Computing and Informatics, Vol. 35, 2016, 241–258

IMPROVING ACCURACY OF VIRTUAL MACHINE POWER MODEL BY RELATIVE-PMC BASED HEURISTIC SCHEDULING

Peng Xiao, Dong-Bo Liu

Department of Computer and Communication
Hunan Institute of Engineering
411100 Xiang-Tan, China
e-mail: {xtanefn, liudongbo1974}@gmail.com

Abstract. Conventional utilization-based power model is effective for measuring the power consumption of physical machines. However, in virtualized environments its accuracy cannot be guaranteed because of the recursive resource accessing among multiple virtual machines. In this paper, we present a novel virtual machine scheduling algorithm, which uses Performance-Monitor-Counter as heuristic information to compensate the recursive power consumption. Theoretical analysis indicates that the error of virtual machine power model can be quantitative bounded when using the proposed scheduling algorithm. Extensive experiments based on standard benchmarks show that the error of virtual machine power measurements can be significantly reduced comparing with the classic credit-based scheduling algorithm.

Keywords: Cloud computing, energy efficiency, data center, heuristic algorithm, workflow

1 INTRODUCTION

In cloud platforms [1], virtualization technology has emerged as a promising approach to reducing the energy consumption of high-performance data centers [2, 3]. From the perspective of cloud providers, virtualization technology provides the mechanisms of server consolidation and virtual machine (VM) migration, which can be used to enhance the capability of cloud platforms without increasing too many IT devices [4]. Consequently, massive efforts have been taken to implement energy management and optimization through various kinds of virtualization-based

technologies [5, 6, 7]. However, the issue of how to efficiently and accurately measure the power consumption on per-VM basis still remains opened.

As VM is the basic unit of scheduling and provision in virtualized environments, knowing per-VM power consumption is the basic prerequisite to achieving the goal of fine-grained power/energy management [8, 9, 10, 11, 12]. Furthermore, by exploring the per-VM power usage, cloud providers can charge their clients with more flexible pricing strategies instead of current fixed pricing schemes [13, 14, 15]. The direct difficulty of measuring VM power consumption is that a VM cannot be connected by hardware power meters. So, researchers have to resort to indirect approaches to measuring the VM power consumption.

Power model technology is the most mentioned approach for VM power measuring [9, 11, 13], which is often based on a simple assumption that the power consumption of a VM is linear to its runtime resource utilization. However, plenty of studies indicate that such utilization-based VM power models are insufficient for implementing fine-grained power/energy management, and the difficulties are summarized as follows:

- As the physical machine is multiplexed by VM hypervisor, multiple 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, 9, 11, 16].
- Many utilization-based power models has several coefficients, which are obtained through empirical approaches and only suitable for certain kinds of underlying resources. So they are difficult to be widely applied in heterogeneous distributed systems [9, 12, 14].
- 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 [8, 10, 14].

To overcome the above difficulties, this work presents a novel VM scheduling algorithm, which uses the Performance-Monitor-Counter (PMC) information as heuristic to compensate the accuracy loss caused by recursive resource accessing. As the proposed algorithm is based on relative PMC metrics empirical coefficients are no longer needed in our VM power model, which makes the proposed approach being independent of the underlying physical resources.

The rest of this paper is organized as follows: in Section 2, the related work is summarized; in Section 3, we describe the problem formally and introduce the relative PMC-based power model; Section 4 presents the proposed heuristic scheduling algorithm and the theoretical analysis; in Section 5, the experimental results and evaluations are presented. Finally, Section 6 concludes the paper with a brief discussion of the future work.

2 RELATED WORK

Early studies on measuring VM power are often implemented by extending a monitoring adaptor between VM hypervisor and driver modules. For instance, Cherkasova et al. presented an approach to measuring the power consumption of virtual CPU (vCPU), which can be considered as time slices of the physical CPU [16]. In [8], Stoess et al. presented a two-layer power managing framework for measuring and controlling the power consumption of virtualized devices. Both studies mainly focused on measuring the overall power overheads of virtualization layer, none of them can provide fine-grained VM power measuring mechanism.

To measure the power consumption on per-VM base, several VM metering techniques have been proposed. 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 various kinds of VM power modeling methodologies. By performing extensive experiments on standard benchmarks, they concluded that application's characteristics are of significant importance when modeling 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 extensively studied. In [17], Bircher et al. presented a comprehensive work on using PMCs to model power consumption in non-virtualized machines. Their experimental results strongly suggested that selecting suitable PMCs is of critical importance when building PMC-based power models. In [14], Lim et al. proposed an empirical VM power model on Intel Core-i7 platform. In this empirical model, PMCs are classified into three classes and the correlation between these counters are obtained through extensive tests on various benchmarks. In [13], 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. Their experiments demonstrated that PMC-based power models are more accurate and stable than utilization-based power models.

3 PROBLEM DESCRIPTION

In non-virtualized machine, the power model is typically formulated as

$$P(t) = P_{static} + k_{cpu}U_{cpu}(t) + k_{ram}U_{ram}(t) + k_{disk}U_{disk}(t) + k_{io}U_{io}(t), \qquad (1)$$

where P_{static} is the fixed power consumption as long as the machine is switched on, U_j is the dynamic utilization of component j, k_j is the dynamic power coefficient which is often obtained by empirical approaches. For the convenience of representation, we note the power consuming components as set $J = \{\text{CPU, RAM, Disk, I/O}\}$ in the following sections.

