Technical Report UPC-DAC-RR-CAP-18
Accurate Energy Accounting for Shared Virtualized Environments using PMC-based Power Modeling Techniques

Ramon Bertran[†1] Yolanda Becerra[*†] David Carrera[*†] Vicenç Beltran[†] Marc Gonzalez[*†]
Xavier Martorell[†] Jordi Torres[†] Eduard Ayguade[†]

[*]*Departament d'Arquitectura de Computadors*
Universitat Politècnica de Catalunya
Barcelona, Spain
Email:{yolandab,dcarrera,marc,xavim,torres,eduard}@ac.upc.edu

[†]*Barcelona Supercomputing Center*
Barcelona, Spain
Email:{ramon.bertran,vincenc.beltran}@bsc.es

***Abstract*—Virtualized infrastructure providers demand new methods to increase the accuracy of the accounting models used to charge their customers. Future data centers will be composed of many-core systems that will host a large number of virtual machines (VMs) each. While resource utilization accounting can be achieved with existing system tools, energy accounting is a complex task when per-VM granularity is the goal.**

In this paper, we propose a methodology that brings new opportunities to energy accounting by adding an unprecedented degree of accuracy on the per-VM measurements. We present a system –which leverages CPU and memory power models based in performance monitoring counters (PMCs)– to perform energy accounting in virtualized systems. The contribution of this paper is twofold. First, we show that PMC-based power modeling methods are still valid on virtualized environments. And second, we introduce a novel methodology for accounting of energy consumption in virtualized systems. In overall, the results for an Intel® Core™ 2 Duo show errors in energy estimations below the 5%. Such approach brings flexibility to the chargeback models used by service and infrastructure providers. For instance, we show that VMs executed during the same amount of time, present more than 20% differences in energy consumption even only taking into account the consumption of the CPU and the memory.

***Keywords*-Energy accounting; Performance Counters; Power Modeling; Virtualization;**

I. Introduction

Advanced service data centers leverage virtualization technologies to run applications from multiple customers in shared environments. Initiatives such as the Amazon EC2 platform have brought to reality the creation of large pools of resources where end users deploy applications and services. Infrastructure providers manage user applications transparently thanks to virtualization operations, and due to the benefits of server and workload consolidation, they offer affordable and virtually unlimited computing resources to bulk customers.

Power and energy consumption plays an important role in the maintenance cost of a large data center, including power consumption from processors, memories, storage and networking resources, and also from other costly infrastructures, such as AC facilities. Infrastructure providers decide their pricing policies based on the costs of the infrastructure, and charge customers based on fixed costs ratios as well as following the usage that they do of the computing resources.

Therefore, detailed measurement of resource consumption is a critical point for shared data centers. Accounting of resource utilization must be done in a per-virtual machine (VM) basis, even when multiple VMs are deployed on top of the same physical hardware. While measuring CPU, storage and network utilization is feasible with existing system tools, per-VM energy consumption can be hardly estimated in resource sharing environments. Power metering devices measure aggregate power consumption metrics for a whole system, but detailed per-VM metrics can only be measured with advanced techniques. The new many-core systems will take this limitation even further, envisioning environments in which large number of VMs will be deployed on top of tenths of cores in a single server.

There are several techniques –working at different levels– addressing power consumption issues. All of them rely on accurate methods to gather information about power consumption. Specifically, for software-based solutions the need to estimate and predict power behavior has justified the research on power modeling strategies. Methods based on performance monitoring counters (PMCs) have been shown to be a good solution to estimate power consumption. As a result, their applicability has been demonstrated on several fields such as power management and application profiling. They are used to perform live predictions of power behavior in order to guide power aware policies [1, 2, 3]. Moreover, they are also used in research for quickly exploring new approaches since they allow to profile real systems and full executions of applications, avoiding the need to perform long-time and limited simulations [4, 5, 6]. In the end, they have been crucial in the process of addressing power issues.

In this paper we join the virtualization technology with

[1]*Corresponding author. Email: ramon.bertran@bsc.es*

power model techniques to derive energy consumption estimates at per-VM level. The major contribution of this paper is a novel methodology that provides accurate model-based per-VM energy accounting on current multicore processors. The technique leverages the power modeling methodology introduced in [7] to derive per-VM power consumption. The approach is able to estimate the energy consumption of each VM in the system, even in the case that multiple VMs are multiplexed on top of the same physical core. To the best of our knowledge, we are the first that validate that PMC-based power models are valid to perform energy accounting on virtualized environments, showing errors below the 5%. Moreover, we demonstrate the applicability of the proposal showing examples of VMs that consumed the same amount of CPU time, but present around 20% variations in energy consumption. As a result, this approach provides accuracy to the chargeback models used by service and infrastructure providers.

The rest of this paper is organized as follows: Section II describes the power modeling and energy accounting methodology. Section III validates the methodology in virtualized systems and demonstrates how this technology can be used to derive energy consumption estimates. Section IV describes the related work. Finally, section V concludes the paper.

II. Methodology

In this section we describe the experimental environment used for the experimentation. Then, the mechanism used for gathering the power and the PMC traces of the workloads is explained and the details of the virtualization environment are presented. Finally, we describe the technique used to produce the power models from the gathered traces and how we perform the per-VM energy accounting. It should be noted that even though the study presented is for a particular architecture, the same approach can be applicable to other architectures.

