

VM Power Metering: Feasibility and Challenges

Bhavani Krishnan, Hrishikesh Amur, Ada Gavrilovska, Karsten Schwan
Center for Experimental Research in Computer Systems (CERCS)
Georgia Institute of Technology, Atlanta
{bhavani.krishnan, amur}@gatech.edu, {ada, schwan}@cc.gatech.edu

Abstract

This paper explores the feasibility of and challenges in developing methods for black-box monitoring of a VM’s power usage at runtime, on shared virtualized compute platforms, including those with complex memory hierarchies. We demonstrate that VM-level power utilization can be accurately estimated, or estimated with accuracy with bound error margins. The use of bounds permits more lightweight online monitoring of fewer events, while relaxing the fidelity of the estimates in a controlled manner. Our methodology is evaluated on core i7 and core2 x86 platforms, running synthetic and SPEC benchmarks.

1. Introduction

The continuing, unsustainable increases in datacenter power consumption are causing researchers in academia and in industry to be heavily invested in addressing the issue [7, 1, 15, 13, 6]. Efforts to develop power-aware datacenter management techniques range from multi-scale methods for managing IT system power usage [10], to power capping to deal with increased server densities [4, 5, 12], to integrating into the management processes information regarding the datacenter cooling infrastructure, the latter aimed at improving facility-level metrics like PUE [14, 13, 6].

A common element of power-aware management is its exploitation of virtualization technology for server consolidation and for dynamic load distribution and/or load balancing (i.e., VM migrations). This is true in private virtualized datacenters as well as in emerging cloud computing systems. The increased freedom virtualization presents to mapping IT loads to machines, however, also implies the need for efficiency in the management actions being taken, to achieve, for instance, some desired power state using a limited number of reconfigurations and VM migrations [9]. Consider, for example, the data presented in Table 1, which shows the power utilization for 3 VMs. When load migration is needed, a random decision may result in the migration of VM1, which may be insufficient for meeting some lower target power cap. The quality of the decision being made may be improved by estimating the VM’s power consumption via its current average CPU utilization, as shown useful in [7], but unfortunately, such estimates can be imprecise. This is because CPU utilization includes the time spent waiting for memory and thus, does not accurately reflect actual CPU usage and thus, CPU power consumption. For instance,

Table 1: Dynamic power for VMs on core2 platform.

	CPU Utilization	Power(W)
vm1	50%	6
vm2	100%	12
vm3	100%	42

VM2 and VM3 in Table 1 both show 100% CPU utilization, but VM3 is memory bound while VM2 is only CPU bound. As a result, despite having the same CPU utilization, the power contributions of the two VMs are quite different.

The data in Table 1 illustrates limitations in using overly simplistic models of per-VM power usage. Having such per-VM information, however, can be of substantial value: (1) power-aware management methods benefit in terms of minimizing the number of VM migrations or other reconfiguration actions required to achieve some given power state; (2) datacenter administrators can use it to develop customer-facing billing or charge-back policies; and (3) environmentally-responsible consumers of datacenter and cloud resources can employ it to minimize their workloads’ carbon footprints. Also evident from the data in Table 1 should be that it is not trivial to accurately capture per-VM power usage in modern datacenter systems. This is because today’s datacenters and emerging compute clouds host a broad range of workloads with diverse resource utilization requirements, which furthermore, often exhibit significant levels of dynamism. In addition, VM-level information regarding the applications it executes and their behavior is typically not available to the operators who are responsible for managing the datacenter/cloud infrastructure. “No one will tell us what their applications are doing” is a common message we have heard from the many datacenter operators with whom we have interacted. What is needed, therefore, are black-box techniques for capturing a VM’s power usage, using existing platform-level monitoring methods and offering degrees of accuracy similar to what past work has shown possible for statically developed VM energy profiles in [7].