When a machine is virtualized, its power model can be rewritten as

$$P(t) = P_{static} + \sum_{i=1}^{M} P_i^{vm}(t), \tag{2}$$

where P_i^{vm} is the dynamic power consumption of VM_i, M is the number of active VMs on this machine. As a VM cannot be connected by hardware power meters, its actual power consumption P_i^{vm} should be measured in an indirect way. The most mentioned per-VM power model is as following

$$P_i^{vm}(t) = \frac{P_{static}}{M} + R_i \sum_{j \in J} [k_j U_j(t)], \tag{3}$$

where R_i is the processor utilization proportion that VM scheduler allocated to VM_i. The VM power model shown in Equation (3) assumes that the static power consumption is equally shared by all VMs, and the dynamic power consumption is strictly proportional to R_i, which is decided when creating the VM and obeyed by VM scheduler during its execution. To keep the power model in Equation (3) accurate, VM scheduler must satisfy the following condition.

$$\left| \frac{U_i(t_1, t_2)}{R_i} - \frac{U_j(t_1, t_2)}{R_j} \right| = 0, \quad \forall i, j \in \{1, 2, \ldots, M\}, \tag{4}$$

where $U_i(t_1, t_2)$ is the actual utilization consumed by VM_i during the time period $[t_1, t_2]$. The Equation (4) implies that VM scheduler must keep the actual utilization $U_i(t_1, t_2)$ strictly consistent with its promise R_i for all active VMs. Unfortunately, none of the current VM schedulers can satisfy this condition, because runtime characteristics of applications have significant effects on the actual utilization. For instance, considering VM_i is data-intensive and VM_j is computation-intensive, then it tends to be $U_i(t_1, t_2)/R_i < U_j(t_1, t_2)/R_j$. Therefore, the more general form of Equation (4) should be as follows:

$$\left| \frac{U_i(t_1, t_2)}{R_i} - \frac{U_j(t_1, t_2)}{R_j} \right| < \omega, \quad \forall i, j \in \{1, 2, \ldots, M\} \tag{5}$$

It is clear that the accuracy of utilization-based power model depends on the parameter ω. The bigger value of ω will result in less accuracy. That is

$$err\,\% = \left| \frac{P_{actual}^{vm} - P_{measured}^{vm}}{P_{actual}^{vm}} \right| \times 100\,\% \propto \psi \tag{6}$$

So, parameter ω can be considered as the upper bound of $err\,\%$ when using Equation (3) to model VM power consumption. To the best of our knowledge, existing VM power models can only provide empirical evaluation in term of accuracy, which means none of them can provide quantitative bounds of measuring error.

Therefore, the objective of this work is to find an approach that cannot only quantitatively evaluate ω, but also reduce it so as to improve the accuracy of VM power measurements.

4 RELATIVE PMC BASED VM SCHEDULING ALGORITHM

4.1 VM Power Model Based on Relative PMC

In this work, we use PMCs facility to log power-consuming events. Typically, these PMCs can be categorized into many classes according to their relationship to specific components, such as CPU, GPU, chipset, RAM, I/O controller and disk. In a virtualized machine, PMC events are firstly propagated between hardware components, and then they can be logged or profiled on per-VM base through VM Hypervisor or other utilities at a virtual layer. The framework of PMC-based power modeling methodology in this work is shown in Figure 1.

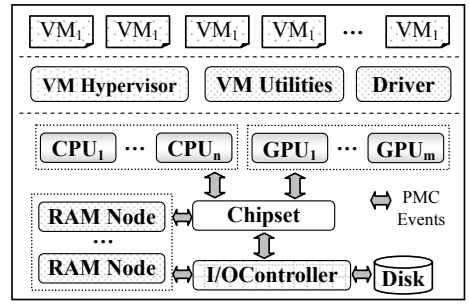


Figure 1. Framework of PMC-based power modeling

As mentioned before, there is only a subset of PMC events that are representative for modeling the device power consumption. In this work, we select {*uOps*, *Halt*, *LLC*, *TLB*, *DMA*, *FSB*, *Interrupt*} as the PMC candidates. Here, *uOps* is the counter of executed micro-operations; *Halt* is the counter of cycles in which clock gating is active; *LLC* is the counter of cache missing that occurs at the last-level cache; *TLB* is the counter of data missing that occurs in the Translation-Lookaside-Buffer; *DMA* is the counter of transactions that originated in an I/O device whose destination is system main memory; *FSB* is the counter of accessing to Front-Side-Bus; *Interrupt* is the count of interrupts received by CPU. The detailed descriptions of each PMC can be seen in [18]. To figure out the co-relation between PMCs and power consumption, a series of tests are performed by using various benchmarks on

four different kinds of servers. The summary of the results are shown as follows:

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

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

$$\begin{cases} E_{cpu}(t_1, t_2) = uOps_{t_1 \to t_2} - Halt^2_{t_1 \to t_2} \\ E_{ram}(t_1, t_2) = LLC_{t_1 \to t_2} + TLB_{t_1 \to t_2} + FSB_{t_1 \to t_2} \\ E_{disk}(t_1, t_2) = Interrupt_{t_1 \to t_2} + DMA^3_{t_1 \to t_2} \\ E_{IO}(t_1, t_2) = Interrupt_{t_1 \to t_2} + DMA_{t_1 \to t_2} \end{cases} \tag{8}$$

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

$$P^{vm}_k(t_1, t_2) = \sum_{j \in J} P^{vm}_{k,j}(t_1, t_2), \tag{9}$$

where $P^{vm}_{k,j}(t_1, t_2)$ is the power consumption of component j. Then for a given machine running multiple VMs, the dynamic power consumption of each VM is linear to their relative PMC. That is

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

where $E^k_j(t_1, t_2)$ is the part of $E_j(t_1, t_2)$ triggered by VM_k.

