prices are communicated to VM agents using VMBus interfaces, after which the VM agents respond with new bids B_i based upon their internal PID controllers.

There are various configurable parameters in our architecture that affect system behavior. The values used for experiments in this paper were found through a trial-and-error process and statically

assigned. In particular, the PID parameters for the power budget controller and VM bid controllers are ($K_P=10, K_I=5, K_D=1$) and ($K_P=1.0, K_I=0.25, K_D=0.05$) respectively. The κ_{over} and κ_{opp} parameters are both set to 0.001. Significant variances from these values were found to provide instability or poor performance.

3.3 Experimental Results

Our first set of experiments contain two VMs with four VPs each. One VM executes the `gobmk` workload from the SPEC suite, while the other runs the WARP application. In the first scenario the QoS-aware resource manager is disabled, and VP caps are assigned in a fair manner. Figure 5 illustrates the resulting power consumption data as the budget is varied from an initially unconstrained setting, to 250W, down to 240W, and back to 250W. The figure provides the power samples at each one second interval, as well as a data curve based upon averaging power values across a five second sliding window. We observe that, as expected with a feedback approach, the power deviates from the budget slightly at times. However, the controller is successfully able to converge to the power budget and adhere to it when averaged across larger time scales, thus validating the power budget controller in the Root agent.

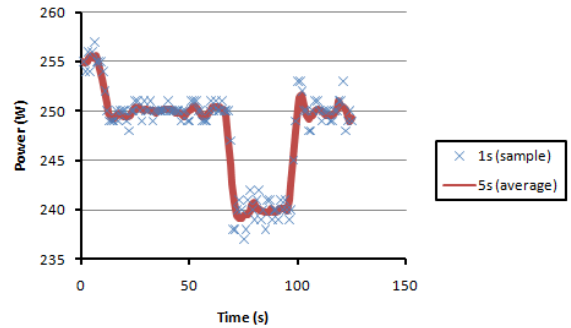


Figure 5: Power Consumption with Varying Budget

Next, we enable the QoS-aware resource manager, and initialize the willingness to pay of the WARP VM, w_{warp} , to be four times w_{gobmk} . After some period of time, the two values are swapped. This scenario reflects the WARP application as requiring a higher level of QoS initially, followed by a period where the performance requirements of the graphics workload are reduced and the demands of the SPEC workload increase. We observe the VP caps (averaged across intervals of 10s) assigned by the resource man-

ager based upon the ensuing feedback control when a power cap of 250W is set in Figure 6. The experimental results illustrate the ability of the QoS-aware resource allocator to react to and improve capping based upon relative workload demands. For example, during its demanding period, the system is able to provision the WARP workload with a cap improvement of 30% compared to the fair allocation (average cap of approximately 70) obtained with the same power budget in Figure 5. Similarly, *gobmk* experiences an improvement of 14% when the QoS requirements are reversed. An interesting observation from the figure is that the total VP cap, r_{cap} , is greater in the first half of the experiment by approximately 16% compared to the second half. This results from the fact that *gobmk* consumes more power than WARP for the same VP allocation. The system is able to account for this difference through its use of feedback control and reduce r_{cap} based upon the current workload mix and allocation.