The goal of our research is to understand the feasibility of and challenges in developing methods for black-box monitoring of a VM’s power usage at runtime, on the shared and virtualized compute platforms used in modern datacenter and cloud computing infrastructures. To carry out this task and similar to other ongoing work on VM power metering [1], we construct power models by correlating a VM’s usage of specific types of resources to system power consumption, so as to later use these

models to continually estimate the power consumed by each VM. Unlike prior work, however, we also aim to understand the challenges due to the complex memory hierarchies present on current and next-generation hardware. This is particularly important as memory is becoming a significant component of the system memory usage on new hardware platforms [2]. Finally, our goal is to understand the feasibility of creating lightweight power metering mechanisms that do not require full system instrumentation and monitoring, so that they can be used efficiently with online management methods.

The resulting technical contributions of this paper are as follows. Using a broad range of workloads exhibiting different CPU and memory usage patterns, derived from a synthetic benchmark and from the SPEC benchmark suite, we demonstrate the ability to accurately estimate an individual VM’s contributions to platform power consumption. The data gathered during our model construction and experimental analysis indicates that estimation is achievable by considering both the VM’s CPU and memory resource utilization. Furthermore, the accuracy of the estimates is strongly dependent on additional insights into the VM’s usage of the memory system, such as their utilization of different levels of the cache hierarchy or attained memory-level parallelism. In the absence of such additional information (due to increasing overheads of finer-grain monitoring of low-level hardware counters or need for additional application instrumentation), we demonstrate that it is still feasible to establish bounds on a VM’s power usage. This is in contrast to observations made in [7] which claim that dynamic estimations are not feasible. We hypothesize that this is mainly due to the fact that they ignore memory hierarchy contributions to power usage. In addition, we experimentally demonstrate that using black box methods, it is possible to estimate the per VM power usage, contrary to arguments made in [7, 8]. Finally, we demonstrate that the model construction and the runtime measurements gathered for online estimation must be performed with consideration of detailed architectural features, including the design of the platform’s cache architecture.

2. Power Metering Methodology

The basic idea behind the VM power metering approach explored in our research is (1) to first establish a power model for the various system resources present on a given platform. We do this by correlating the utilization level of the specific resource to the overall system power, when other types of resources are maintained at extremely low utilization levels. Next, (2) at runtime, using lightweight monitoring tools, we measure the per-VM utilization of various resources. Our current implementation targets platforms virtualized with the Xen hypervisor, and relies on *xenoprof* and the available hardware counters for online profiling. Finally, (3) the VMs power usage is estimated based on the power levels corresponding to the appropriate resource utilization, as derived in the resource’ power models, and few additional factors, such as number of active cores or sockets, to include the often significant transition costs from activating a system component.

The total system power consumed by the server can be written as follows.

$$Dynamic P_{server} = P_{idle} + P_{cpu} + P_{mem} + P_{disk} + P_{io}$$

We measure the idle power by keeping all cores and the memory subsystem idle. Disk has been modeled successfully in pre-

vious work [7, 1] and hence we do not focus on it in this paper. In addition, for the target hardware and using benchmarks such as *iPerf*, we determine that the contribution of network I/O utilization to the total system power is very low. Therefore, we focus on the CPU and memory subsystems and show estimations for benchmarks which are CPU bound, memory bound or combinations of those.

Since our goal is to explore the feasibility of lightweight VM power metering methods, we limit the amount of information we monitor to only few type of events. Specifically, we use hardware counter values for instructions retired per second (*inst_ret/s*) and last-level cache misses per second (*llc_miss/s*). As a result, a single type of event may represent more than one resource utilization state. For instance, the same value for *inst_ret/s* may correspond to certain CPU utilization and its accompanying power level, or, in the event of a cache-bound workload, it may represent cache plus different CPU utilization value, resulting in different power usage. Similarly, the same *llc_miss/s* value may correspond to different levels of memory utilization depending on memory-level parallelism and memory overlap which exists on a cache miss.