As shown in Equation (10), we use relative PMC (also called PMC ratio) to describe VM power model instead of absolute PMC accounts. It is well-known that mapping the absolute PMC accounts to power consumption is a notoriously difficult work, which requires not only comprehensive architecture knowledge but also long training time to obtaining sufficient accurate model [10, 14, 17]. However, by using the relative PMC, we can overcome these difficulties in a simple manner. It is noteworthy that the validity of Equation (10) relies on an assumption that all the workloads on a machine are executed on VMs. This assumption is acceptable for most modern virtualized servers in cloud datacenters.

4.2 Algorithm Implementation

As mentioned before, classical VM schedulers (i.e. Xen scheduler [19]) mainly focus on fairness in term of processor utilization. This strategy tends to underestimate VM's recursive operation, which consequently results in fact that the recursive

PMC-RPC Algorithm Implementation

1: **if** VM_i is newly created or migrated **then**
2: $H_i := R_i; L_i := 1;$
3: **end if**
4: **for all** active VM_i **do**
5: $W_i := \frac{L_i - H_i}{R_i};$
6: **end for**
7: **while** processor is available **do**
8: sort $<\mathrm{VM}_1, \mathrm{VM}_2, \ldots, \mathrm{VM}_m>$ in ascending order of H_i, and the results are stored as $<\mathrm{VM}_{k1}, \mathrm{VM}_{k2}, \ldots, \mathrm{VM}_{km}>$;
9: assign processor to VM_{k1} and set its utilization ratio as $L_{k1} - H_{k1}$;
10: **for all** un-scheduled VM_{kn} **do**
11: $H_{k_n} := H_{k_n} + \frac{L_{k_1} - H_{k_1}}{R_{k_1}} \cdot R_{k_n};$
12: $L_{k_n} := L_{k_n} + \sum_{j \in J - \{cpu\}} E_j^{k_n} \Big/ \sum_{j \in J - \{cpu\}} E_j;$
13: **end for**
14: $H_{k_1} := 0;$
15: $L_{k_1} := L_{k_1} + \sum_{j \in \{cpu\}} E_j^{k_1} \Big/ \sum_{j \in \{cpu\}} E_j;$
16: **end while**

power consumption is often ignored when building VM power model. As a result, many existing VM power management framework have to use other mechanisms to measure the non-processor power consumption. For instance, the VM power metering system proposed in [8] has to resort to VM driver to measure disk power consumption. To take into recursive power consumption, we design a heuristic VM scheduling algorithm, namely *PMC-based Scheduling with Recursive Power Compensation* (PMC-RPC), which is based on classical credit scheduling strategy and uses both the relative PMC account and utilization ratio as heuristic.

In PMC-RPC algorithm, each VM_i is associated with a credit value H_i and its initial value is set as the utilization ratio R_i. L_i is an accumulator that records the relative PMC of VM_i, and its initial value is 1. According to the algorithm implementation, a VM with lowest H_i has the highest scheduling priority. After each scheduling, the credit value of scheduled VM will be proportional shared by other VMs (as shown in step 11).

In each scheduling, all VMs will be sorted in ascending order of H_i, which is used as heuristic of recursive power consumption. More specifically, when a VM is scheduled, L_i will accumulate CPU-related PMC events (as shown in step 15). So, if the scheduled VM is computation-intensive, the increased L_i value will result in more utilization ratio $L_i - H_i$ in its next scheduling. On the other side, if the scheduled VM is data-intensive, the PMC-RPC algorithm will reduce its 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 12. So, PMC-RPC 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.

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 keeps 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 PMC-RPC implementation, and its future processor utilization ratio will be increased so as to compensate such delaying.

4.3 Algorithm Analysis and Discussion

In this section, we firstly present the accuracy that the PMC-RPC algorithm can provide.

