Towards Accurate Accounting of Energy Consumption in Shared Virtualized Environments

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 VMs each. While resource utilization accounting can be achieved with existing system tools, power metering is a complex task when per-VM granularity is the goal. In this paper we propose a novel methodology that brings new opportunities to power consumption accounting by adding an unprecedented degree of accuracy on the per-VM measurements. We present a system prototype that leverages power models based in performance monitoring counters (PMCs) to be used for energy accounting in virtualized systems. We validate the power modeling methodology in virtualized systems, by comparing the power predictions in both virtualized and non virtualized systems. The validation process has been performed as a case study for the Intel® Core™ 2 Duo architecture. The validation follows two steps: first we validate the power model for one core, and second, we proceed on the validation of the entire processor. The resulting model is able to account for the power consumption for CPU and memory at process level. The main contribution of this paper is the introduction of a novel methodology that allows accurate accounting of energy consumption in virtualized systems. Accounting is done on a per-VM basis, even in the case that multiple VMs are deployed on top of the same physical hardware, overpassing the limitations of per-server aggregated power metering. Such approach can bring an unprecedented level of flexibility to the chargeback model used by service and infrastructure providers.

Keywords

power model, virtualization, accounting

1. INTRODUCTION

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

Power consumption plays an important role in the maintenance cost of a large data center, including power consumption not only from computing, storage and networking resources, but 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 [17, 25, 5]. 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 [3, 12, 16, 7, 15, 6]. At the end, they have been crucial in the process of addressing power issues.

In this paper we join the virtualization technology with power model techniques to derive energy consumption estimates at per VM level. We present a methodology that provides accurate model-based power consumption accounting for native and virtualized platforms. The technique leverages the power modeling methodology introduced in [4] to derive per-VM power consumption. The proposed technique uses existing performance monitoring counters and the above-mentioned model 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.

The rest of this paper is organized as follows: section 2 describes the power modeling methodology, section 3 validates the methodology in virtualized systems and demonstrates how this technology can be used to derive energy consumption estimates. Section 4 describes the related work. Finally, section 5 concludes the paper.

2. METHODOLOGY

In this section, we describe our methodology for producing power models. The discussion is guided through the presentation of an example for a particular architecture, an Intel® Core™ 2 Duo [10]. The final outcome of this section is a power model of this particular processor, exposing a small average error in power prediction. Combining the model predictions with actual execution times, we will prove that is possible to account for the energy consumption at VM level.

In short, our methodology for producing decomposable power models follows the common modeling steps, already addressed in previous works [12, 27, 25]. First, we define the model inputs which in our case they are the power components activity ratios. Second, we define the training data which in our case will be generated using microbenchmarks. Third, we collect the required data in order to train and validate the model respectively. And fourth, the model is built using our method.

It should be noted that even though the example presented is for a particular architecture, the methodology can be applicable to other architectures. Particularly, one should redefine and redesign the power components and the microbenchmarks in order to fulfill the requirements of our method. The rest of the section presents in detail the previous steps and justifies their rationale.

2.1 Defining power components

The inputs of our model are component activity ratios, which have been shown to be good proxies for power estimation [3, 12]. As a result, the first step towards the final power model is to define the power components and their associated activity ratio formula.

Our objective is to reach a component definition with as many as possible power components that represent a single microarchitectural component in order to have a more detailed power model. However, there are two main restrictions to this direct-mapping between microarchitectural components and power components. First, some microarchitectural components are tightly related in the sense that even if they are monitored by a different set of PMCs, they expose very similar levels of activity. For such components it is not possible to discern directly their power contribution because their activity can not be isolated or decoupled from the activity of other microarchitectural components. And second, there are also microarchitectural components that do not have any PMC that reports their activity. Therefore, due to these restrictions, in our approach a power component can represent one or a set of microarchitectural components. This approach is a reasonable tradeoff between the undecomposability of simpler approaches [25] and the complexity of the microarchitectural-level decomposable ones [12].

As a case study, we explain the power component definition for an Intel® Core™ 2 Duo. We have identified more than 25 microarchitectural components on this architecture. Figure 1 shows an schematic view of them classified into three categories in order to facilitate the explanation: the in-order engine (dark gray), the memory subsystem (white) and the out-of-order engine (light gray).

The in-order engine includes some microarchitectural components that do not have PMCs that report their activity. Moreover, the activity of each of these components can not be isolated since they are in the in-order part of the pipeline, i.e. activity in the `FETCH` unit, means activity in the `PRE-DECODE` unit. Therefore, we group the whole in-order engine components –except the branch prediction unit (`BPU`)– as one power component, namely, the frontend (`FE`) of the processor. The decision to define the `BPU` power component separately is based in the fact that it includes several predictors that can consume between 10% and 40% of the whole processor power budged [19].

For the memory subsystem, it is not possible to generate activity in one memory level without causing activity in the previous one due to its stacked configuration. However, it is possible to decouple their activity. As a result, the whole memory subsystem is divided in three power components: the `L1` cache, the `L2` cache and the Front Side Bus (`FSB`), which also represents the main memory.

The parts of the architecture in the out-of-order engine define the `INT`, `FP` and `SIMD` power components. At the end, eight power components are defined. Table 1 summarizes them in conjunction with their related microarhictectural components and PMC-based activity formulas.