A. Experimental environment

All the experiments have been carried out on a workstation that features an Intel® Core™ 2 Duo T9300 processor [8] and two modules of 1GB of memory. The embedded controller firmware of the platform provides information about the power source, which fulfills the SMAPI specification [9]. This specification defines an interface to obtain power consumption measurements with a guaranteed granularity and accuracy. As a result, we gather power measurements in a granularity of milliwatts with a maximum error of 2%.

The platform runs a Linux kernel 2.6.28 with virtualization support enabled (KVM) [10] and the required patches to allow PMC readings. The `tp_smapi` [11] module is loaded in order to be able to gather the power information. This module creates some entries in the `/sys` filesystem that provide information such as the current power consumption. We configure the system to avoid power interferences –e.g. minimizing the number of running processes, switching off peripherals, ...–.

B. Power metering and data gathering

We use a modified version of `pfmon` [12] to access to the PMCs and power consumption simultaneously. This tool allows to perform system-wide or per-process PMC readings. For validating the model, we configure `pfmon` in system-wide mode, retrieving PMCs counters and current power consumption every 2 seconds. Whereas for performing per-VM accounting, we use the per-process mode (specifying the PID[1] to follow), retrieving PMCs counters and last minute average power consumption[2] every 60 seconds. In both cases, at the end of the experiments we obtain two traces of information: one contains the PMC readings and the other one provides the trace of the real power consumption. PMC-traces are processed offline in order to perform the power estimations. Then, we compare them against the real power consumption readings during the validation in Section III.

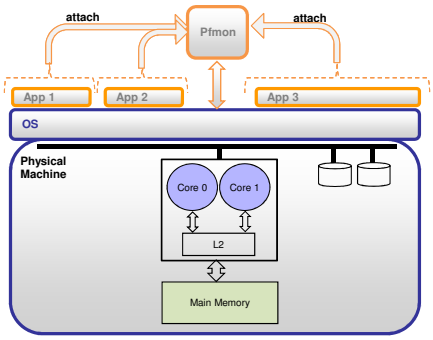
C. Virtualization environment

We run the experiments on a non virtualized environment and into a virtualized one. Fig. 1(a) describes the non virtualized environment. In this case, the applications run on the native operating system (OS) and `pfmon` gathers information about their activity in the processor. In contrast to Fig. 1(a), the virtualized environment introduces one layer of virtualization represented by the VMM (Virtual Machine Monitor). Fig. 1(b) shows this scenario where two VMs coexist, and within every VM two applications are running on top of virtual hardware. In this case, `pfmon` gathers information about the activity in the VMs. One of the main objectives of this paper is to confirm that PMC-based power models are valid in this scenario. Then, they can be used as a mechanism to account per-VM energy consumption.

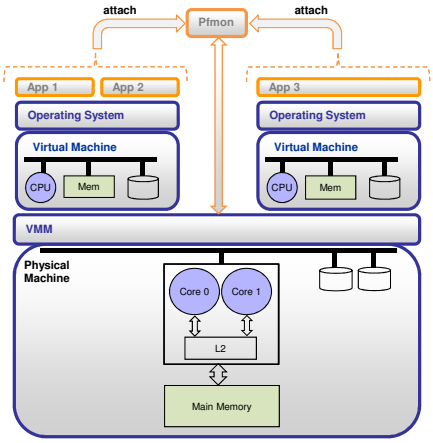
In particular, our virtualization system uses QEMU 0.10.0 [13] and KVM[10]. QEMU is a generic and open source machine emulator and virtualizer. When used as a virtualizer achieves near native performances by executing the guest code directly on the host CPU. However, for that purpose it needs KVM. KVM (Kernel Virtual Machine) is a Linux kernel module that allows a user-space program to utilize the hardware virtualization features of the processor. We use libvirt 0.6.1 [14] to interact with the virtualization capabilities of our host system and to manage the VMs. We set up a set of VMs which feature a Linux kernel 2.6.28 and 512MB of memory. The number of virtual cores per VM is one or two depending on the experiment performed.

[1]PID stands for 'Process IDentifier', which is a number used by the operating system to uniquely identify a process.

[2]Provided by the `tp_smapi` module through an entry in the`/sys` filesystem.



(a)



(b)

Figure 1. Non Virtualized and Virtualized system for gathering data.

D. Power modeling

Our methodology for producing the power model follows the well-known modeling steps performed in previous works [4, 7]. Concretely, we followed the same methodology already validated in [7]. The methodology ensures the generation of responsive, accurate, decomposable and benchmark-independent PMC-based power models. The outcome of that methodology is a formula that predicts power consumption of the CPU and the memory from a set of performance monitoring counters. The four main steps performed are the following:

- Define power model inputs from microarchitectural components. They are PMC-based formulas that represent ratios of activity of microarchitectural components.
- Design a specific set of microbenchmarks that stress each of the inputs defined for a wide set of situations.
- Execute the microbenchmarks in order to gather the required data to train the model.
- Produce the model by applying incrementally linear regression techniques on the training data for deriving the weight in power consumption of each power component defined.