In order to avoid the development of fine-grain models for every single platform element, and the subsequent need for runtime monitoring of all of the corresponding hardware events, we explore the possibility of establishing bounded models. The goal is to limit the required monitoring state and runtime overheads, while at the same time understanding the level of estimation accuracy.

2.1 Modeling

We next apply the methodology described in the previous section to establish the CPU and memory power models for two different platforms, a dual-socket quad core Nehalem core i7-based system, and a quad-core core2-based Xeon system, termed Nehalem Core i7 and Xeon Core2 in the remainder of this paper, respectively. The Nehalem Core i7 has 12GB RAM and 8MB last-level (L3) cache. The Xeon Core 2 has a 6MB last-level (L2) cache.

2.1.1 CPU Model

The goal of the CPU model is to accurately model the power consumed by the CPU subsystem. This includes the processor power and the cache power. As explained previously, we monitor the instructions retired per second (*inst_ret/s*) to understand the CPU usage of a VM. We build the model by correlating *inst_ret/s* and the dynamic server power. The mapping from instructions retired per second to CPU power is complicated by the fact that memory references that hit in different levels of the cache hierarchy consume different amounts of power. Therefore, an observation of *x* instructions retired per second will consume much less power if each of those instructions executes on the CPU or hits in L1 as compared to the case where each of the references hits only in the last level cache (LLC). This is shown in Figure 1. In [7] the authors also present experimental evidence that variable power usage is observed for the same processor utilization levels, without providing rationale for these observations. We hypothesize that the variability they observe is due to the same cache-related argument as above. In order to limit the need for fine-grain monitoring of the utilization for each level cache, we explore the utility of building a CPU power model using the *inst_ret* counter only. Our approach is to provide bounds on CPU power for any executing

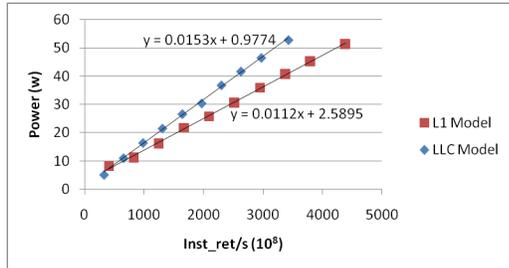


Figure 1: Cpu Model for L1 and LLC

workload by considering the extreme cases. To build the baseline model, we build a custom benchmark, which allows us to vary the CPU utilization on a core and the level of memory hierarchy being accessed (cache or memory). It achieves a desired CPU utilization by performing strided access to an array for $x\%$ of time and sleeping for the rest of the time in a given time interval. By varying the size of the array, we can vary its working set size and make it L1-bound vs. L2-bound and so on, up to the system’s last level cache (LLC). We use `oprofile` to read the hardware counter values and obtain the `inst_ret/s`.

Figure 1 shows the results of the benchmark execution with an L1- vs. LLC-bound (L3-bound) workload on the Nehalem platform. As expected, the L1- and LLC-bound workload exhibit linear relationship between `inst_ret/s` and power, with the LLC graph having greater slope due to the more power-hungry nature of last level caches. We observe that 100% cpu utilization for a LLC-bound process results in 22% fewer `inst_ret/s` than a 100% cpu utilization for L1 bound process. In addition, for the same value of `inst_ret/s`, we observe different power values. Therefore, using the single `inst_ret/s` counter, we build two models, one for the first level of cache and another for the last level of cache. Any program would lie between these two extremes, thus they represent the bounds for the CPU power contribution of a VM with the given `inst_ret/s` value. At runtime, through use of lightweight heuristics, it may be possible to further determine which bound the VM power is closer to.