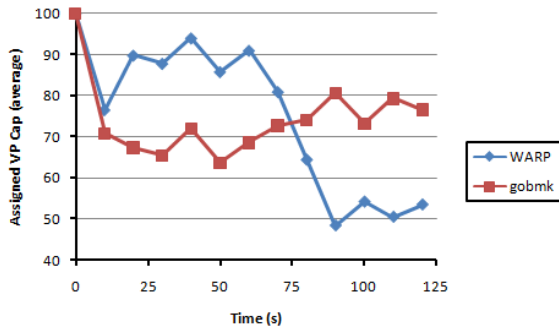


Figure 6: VP Caps with Varying VM QoS Requirements

Our final experimental scenario consists of four deployed VMs, each with two virtual processors. In addition to the WARP and SPEC VMs, we include two image processing server VMs. The VM agents are statically configured with w_i such that the ratio compared to w_{warp} are one, eight, and four for the WARP, SPEC, and two server VMs respectively. Figure 7 provides the performance experienced by each of these VMs during unconstrained execution (i.e., no power budget), as well as during enforcement of various power budgets. The performance numbers are based upon the respective workload metrics and normalized for each VM based upon the unconstrained case. An interesting result in the figure is that the *gobmk* VM actually experiences improved performance as budgets are introduced. This can be attributed to performance interference effects between VMs [10], where there is a performance penalty of running consolidated. As other VMs are capped, these effects diminish improving performance. In general, we observe that the distributed feedback controllers are able to guide the system in a desirable manner based upon the QoS tradeoffs described by willingness to pay values. The SPEC VM obtains preferential performance across power constraints while the two server guests have similar performance trends. This is due to the fact that their w_i values are equal, causing both VM agent PID controllers to behave similarly. Finally, the WARP VM experiences the most severe performance degradation as the power budget is reduced. These results support the ability of distributed feedback control to perform QoS-aware resource management under power budgets.

4. RELATED WORK

Feedback designs have been shown to be beneficial for managing computing systems in previous work. Improvements in system

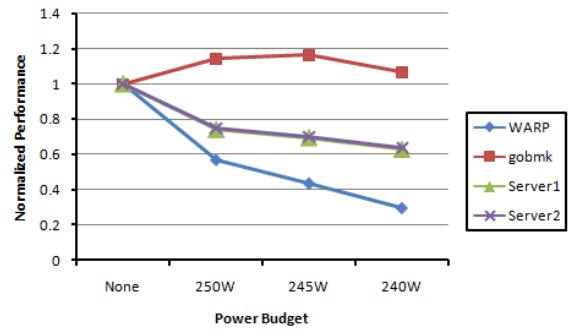


Figure 7: VM Performance with QoS-aware Power Budgets

throughput under deadline miss ratio constraints have been demonstrated for soft real time schedulers using PID controllers [12]. Other work has illustrated the benefits of using memory miss ratios to drive dynamic frequency scaling for energy efficient execution of real-time workloads [20]. These approaches effectively use system metrics as feedback signals. In this paper, we combine the use of power monitoring, as well as shadow prices and resource bids proposed in prior work [9, 18], as feedback mechanisms to drive power budgeting.

Power management has become a serious problem in server environments, motivating a significant amount of work. The ability to power budget servers, or sets of servers, has been shown to improve datacenter level provisioning by allowing increased server deployment under power delivery constraints [5, 7]. Previous work on power budgeting includes feedback controllers in hardware that manage performance states at fine time granularities [11]. Other approaches have considered implementing power budgets across nodes using non-uniform allocations across nodes in a datacenter [6, 25] or blades in a chassis [22]. The ability to coordinate hierarchical feedback controllers for power management has been demonstrated as well [21]. The work presented in this paper complements these solutions by considering how power budgeting can be performed in a QoS-aware manner for virtualized servers.

The use of virtualization requires extending power management mechanisms for virtual machines. Recent work has shown how energy accounting methods can be used to enforce energy consumption limits among guests based upon a server budget [23]. Other methods of incorporating VM-awareness include exposing virtualized ACPI states, and monitoring changes made by VM policies to manage workloads [15]. Power budgeting can be built on top of such a solution as well [16]. Compared to these techniques, the solution presented here uses a control theoretic framework to manage power budgets with a decentralized feedback model and paravirtualized interfaces for communicating with virtual machines.

5. CONCLUSIONS AND FUTURE WORK

Power management has emerged as a fundamental problem in modern datacenter environments. The costs associated with energy consumption, along with the power delivery and cooling infrastructure can constitute significant overheads. Therefore, it is necessary to deploy adaptive management mechanisms into datacenter software and platforms. In this paper, observing that virtualization is becoming increasingly predominant in enterprise systems, we extend an approach for power budgeting on virtualized server platforms. Our solution is based upon multiple feedback control loops that allow the server to maintain a power budget while pro-

viding resources to guest VMs in a QoS-aware manner. We maintain a decentralized architecture by utilizing a congestion pricing based scheme that uses shadow prices as signals to agents deployed within VMs. Our experimental results highlight the capability and benefits of our system.

As future work, we plan on extending the system in multiple ways. First, the evaluation performed in this paper focused on CPU bound workloads. Applications that make heavy use of I/O can cause greater variance in power consumption. As mentioned, the parameters in our system required manual tuning to obtain good performance and stability. It is unclear, however, whether these parameters behave well across different workloads and scenarios. We would therefore like to determine what system changes might be necessary in order to address these issues. We also plan to investigate how to generalize the use of shadow prices beyond VP caps, including, for example, allocating asymmetric physical processing units amongst VMs in a manner that improves the overall efficiency of the system. Finally, we will consider how to extend the paradigm to simultaneously manage additional types of resources, including memory and I/O capacity.