2.2 Designing microbenchmarks

We need empirical data in order to train our power model. For that purpose, a set of microbenchmarks is designed. This is an important step in the methodology because not considering a significant set of possible scenarios incurs in inaccuracy of the final model [27]. We do not use real applications to train the model due to two reasons. First, we want a model suitable for any workload; therefore, the training set should be workload independent. And second, in order to be able to derive the contribution of each power component to the total power consumption it is mandatory to be able to isolate or decouple their activity. This is required in order

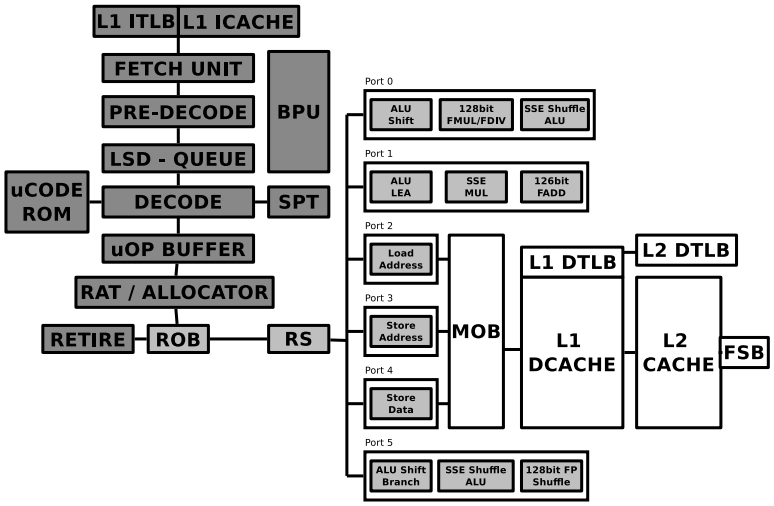

Figure 1: Single core view of Intel® Core™ 2 Duo microarhictectural components.

Power Component	Activity formula	Modeled components
FE	UOPS_RETIRED:ANY/ CPU_CLK_UNHALTED:CORE_P	L1_ITLB, L1_ICACHE, FETCH_UNIT, PREDECODE, LSD, DECODE, uCODE ROM, SPT, uOP BUFFER, RAT, ROB, RETIRE
INT	(RS_UOPS_DISPATCHED_CYCLES:PORT_0 + RS_UOPS_DISPATCHED_CYCLES:PORT_1 + RS_UOPS_DISPATCHED_CYCLES:PORT_5 - FP_COMP_OPS_EXE - SIMD_UOPS_EXEC - BR_INST_RETIRED:ANY)/ CPU_CLK_UNHALTED:CORE_P	Integer arithmetic units
FP	FP_COMP_OPS_EXE/ CPU_CLK_UNHALTED:CORE_P	Floating point arithmetic units
SIMD	SIMD_UOPS_EXEC/ CPU_CLK_UNHALTED:CORE_P	SIMD arithmetic units
BPU	BR_INST_DECODED/ CPU_CLK_UNHALTED:CORE_P	BPU and branch execution
L1	L1D_ALL_REF/ CPU_CLK_UNHALTED:CORE_P	LD/ST execution, MOB, L1, L1 DTLB, L2 DTLB
L2	L2_RQSTS:BOTH_CORES:ANY:MESI/ CPU_CLK_UNHALTED:CORE_P	L2
FSB	BUS_DRDY_CLOCKS:ALL_AGENTS/ CPU_CLK_UNHALTED:BUS	FSB and memory

Table 1: Power components defined for a Intel® Core™ 2 Duo in conjunction with their PMCs-based activity formulas and microarhictectural components that represent.

to apply multiple regression techniques without collinearity problems [20].

The microbenchmark design is always a source of inaccuracy since it is not possible to cover all possible scenarios of power consumption and activity. For example, for exercising the `INT` units we can use different instructions that will consume differently. Similarly, different inputs produce different power consumption. In conclusion, the microbenchmark suite should cover enough variety of power behavior in order to minimize all these sources of inaccuracy.

For our case study, we have designed 97 microbenchmarks. Their structure is an infinite loop containing a specific sequence of assembly instructions with distinct dependency chains in order to explore different activity ratios. However, we had to face the problem of the inherent correlation between the `FE` component and the rest of components. We solved it by introducing diverse combinations of `fxch`[1] instruction sequences. The instruction sequence within the infinite loop is long enough to minimize the effect of the branch instruction at the end of it. Table 2 summarizes the range of activity ratios that are explored for each power component. For example, for the `INT` component we have designed 13 benchmarks with `INT` activity ratios ranging from 1 to 3 and `FE` activity ratios from 1 to 3.45. Notice that we are able to have a `FE` activity of 3.45 uops/cycle with only 3 uops/cycle on the `INT` component and the rest of components unused. This decoupling of activity is the effect of the `fxch` instruction and it is mandatory to be able to form a decomposable power model afterwards. Next section describes the experimental framework and the data gathering process used to obtain the power consumption.

2.3 Collecting Data

All the experiments have been carried out on a workstation that features an Intel® Core™ 2 Duo T9400 processor [10] and two modules of 2GB of memory. The embedded controller firmware of the platform provides information about the power source, which fulfills the SMAPI specification [24]. 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 the required patches to allow PMC readings. The `tp_smapi` [1] 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 use a modified version of `pfmon` [2] to access to the PMCs and power consumption simultaneously. All measurements are obtained running the experiments in standalone mode to minimize interferences. We have switched off all sources of power consumption –i.e. the display– that we do not want

[1]As far as we know, the *fxch* is the only one instruction that does not dispatch any operation to the out-of-order engine. It only performs register renaming, therefore it allows to decouple the activity on the `FE` and the components in the out-of-order engine.