We also observe that since the machine we are using has deep low power states (up to C6) [16] the power it consumes to wake up from deep sleep state to active state is significant. The models shown above include that cost. We measure the power consumed by a core at 100% L1 utilization to be 51W but 41W of this is attributed to the wakeup cost of the package. If we activate another core on the same socket and run it at 100% CPU utilization with all accesses hitting L1 cache, the power increases by 11W. Instead, if we activate a core on another socket, the resulting increment is 11+2W. The same applies to LLC model also. This information is used to make the aforementioned adjustments to the VM power estimation process.

2.1.2 Memory Model

The goal of the baseline memory model is to accurately model the power consumed by the memory subsystem. As with the CPU model, we limit the amount of monitoring information used. We look only at the last level cache misses per second (`llc_misses/s`) to understand the memory usage of a VM, and build the memory power model by correlating `llc_misses/s` and the dynamic server power.

Similarly to the CPU case, the use of this one event only does

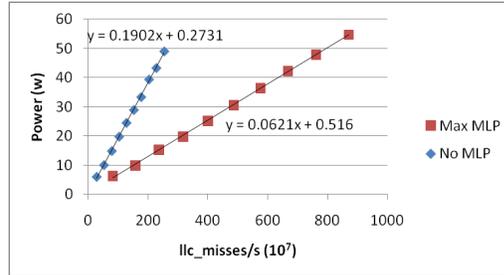


Figure 2: Memory Model for MAX MLP and NO MLP.

not accurately represent the exact memory usage of a workload. The memory subsystem, consisting of the front-side bus (FSB) in some architectures, the memory controllers and the DRAM modules, typically supports a fixed maximum number of outstanding memory accesses at any given time. Therefore, depending on the nature of the application, this value may be reached in practice. But the memory access pattern in some applications may be such that a memory access depends on the value returned by the previous memory access in which case the parallelism offered by the memory subsystem cannot be exploited. While this is rare, we observe this pattern in the 471.omnetpp SPEC benchmark which uses the OMNet++ discrete event simulator to model a large Ethernet campus network. This complicates the mapping from the rate of LLC misses to memory power as shown in Figure 2. For example, at 2.3×10^9 LLC misses/sec the memory power can be either 45W or 15W for the cases of no parallelism vs. maximum memory-level parallelism (MLP) respectively. Therefore, we provide bounds on the memory power for a workload by considering the two extreme cases. We use the same custom benchmark as for the CPU model, but allocate a large memory array and make strided access beyond the cache-line boundary to ensure that every access misses the LLC. We also turn hardware prefetching off.

Figure 2 shows the two models for the memory baseline for the Nehalem platform – one for maximum MLP and another for no MLP. For any real program, the MLP would lie between these extremes and thus we can use these two models to estimate bounds for the power consumption. We observe that both models exhibit linear behavior, with drastically different slopes, since for a given `llc_miss/s` value, the Max MLP model results in higher memory utilization compared to the No MLP model. To address this, we can rely on the use of a staging server [7, 11] to run the VM standalone and learn which bound is it closer to. Techniques such as those described in [3] can also be used to determine the MLP of a given workload and to further narrow down the bounds. As in the case of the cpu model, the above graph includes the cost for waking up the package.

For brevity, we do not include in the paper the CPU and memory models for the Xeon Core2 platform.

2.1.3 Validation

Next, we validate the models by conducting a set of simple experiments. We refer to a core running a CPU-bound workload with 100% cpu utilization – `1cpu`, and a memory-bound workload with 100% memory utilization – `1mem`. In the first experiment, we increase the number of cores at `1cpu` from 1 to 8. The goal of this experiment is to show that CPU power is additive as we keep increasing the number of workloads. Note

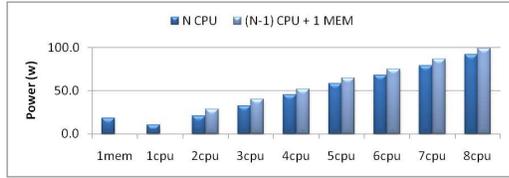


Figure 3: NCPU + 1MEM for Core i7.