Theorem 1. When using PMC-RPC algorithm, for any VM_i and VM_j in any time period $[t_1, t_2]$, they are satisfying

$$\omega \leq \frac{\max_{t\in[t_1,t_2]}\{H_i(t), H_i(t)\}}{\min\{R_i, R_j\}}. \tag{11}$$

Proof. Assume that the PMC-RPC algorithm has been executed k times during time interval $[t_1, t_2]$. Accordingly, the scheduling sequence during $[t_1, t_2]$ can be noted as $< \mathrm{VM}_{q1}, \mathrm{VM}_{q2}, \ldots, \mathrm{VM}_{qk} >$. For any VM_i, its $H_i(t)(t \in [t_1, t_2])$ can be noted as $H_i(n)(n \in [1, 2, \ldots, k])$. According to the PMC-RPC algorithm, for any VM_i, $H_i(k)$ after the k^{th} iteration is formulated as

$$H_i(k) = \sum_{n=1}^{k-1}\left(\frac{L_{q_n}(n) - H_{q_n}(n)}{R_{q_n}} R_i\right) + H_i(1) - U_i(t_1, t_2), \tag{12}$$

where q_n is the index of the VM being scheduled in the n^{th} iteration during $[t_1, t_2]$. So

$$\frac{U_i(t_1, t_2)}{R_i} = \frac{H_i(1) - H_i(k)}{R_i} + \sum_{n=1}^{k-1}\frac{L_{q_n}(n) - H_{q_n}(n)}{R_{q_n}}. \tag{13}$$

Therefore, we have

$$\begin{aligned}\left|\frac{U_i(t_1, t_2)}{R_i} - \frac{U_j(t_1, t_2)}{R_j}\right| &= \left|\frac{H_i(1) - H_i(k)}{R_i} - \frac{H_j(1) - H_j(k)}{R_j}\right| \\ &\leq \left|\frac{H_i(1) - H_i(k)}{R_i}\right| + \left|\frac{H_j(1) - H_j(k)}{R_j}\right|\end{aligned}$$

$$\leq \frac{\max_{n\in\{1\ldots k\}}\{H_i(n)\}}{R_i} + \frac{\max_{n\in\{1\ldots k\}}\{H_j(n)\}}{R_j} \tag{14}$$

$$\leq \frac{\max_{n\in\{1\ldots k\}}\{H_i(n), H_j(n)\}}{\min\{R_i, R_j\}}$$

$$= \frac{\max_{t\in[t_1,t_2]}\{H_i(t), H_j(t)\}}{\min\{R_i, R_j\}}.$$

□

According to Theorem 1, it is clear that the error of utilization-based VM power model is bounded when using PMC-RPC algorithm. Such an error-bound is related to utilization ratio R_i and scheduling rank $H_i(t)$ of VMs. Based on the conclusion of Theorem 1, we can easily obtain the following theorem about PMC-RPC algorithm.

Theorem 2. If $R_1 = R_2 = \ldots = R_n$, then the overall error bound of per-VM power measurements can be minimized when using PMC-RPC algorithm.

Proof. Due to $\sum R_i = 1$, it is clear that $\forall i, j\ min\{R_i, R_j\}$ will be reduced if $R_1 = R_2 = \ldots = R_n$. By Theorem 1 and Equation (6), the proof is completed. □

Theorem 2 indicates that if all VMs are configured to equally share processor capability, PMC-RPC algorithm can improve the accuracy of per-VM power measurements. This feature is especially useful for those virtualized platforms, which use fair allocation strategy for VM provision and consolidation.

Next, we present the scheduling strategy of PMC-RPC algorithm by following analysis.

Theorem 3. In PMC-RPC algorithm, if VM_i is scheduled, it must satisfying

$$\min\left\{\frac{1}{R_i}\cdot\frac{E_J^i(t_1,t_2)}{E_J(t_1,t_2)}\right\},$$

where $[t_1, t_2]$ is the time interval between two successive scheduling of VM_i.

Proof. According to the PMC-RPC algorithm, we have the following recursive formula:

$$L_i(k) = L_i(k-1) + \frac{E^i_{J-\{cpu\}}(k-1)}{E_{J-\{cpu\}}(k-1)}, \tag{15}$$

where $L_i(k)$ is the relative PMC account at the k^{th} scheduling, $E^i_{J-\{cpu\}}(k-1)$ is the non-processor PMC account triggered by VM_i between the $(k-1)^{\text{th}}$ and k^{th}, and $E_{J-\{cpu\}}(k-1)$ is the total non-processor PMC account of the sever. By recursively deducting Equation (15), we can obtain

$$L_i(k) = L_i(1) + \sum_{n=1}^{k-1}\frac{E^i_{J-\{cpu\}}(n)}{E_{J-\{cpu\}}(n)}, \tag{16}$$

where $L_i(1)$ is the relative PMC account of VM_i at time t_1. As VM_i has the processor before time t_1, $L_i(1)$ is the processor-related relative PMC value of VM_i during its last execution. That is

$$L_i(1) = \frac{E^i_{\{cpu\}}(1)}{E_{\{cpu\}}(1)}. \tag{17}$$

By Equation (17), we can rewrite Equation (16) as

$$L_i(k) = \frac{E^i_{\{cpu\}}(1)}{E_{\{cpu\}}(1)} + \sum_{n=1}^{k-1} \frac{E^i_{J-\{cpu\}}(n)}{E_{J-\{cpu\}}(n)} = \sum_{n=1}^{k-1} \frac{E^i_J(n)}{E_J(n)}. \tag{18}$$

As VM_i is not scheduled during $[t_1, t_2)$, then $U_i(t_1, t_2) = 0$ and $R_i(1) = 0$. By Equation (17) and Equation (18), it has

$$L_i(k) - H_i(k) = \sum_{n=1}^{k-1} \frac{E^i_J(n)}{E_J(n)} - \sum_{n=1}^{k-1} \frac{L_{q_n}(n) - H_{q_n}(n)}{R_{q_n}} R_i. \tag{19}$$