At the end the model obtained for our architecture is the following:

$$\begin{aligned} Power &= 789\times AR_{Fe} + 261\times AR_{Int} + 1908\times AR_{Bpu} \\ &+ 502\times AR_{Fp} + 856\times AR_{L1} + 24437\times AR_{L2} \\ &+ 8852\times AR_{Mem} + \begin{cases} 8701 & \text{if 1 core enabled} \\ 6272 & \text{if 2 cores enabled} \end{cases} \end{aligned}$$

where each AR_i is a PMC-based formula accounting for the activity ratio of the i component. For instance, the AR_{L1} is the activity ratio of the first level cache, and its translation to a PMC-based formula is the number of L1 accesses per cycle [3]. For sake of simplicity and due to space limitations, we only summarized the main steps followed. Please refer to the related work for a more detailed description of the power components and the model generation.

E. Per Virtual Machine Energy accounting

In order to perform per-VM energy accounting, we profit from the abstraction provided by the OS in our experimental environment. From the host OS, each VM is seen as an independent process. Moreover, each of the processes within a given VM is seen as a thread of the VM process. In the end, this abstraction in conjunction with the capability of `pfmon` to gather per process[4] PMCs allows us to track the activity of each VM running on the host OS. Afterwards, we estimate their power consumption using the model produced and then, we derive the energy by multiplying the power by the execution time.

However, when performing per-VM accounting, it is not possible to validate the estimations against empirical data because the platform only provides aggregated power readings. Hence, for validation purposes, we assume that the sum of the predicted energy consumption of each virtual machine running on the system should be the same as the overall platform energy consumption. In Section III-B we validate this assumption.

III. Validation and Evaluation

The model and methodology evaluation is organized in three main sections. The first one covers the power model validation in virtualized systems. First, we validate the single-core model and then, we validate the dual-core model. In both cases, we use accuracy –the average error in power predictions– to validate the power model. We see that the power model is valid for both scenarios, showing the fact that the virtualization layer does not affect the model accuracy since the VMM does not interfere with the virtual domains too often. In the second part of this section, we validate the energy accounting methodology presented in

[3]The rest of activity ratios represent the following: AR_{Fe}: Front-End, AR_{Int}: Integer units, AR_{Bpu}: Branch predictor unit, AR_{Fp}: Floating point units, AR_{L2}: level 2 cache and AR_{Mem}: main memory.

[4]In fact, `pfmon` generates one trace for the process and one for each of its children processes/threads.

Table I
SUMMARY OF THE SINGLE-CORE MODEL RESULTS.

	VIRTUALIZED			NON VIRTUALIZED			
Benchmark	Power (mW)	%error	%std	Power (mW)	%error	%std	Power Difference
401.bzip2	11329,77	1,55	2,62	11598,69	1,71	2,28	-2,37%
403.gcc	10858,97	3,83	20,81	11255,01	3,34	3,54	-3,65%
416.gamess	11668,49	1,1	1,44	11875,83	2,31	1,82	-1,78%
435.gromacs	9974,62	6,59	2,98	10169,89	3,98	2,75	-1,96%
436.cactusADM	10582,61	3,75	2,96	10903,2	1,49	2,35	-3,03%
437.leslie3d	10908,12	3,64	1,32	11124,86	2,11	0,96	-1,99%
444.namd	11643,84	2,72	2,06	11873,42	4,8	1,86	-1,97%
445.gobmk	11781,7	1,18	1,99	12124,89	4,11	1,79	-2,91%
450.soplex	10866,13	3,77	1,25	11005,27	2,66	1,07	-1,28%
453.povray	11579,81	0,3	1,89	10899,88	5,65	0,97	5,87%
454.calculix	12432,47	1,17	3,09	11725,84	4,02	3,17	5,68%
456.hmmer	11829,63	1,14	1,74	12127,58	3,7	1,6	-2,52%
458.sjeng	11811,84	2,5	1,23	12121,78	4,84	1,61	-2,62%
464.h264ref	10603,2	6,05	2,63	11855,47	0,34	1,36	-11,81%
465.tonto	11447,38	1,2	2,52	10717,01	6,38	3,48	6,38%
470.lbm	13276,27	5,42	2,05	13431,23	6,19	1,51	-1,17%
471.omnetpp	11517,52	0,34	1,45	11697,07	1,6	1,41	-1,56%
473.astar	11635,48	1,1	2,33	12045,38	3,78	2,34	-3,52%
482.sphinx3	10976,9	2,02	2,24	11275,01	1,62	2,03	-2,72%

Section II-E. We see that the aggregated energy accounted for each VM present on the system matches with the overall platform energy consumption. As a result, we validate one of the contributions of this paper: the proof that power models allow for energy accounting at process level, which in virtualized environments translate to energy accounting at VM level. The last part of this section discusses an applicability case. We show that VMs that run during the same amount of CPU cycles do not consume the same amount of energy. Hence, this methodology can be used to introduce new chargeback models that take into consideration fine grained energy consumption measurements.

A. Power model validation

This section addresses the validation of the power model in virtualized environments. For this purpose, we will compare the average error of the model in both virtualized and non virtualized systems. In the non virtualized system, applications run on top of the native OS and hardware whereas in virtualized tests, applications run within a VM. All the applications used are part of the `SPECcpu2006`[15] benchmark suite, which stress the system's processor and the memory subsystem.