Microbench set	#	FE Activity	INT Activity	FP Activity	SIMD Activity	BPU Activity	L1 Activity	L2 Activity	FSB Activity	Power Range
FE	1	1	0	0	0	0	0	0	0	5587mW
INT	13	1-3.45	1-3	0	0	0	0	0	0	5792mW-8114mW
FP	9	0.2-1.98	0	0.2-1	0	0	0	0	0	5087mW-6887mW
SIMD	12	1.85-3.29	0	0	0.99-2.63	0	0	0	0	6966mW-9274mW
BPU	5	0.42-1.14	0-0.42	0	0	0.46-1	0	0	0	5761mW-8521mW
L1	16	1-2.97	0	0	0	0	0.66-2	0	0	6571mW-8160mW
L2	12	0.12-0.42	0	0	0	0	0.11-0.22	0.11-0.21	0	8600mW-10112mW
FSB	18	0.02-0.14	0	0	0	0	0.02-0.04	0.02-0.04	0.58-0.71	10976mW-12205mW
MIX	11	1.63-3.95	0-1	0-0.8	0-1.97	0-0.34	0-1.97	0-0.07	0-0.34	7318mW-11298mW
Total	97	0.02-3.95	0-3	0-1	0-2.63	0-1	0-2	0-0.21	0-0.71	5087mW-12205mW

Table 2: Microbenchmark characteristics. Activity ratios and power consumption for an Intel® Core™ 2 Duo.

to interfere to the power model generation. For platform components where it is not possible to switch off, we configured them at constant operation mode. For example, we found that the fan of the processor introduces up to 0.5W variations on power consumption. Hence, we configured it to always operate at full-speed. Under this conditions, we tracked the power consumption during five minutes before each experiment and we found the baseline power consumption for the idle system to be 9.8W. The fact that for all the experiments carried out we found a fairly constant idle power consumption demonstrated that we nullified possible interferences. For the purpose of the model formation only, we assumed that baseline, 9.8W, to be the power consumption of the platform except the processor and the memory. This assumption and the accuracy ensured by the SMART specification avoids the need of more complicated measuring techniques [25].

The performance monitoring unit of the processor is not able to track all the selected PMCs simultaneously. Therefore, we grouped them in sets that can be sampled at the same time. We programmed `pfmon` to switch PMC sets every 10ms and print the PMCs values and power measurements every 2 seconds. Although it is widely accepted that the PMC sampling does not introduce unacceptable inaccuracies [14, 12], we measured its effect, resulting in an interference of less than 150mW.

To generate the input data for training the power model we have run every microbenchmark during three minutes with only one core enabled, collecting a trace of 90 samples. We have validated that each microbenchmark stress the component it was designed to do so by applying the formulas in Table 1. Moreover, we checked that during the entire trace of each microbenchmark the PMC activities and power consumption remain nearly constant. We take the average over the 90 samples so that the possible errors and noise are minimized as much as possible [27].

In order to validate the applicability for energy accounting, it is necessary to prove that the model behaves similarly in virtualized and non virtualized environments. According to that, we set up two type of scenarios. On one side, we configure a system where we run applications on top of native hardware with no virtualization. Then we run the same applications but in a virtualized system. In this case we force the pfmon utility to attach to the process that represents a virtual machine, and to follow any thread of the VM. We take power measurements of both environments and we expect them to be similar, and moreover, we expect the model

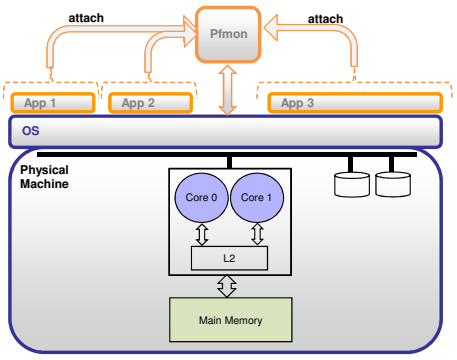

Figure 2: Non virtualized system for gathering data.

to correctly predict the power consumption. In both cases, pfmon is responsible for the data gathering. It should be noticed that the usage of the pfmon application is related to the fact that there is no explicit support in the host system to monitor the activity of the processor, and define particular frequencies of sampling and relate the samples to actual power measurements. If all this support were available in the host system, the pfmon utility will be unnecessary.

Figure 2 describes the non virtualized environment. In this case the applications run on the native OS and pfmon gathers information about information about their activity in the processor.

In contrast to Figure 2, the virtualized environment introduces one layer of virtualization represented by the VMs. Figure 3 shows this scenario where two VMs coexist, and within every VM two applications are running on top of virtual hardware represented by two virtual cores. In this case pfmon gathers information about the activity in the VMs. One of the main objectives of this paper is to validate the power model in this scenario, so that we open the power modelling as a mechanism to account for energy consumption.

For the two of the aforementioned scenarios, we have collected the data for 26 applications of the SPECcpu2006 [11] benchmark suite. Every benchmark runs for 30 minutes or less if it ended earlier. This data is going to be used to validate the power model for one core and for the entire architecture (two cores). This is addressed in section 3.

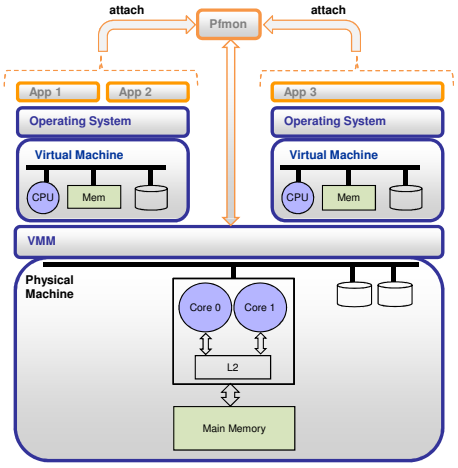

Figure 3: Virtualized system for gathering data.