Dividing both sides with R_i, we further have

$$\frac{L_i(k) - H_i(k)}{R_i} = \frac{1}{R_i} \sum_{n=1}^{k-1} \frac{E^i_J(n)}{E_J(n)} - \sum_{n=1}^{k-1} \frac{L_{q_n}(n) - H_{q_n}(n)}{R_{q_n}}. \tag{20}$$

As $< VM_{q_1}, VM_{q_2}, \ldots, VM_{q_k} >$ is the scheduling sequence that happened during $[t_1, t_2)$, so at time t_2, $\sum_{n=1}^{k-1} \left(L_{q_n}(n) - H_{q_n}(n)\right) / R_{q_n}$ is identical for all VMs. Therefore, the scheduling strategy of PMC-RPC algorithm can be noted as

$$\min \left\{ \frac{1}{R_i} \sum_{n=1}^{k-1} \frac{E^i_J(n)}{E_J(n)} \right\}. \tag{21}$$

The Equation (21) can be rewritten in time manner, which is shown in the original theorem. □

Theorem 4. If $R_1 = R_2 = \ldots = R_n$, then the scheduling strategy of PMC-RPC algorithm is *Recent Lowest Power First*.

Proof. According to Theorem 3, $E^i_J(t_1, t_2)/E_J(t_1, t_2)$ is the most recent relative PMC of VM_i. As shown in Equation (10), the power consumption of VM_i will be linear to $E^i_J(t_1, t_2)/E_J(t_1, t_2)$, so the VM with the recently lowest power consumption will be selected by PMC-RPC algorithm. □

5 PERFORMANCE EVALUATION IN EXPERIMENTS

5.1 Experimental Settings

In our experiments, two kinds of platforms are used with aiming to comparing the performance on different architectures. The first platform is Intel Xeon E5606 server

machine with four 2.13 GHz cores, 8 M LLC cache, 16 G memory, and 2.0 T SATA hard disk. The second platform is Pentium D830 desktop-PC, with of two 3.0 GHz cores, 2 M LLC cache, 2 G memory, and 160 G IDE hard disk. In both platforms, the VM hypervisor is Xen version 4.1.2 [21] and operation system is Linux with kernel version 2.6.2. In the experiments, we use Oprofile [20] to log and statistic PMC events, and the original reports produced by Oprofile are categorized for individual VMs.

Five benchmarks are used as basic workloads, including bzip2 [22], mcf [22], TPC-W [23], Cachebench [24], and IOZone [25]. In these benchmarks, bzip2 and mcf come from SPECcpu2006 benchmark suite; TPC-W is a representative benchmark for testing the performance of web servers; Cachebench is designed for measuring the performance of cache subsystem; IOZone is a file system benchmark that generates variety of file system operations.

5.2 Comparison of Experimental Performance

To investigate the accuracy of VM power measurements, we need to obtain the power baseline of each benchmark at first. Therefore, five benchmarks are executed on two platforms one by one. As the benchmark is the only VM on the platform, the actual power consumption of each benchmark can be approximately estimated as the baseline power consumption, which is calculated by Equation (3) with $M = 1$. Since the power consumptions of individual VMs will dynamically change during their runtime period, we sample the CPU's utilization in each second for measuring the VM's power consumption, and the statistical results of each benchmark are shown in Figure 2.

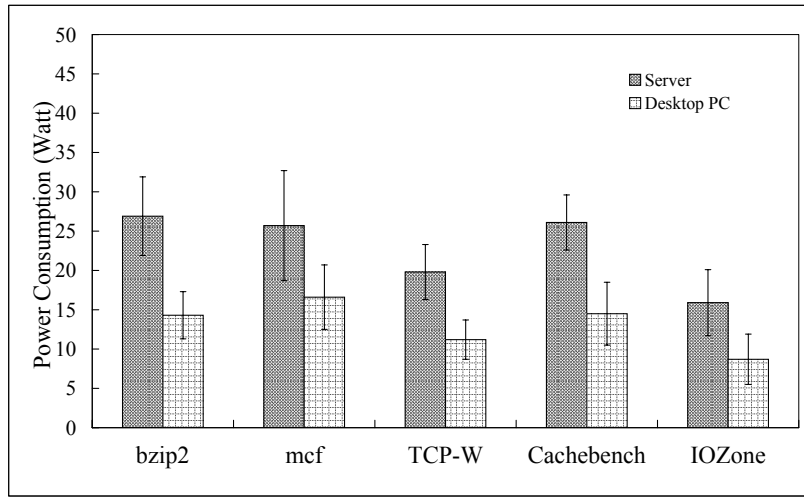


Figure 2. Power baseline of experimental benchmarks

In the following experiments, multiple VMs will be executed concurrently, and the error of VM power measurements are calculated by Equation (6). We use the above baseline (shown in Figure 2) to approximate the actual power consumption. It is noteworthy that VM power consumption can be measured by Equation (3) or Equation (10). If using Equation (3), VM power is measured by conventional utilization-based model (UM); if using Equation (10), it is measured by relative PMC model proposed in this paper. Our experiments will test both power models and compare them. As to VM scheduling algorithm, we compare PMC-RPC algorithm with the Xen's default VM scheduling algorithm [19], which uses utilization ratio as scheduling strategy (URS). Therefore, the experimental results can be categorized into following groups:

- **URS + UM**: using Xen's default scheduler and utilization-based power model.
- **URS + rPMC**: using Xen's default scheduler and relative PMC based power model.
- **PMC-RPC + UM**: using PMC-RPC algorithm and utilization-based power model.
- **PMC-RPC + rPMC**: using PMC-RPC algorithm and relative PMC based power model.