1) One core validation: During this validation, only one core of the Intel® Core™ 2 Duo is enabled and the VM is configured also with a single virtual CPU. Table I, shows the average power consumption for each tested application in both virtualized and non virtualized systems. Columns labeled '*%error*' and '*%std*' correspond to the average percentage error and standard deviation that the model incurs in both environments. In general, for both cases the average error never exceeds a 7% of error. Graphically, this can be observed in Fig. 2. In this figure we compare the model average error for every application in virtualized and non virtualized systems. The *y* axis is the percentage of error of the estimation, which corresponds to the third and fifth columns in Table I. The differences are around 3 points of percentage, unless for specific cases like the `453.povray`, the `464.h264` and the `465.tonto`, where there is a difference around 6 points of percentage.

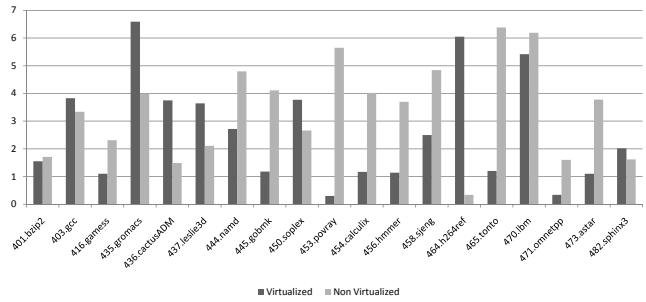


Figure 2. Comparison of average percentage error of the estimations for the single-core model in virtualized and non virtualized environments.

Table II
SUMMARY OF THE DUAL-CORE MODEL RESULTS.

	VIRTUALIZED			NON VIRTUALIZED			
Benchmark	Power (mW)	%error	%std	Power (mW)	%error	%std	Difference
400.perlbench-453.povray	17697,95	0,41	1,4	17755,29	0,46	1,19	-0,32%
400.perlbench-454.calculix	18379,85	1,22	1,78	18506,85	0,68	1,81	-0,69%
462.libquantum-436.cactusADM	14863,66	3,06	2,23	14902,06	2,51	2,34	-0,26%
462.libquantum-453.povray	15974,55	1,53	1,88	15987,36	1,2	1,9	-0,08%
462.libquantum-454.calculix	16467,04	1,8	2,47	15574,43	6,69	3,66	5,42%
473.astar-453.povray	17488,31	1,42	1,95	17442,28	1,39	1,61	0,26%
473.astar-454.calculix	18221,76	1,7	2,13	17196,07	3,72	2,54	5,63%

We have also analyzed the differences in the power consumption between the virtualized and non virtualized systems (last column in Table I). In general, the differences are below 7%, which in actual power corresponds to less than 1W. We observe a trend that indicates that in the virtualized system less power is consumed. Only three applications require more power (`465.tonto`,`453.povray` and `454.calculix`). The slight decrement in power consumption is related to eventual interferences of the VM with the actual execution of the application running within it.

In conclusion, the power model generated for one core is validated in both virtualized and non virtualized systems. The layer corresponding to the VM that encapsulates the test application does not introduce any significant distortion over the power predictions obtained through the model.

2) Two Core Validation: This section addresses the validation of the power model for the entire chip in virtualized environments. Similarly as in the previous section, we will compare the average error of the model in both virtualized and non virtualized systems. In the non virtualized system, applications run on top of the native OS and hardware, where two cores are active. In virtualized tests, applications run within one dual-core VM, that runs as a process on top the native OS and hardware.

We have selected a subset of 6 application of the `SPECcpu2006` benchmarks to evaluate the power model. The applications have been chosen according to the classification and characterization of the `SPECcpu2006` for multiprogramming evaluation [16, 17]. We paired them in order to be executed concurrently in both scenarios –virtualized and non-virtualized–. In both cases no process pinning or NUMA control is activated.

Table II describes the data obtained in both scenarios, similarly as in the case with just one core being active. Both environments expose similar average errors and power

consumption in every case. The [0.40-3.06]% range of error demonstrates the reliability of the power model for both environments –single-core and dual-core–.

In conclusion, the power model generated for the entire chip is validated in both the virtualized and the non virtualized systems. The layer corresponding to the VM that encapsulates the test application does not introduce any significant distortion over the power predictions obtained through the model. This validates one of the main contributions of this paper: PMC-based power models can be applicable on virtualized environments. Next section validates the applicability of this power model to perform per-VM energy accounting.

B. Per Virtual Machine energy accounting validation

In this section, we validate the methodology used to perform the energy accounting. We confirm that `pfmon` is able to track the activity of the VMs and also that the power model estimations are accurate.