In order to show the type of gathered data, Figure 4 depicts a time slice of the activity evolution of all components for three applications of the SPECcpu2006 benchmark suite. We accompany this data with actual power measurements and the instruction distribution between load, store, integer, floating point and branch instructions which can be computed from the activity ratios. This data has been collected in a non virtualized environment. The basis for the power modelling are the curves of activity in every component in the architecture: the front-end (FE), the INT and FP units, the branch predictor unit (BPU), the L1 and L2 caches, and the memory activity, represented by activity in the Front Side Buffer (FSB). According to the observed activity, it is possible to estimate the power consumption at particular instants. As first observation, notice that it is worth to decompose the architecture in components because the power behavior in response to the component activity might be significantly dependant on different components. For instance, the 473.astar exposes an increment of power consumption for a particular time phase. Looking at the component activity, the increment is totally related to an increment in the L2 cache, the L1 cache and the BPU, but without activity in main memory (FSB). On the contrary, the 482.sphinx presents no significant change in the power consumption, but the application presents a significant increment on the activity in main memory. Moreover, the 416.gamess has no activity in the FSB, but its power also remains constant. In the same direction, notice that the 482.sphinx and 416.gamess applications present similar power levels, but very different activity in the FE, that mostly represents IPC. In conclusion, the aim of this explanation is to show the type of data it is necessary to gather, and to expose the necessity of know about the activity in the architecture to perform accurate power predictions.

2.4 Modeling power

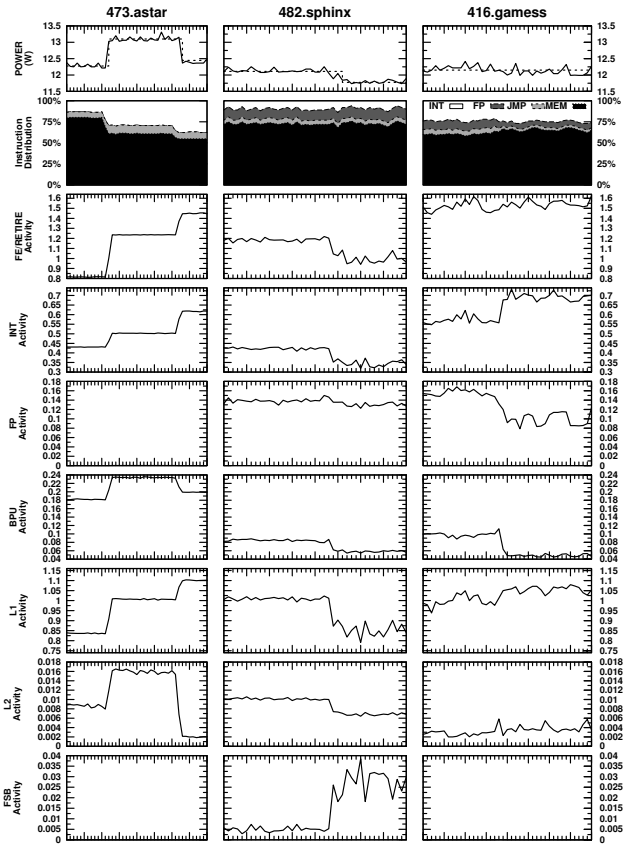

Figure 4: Power consumption, instruction distribution and component activity ratios during three time slices of the `473.astar`, `482.sphinx` and `416.gamess` SPECcpu2006 benchmarks.

2.4.1 *Single Core modeling*

We model the power consumption using a multiple linear equation with seven input variables, one for each power component defined. Therefore, the total power consumption is expressed as:

$$P_{total} = \left(\sum_{i=1}^{i=numcomp} AR_i \times P_i \right) + P_{static} \qquad (1)$$

where P_i is the weight of component i, which we need to solve, and AR_i is its activity ratio. The $AR_i \times P_i$ represents the dynamic power consumption of component i, and P_{static} represents the overall static power consumption of all components. This approach is commonly used for generating PMC based power models [12, 27, 25]. However, instead of applying directly linear regression techniques [9, 20], we derive the marginal effect of each component to the overall power consumption. Firstly, we compute the P_i of each component –except the `FE`– separately using a multiple linear regression with the component related microbenchmark set as input. For example, we computed P_{INT} using only the `INT` microbenchmark set, which only stresses the `INT` component. The components which activity is not completely isolated -i.e. `L2` and `FSB`- are calculated incrementally. For example, before deriving P_{L2} from the `L2` microbenchmark subset, we derive first P_{L1} and use that value for the P_{L2}

estimation. The rationale that justifies this methodology is that the collinearity that may exist between components that are not stressed together is canceled. Therefore, we get better estimates about what is the contribution of each component. The goodness of the estimates also relies on the collinearity between the exercised component and the `FE`, which we minimized when we designed the microbenchmarks. Finally, we use again a multiple linear regression using as input the `MIX` microbenchmark subset to derive P_{static} and P_{FE}. We use that set because it represents the common activity scenario, when there is activity across all processor components, canceling the known effect of under-estimate their contribution [27]. Moreover, this process have been automatized using a R script [23] in order to be applicable systematically. This is possible because no manual tuning is required, demonstrating that with an appropriate set of microbenchmarks it is possible to generate decomposable and accurate power models. Table 3 describes the coefficients of the generated model.

