International Journal of Grid and Distributed Computing
Vol. 6, No. 3, June, 2013

A Low-Cost Power Measuring Technique for Virtual Machine in Cloud Environments

Xiao Peng[1*] and Zhao Sai[2]

[1]Department of Computer and Communication, Hunan Institute of Engineering
[2] School of Computer Engineering and Applied Science,
Memorial University of Newfoundland

[*]xtanefn@gmail.com; adventurs@gmail.com

Abstract

With the development of cloud computing, high energy consumption issue has attracted more and more attentions recently. To achieve the goal of energy conservation, accurately measuring the power consumption of distributed resource is of significant importance. Conventional power models can only provide fine-grained power measurement for physical devices instead of virtualized resources. In this paper, we proposed a novel power measuring technique, which uses performance monitoring counters (PMC) as basic indicator of virtual machine's power consumption. Meanwhile, we also designed a scheduling algorithm to reduce the measuring errors caused by recursive power consumption. Theoretical analysis indicates that the proposed algorithm can provide bounded error when metering virtual machine's power. Massive experiments are conducted by using various benchmarks on different platforms, and the results shown the proposed technique is effective and low-cost for measuring the power consumption of virtualized resources in large-scale cloud systems.

Keywords: *Cloud Computing; Virtual Machine; Power Consumption; Performance Monitor Counter*

1. Introduction

With the development of cloud computing [1], more and more high-performance datacenters have deployed their resources on virtualization platforms (*i.e.*, Xen, VMWare and KVM). It is well-recognized that cloud computing will be the next generation of network-based distributed computing paradigm in the future.

From the perspective of cloud providers, virtualization technology provides an effective approach to extending the capability of their cloud infrastructures without increasing too many IT devices. Meanwhile, server consolidation and live migration have been commonly considered as two effective mechanisms for reducing the power consumption in large-scale datacenters without significant performance degradation [2-5]. However, virtualization technique also brings many challenges on power management in those high-performance datacenters. In conventional data centers, the power consumption of a device can be directly metered using hardware meters or sensors. While in virtualized datacenters, the basic unit of resource management is virtual machine, which can not be directly connected by hardware power meters. Therefore, one of the most mentioned challenges is how to accurately meter the power consumption on the base of per-VM [2, 6-13]. Many efforts have been taken to construct the power model of VMs, and most of them are based on a simple assumption that the power consumption of a VM is linear to the utilization of its occupied physical resources [6-9, 11]. Although the assumption itself is correct, existing power models still can not provide fine-grained per-VM power metering, and the main difficulties are shown as following:

- In existing VM power models, the coefficient of each component is often obtained through empirical studies, which makes the models only suitable for certain kinds of the underlying physical resources. In heterogeneous datacenters, such empirical model is difficult to be widely applied [2, 9, 13].
- As the capability of a virtualized server is multiplexed by VM hypervisor, several VMs will compete for common physical devices. Therefore, VM scheduler's decision will have significant effects on the per-VM power consumption at runtime [8, 10-11].
- Recursive power consumption may be complicated when measuring per-VM power. For example, I/O requests often involve encrypt and decrypt operations, which often consume a great deal of CPU power. Therefore, it is difficult to distinguish the power spent on I/O operation from that spent on processor [9, 12, 14].

To address the above challenges, in this work we firstly formulize the relationship between the resource utilization and the accuracy of power metering. Then, we introduce the conception of relative PMC (also called PMC ratio) to measure the power consumption of a VM. In this way, we can avoid using empirical coefficients to construct VM power model. Based on the conception of relative PMC, we propose a novel VM scheduling algorithm, which uses the relative PMC to compensate the recursive power consumption of a VM. Also, we theoretically proofed that the proposed scheduling algorithm can provided bounded error when metering the VM power consumption.

The remainder of this paper is organized as following: in Section 2, we summarize the related work; in Section 3, we present the formulated relationship between the resource utilization and the accuracy of power metering; in Section 4, we introduce the conception of relative PMC and the PMC-based scheduling algorithm; Section 5 presents the experimental results and evaluation. Finally, Section 6 concludes the paper with a brief discussion of the future work.

2. Related Work

VM hypervisor has comprehensive information on individual VMs, while few of them are implemented with power-measuring functions. On the other hand, device drivers often expose interfaces to upper-level software for power consumption monitoring and metering, however they can not distinct different power consumers especially in virtualized environments. Therefore, early studies on measuring VM power are often implemented by extending a power monitoring adaptor between VM hypervisor and device driver modules. For instance, Cherkasova *et al.*, presented an approach to measuring the vCPU power consumption on Xen platforms [8]. In [14], Stoess *et al.*, presented a two-layer power managing framework for metering, evaluating and controlling the power consumption of virtualized devices. Unfortunately, both studies mainly focused on measuring the overall power overheads caused at virtualization layer, none of them can provide fine-grained per-VM power metering mechanism.