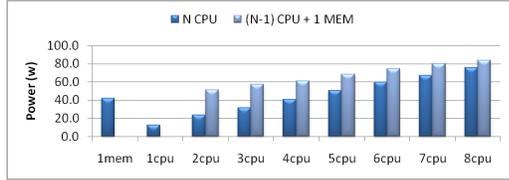


Figure 4: NCPU + 1MEM for Core2.

that we measure the performance of the workloads, and determine that there is no change in their execution time and performance. We run the second experiment on two cores one with 1cpu and another with 1mem load. Then, we keep increasing the number of cores at 1cpu from 1 to 7 in addition to the core running the memory-bound 1mem VM. The goal of this experiment is to show that the CPU and memory power is additive as we keep increasing the workload.

Figure 3 shows the ncpu + 1mem data for the Nehalem platform. We observe that 1cpu + 1mem gives a power value which is equal to the addition of running them individually. Similarly, each additional core running 1cpu increases the power by 11W (except when we activate the other socket as previously explained).

We repeat the validation for the Core2 platform, using the corresponding CPU and memory models. The first experiment shows the expected results with each additional core running 1cpu increasing the total power by 9w. The second experiment however yields unexpected results. We observe that the ncpu + 1mem power does not equal the addition of dynamic power values of running them individually, in spite of the fact that the benchmarks’ execution times and performance remain unchanged. Furthermore, with each additional core running 1cpu, the increase in total power is reduced, which leaves an impression of diminished contribution of the memory VM (see Figure 4).

To understand the cause of this behavior, we consider the architectural differences between the two platforms, particularly with respect to the cache design. Core2 has exclusive caches, where running a single memory intensive benchmark causes large amounts of snooping in the L1 caches of the other cores. This means that the L1 caches of cores that do not run any load are still ‘turned on’ when another core is running a memory-intensive benchmark, which consumes a surprising amount of power. This makes an accurate estimation of the power consumption of the individual VMs difficult. In contrast, the Nehalem Core i7 has an inclusive cache.

To further verify that the design of system caches has implications on the system power, we measure the snooping traffic for Xeon Core2. We use the EXT_SNOOP hardware counter, which measures the responses to external snoop requests (at the LLC level). Every LLC miss is snooped by the other cores and a response is sent to the requesting core. Figure 5 shows the values for EXT_SNOOP when we run first 1mem, then 1cpu

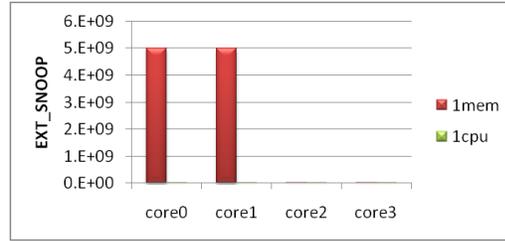


Figure 5: Snoop traffic for 1mem and 1cpu.

on a single socket quad-core Core2 platform with two 3MB L2 caches, each shared by two cores. As we can see, the snoop traffic generated by the 1mem is substantial. This traffic represents the snoop responses sent by core2 and core3 (which share a LLC) to the LLC misses in the other cache. This means that due to the high number of snoops in the exclusive caches of core2 and core3, those cores are not idling even when the memory intensive process is running on core1. This leads to a problem, since the memory power baseline includes this additional cost. Therefore, when another process is run on core2 or core3, the expected increase in power does not occur because the caches are already on and performing lookups due to snooping.

3. Evaluation

Experimental Testbed. We run our experiments on the Nehalem architecture described in the previous section. We measure the power by directing the power supply through a WattsUp power meter. We then used the provided Linux utilities to get the power dump. VM Profiling was done using oprofile v0.9.3 which includes support for passive profiling of VMs. We use the SPEC2006 benchmarks for evaluating the baseline models described in the previous section. For each benchmark we estimate four power values based on: the L1 cpu baseline model (PL1), the LLC cpu baseline model (PLLC), the MLP memory baseline model (PMLP), and the NO_MLP memory baseline model (PNO_MLP). Then we compute the following four bounds.

$$\begin{aligned}
 B1 &= PL1 + PMLP & B2 &= PLLC + PMLP \\
 B3 &= PL1 + PNO_MLP & B4 &= PLLC + PNO_MLP
 \end{aligned}$$

We observe that most benchmarks are close to one of these bounds. The following subsections discuss the results for various SPEC2006 benchmarks.