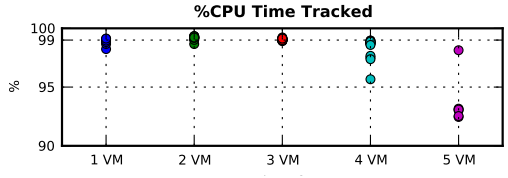
The experiments performed to validate the methodology are summarized in Table III. We choose three `SPECcpu2006` applications (shown at the top of Table III) with high ('+'), medium ('=') and low('−') power consumption. The rationale of that choice is to be able to generate different heterogeneous workloads on the VMs. Two applications run inside each dual-core VM, and depending on the number of VMs explored, we generated a wide set of workloads. They are shown at the bottom of Table III, sorted by the number of '−', '=' and '+' respectively (from low to high power consumption). For instance, the test 10 with 2 VMs runs two `454.calculix` in one VM and two `435.gromacs` in the other one. We executed each of the 62 different workloads during five minutes in order to gather their PMC activity and power trace.

1) Activity tracking validation: As we explained in Section II-B, `pfmon` is used to track the activity of each VM. Ideally, the activity tracking could be done by the host OS. Then, inaccuracies that might be introduced by having another user-level application running on the system could be avoided. However, we see that `pfmon` is able to track most of the VM activity. Fig. 3(a) shows the % of time tracked by `pfmon` depending on the number of VMs. The time tracked is derived from the sum of the CPU cycles that `pfmon` accounted for each VM. Therefore, during the 300 seconds duration of the experiments, and assuming no other processes running, we need to account a total of about 600 seconds[5] of CPU time. We see that unless more than 4 VMs are configured on the system, `pfmon` is able to track about 99% of VM activity. Notice that we stress our mechanism by deploying fairly more than the usual number of VM on

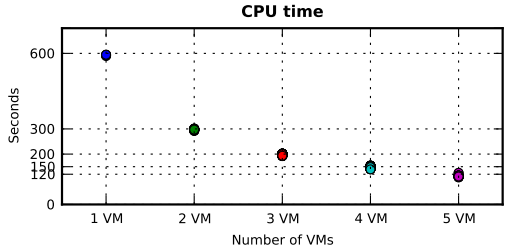
Table III
CONFIGURATIONS EVALUATED

Benchmark	Power Consumption	Avg.Power(mW)	Symbol
454.calculix	High	12432.47	+
482.sphinx	Medium	10976.9	=
435.gromacs	Low	9974.62	-

Test	1 VM	2 VMs	3 VMs	4 VMs	5 VMs
1	−−	=-,−−	==,=-,−−	+-,==,=-,−−	+=,+-,==,=-,−−
2	=-	==,−−	+-,=-,−−	+=,==,=-,−−	++,+-,==,=-,−−
3	==	==,=-	+-,==,−−	+=,+-,=-,−−	++,+=,==,=-,−−
4	+-	+-,−−	+=,=-,−−	+=,+-,==,−−	++,+=,+-,=-,−−
5	+=	+-,=-	+-,==,=-	++,==,=-,−−	++,+=,+-,==,−−
6	++	+=,−−	+=,==,−−	+=,+-,==,=-	++,+=,+-,==,=-
7		+-,==	+=,==,=-	++,+-,=-,−−	
8		+=,=-	++,=-,−−	++,+-,==,−−	
9		+=,==	+=,+-,−−	++,+=,=-,−−	
10		++,−−	++,==,−−	++,+=,==,−−	
11		++,=-	+=,+-,=-	++,+-,==,=-	
12		+=,+-	++,==,=-	++,+=,==,=-	
13		++,==	+=,+-,==	++,+=,+-,−−	
14		++,+-	++,+-,−−	++,+=,+-,=-	
15		++,+=	++,+-,=-	++,+=,+-,==	
16			++,+=,−−		
17			++,+-,==		
18			++,+=,=-		
19			++,+=,==		
20			++,+=,+-		



(a)



(b)

Figure 3. Percentage of CPU time tracked by `pfmon` and CPU time consumed per-VM.

the same system[6]. Even so, our mechanism is able to track fairly well the system activity.

Moreover, we want to validate that the scheduler performs correctly and assigns the same CPU time to each VM. In particular, our 600 seconds of CPU time should be split equally among the VMs. For example, when 3 VMs are present on the system, their execution time should be around 200 seconds (600s / 3 VMs). We confirm that the host scheduler assigns the same amount of CPU time to each VM in Fig. 3(b). Notice that all the points are concentrated on their expected value, without any significant variation.

[5]We have 2 CPUs. Hence, during 300 seconds, we have 600 seconds of CPU time.

[6]Usually, a 1 to 1 mapping is deployed between physical CPUs and virtual CPUs. Whereas we perform a maximum of 5 to 2 mapping.

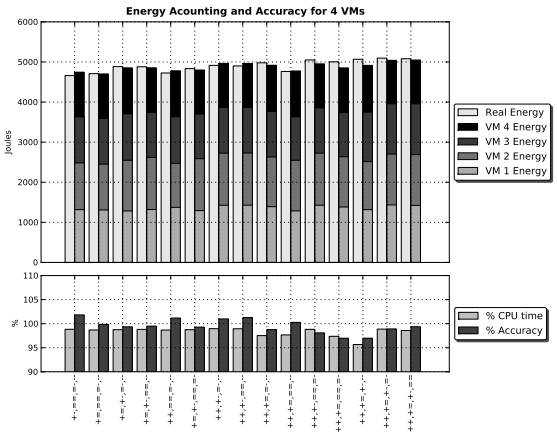


Figure 4. Results summary for 4 VMs.

From that observation, one can think that the chargeback that a provider must perform for each VM machine should be the same. We see later on, in Section III-C, that in terms of energy this is not true.