2.4.2 Multiple Core modeling

We use an accumulative approach for modeling multiple cores assuming that each core behaves equally. This assumption have been already done in similar works [25]. Hence, we apply the single core model to each core in the architecture. Therefore, we express the total power consumption as:

$$P_{total} = \sum_{j=1}^{j=numcores} ((\sum_{i=1}^{i=numcomp} AR_{ij} \times P_i) + P_{static}) \quad (2)$$

where the P_i of each component is the same of the single core model but the formulas to calculate the AR_ij should be modified to perform per core accounting. This is straightforward since PMCs already support per core event masks. Besides, it is needed to redefine the P_{static} component of the model because that component represents the static power of the entire processor. As a result, the P_{static} value obtained from single-core model training set is not valid because it accounts for shared resources which static power should not be replicated –i.e. the `L2` cache–. To recalculate it, we need an input data where all the cores modeled in the architecture are active. For that purpose, we re-executed the `MIX` subset of the microbenchmark suite on both cores at the same time in order to get a suitable training data. Then, a new P_{static} was generated by using a linear regression and dividing the obtained value by the number of cores.

3. VALIDATION AND EVALUATION

The model evaluation is organized in three main sections. The first one covers the model validation for one core in virtualized and non virtualized systems. In both cases, accuracy is measured in average error in power predictions. The second part does the same but for the entire architecture, that is, for two cores. Finally, the third part of the evaluation corresponds to the main outcome of this section: the proof of concept that power models allow for energy accounting at process level, which in virtualized environments translate to energy accounting at VM level.

3.1 One Core Validation

3.1.1 Model Error

	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%

Figure 5: Average error of the single core model for the SPECcpu2006 suite.

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. In virtualized tests, applications run within a VM, that runs as a process on top the native OS and hardware.

Figure 5, 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 error and standard deviation that the model incurs in for both environments. In general, for both cases the average error never exceeds a 7% of error. Graphically, this can be observed in Figure 6. In this figure we compare the model average error for every application in virtualized and non virtualized systems. This corresponds to the third and fifth columns in Figure 5. 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.

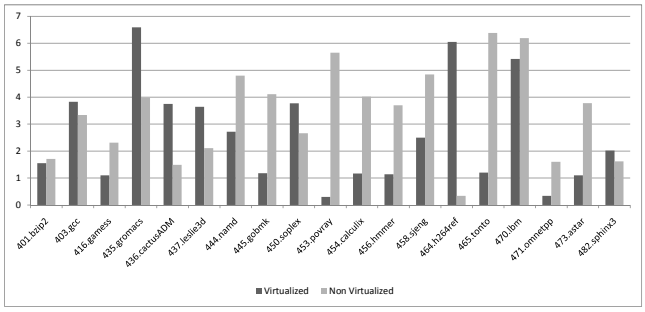

Figure 6: Comparison of average error of the single core model for the SPECcpu2006 suite.

We have also measured the differences in the power consumption between the virtualized and non virtualized systems. Figure 7 shows these differences in percentage with respect the observed value in the non virtualized system. Negative values imply less power in the virtualized system, and the magnitude of the value indicates how much less power in percentage. In general, that the differences are in the range

Model	Method	training set	Frequency	P_{FE}	P_{INT}	P_{FP}	P_{BPU}	P_{L1}	P_{L2}	P_{FSB}	P_{STATIC}
MICRO	inc.	micro	2.54GHz	789	261	502	1908	856	24437	8852	8701

Table 3: Model generated for an Intel® Core™ 2 Duo.

of 1 and 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). In all the other cases, the differences always range between -1.20 and -3.70. The slight decrement in power consumption is related to eventual interferences of the VM with the actual execution of the application running within it.

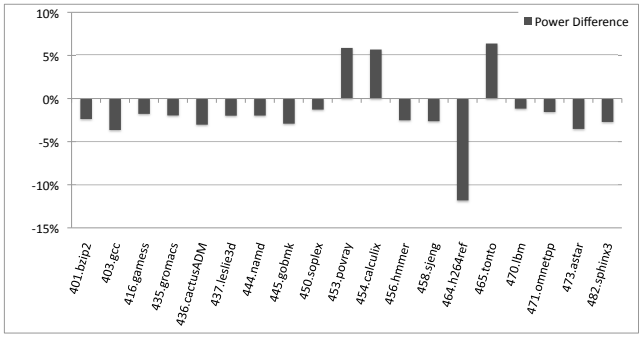

Figure 7: Difference in power consumption in virtualized and non virtualized environments.

In the virtualized experiments, we have also measured the response of the power model during the boot and shutdown phases. The experiment consists on booting a VM, launch the application within it and run it for a period of time while we gather activity samples. Then shutdown the VM. Figure 8 shows the power evolution for the 454.calculix application. The three phases of execution are clear: the VM boot, execution of the application, and the VM shutdown. During the application's execution the model succeeds on accurately predict the power consumption level. But this is not the case for the boot and shutdown phases. These two phases are intensive in IO operations that are not captured by the model. The model just accounts for activity in the processor and main memory. The modelling of IO operations is out of the scope of this paper and it is one subject of future study.

In conclusion, the power model generated for one core is

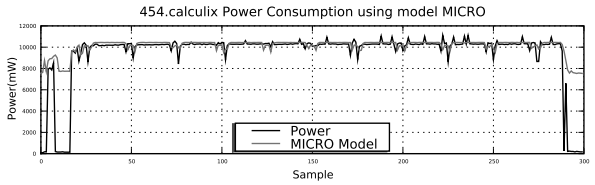

Figure 8: Complete power modelling for the 454.calculix application.