With the increasing requirements of fine-grained power management in modern data centers, plenty of efforts have been taken to address the issue of per-VM power metering. For instance, Kansal *et al.*, proposed a VM power metering mechanism, namely Joule-meter, which uses software-based power models to track per-VM power consumption [9]. In [10], Koller *et al.*, investigated the power modeling methodology in virtualization environments. By performing extensive experiments on various real benchmarks, they suggested that the application’s characteristics are of significant importance when modeling a VM power consumption. In [11], Bohra *et al.*, presented an empirical VM power model called VMeter, which is based on an experimental observation that the power consumption of different hardware components is highly correlated with each other.

Recently, PMC-based power metering techniques have been extensive studied. In [13], Bircher *et al.*, presented a comprehensive work on using PMCs to model power consumption for both server and desktop machines. Their experimental results strongly suggested that selecting suitable PMCs is of critical importance when building PMC-based power models. In [15], Bertran *et al.*, conducted massive experiments to investigate the effectiveness and accuracy of PMC-based power modeling technique in both virtualized and non-virtualized environments, and their experiments also demonstrated that PMC-based power models are more accuracy and stable than utilized-based power models. In [16], Lim *et al.*, demonstrated an empirical VM power model on Intel Core i7 platform. In this empirical model, performance counters are classified into three classes according to their relationship to CPU, memory or others, and the correlation between these counters are obtained through extensive tests on various benchmarks.

Our work also uses PMCs to collect the basic VM power consuming events. The difference is that our work does not rely on the empirical approach to obtaining the coefficients of various PMC events like the above studies, because it is highly couple with the underlying platform and is difficult to be employed in heterogeneous datacenters. Instead, we only record the PMC events on per-VM basis and calculate the distributions of various kinds of PMC events. Combining with our energy-credit VM scheduler and the PMC event distribution, we proposed an efficient online per-VM power measuring mechanism, which is independent on the architecture of underlying platform.

3. Power Model for Virtual Machine

3.1 Utilization-based power models

In a typical virtualized server, there exists a multi-layered software stack as shown in Figure 1, which consists of physical devices, native OS, drivers, VM hypervisor, VM utilities and multiple VMs.

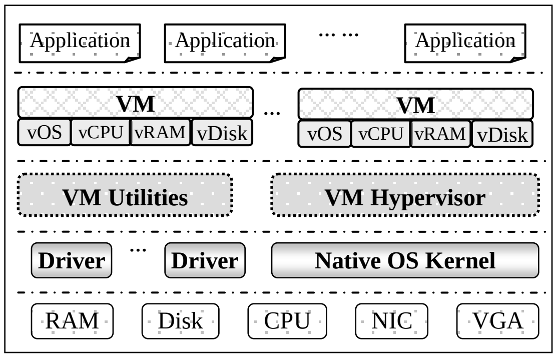


Figure 1. Software Framework in Virtualized Servers

In this architecture, the privileged VM hypervisor and VM utilities can control over physical devices through driver modules or hardware interfaces directly. Therefore, they can obtain coarse-grained power consumption information of physical components, such as average or peak power consumption. However, fine-grained per-VM power consumption can not be obtained in this approach, because the capability of physical devices are multiplexed by VM hypervisor across many VMs, whose actual power consumption is decided by the characteristic of the current application that running on it. Furthermore, the virtualization layer may lead to a series of complicated calling-chain, which make it difficult to separate the application's power consumption from the overall power consumption [8-9, 14]. For example, creating or migrating a VM will lead to intensive I/O operations and NIC operations [3, 8], this part of power consumption should be considered as virtualization overheads instead of application's power consumption.