CPU-bound benchmarks. 401.bzip2, 444.namd, and 464.h264-ref are CPU intensive benchmarks with high L1 hit rate. The measured power is expected to be close to the estimated B1 (PL1 + PMLP) or B3 (PL1 + PNO_MLP) bounds. Figure 6 shows that using the L1 model the VM power can be estimated within 6% of measured value. 445.gobmk, 458.sjeng, and 453.povray are CPU intensive benchmarks which hit in LLC. The measured power is expected to be close to the estimated B2 (PLLC + PMLP) or B4 (PLLC + PNO_MLP) bounds. Figure 6 shows that the LLC estimations are within 10% of the measured power whereas the L1 model results in 20% underestimated values for these benchmarks. The memory power contribution for these VMs is insignificant.

Memory-bound benchmarks. 462.libquantum, 433.milc, and 470.lbm are memory bound benchmarks with high MLP. The measured power is expected to be close to the estimated B1 (PL1

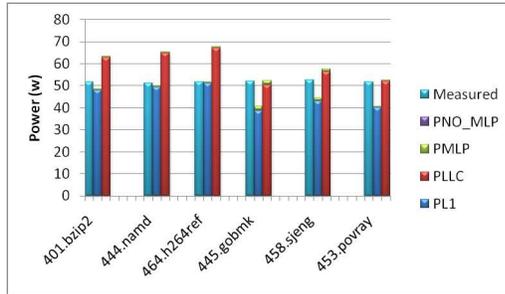


Figure 6: Estimations for CPU bound SPEC2006 benchmarks.

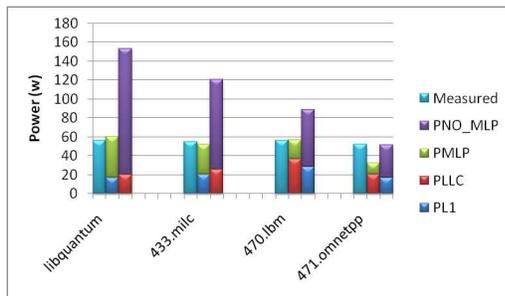


Figure 7: Estimations for SPEC2006 benchmarks with MLP.

+ PMLP) or B2 (PLLC + PMLP) bounds. Figure 7 shows that we can estimate the VM Power within 7% of measured value.

471.omnetpp is a memory bound benchmark which shows no MLP. The measured power is expected to be close to the estimated B3 (PL1 + PNO_MLP) or B4 (PLLC + PNO_MLP) bounds. The NO_MLP estimations are within 1% of the measured power whereas the max MLP model results in 38% underestimated values for this benchmark.

Multiple VMs. Next, we run combinations of CPU intensive and memory intensive benchmarks and estimate the power. Figure 8 shows the measured vs. estimated power for several workload mixes. EXPT1 consists of 444.namd + 470.lbm, EXPT2 of 453.povray + 471.omnetpp, and EXPT3 includes 4 VMs running 453.povray + 444.namd + 464.h264ref + 470.lbm. We find that we are able to estimate the power contribution of each VM within 5% accuracy in each of the described experiments.

Monitoring Overheads. Finally, we also compare the overhead imposed by the continuous access to the `inst_ret/s` and `llc_miss/s` hardware counters, necessary to perform online profiling. We establish that the use of online monitoring results in only slight degradation (up to 3%).