	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%

Figure 9: Average error of the two core model for the SPECcpu2006 suite.

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.

3.2 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 VM (with two virtual cores), that runs as a process on top the native OS and hardware.

We have selected a subset of the SPECcpu2006 benchmarks to evaluate the power model. In virtualized tests, one VM is created and two of these applications run within the VM. In non virtualized tests, we compare with the same pair of applications running on top of the native OS and hardware. In both cases no process pinning or numa control is activated. The applications have been chosen according to the classification and characterization of the SPECcpu2006 for multi-programming evaluation [22, 21]. We have selected a total number of 6 applications and paired them to produce a workload within the VM.

Figure 9 describes the data obtained in both scenarios, similarly as in the case with just one core being active. Both environments expose similar average errors in every case. These range between 0.40 and 3.06 percentages of error which demonstrates the reliability of the power model for both environments. Figure 10 shows the comparison between the obtained average errors. Notice that the general trend is that they are very similar and always down the 4. unless for the case of the 462.libquantum and 454.calculix. We observe a 1.8 average error for the virtualized test and a 6.69 average error for the non virtualized test.

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

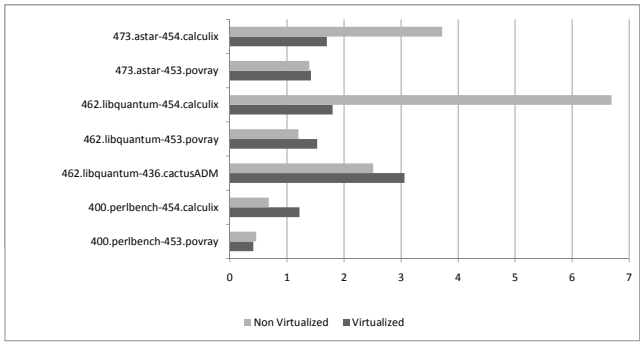

Figure 10: Comparison of average error of the two core model for the SPECcpu2006 suite.

Configuration		Workload	CPU Cycles	CPU Time (Secs)	Avg . Power (Watts)	Energy (Joules)
test 1	VM1	400.perlbench-436.cactusADM	1.21E+12	451.88	7.70	3478.31
	VM2	400.perlbench-436.cactusADM	9.40E+11	350.29	7.36	2566.29
test 2	VM1	400.perlbench-453.povray	1.29E+12	481.80	8.65	4184.84
	VM2	400.perlbench-453.povray	1.29E+12	480.67	8.66	4173.80
test 3	VM1	400.perlbench-454.calculix	1.41E+12	523.61	8.89	4655.36
	VM2	400.perlbench-454.calculix	1.40E+12	522.42	8.90	4653.57
test 4	VM1	462.libquantum-436.cactusADM	1.40E+12	523.39	7.45	3897.28
	VM2	462.libquantum-436.cactusADM	1.40E+12	521.87	7.44	3883.91
test 5	VM1	462.libquantum-453.povray	1.34E+12	499.33	8.06	4024.63
	VM2	462.libquantum-453.povray	1.34E+12	498.29	8.05	4014.95
test 6	VM1	462.libquantum-454.calculix	2.88E+11	107.33	7.72	829.22
	VM2	462.libquantum-454.calculix	1.41E+12	525.18	8.29	4357.34
test 7	VM1	473.astar-436.cactusADM	1.17E+12	434.29	7.36	3180.20
	VM2	473.astar-436.cactusADM	1.25E+12	466.25	7.44	3460.28
test 8	VM1	473.astar-453.povray	1.27E+12	474.41	8.58	4079.57
	VM2	473.astar-453.povray	1.27E+12	473.77	8.58	4073.24
test 9	VM1	473.astar-454.calculix	1.40E+12	520.44	8.75	4553.65
	VM2	473.astar-454.calculix	1.40E+12	520.61	8.71	4538.23

Table 4: Applicability of the power model for accounting purposes

model. This allows for generating the main outcome of this section and what corresponds to the main contribution of this paper, that of accounting for energy consumption at VM level. Next section addresses this particular issue.

3.3 Applicability

In this section we present some experiments that illustrate how the methodology introduced in this paper can be leveraged for accurate accounting of energy consumption. We used our 2-core system to host 2 VMs, each configured to use two virtual CPUs on top of the two cores. Two applications were simultaneously run inside each VM, so 4 applications were run concurrently in the system. No explicit CPU-binding nor resource management was enforced, allowing the OS to decide how to share resources across VMs and virtual CPUs. The experiment leverages the model validated in the previous sections to estimate the power consumption of each VM in the system based on the processor and memory activity produced by the VMs. Results obtained for this experiment are summarized in Table 4.

As Table 4 shows, we have ran 9 different tests for this experiment, resulting in 9 different combinations of applications running inside the VMs. The length of each test was of 600s, after what, each VM was shutdown and consequently the hosted applications stopped. For each VM, we used pfmon to collect information about CPU activity. Such data was used both to build the per-VM power consumption model (Avg. Power column) and to monitor the real CPU utilization time for each VM (CPU time column) over the 600s duration of each test. Notice that while CPU time is directly read from performance counters, energy consumption is derived by applying the model presented and validated previously in this paper. It can be calculated provided that we have information from performance counters in 2-seconds samples, and we use the power model to estimate the average power consumption per-VM during the sampling period. Therefore, energy consumption is obtained straightforward by multiplying the average power consumption in a sample and the sample length.