For the convenience of representation, we note the power consuming components as J = {CPU, RAM, Disk, I/O}. Typically, the power consumption of a server is formulated as:

$$P_{server} = P_{static} + \sum_{j \in J} (k_j \cdot U_j) \tag{1}$$

where P_{static} is the fixed power consumption when there is no workload on it, U_{j} is the utilization of each kind of components, k_{j} is the dynamic power coefficient. When a server is virtualized, its power model can be rewritten as:

$$P_{server} = P_{static} + \sum_{i=1}^{M} P_i^{vm} \tag{2}$$

where P_i^{vm} is the dynamic power consumption of VM_{i}, M is the active VM number on this server. As the VMs can not be connected by hardware power meters, their actual power consumption P_i^{vm} should be measured in an indirect way. The most mentioned per-VM power model is as following:

$$P_i^{vm} = \frac{P_{static}}{M} + W_i \cdot \sum_{j \in J} (k_j \cdot U_j) \tag{3}$$

where W_{i} is the processor utilization ratio that allocated to VM_{i}. The above VM power model assumes that the static power consumption is equally shared by all VMs, and the dynamic power consumption is strictly proportional to W_{i}, which is decided when creating the VM and obeyed by VM scheduler. In order to keep the power model accurate, VM scheduler must satisfy the following equation.

$$\forall i, j, \quad \left| \frac{U_i(t_1,t_2)}{W_i} - \frac{U_j(t_1,t_2)}{W_j} \right| = 0 \tag{4}$$

where $U_{\mathrm{i}}(t_1,t_2)$ is the actual utilization of resource consumed by VM_{i} during the time period $[t_1,t_2]$. This condition implies that the VM scheduler must keep the actual utilization $U_{\mathrm{i}}(t_1,t_2)$ strictly consistent with its promise W_{i} among all VMs. Unfortunately, none of the current VM schedulers can satisfy this condition, because runtime characteristics of applications have significantly effects on the actual utilization. For instance, considering VM_{i} is data-intensive and VM_{k} is computation-intensive, then it tends to be $U_{\mathrm{i}}(t_1,t_2)/W_{\mathrm{i}} < U_{\mathrm{k}}(t_1,t_2)/W_{\mathrm{k}}$. Therefore, the more general form of the condition can be noted as:

$$\forall i, j \quad \left| \frac{U_i(t_1,t_2)}{W_i} - \frac{U_j(t_1,t_2)}{W_j} \right| \le \psi \tag{5}$$

It is clear that the accuracy of per-VM power model depends on the parameter ψ. The bigger value of ψ will result in less accuracy when modeling the VM power consumption. That is

$$\left| P^{VM}_{actural} - P^{VM}_{measured} \right| \propto \psi \tag{6}$$

So, parameter ψ can be considered as the upper bound of error when metering the VM power consumption.

3.2 PMC-based power models

In modern servers, performance monitoring counter (PMC) has been widely supported in various platforms. Typically, these PMCs can be categorized into many classes according to their relationship to the specific components, such as CPU, GPU, chipset, RAM, I/O controller, and disk. In a server, the PMC events are propagated between different subcomponents. As only a few of PMCs are representative for modeling power consumption, we select these representative PMCs through performing extensive experiments on various benchmarks. In this work, the PMC set that selected includes {uOps, Halt, LLC, TLB, DMA, FSB, Interrupt}, which are used for accounting the power consumption of four components including processor, memory, disk and I/O controller. The detail descriptions of each PMC can be found in [17]. To figure out the co-relation between these PMCs and the power consumption of different subcomponents, a series of tests are performed by using various benchmark on four different kinds of servers. The summary of the experiments are as following:

$$\begin{cases} P_{cpu} \propto uOps - Halt^2 \\ P_{ram} \propto LLC + TLB + FSB \\ P_{disk} \propto Interrupt + DMA^3 \\ P_{IO} \propto Interrupt + DMA \end{cases} \tag{7}$$

Based on the above summary, it is clear that the power consumption of different subcomponents may be co-relative to two or three kinds of PMC events. It is noteworthy that we do not impose any empirical coefficients in the above formulas (7), which is of significant importance for architecture-independence. For the convenience of representation, we give the following notations

$$\begin{cases} E_{cpu}(t_1,t_2) = uOps_{t_1 \to t_2} - Halt_{t_1 \to t_2} \\ E_{ram}(t_1,t_2) = LLC_{t_1 \to t_2} + TLB_{t_1 \to t_2} + FSB_{t_1 \to t_2} \\ E_{disk}(t_1,t_2) = Interrupt_{t_1 \to t_2} + DMA_{t_1 \to t_2} \\ E_{IO}(t_1,t_2) = Interrupt_{t_1 \to t_2} + DMA_{t_1 \to t_2} \end{cases} \tag{8}$$

where $E_j(t_1,t_2)$ is the PMC events related to component j ($j \in J$) in the time duration $[t_1,t_2]$. If we defined the power model of VM_k as:

$$P_k^{vm}(t_1,t_2) = \sum_{j \in J} P_{k,j}^{vm}(t_1,t_2) \tag{9}$$

where $P_{k,j}^{vm}(t_1,t_2)$ is the power consumption consumed by physical component j. So, for a given server that running multiple VMs, the dynamic power consumption of each VM is linear to their relative PMC ratio. That is

$$P_{k,j}^{vm}(t_1,t_2) \propto \frac{E_j^k(t_1,t_2)}{E_j(t_1,t_2)}, \quad \forall j \in J \tag{10}$$

where $E_j^k(t_1,t_2)$ is the part of $E_j(t_1,t_2)$ that produced by VM_k.