2) Energy accounting validation: Once we have validated that our methodology is able to track the VM activity, we apply the power model generated in Section II-D. Fig. 4 shows the results for the workloads studied with 4 VMs enabled. The *x* axis denotes the experiment workload. At the top, the real energy consumption and a stacked bar of energy contribution of each VM is shown. The energy is derived from the average power consumption predicted by the model multiplied by the VM execution time. We see that we are able to accurately predict overall energy consumption from the sum of per-VM energy consumption. This is summarized at the bottom of Fig. 4, where the % of accuracy and CPU time tracked are shown. First, we see again that we are able to track almost 100% of the CPU time. And second, the error in energy prediction is always below 5%.

We do not show absolute energy workload results for the other number of VMs due to space limitations. However, we summarize the error in energy prediction for all configuration studied in Fig. 5(a). The main observation is that the error in prediction is not affected by the number of VMs present on the system –keeping the same magnitude–. From that observation we can expect this methodology to be valid on future many-core processors.

We also show the relation between energy prediction and time tracked in Fig. 5(b). We see that if we are able to track most of the VM activity, the error in prediction is low. Once we loose some accuracy in tracing system –the case of 5 VMs–, we obtain higher errors in energy prediction. The reason is that in that configuration the system starts to be overloaded –including some *thrashing*– and as a result, our mechanism based on the user-space process `pfmon` do not accounts all the VM activity. This supports the idea of that the implementation of an activity tracker mechanism should be done into the host OS in order to avoid losses in activity

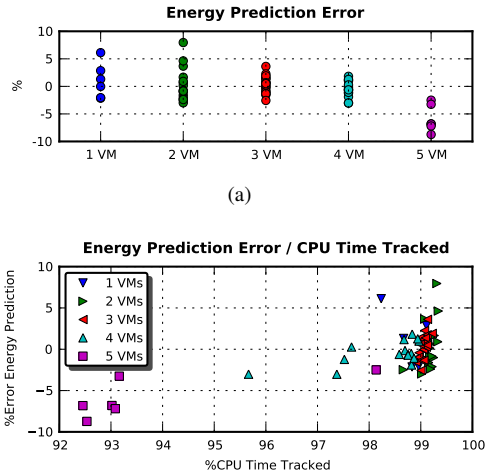


Figure 5. Error in energy prediction and relation between energy prediction error and time tracked by `pfmon`.

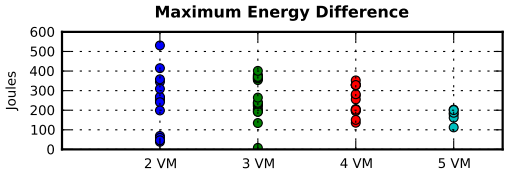


Figure 6. Maximum difference in joules between VM executed concurrently.

accounting.

In this section, we have validated and shown that we are able to perform per-VM energy accounting while keeping acceptable error levels. The following section explains a case of applicability.

C. Applicability

It is obvious that it is possible that VMs that spend the same number of CPU cycles, do not always consume the same amount of energy due to the differences of workloads. In this section, we quantify these differences, showing the fact that our approach can be used to increase the flexibility and the fairness of the chargeback models that service and infrastructure providers use.

Fig. 6 shows the maximum difference in joules between VMs executed concurrently. As we showed in Fig. 3(b), the OS scheduler does a good job assigning a similar amount of CPU cycles to each VM. However, as we see in Fig. 6, in most cases there exist differences in energy consumption among the VMs. We also see that the absolute difference between VMs decreases when the number of concurrent VMs increases. However, that does not mean that the percentage of difference between VMs decreases. For instance, having 500 joules difference in the 2VMs workloads, is the same percentage of difference of having 200 joules difference when executing 5 concurrent VMs on the system. As a result, we have evidenced that energy

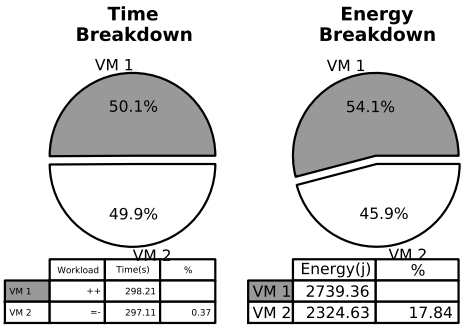
differences between VMs do not depend on the number of VMs running on the system. Hence, they only depend on the workload characteristics.

In order to quantify the magnitude of the differences we present four cases in detail, Fig. 7 shows them. Each subfigure shows the time and energy breakdown pies in conjunction with the information of the workload executed in each VM and the absolute numbers of execution time and energy consumption. We also show the percentage difference in execution time and energy consumption with respect to the VM1. In all the cases, we see that VMs were executed during a fairly similar amount of time. For instance, in Fig. 7(a) the VM1 only was running a 0.37% more time than VM2. The bigger difference in execution time is shown in Fig. 7(d). In that experiment, the VM1 was running 9.91% more time than VM4. In any case, a chargeback model based on execution time will charge similarly all the VMs. However, if we take into account the energy consumption we see that the differences are quite higher. For instance, in Fig. 7(b) and 7(c), the VM1 consumes >20% more energy than VM2 and VM3, even with execution time differences below 5%. The smaller difference is seen in Fig. 7(d), where there is 9.89% energy difference between VM1 and VM2, even there only exists a 0.52% difference in execution time. If we analyze the workloads being executed and the differences, we clearly see the expected results, which are that the '++' workloads consume more energy than the rest.