Two main results are obtained from data listed in Table 4. First, it is shown that the power model presented in this paper can be leveraged to perform accurate per-VM accounting of energy consumption in shared and virtualized execution environments. Second, the proposed methodology can be used to introduce new chargeback models that take into consideration fine grained energy consumption measurements. One particularly illustrative example of applicability of the proposed proposed technique is shown in the results corresponding to tests 3, 4 and 9 listed in Table 4. As it can be observed, the 6 VMs that were involved in these 3 tests showed the same CPU utilization over the 600s run, resulting in around 520s of CPU time each. But given the different characteristics of the hosted applications, VMs for test 3 consumed 8.9W, 7.4W for test 4 and 8.7W for test 9. Therefore, the energy consumption (and its electricity cost) were clearly different. As the three tests ran for 600s, current pricing models would charge the customers owning VMs for test 3, 4 and 9 equally, while in terms of energy consumption, their applications had a clearly differentiated behavior. This is true for both infrastructure providers charging customers on only at utilization time (600s in all cases) and for those charging in a per-utilization basis (around 520 of CPU time in all cases).

Finally, recall that all the applications in the set of benchmarks used in the experiments of this paper are CPU-intensive; but in the case of running mixed workloads, i.e. a softly loaded web server an a heavily loaded database server, the benefits of accurate energy consumption accounting would be much more remarkable.

4. RELATED WORK

In general, previous works on power modeling focus in two main aspects: power model generation and its usage to guide power aware policies. For instance, Tao Li et al. [16] study the power consumption at the OS level. For this purpose, an IPC based model is built based on the broadly accepted observation that there is a linear relation between IPC and power consumption. In [3], 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. K. Singh et al. [25] describe a methodology based on a set of microbenchmarks that stress particular components of the processor architecture being modeled. As a result, a model suitable for any workload -generic-. Our main difference with all these approaches is that they use the PMCs –one or several– to predict the power consumption of the whole processor, treating it as a black-box.

Regarding the consideration of energy-costs in virtualized data-center management, some research has been conducted in order to save energy, but there is not previous work about per-process energy accounting. For example, in [26] the authors propose a placement optimization process that considers both performance and energy-costs. The proposed placement algorithm aims 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.

A. Miyoshi et al. [18] introduce the concept of critical power slope. This new metric is used to determine the effectiveness of activating power saving techniques on scheduling points. A contribution of this work is that depending on the workload characteristics lowering the processor frequency results in power savings. Our proposal agrees with this result, and points out the fact that power models have to be validated at inflection points of power in order to be used by power aware policies.

The works of R.Joseph, C.Isci, G.Contreras and M.Martonosi [14, 12, 8] present many similarities to our proposal because we follow the same objective: generate an accurate power model capable to breakdown the power consumption. The most similar work to our is [12]. In that work, the Pentium IV architecture is decomposed in microarchitectural components in a similar fashion. But the model generation methodology differs: they use component area as the main heuristic to derive each component power consumption. Some microbenchmarks are also used to guide the manual tuning of the model. In contrast, we derive the marginal effect of each component on power from data gathered using a specifically designed set of microbenchmarks. Thus, we avoid their test-and-evaluate manual tuning and the need of the floor-plan information. In [8] they also use statistical linear regression to generate the model but they validate it against a similar set of applications. In overall, these works are able to predict power consumption in the same order of accuracy than the ones that use less PMCs but, they can breakdown the power consumption. These works proved that the usage of sampling techniques to read a bigger set of PMCs does not incur in inaccuracies. In [13], C.Isci and M.Martonosi present a study comparing power phase classification techniques based on PMCs and control flow information, concluding that PMCs detect more power phases.

Bircher et al. [6] model the power consumption at a very coarse granularity. A full system is considered and decomposed in high level components (e.g: CPU, Memory, I/O and Chipset). The activity in each component is gathered through the PMCs. A power model is derived for each component, with errors ranging 6% and 17% depending on the component. The proposed model for the processor activity is based on counts of fetched micro operations, something that has been proved to be a fair index of activity, but not enough to account for isolated elements in the processor architecture. This work models I/O activity, but has not yet been ported to virtualized environments.

Regarding the consideration of energy-costs in virtualized data-center management, some research has been conducted in order to save energy, but there is not previous work about per-process energy accounting. For example, in [26] the authors propose a placement optimization process that considers both performance and energy-costs. The proposed placement algorithm aims 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.

There is not doubt about the applicability of PMC-based power models since they provide accurate insight into processor power consumption and that they can be used to breakdown the power consumption of a given platform. While this work is based on the same idea of using the PMCs for power estimation, there are some key characteristics that differentiate our proposal from all previous research:

5. CONCLUSION

This paper joins the virtualization technology with specific power modeling techniques. A proof of concept is made for power models based in performance monitoring counters (PMCs) to be used for energy accounting in virtualized systems. We apply a systematic power modeling methodology based on PMCS to derive energy consumption estimates on a per VM basis.

We validate the power modeling methodology in virtualized systems, by comparing the power predictions in both virtualized and non virtualized systems. The validation process has been performed as a case study for the Intel® Core™ 2 Duo architecture. The validation follows two steps: first we validate the power model for one core, and second, we proceed on the validation of the entire processor. The resulting model is able to account for the power consumption for CPU and memory at process level.

Finally, as the main contribution of this paper, we have applied the power model to account for the energy consumption of VMs in a virtualized system. The accounting is based on collecting the number of cycles that a VM runs on the native hardware, plus the PMCs values that describe its activity to produce power estimates of power consumption along the periods the VM runs on the architecture. With that we derive energy consumption estimates at VM level.