It is well-known that mapping the absolute PMC accounts to power consumption is a notorious difficult work [13, 15-16], which requires not only comprehensive architecture knowledge but also long training time to obtain sufficient accurate model. Therefore, we use relative PMC value (also called PMC ratio) to describe VM power model instead of the absolute PMC accounts, as shown in (10). It is noteworthy that the validity of formula (10) relies on an assumption that all the workloads on a server are executed by VMs. This assumption is acceptable for most modern virtualized servers.

4. PMC-based Virtual Machine Scheduler

Classical VM schedulers mainly focus on fairness in term of processor utilization. This strategy tends to underestimate VM's recursive operation, which consequently results in that the recursive power consumption is often ignored when constructing VM power model. So, it has to use other mechanisms to measuring the non-processor power consumption. For instance, the VM power metering system proposed in [14] has to resort to VM driver to measure the disk power consumption. To taken the recursive power consumption into account, we design a novel scheduling algorithm, namely PMC Accounting based Credit Scheduling (PACS), which uses both the relative PMC account and the utilization ratio as credits when scheduling VMs. The implementation of PACS algorithm is shown as following.

PACS: PMC Accounting based Credit Scheduling

```
Begin
  1.  for each arrival $VM_i$ do
  2.    $R_i := W_i$; $L_i := 1$;
  3.  end for
  4.  while the processor is idling do
  5.    Sort VMs as $\{VM_{k1}, VM_{k2}, \ldots, VM_{km}\}$ according to
            $\frac{L_{k_1} - R_{k_1}}{W_{k_1}} \le \frac{L_{k_2} - R_{k_2}}{W_{k_2}} \le \cdots \le \frac{L_{k_m} - R_{k_m}}{W_{k_m}}$;
  6.    Assign processor to $VM_{k1}$ with utilization $L_{k1}$-$R_{k1}$;
  7.    for $n = 2$ to $m$ do
  8.      $R_{k_n} := R_{k_n} + \frac{L_{k_1} - R_{k_1}}{W_{k_1}} \cdot W_{k_n}$;
  9.      $L_{k_n} := L_{k_n} + \sum_{j \in J - \{cpu\}} E_j^{k_n} \Big/ \sum_{j \in J - \{cpu\}} E_j$;
  10.   end for
  11.   $R_{k_1} := 0$;
  12.   $L_{k_1} := L_{k_1} + \sum_{j \in \{cpu\}} E_j^{k_1} \Big/ \sum_{j \in \{cpu\}} E_j$;
  13. end while
End.
```

In PACS algorithm, each VM_i is associate with a credit value R_i and its initial value is set as the utilization ratio W_i. L_i is an accumulator that records the relative PMC value of VM_i, and its initial value is set 1. In each scheduling, all the VMs will be sorted according to the criterion as shown in step 5, and the first VM will be scheduled. With respect to the scheduled VM, its credit value will be proportional shared by other VMs as shown in step 8.

PACS uses the relative PMC value L_i as a heuristic of recursive operation. More specific, when a VM is scheduled, its L_i will accumulate those CPU-related PMC events (as shown in step 12). So, if the scheduled VM is computation-intensive, the increased L_i value will result in more utilization ratio (L_i-R_i) in its next scheduling. On the other side, if the scheduled VM is data-intensive, the PACS algorithm will reduce its future utilization ratio in its future scheduling. As to other un-scheduled VMs, their non-processor PMC events will be continually accumulated to L_i as shown in step 9. There are two cases that need to be discussed here: (1) If the un-scheduled VM is in waiting queue, it will not trigger any PMC event. So its L_i keep unchanged. (2) If it is performing recursive operation (*i.e.*, DMA, disk access or other I/O operation), the increased L_i will delay its future scheduling according to the criterion in step 5, and its future processor utilization ratio will be increased so as to compensate such delaying.

In the Figure 2, we present an example of scheduling sequence when using PACS algorithm for VM scheduling.