In the experiments, we set all VMs equally sharing processor capabilities, that is $R_1 = R_2 = R_3 = R_4 = R_5 = 20\,\%$. According to Theorem 2, such fairness allocation strategy will lower down the error bound of power measurements when using PMC-RPC algorithm. The experiments are categorized as four groups, each using one of the above four combinations as VM scheduling policy and power consumption model. In each experimental group, the experiment is conducted two times on different platforms and the results are compared with the baseline results (as shown in Figure 2) to calculate the error of power consumption. The experimental results are shown in Figures 3 to 6.

It is clear that the most distinguishing result is that the error of VM power measurements 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 URS + UM technique. It is because those VMs are often blocked by VM hypervisor when performing massive I/O operations, however the URS + UM technique cannot take such recursive power into account. When using rPMC model, 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. For instance, the error of TPC-W is reduced from 11.9 % to 5.4 %, and the improvement on IOZone is more significant.

The other experimental result is that the error on desktop platform is higher than that on server platform in most cases. Two exceptions happen on mcf and IOZone benchmarks. For the mcf, we notice that it is mostly the cpu-intensive in all benchmarks, while IOZone is the most disk-intensive benchmark. When running the

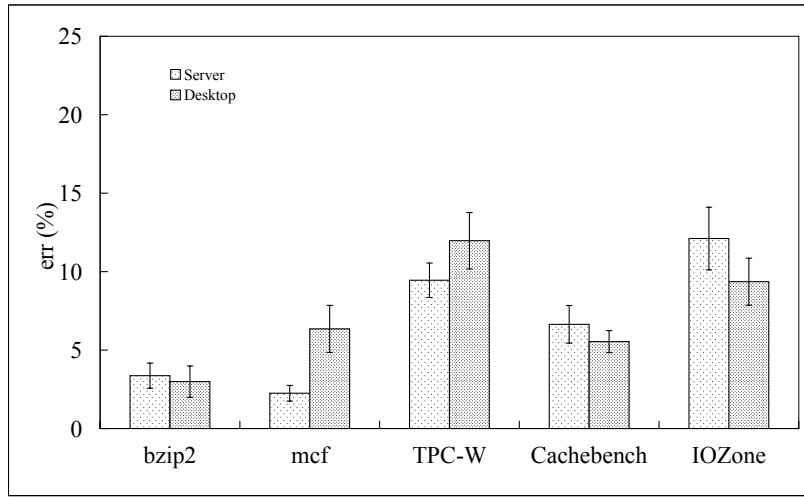


Figure 3. URS + UM

benchmarks on desktop platform, only two VMs can be concurrently 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 consumption into the current VMs, which make its error very high. While on the server platform, there are four cores which allow four VMs

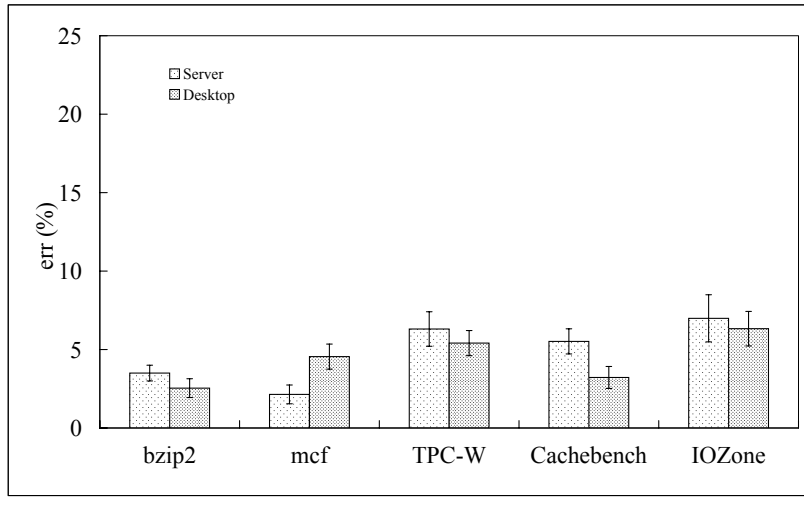


Figure 4. URS + rPMC

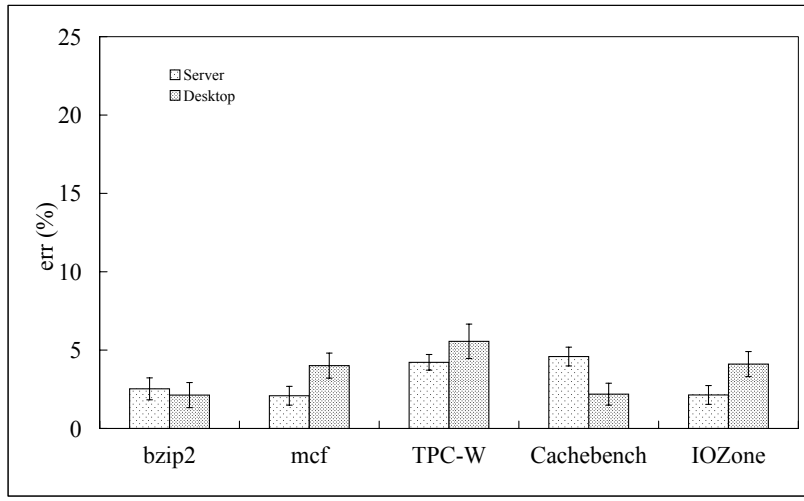


Figure 5. PMC-RPC + UM

concurrently running at most. Therefore, this significantly balances the measuring error when using UM power model. While using rPMC model, this increased error can be reduced by about 25 % for mcf benchmark and 30 % for IOZone benchmark.