4. Conclusion

The measurements presented in this paper demonstrate that our power metering methodology can result in accurate estimates, or in estimates with bound fidelity, if no additional information is available regarding the VMs utilization of different levels of cache hierarchy or attained MLP. This shows that black-box power metering is feasible and its accuracy can be established. In addition, our approach can be supplemented with techniques which use heuristics to classify a workload, or which dynamically ‘zoom in’ and turn on fine grain event moni-

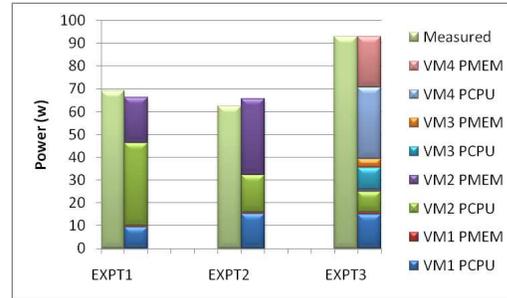


Figure 8: Estimations for mixed experiments.

toring, when interesting trends are detected or when the fidelity of the estimation drops below target accuracy levels. Our future work will continue to explore the tradeoffs associated with dynamic power-metering methods for virtualized platforms.

5. References

- [1] BOHRA, A., AND CHAUDHARY, V. Vmeter: Power Modelling for Virtualized Clouds. In *IPDPS* (2010).
- [2] CARTER, J. Looming Challenges in Server and Data Center Energy Efficiency, One Researcher’s Perspective. CERCS Energy Workshop. 2010.
- [3] CHOU, Y., FAHS, B., AND ABRAHAM, S. Microarchitecture Optimizations for Exploiting Memory-Level Parallelism. In *HPCA* (2005).
- [4] HP Data Center Management Solution Reduces Costs by 34 Percent. www.hp.com/hpinfo/newsroom/press/2007/070625xa.html.
- [5] IBM Helps Clients “Meter” Datacenter Power Usage to Help Lower Energy Costs. www-03.ibm.com/press/us/en/pressrelease/19695.wss.
- [6] IBM, Syracuse University, New York State to Build One of the World’s Most Energy-Efficient Data Centers. www-03.ibm.com/press/us/en/pressrelease/27612.wss.
- [7] KANSAL, A., KOTHARI, N., AND BHATTACHARYA, A. Virtual Machine Power Metering and Provisioning. In *ACM SOCC* (2010).
- [8] KOLLER, R., VERMA, A., AND NEOGI, A. WattApp: An Application Aware Power Meter for Shared Data Centers. In *ICAC* (2010).
- [9] KUMAR, S., TALWAR, V., ET AL. vManage: Loosely-coupled Platform and Virtualization Management in Datacenters. In *ICAC* (2009).
- [10] KUSIC, D., KEPHART, J., ET AL. Power and Performance Management of Virtualized Computing Environments via Lookahead Control. In *ICAC* (2008).
- [11] NATHUJI, R., AND KANSAL, A. Q-Clouds: Managing Performance. Interference Effects for QoS-Aware Clouds. In *Eurosys* (2010).
- [12] NATHUJI, R., AND SCHWAN, K. VPM Tokens: Virtual Machine-Aware Power Budgeting in Datacenters. In *HPDC* (2008).
- [13] NATHUJI, R., SOMANI, A., SCHWAN, K., AND JOSHI, Y. CoolIT: Coordinating Facility and IT Management for Efficient Datacenters. In *USENIX HotPower* (2008).
- [14] SAMADIANI, E., AMUR, H., KRISHNAN, B., AND SCHWAN, K. Coordinated Optimization of Cooling and IT Power in Data Centers. *ASME Journal of Electronics Packaging* (2010).
- [15] STOESS, J., LANG, C., AND BELLOSA, F. Energy Management for Hypervisor-based Virtual Machines. In *USENIX* (2007).
- [16] STRONG, B. A Look Inside Intel: The Core (Nehalem) Microarchitecture.