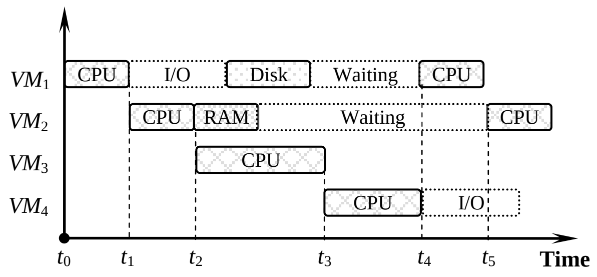


Figure 2. An Example of Scheduling Scheme of PACS

In this example, VM_1 is scheduled at time t_0, however it is blocked at time t_1 because of I/O waiting and disk access. So, the PACS algorithm will compensate its utilization loss by accounting its non-processor PMC event since time t_1. In this case, most of PMC events are belong to {Interrupt, DMA} as shown in Eqs.(7). VM_2 is scheduled at time t_1 because the VM scheduler tries to make the processor as busy as possible. However, massive data transferring (*e.g.*, DMA operation between RAM and disk) makes the scheduler preempt the processor from VM_2 for other VMs. It is noteworthy that simple memory access operation will not make VM_2 preempted. So most of PMC events for VM_2 are belonging to {LLC, TLB, DMA}.

In this example, VM_3 is the only one that completely uses out its utilization ratio. This usually means that it is a computation-intensive application. According to our previous analysis, VM_3 will obtain more utilization ration in its future scheduling. At time t_3, there are four VMs (VM_1 ~ VM_4) in the waiting queue. As VM_1 and VM_2 are compensated with more utilization ratio (increased L_i value), their scheduling rank will be lower than VM_4. As to the VM_3, its scheduling rank also is lower than VM_4 because it uses out its credits R_3. So, VM_4 obtained the processor until it is blocked by I/O at time t_4. Based on the above illustration, it can be seen that the PACS algorithm not only uses the PMCs information to compensate the recursive power consumption, it also dynamically adjusts the priority of waiting VMs in a fairness manner.

5. Experiments and Performance Comparison

5.1. Experimental setting

To investigate the performance of PACS algorithm, we conducted series of experiments on two platforms with various benchmarks. In the experiments, we use Oprofile [18] to sample the PMC events, and the original reports produced by Oprofile are categorized for individual VMs. All the benchmarks used in our experiments are obtained from standard testing suits. For instance, bzip2 and mcf are come from SPECcpu2006 benchmark suite [19]; TPC-W [20] is a representative benchmark for testing the performance of web servers; Cachebench [21] is benchmark for measuring the performance of cache subsystem performance; IOZone [22] is a file system benchmark that generates variety of file system operations.

5.2. Power measurement error comparison

In this experiment, we mainly concentrate on investigating the accuracy of per-VM power. To do this, we need to obtain the power baseline of each benchmark at first. So, we run the benchmarks on two platforms one by one. Due to the benchmark is the only VM on the platform, the actual power of each benchmark can be approximately estimated by the baseline power which is calculated by formula (3) where M=1. The results are shown in Table 1. When running multiple VMs concurrently, we use the following formula to calculate the error of per-VM power.

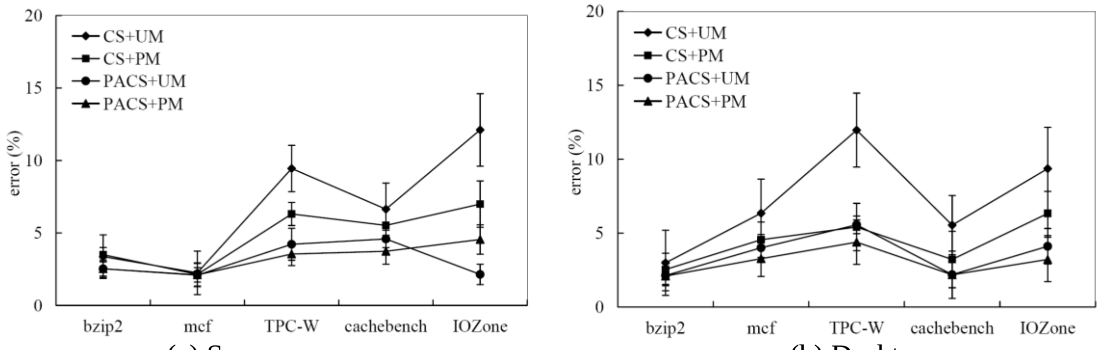
$$err\% = \left| \frac{P^{VM}_{measured} - P^{VM}_{baseline}}{P^{VM}_{baseline}} \right| \times 100\% \tag{11}$$

where $P^{VM}_{measured}$ can be obtained by formula (3) or (10). If using formula (3), the per-VM power is measured by the conventional utilization-based technique; if using formula (10), it is measured by relative PMC based technique that proposed in this paper. Our experiments will test both techniques and compare with each other. As to the VM scheduler, we compare out PACS algorithm with the Xen's default Credit Scheduler (CS) [23], which only uses utilization ratio as scheduling credits. Therefore, the experimental results can be categorized into four groups, which includes: CS+UM, CS+PM, PACS+UM, and PACS+PM.