A most interesting result happens on the IOZone benchmark running on the server platform, when we use PMC-RPC scheduling algorithm and utilization-based measuring model. Its error is dramatically reduced from 12.11 % to 2.14 %. To

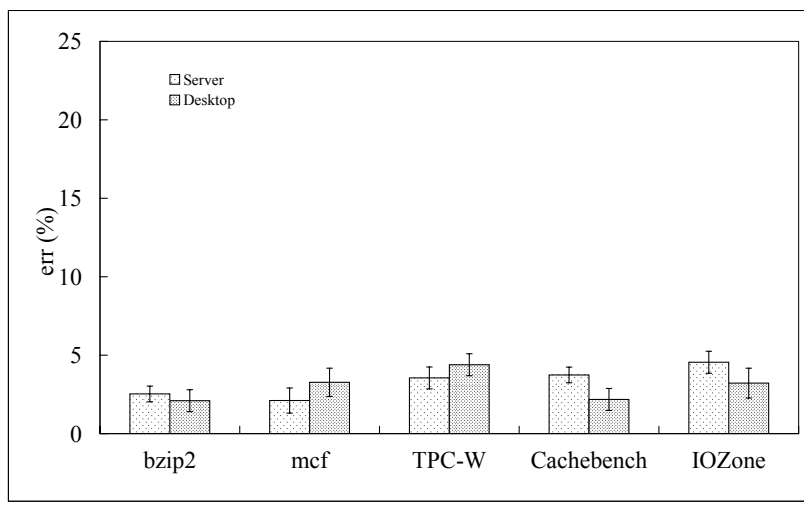


Figure 6. PMC-RPC + rPMC

figure out the reason, we check the intermediate data collected during experiments. The data show that the IOZone benchmark requires almost no processor during all its execution. So, UM based technique results in very high error, which is especially worse on server platform. Although rPMC model 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 PMC-RPC + UM (shown in Figure 5), as the PMC-RPC algorithm tends to select the VM with least power consumption at recent duration (as shown in Theorem 3 and Theorem 4), the IOZone benchmark 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 model can accurately measure the power consumption of disk device. So, we obtain the most accurate power measurements in this case.

By the above experimental results and analysis, we draw the following conclusions:

1. Conventional utilization-based power model is only suitable for pure computation-intensive workloads (i.e. bzip2), because the VM scheduler can strictly obey its scheduling principle only in this case;

2. The proposed PMC-base power model is effective to reduce the error of power consumption measurements, because it enables to mitigate the negative effects caused by recursive power consumption especially for data-intensive workloads;

3. By combing the PMC-RPC algorithm with PMC-based power model, we cannot only reduce the power measuring error caused by conventional scheduler but also mitigate the negative effects of data-intensive workloads on power measuring.

6 CONCLUSIONS AND FUTURE WORKS

In this work, we take efforts on the issue of VM power measuring technology. By introducing the concept of relative PMC power model, a novel VM scheduling algorithm called PMC-RPC is proposed, which uses relative PMC information to compensate the recursive power consumption aiming to improve the accuracy of power measurements. Theoretical analysis indicates that the algorithm can provide quantitative error bound for VM power measuring. The experimental results obtained from various benchmarks show that the practical error of power measuring can be significantly reduced when using PMC-RPC algorithm. In the future, we plan to enhance the PMC-RPC algorithm with QoS-aware capability, because plenty of non-trivial applications have required cloud platforms providing better QoS performance, especially those user-oriented QoS measurements including price, security, reliability and etc.

Acknowledgements

This work was supported by a grant from the National Natural Science Foundation of China (No. 61402163). Also, it is supported by the Provincial Science & Technology Plan Project of Hunan (No. 2012GK3075).

REFERENCES

[1] Rings, T.—Caryer, G.—Gallop, J.—Grabowski, J.—Kovacikova, T.—Schulz, S.—Stokes-Ree, I.: Grid and Cloud Computing: Opportunities for Integration with the Next Generation Network. Journal of Grid Computing, Vol. 7, 2009, No. 3, pp. 375–393.

[2] Jang, W.—Jeon, M.—Kim, H. S. et al.: Energy Reduction in Consolidated Servers Through Memory-Aware Virtual Machine Scheduling. IEEE Transaction on Computers, Vol. 60, 2011, Vol. 4, pp. 552–546.

[3] Beloglazov, A.—Abawajyb, J.—Buyya, R.: Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing. Future Generation Computer Systems, Vol. 28, 2012, No. 5, pp. 755–768.