In conclusion, we quantified the differences in energy consumption of VMs, showing that chargeback models based only on CPU execution time may fail assigning the same charge to VMs that actually consumed fairly different amount of energy (>20%). The reason is that such chargeback models assume the same average power consumption for all workloads, which will result in the same differences in execution time than in energy. However, as seen in Fig. 7, that assumption introduces huge errors because the average power consumed by each workload can differ substantially. As a result, this methodology can be used to obtain more accurate and fair chargeback models.

IV. Related work

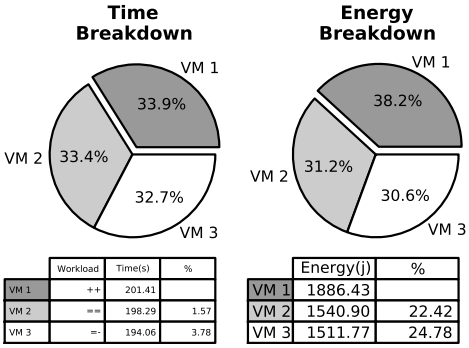
In general, previous works on PMC-based power modeling focus in two main aspects: power model generation and its usage to guide power aware policies. For instance, Tao Li et al. [18] study the power consumption at the OS level using IPC as a predictor. In [19], F. Bellosa implements an OS power aware policy based on data collected at runtime through the PMCs. In this work, overall power consumption is being modeled using a reduced set of PMCs. The works of R.Joseph, C.Isci, G.Contreras and M.Martonosi [20, 4, 21] describe PMC-based power models of other architectures, but following a similar methodology as the one used to derive the Intel® Core™ 2 Duo power model. In conclusion there have been sufficient prior work that proves power



	Workload	Time(s)	%
VM 1	++	298.21	
VM 2	=-	297.11	0.37

	Energy(j)	%
VM 1	2739.36	
VM 2	2324.63	17.84

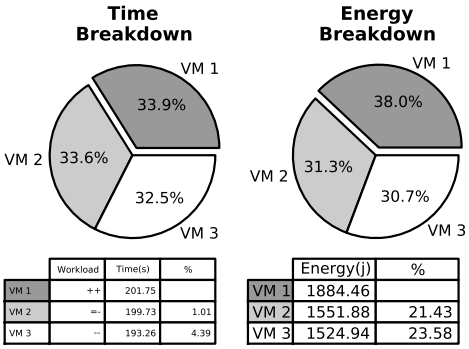
(a)



	Workload	Time(s)	%
VM 1	++	201.41	
VM 2	==	198.29	1.57
VM 3	=-	194.06	3.78

	Energy(j)	%
VM 1	1886.43	
VM 2	1540.90	22.42
VM 3	1511.77	24.78

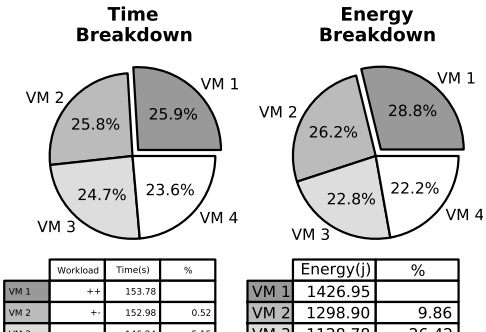
(b)



	Workload	Time(s)	%
VM 1	++	201.75	
VM 2	=-	199.73	1.01
VM 3	--	193.26	4.39

	Energy(j)	%
VM 1	1884.46	
VM 2	1551.88	21.43
VM 3	1524.94	23.58

(c)



	Workload	Time(s)	%
VM 1	++	153.78	
VM 2	+-	152.98	0.52
VM 3	==	146.24	5.15
VM 4	=-	139.92	9.91

	Energy(j)	%
VM 1	1426.95	
VM 2	1298.90	9.86
VM 3	1128.78	26.42
VM 4	1099.23	29.81

(d)

Figure 7. Time and Energy breakdowns for four configurations studied.

models based on PMCs are functional and offer more than satisfactory levels of accuracy. However, as far as we know, they have not been validated on virtualized systems.

Regarding the consideration of energy-costs in virtualized data-center management, some research has been conducted in order to manage energy. In [22, 23] the authors propose a placement optimization process that considers both performance and energy-costs. The proposed placement algorithms aim to satisfy the performance goal of each application and, at the same time, to save energy through workload consolidation, which enables to minimize the number of physical machines required to support the execution.

Finally, there exists some works that perform per-VM energy accounting. In [24], the authors propose an experimental OS prototype to perform per-VM energy accounting. For that purpose, they propose modifications on host and virtualized OS to achieve per-VM energy accounting and assume the PMC-based models are still valid on such environments. In a very recent proposal [25], the authors perform per-VM energy accounting. However, they use simple modeling methods –not PMC-based– for the CPU and memory and as a result, they face the problem evidenced in this paper: same CPU time usage does not imply same energy consumption. They overcome the problem proposing adaptative techniques. In the end, our main difference over previous works is that we do not assume that PMC-based models are valid on virtualized systems. Instead, we firstly validate that assumption, and we propose a mechanism –that do not require any code modification– to track per-VM power consumption. Moreover, the mechanism is also validated in terms of accuracy for a wide set of configurations and workloads.