Table 1. Power Baseline of Five benchmarks (Watt)

	bzip 2	mcf	TCP-W	Cache bench	IOZone
Server	26.9	25.7	19.8	26.1	15.9
Desktop	14.3	16.6	11.2	14.5	8.7

The experimental results are shown in Figure 3. In this experiment, we set all VMs equally sharing processor, that is VM_1= VM_2= VM_3= VM_4= VM_5=20%. According to the analysis mentioned above, such fairness allocation strategy will lower down the error bound of power measurement when using PACS algorithm. The most distinguishing result is that the error of measuring per-VM power exhibits highly co-relationship with the characteristics of benchmarks. For example, those cpu-intensive benchmarks (bzip2 and mcf) have very lower error in all cases, while the disk or I/O intensive benchmarks (TPC-W and IOZone) are difficult to be accurately measured when using CS+UM technique. That is because those VMs are often blocked by VM hypervisor when performing massive I/O operations, however the CS+UM technique can not take such recursive power into account. When using PMC measurements, all the power consumption of physical components will be accounted. So, it can significantly reduce the error for those disk and I/O intensive benchmarks. As shown in Figure 3(a), the error of TPC-W is reduced from 11.9% to 5.4%, and the improvement on IOZone is more significant.



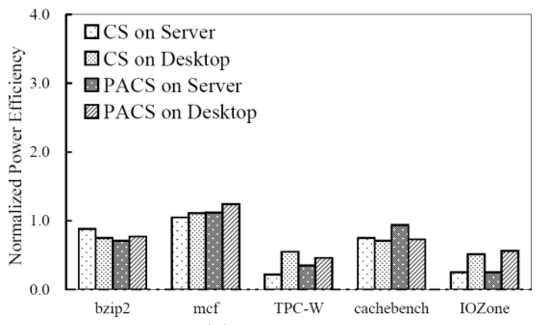
(a) Server (b) Desktop

Figure 3. Power Measurement Error of Benchmarks

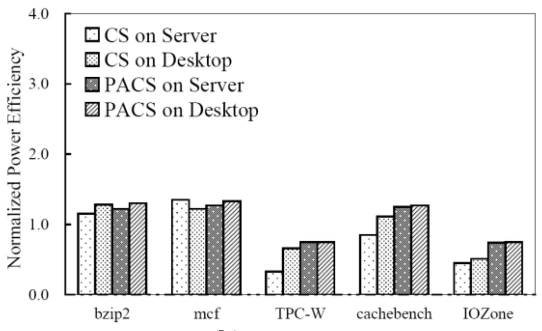
General speaking, the error on desktop platform is higher than that on server platform in most cases. Two exceptions happen on mcf and IOZone. For the mcf, we notice that it is most the cpu-intensive in all benchmarks, while IOZone is the most disk-intensive benchmark. When running the benchmarks on desktop platform, only two VMs can concurrently be executed. So, the utilization of individual devices (*i.e.*, cpu, ram, disk, I/O) is highly imbalanced when running mcf and IOZone. As mentioned before, UM based technique has to account the overall power into the current VMs, which make its error very high. While on the server platform, there are four cores which allow four VMs concurrently running at most. Therefore, this significantly balances the measuring error when using UM based technique. While using PMC based technique, this increased error can be reduced about 25% for mcf benchmark and 30% IOZone benchmark. Although PMC based technique can reduce part of error, the disk-related PMC events fail to accurately measure the actual power consumption when the disk is in overload working state. When using the combination of PACS+UM, as the PACS tends to select a VM with lest power consumption at recent duration, the IOZone is frequently scheduled and allocated with a very small processor utilization ratio. Such a scheduling decision is very effective for those disk or I/O intensive workloads. On the other side, UM based technique can accurately measures the power consumption of disk device. So, we obtain the most accurate power measurement in this case.