[4] Liao, X.—Jin, H.—Liu, H.: Towards a Green Cluster Through Dynamic Remapping of Virtual Machines. Future Generation Computer Systems, Vol. 28, 2012, No. 2, pp. 469–477.

[5] Dhiman, G.—Marchetti, G.—Rosing, T.: vGreen: A System for Energy-Efficient Management of Virtual Machines. ACM Transaction on Design Auto Electronic Systems, Vol. 16, 2010, No. 1, pp. 1–27.

[6] Liao, X.—Hu, L.—Jin, H.: Energy Optimization Schemes in Cluster with Virtual Machines. Cluster Computering, Vol. 13, 2010, No. 2, pp. 113–126.

[7] Wang, Y.—Wang, X.—Chen, M.—Zhu, X.: PARTIC: Power-Aware Response Time Control for Virtualized web Servers. IEEE Transaction on Parallel and Distributed Systems, Vol. 11, 2011, No. 2, pp. 323–336.

[8] Stoess, J.—Lang, C.—Bellosa, F.: Energy Management for Hypervisor-Based Virtual Machines. Proceedings of USENIX Annual Technique Conference, USENIX Association, 2007, pp. 1–14.

[9] Kansal, A.—Zhao, F.—Liu, J.: Virtual Machine Power Metering and Provisioning. Proceedings of ACM Symposium on Cloud Computing, ACM Press, 2010, pp. 39–50.

[10] Koller, R.—Verma, A.—Neogi, A.: WattApp: An Application Aware Power Meter for Shared Data Centers. Proceedings of International Conference on Automatic Computing, IEEE Press, 2010, pp. 31–40.

[11] Bohra, A.—Chaudhary, V.: Vmeter: Power Modelling for Virtualized Clouds. Processing of International Parallel and Distributed Processing Symposium, IEEE Press, 2010, pp. 1–8.

[12] Krishnan, B.—Amur, H.—Gavrilovska, A. et al.: VM Power Metering: Feasibility and Challenges. ACM SIGMETRICS Performance Evaluation Review, Vol. 38, 2010, No. 3, pp. 56–60.

[13] Bertran, R.—Becerra, Y.—Carrera, D. et al.: Energy Accounting for Shared Virtualized Environments under DVFS Using PMC-Based Power Models. Future Generation Computer Systems, Vol. 28, 2012, No. 2, pp. 457–468.

[14] Lim, M. Y.—Porterfield, A.—Fowler, R.: SoftPower: Fine-Grain Power Estimations Using Performance Counters. Proceedings of International Symposium on High Performance Distributed Computing, ACM Press, 2010, pp. 308–311.

[15] Berl, A.—Meer, H.: An Energy Consumption Model for Virtualized Office Environments. Future Generation Computer Systems, Vol. 27, 2011, No. 8, pp. 1047–1055.

[16] Cherkasova, L.—Gardner, R.: Measuring CPU Overhead for I/O Processing in the Xen Virtual Machine Monitor. USENIX Annual Technique Conference, USENIX Association, 2005, pp. 387–390.

[17] Bircher, W. L.—John, L. K.: Complete System Power Estimation Using Processor Performance Events. IEEE Transactions on Computers, Vol. 60, 2012, No. 4, pp. 563–577.

[18] Bertran, R.—González, M.—Martorell, X.: Decomposable and Responsive Power Models for Multicore Processors Using Performance Counters. Proceedings of International Conference on Supercomputing, ACM Press, 2010, pp. 147–158.

[19] Cherkasova, L.—Gupta, D.—Vahdat, A.: Comparison of the Three CPU Schedulers in Xen. ACM SIGMETRICS Performance Evaluation Review, Vol. 35, 2007, No. 2, pp. 42–51.

[20] Oprofile. `http://oprofile.sourceforge.net`.

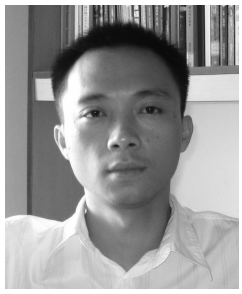
[21] Xen. `http://www.xen.org/products/xen_source.html`.

[22] SPEC CPU2006. `http://www.spec.org/cpu2006/`.

[23] TPC-W. `http://www.tpc.org/tpcw/default.asp/`.

[24] Cachebench. `http://icl.cs.utk.edu/projects/llcbench/cachebench.html/`.

[25] IOZone. `http://www.iozone.org/`.



Peng Xiao received his Ph.D. degree in computer science from the Central South University (China) in 2009. He is currently Associate Professor in the Hunan Institute of Engineering. Also, he works as a senior network engineer in HP High-Performance Network Center in Hunan Province. His research interests include cloud computing, resource virtualization technology, HPC energy efficiency management, large-scale workflow management. He is a member of ACM, IEEE, and IEEE Computer Society.

Dong-Bo Liu received his Ph.D. degree in computer science from the Hunan University (China) in 2011. He is currently Associate Professor in the Hunan Institute of Engineering. Currently, he is doing his postdoctoral research in IBM Mobile Communication Center in China. His research interests include grid computing, cloud computing, green computing, multi-agent intelligent systems, mobile communication protocol, and embedded system. He is a member of ACM, IEEE, IEEE Computer Society, and IEEE Communication Society.