V. Conclusion

This paper joins the virtualization technology with specific power modeling techniques. We validate the power modeling methodology in virtualized systems. The validation process has been performed on an Intel® Core™ 2 Duo platform. The resulting models are able to account for the power consumption for CPU and memory at process level, corroborating that the virtualization layer does not introduce inaccuracies on predictions. Then, we have applied the power model to account for the energy consumption of each VM in a virtualized system. The accounting is based on collecting the number of cycles that a VM runs, plus the PMCs values that describe its activity. With that we derive energy consumption estimates at VM level. At the end, we are able to obtain estimations with an error below 5%. This estimations can be used to generate more accurate chargeback models, since, as we shown, VMs machines executed during the same amount of time, presented more than 20% variations in energy consumption.

Although this is a promising technology for energy accounting, there are several open issues. Mainly, the current model is restricted to the processor and memory, with no support for modeling the IO operations. This corresponds to future work to come.

References

[1] A. Merkel and F. Bellosa, "Balancing power consumption in multiprocessor systems," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 4, pp. 403–414, 2006.

[2] K. Singh, M. Bhadauria, and S. A. McKee, "Real time power estimation and thread scheduling via performance counters," *SIGARCH Comput. Archit. News*, vol. 37, no. 2, pp. 46–55, 2008.

[3] A. Bhattacharjee and M. Martonosi, "Thread criticality predictors for dynamic performance, power, and resource management in chip multiprocessors," in *ISCA '09*, Austin, TX, USA, 2009, pp. 290–301.

[4] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: Methodology and empirical data," in *MICRO 36*, San Diego, CA, 2003, p. 93.

[5] K.-J. Lee and K. Skadron, "Using performance counters for runtime temperature sensing in high-performance processors," in *IPDPS '05*, Denver, CO, USA, 2005, p. 232.

[6] W. Bircher and L. John, "Complete system power estimation: A trickle-down approach based on performance events," *ISPASS '07*, pp. 158–168, 2007.

[7] R. Bertran, M. Gonzàlez, X. Martorell, N. Navarro, and E. Ayguadé, "Decomposable and responsive power models for multicore processors using performance counters," in *ICS'10*, Tsukuba, Japan, 2010.

[8] V. George, S. Jahagirdar, C. Tong, K. Smits, S. Damaraju, S. Siers, V. Naydenov, T. Khondker, S. Sarkar, and P. Singh, "Penryn: 45-nm next generation Intel® Core™ 2 processor," in *ASSCC'07*, 2007.

[9] SBSIF. (1998, December) Smart specifiaction rev.1.1. [Online]. Available: http://sbs-forum.org

[10] "Kvm." [Online]. Available: http://www.linux-kvm.org

[11] "Thinkpad smapi kernel module version 0.40." [Online]. Available: http://tpctl.sourceforge.net/

[12] Perfmon2. [Online]. Available: http://perfmon2.sourceforge.net/

[13] "Qemu." [Online]. Available: http://www.qemu.org

[14] "Libvirt." [Online]. Available: http://www.libvirt.org

[15] J. L. Henning, "Spec cpu2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, 2006.

[16] A. Phansalkar, A. Joshi, and L. K. John, "Subsetting the spec cpu2006 benchmark suite," *SIGARCH Comput. Archit. News*, vol. 35, no. 1, pp. 69–76, 2007.

[17] A. Phansalkar and et al., "Analysis of redundancy and application balance in the spec cpu2006 benchmark suite," *SIGARCH Comput. Archit. News*, vol. 35, no. 2, pp. 412–423, 2007.

[18] T. Li and L. K. John, "Run-time modeling and estimation of operating system power consumption," *SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 1, pp. 160–171, 2003.

[19] F. Bellosa, "The benefits of event: driven energy accounting in power-sensitive systems," in *ACM SIGOPS - EW 9*, Kolding, Denmark, 2000, pp. 37–42.

[20] R. Joseph and M. Martonosi, "Run-time power estimation in high performance microprocessors," in *ISLPED '01*, Huntington Beach, CA, USA, 2001, pp. 135–140.

[21] G. Contreras and M. Martonosi, "Power prediction for Ontel XScale processors using performance monitoring unit events," in *ISLPED '05*, San Diego, CA, USA, 2005, pp. 221–226.

[22] M. Steinder, I. Whalley, J. E. Hanson, and J. O. Kephart, "Coordinated management of power usage and runtime performance." in *APNOMS '08*. IEEE, 2008, pp. 387–394.

[23] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 103–116, 2001.

[24] J. Stoess, C. Lang, and F. Bellosa, "Energy management for hypervisor-based virtual machines," in *ATC'07*. Berkeley, CA, USA: USENIX Association, 2007, pp. 1–14.

[25] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. Bhattacharya, "Virtual machine power metering and provisioning," in *SOCC*, June 2010. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=120435