5.3. Power efficiency comparison

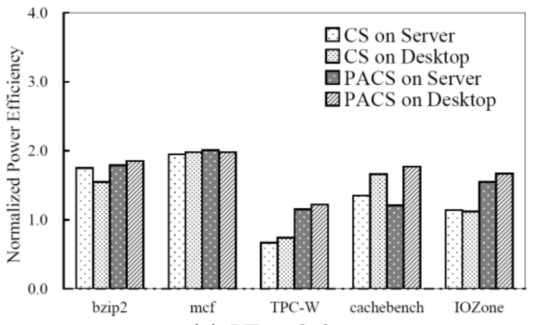
It is well-known that the virtualization ratio (the ratio of concurrent VMs to physical processor number, VR) metric has significant effects on the PE of virtualized servers. To comparing the performance, we use VR metric to normalize the PE. The experiment is conducted four times on each platforms with *VR*=1.0, 2.0, 3.0, 4.0, and the results are shown in Figure 4(a)~(d).



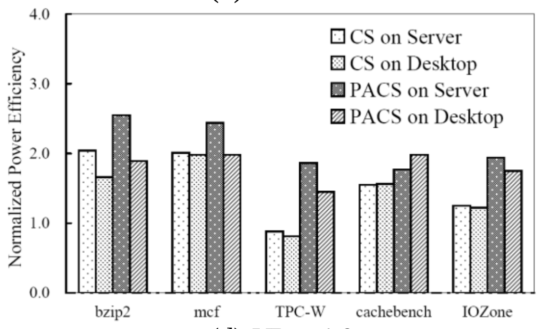
(a) *VR* = 1.0



(b) *VR* = 2.0



(c) *VR* = 3.0



(d) *VR* = 4.0

Figure 4. Power Efficiency with various virtualization ratios

Based on the results shown in Figure 4, we notice that the PE metric of cpu-intensive benchmarks (bzip2 and mcf) is generally higher than other benchmarks. Furthermore, it is significantly improved with higher value of *VR*. That is because the power consumption of processor is the most dominate in the tested platforms. As the cache-bench, its normalized PE is slightly lower than bzip2 and mcf, but significantly higher that TPC-W and IOZone. By examining the PMC logs, we find that the cache-bench triggered plenty of LLC-miss events during its execution. Unlike the DMA event, LLC-miss event will not result in long-term blocking. So, both processor and memory are kept in working state, which makes the power consumption of cache-bench very similar to those cpu-intensive benchmarks. As to the TPC-W, its performance is the worst in all cases. For instance, its normalized PE is always lower than 1.0 when using CS scheduling algorithm.

We noticed that the normalized PE of desktop is higher than that of the server in many cases. By checking datasheets of the two platforms, we find the server is designed to provide high performance as much as possible. For example, the processor on the desktop platform will gate the clock signal when it detects 'idle phases', which has a significant effect on power saving. While the server platform does not has such a mechanism on its processors. So, when the workload is not heavy, the desktop platform seems more power efficient than the server platform. With the increasing of *VR*, the power efficiency of server increases quick especially for those cpu-intensive benchmarks. As the effects of scheduling algorithm on the normalized PE metric, we can see that the performance difference of two algorithms is very small when *VR*=1.0, 2.0. When *VR*=3.0, 4.0, the normalized PE of TPC-W and IOZone is improved more quickly by using the PACS algorithm. As to bzip2 and mcf, the performance of two algorithms keep the same when VR=3.0, however, the PE significantly increased from about 1.81~1.95 to 2.50 when using the PACS on server platform. Such an improvement indicates that PACS algorithm is more effective to improve the power efficiency that conventional CS algorithm.

6. Conclusion

In this work, we address the issue of VM power measuring technology. By using relative PMC value as scheduling heuristic, we proposed a PMC Accounting based Credit Scheduling (PACS) algorithm, which uses the PMCs information to compensate the recursive power consumption. The experimental results that obtained from various benchmarks show that the practical error of power measuring can be significantly reduced when using PACS algorithm. In addition, experimental results also indicate that such a scheduling strategy is effective to improve the power efficiency when the virtualization ratio of a server is high. In the future, we plan to enhance the PACS algorithm with deadline-guarantee functionality, since plenty of real-time applications have emerged in many cloud-based systems. In addition, we will take more efforts on the other QoS metrics when using the PACS algorithm.

References

[1] M. Sedaghat, F. Hernández and E. Elmroth, "Unifying Cloud Management: Towards Overall Governance of Business Level Objectives", IEEE/ACM Int'l Symp. on Cluster, Cloud and Grid Comp., **(2011)**, pp. 591-597.

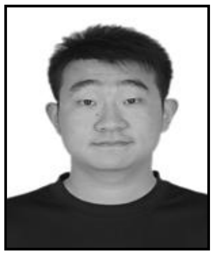
[2] G. Dhiman, G. Marchetti and T. Rosing, "vGreen: a System for Energy-efficient Management of Virtual Machines", ACM Trans. on Design Automation of Electronic Systems, vol. 1, no. 16, **(2010)**, pp. 1-27.