6. REFERENCES

- [1] Amazon Elastic Compute Cloud. <http://aws.amazon.com/ec2>.
- [2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2003.
- [3] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle. Managing energy and server resources in hosting centers. In *Proceedings of the 18th Symposium on Operating Systems Principles (SOSP)*, 2001.
- [4] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, May 2005.
- [5] X. Fan, W.-D. Weber, and L. Barroso. Power provisioning for a warehouse-sized computer. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, June 2007.
- [6] M. Femal and V. Freeh. Boosting data center performance through non-uniform power allocation. In *Proceedings of the IEEE International Conference on Autonomic Computing (ICAC)*, 2005.
- [7] S. Govindan, J. Choi, B. Urgaonkar, A. Sivasubramaniam, and A. Baldini. Statistical profiling-based techniques for effective power provisioning in data centers. In *EuroSys*, March 2009.
- [8] Hewlett-Packard, Intel, Microsoft, Phoenix, and Toshiba. Advanced configuration and power interface specification. <http://www.acpi.info>, September 2004.
- [9] P. Key, D. McAuley, P. Barham, and K. Laevens. Congestion pricing for congestion avoidance. Technical Report MSR-TR-99-15, Microsoft Research, February 1999.
- [10] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu. An analysis of performance interference effects in virtual environments. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2007.
- [11] C. Lefurgy, X. Wang, and M. Ware. Server-level power control. In *Proceedings of the IEEE International Conference on Autonomic Computing (ICAC)*, June 2007.
- [12] C. Lu, J. Stankovic, G. Tao, and S. Son. Design and evaluation of a feedback control edf scheduling algorithm. In *Proceedings of the 20th IEEE Real-Time Systems Symposium*, December 1999.
- [13] Microsoft Azure Services Platform. <http://www.microsoft.com/azure>.
- [14] J. Moore, J. Chase, P. Ranganathan, and R. Sharma. Making scheduling cool: Temperature-aware workload placement in data centers. In *Proceedings of the USENIX Annual Technical Conference*, June 2005.
- [15] R. Nathuji and K. Schwan. Virtualpower: Coordinated power management in virtualized enterprise systems. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP)*, October 2007.
- [16] R. Nathuji and K. Schwan. Vpm tokens: Virtual machine-aware power budgeting in datacenters. In *Proceedings of the ACM/IEEE International Symposium on High Performance Distributed Computing (HPDC)*, June 2008.
- [17] G. Neiger, A. Santoni, F. Leung, D. Rodgers, and R. Uhlig. Intel virtualization technology: Hardware support for efficient processor virtualization. In *Intel Technology Journal* (<http://www.intel.com/technology/itj/2006/v10i3/>), August 2006.
- [18] R. Neugebauer and D. McAuley. Congestion prices as feedback signals: An approach to qos management. In *Proceedings of the 9th ACM SIGOPS European Workshop*, September 2000.
- [19] R. Neugebauer and D. McAuley. Energy is just another resource: Energy accounting and energy pricing in the nemesis os. In *Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS)*, May 2001.
- [20] C. Poellabauer, L. Singleton, and K. Schwan. Feedback-based dynamic frequency scaling for memory-bound real-time applications. In *Proceedings of the 11th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, March 2005.
- [21] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu. No “power” struggles: Coordinated multi-level power management for the data center. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, March 2008.
- [22] P. Ranganathan, P. Leech, D. Irwin, and J. Chase. Ensemble-level power management for dense blade servers. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2006.
- [23] J. Stoess, C. Lang, and F. Bellosa. Energy management for hypervisor-based virtual machines. In *Proceedings of the USENIX Annual Technical Conference*, June 2007.
- [24] J. Sugerman, G. Venkitachalam, and B.-H. Lim. Virtualizing i/o devices on vmware workstation’s hosted virtual machine monitor. In *Proceedings of the USENIX Annual Technical Conference*, 2001.
- [25] X. Wang and M. Chen. Cluster-level feedback power control for performance optimization. In *Proceedings of the 14th International Symposium on High-Performance Computer Architecture (HPCA)*, February 2008.
- [26] Windows Server 2008 Hyper-V. <http://www.microsoft.com/hyperv>.