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] Thinkpad SMAPI kernel module version 0.40. http://tpctl.sourceforge.net/ .

[2] Perfmon2. http://perfmon2.sourceforge.net/ .

[3] F. Bellosa. The benefits of event: driven energy accounting in power-sensitive systems. In *EW 9: Proceedings of the 9th workshop on ACM SIGOPS European workshop*, pages 37–42, New York, NY, USA, 2000. ACM.

[4] R. Bertran, M. Gonzàlez, X. Martorell, N. Navarro, and E. Ayguadé. UPC-DAC-RR-CAP-2010-2 (Grup de Computació d'Altes Prestacions) Decomposable and Responsive Power Models for Multicore Processors using Performance Counters. Technical report, Departament d'Arquitectura de Computadors (DAC) - UPC, 2010.

[5] A. Bhattacharjee and M. Martonosi. Thread criticality predictors for dynamic performance, power, and resource management in chip multiprocessors. In *ISCA '09: Proceedings of the 36th annual international symposium on Computer architecture*, pages 290–301, New York, NY, USA, 2009. ACM.

[6] W. Bircher and L. John. Complete system power estimation: A trickle-down approach based on performance events. *Performance Analysis of Systems and Software, IEEE International Symmposium on*, 0:158–168, 2007.

[7] W. L. Bircher, M. Valluri, J. Law, and L. K. John. Runtime identification of microprocessor energy saving opportunities. In *ISLPED '05: Proceedings of the 2005 international symposium on Low power electronics and design*, pages 275–280, New York, NY, USA, 2005. ACM.

[8] G. Contreras and M. Martonosi. Power prediction for Ontel XScale®processors using performance monitoring unit events. In *ISLPED '05: Proceedings of the 2005 international symposium on Low power electronics and design*, pages 221–226, New York, NY, USA, 2005. ACM.

[9] N. Draper and H. Smith. *Applied Regression Analysis*. Wiley, New York, NY, second edition, 1981.

[10] 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 IEEE Asian Solid-State Circuits Conference*, 2007.

[11] J. L. Henning. Spec cpu2006 benchmark descriptions. *SIGARCH Comput. Archit. News*, 34(4):1–17, 2006.

[12] C. Isci and M. Martonosi. Runtime power monitoring in high-end processors: Methodology and empirical data. In *MICRO 36: Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, page 93, Washington, DC, USA, 2003. IEEE Computer Society.

[13] C. Isci and M. Martonosi. Phase characterization for power: Evaluating control-flow-based and event-counter-based techniques. In *HPCA-12*. Princeton University, February 2006.

[14] R. Joseph and M. Martonosi. Run-time power estimation in high performance microprocessors. In *ISLPED '01: Proceedings of the 2001 international symposium on Low power electronics and design*, pages 135–140, New York, NY, USA, 2001. ACM.

[15] K.-J. Lee and K. Skadron. Using performance counters for runtime temperature sensing in high-performance processors. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 11*, page 232.1, Washington, DC, USA, 2005. IEEE Computer Society.

[16] T. Li and L. K. John. Run-time modeling and estimation of operating system power consumption. *SIGMETRICS Perform. Eval. Rev.*, 31(1):160–171, 2003.

[17] A. Merkel and F. Bellosa. Balancing power consumption in multiprocessor systems. *SIGOPS Oper. Syst. Rev.*, 40(4):403–414, 2006.

[18] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, and R. Rajkumar. Critical power slope: understanding the runtime effects of frequency scaling. In *ICS '02: Proceedings of the 16th international conference on Supercomputing*, pages 35–44, New York, NY, USA, 2002. ACM.

[19] D. Parikh, K. Skadron, Y. Zhang, M. Barcella, and M. R. Stan. Power issues related to branch prediction. In *Proceedings of the 8th International Symposium on High-Performance Computer Architecture*, pages 233–, Washington, DC, USA, 2002. IEEE Computer Society.

[20] M. J. Pazzani and S. D. Bay. The independent sign bias: Gaining insight from multiple linear regression. In *In Proceedings of the Twenty First Annual Conference of the Cognitive Science Society*, pages 525–530, 1999.

[21] A. Phansalkar, A. Joshi, and L. K. John. Analysis of redundancy and application balance in the spec cpu2006 benchmark suite. *SIGARCH Comput. Archit. News*, 35(2):412–423, 2007.

[22] A. Phansalkar, A. Joshi, and L. K. John. Subsetting the spec cpu2006 benchmark suite. *SIGARCH Comput. Archit. News*, 35(1):69–76, 2007.

[23] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2005. ISBN 3-900051-07-0.

[24] SBSIF. SMART specifiaction rev.1.1 Dec 11, 1998. [Online] Available: http://sbs-forum.org.

[25] K. Singh, M. Bhadauria, and S. A. McKee. Real time power estimation and thread scheduling via performance counters. *SIGARCH Comput. Archit. News*, 37(2):46–55, 2008.

[26] M. Steinder, I. Whalley, J. E. Hanson, and J. O. Kephart. Coordinated management of power usage and runtime performance. In *Proceedings of the 11th Network Operations and Management Symposium*, pages 387–394. IEEE, 2008.

[27] W. Wu, L. Jin, J. Yang, P. Liu, and S. X.-D. Tan. A systematic method for functional unit power estimation in microprocessors. In *Proceedings of the 43rd annual Design Automation Conference*, pages 554–557, New York, NY, USA, 2006. ACM.