[3] L. Lefevre and A. C. Orgerie, "Designing and Evaluating an Energy Efficient Cloud", Journal of Supercomputing, vol. 3, no. 51, **(2010)**, pp. 352-373.

[4] X. Liao, H. Jin and H. Liu, "Towards a Green Cluster through Dynamic Remapping of Virtual Machines", Future Generation Computer Systems, vol. 2, no. 28, **(2012)**, pp. 469-477.
[5] K. Li, "Optimal Configuration of a Multicore Server Processor for Managing the Power and Performance Tradeoff", Journal of Supercomputing, vol. 1, no. 61, **(2012)**, pp. 189-214.
[6] P. Mahadevan, S. Banerjee, P. Sharma, *et al.*, "On Energy Efficiency for Enterprise and Data Center Networks", IEEE Communications Magazine, vol. 8, no. 49, **(2011)**, pp. 94-100.
[7] M. Mazzucco and D. Dyachuk, "Optimizing Cloud Providers Revenues via Energy Efficient Server Allocation", Sustainable Computing: Informatics and Systems, vol. 1, no. 2, **(2012)**, pp. 1-12.
[8] L. Cherkasova and R. Gardner, "Measuring CPU Overhead for I/O Processing in the XEN Virtual Machine Monitor", USENIX Annual Technical Conference, **(2005)**, pp. 387-390.
[9] A. Kansal, F. Zhao, J. Liu, *et al.*, "Virtual Machine Power Metering and Provisioning", ACM Symposium on Cloud Computing, **(2010)**, pp. 39-50.
[10] R. Koller, A. Verma and A. Neogi, "WattApp: an Application Wware Power Meter for Shared Data Centers", International Conference on Autonomic Computing, **(2010)**, pp. 31-40.
[11] A. Bohra and V. Chaudhary, "VMeter: Power Modelling for Virtualized Clouds", International Parallel and Distributed Processing Symposium, **(2010)**, pp. 1-8.
[12] B. Krishnan, H. Amur, A. Gavrilovska, *et al.*, "VM Power Metering: Feasibility and Challenges", ACM SIGMETRICS Performance Evaluation Review, vol. 3, no. 38, **(2010)**, pp. 56-60.
[13] W. L. Bircher and L. K. John, "Complete System Power Estimation using Processor Performance Events", IEEE Trans. on Computers, vol. 4, no. 61, **(2012)**, pp. 563-577.
[14] J. Stoess, C. Lang and F. Bellosa, "Energy Management for Hypervisor-based Virtual Machines", USENIX Annual Technical Conference, **(2007)**, pp. 1-14.
[15] R. Bertran, Y. Becerra, D. Carrera, *et al.*, "Energy Accounting for Shared Virtualized Environments under DVFS using PMC-based Power Models", Future Generation Comp. Syst., vol. 2, no. 28, **(2012)**, pp. 457-468.
[16] M. Y. Lim, A. Porterfield and R. Fowler, "SoftPower: Fine-grain Power Estimations using Performance Counters", International Symposium On High-Performance Distributed Computing, **(2010)**, pp. 308-311.
[17] M. Sharifi, H. Salimi and M. Najafzadeh, "Power-efficient Distributed Scheduling of Virtual Machines using Workload-aware Consolidation Techniques", Journal of Supercomputing, vol. 1, no. 61, **(2012)**, pp. 46-66.
[18] Oprofile. http://oprofile.sourceforge.net/.
[19] SPEC CPU2006. http://www.spec.org/cpu2006/.
[20] Transaction Processing Performance Council TPC-W. http://www.tpc.org/tpcw/default.asp/.
[21] Cachebench. http://icl.cs.utk.edu/projects/llcbench/cachebench/.
[22] IOZone. http://www.iozone.org/.
[23] L. Cherkasova, D. Gupta and A. Vahdat, "Comparison of the Three CPU Schedulers in XEN", ACM SIGMETRICS Performance Evaluation Review, vol. 2, no. 35, **(2007)**, pp. 42-51.

Authors

Xiao Peng received the Ph.D degree in computer science in CSU at 2010. He is Currently an associate professor in the Hunan Institute of Engineering. His research interests include cloud computing, distributed resource management. He is a member of ACM and IEEE.



Zhao Sai received his bachelor degree in Hunan Institute of Engineering, and now is persuading master degree in Memorial University of Newfoundland at Canada. His research interests include cloud computing, green computing, virtualization technology. He is a student member of IEEE.