Power Consumption Estimation Models for Processors, Virtual Machines, and Servers

Christoph Möbius, Waltenegus Dargie, *Senior Member, IEEE,* and Alexander Schill

***Abstract*—The power consumption of presently available Internet servers and data centers is not proportional to the work they accomplish. The scientific community is attempting to address this problem in a number of ways, for example, by employing dynamic voltage and frequency scaling, selectively switching off idle or underutilized servers, and employing energy-aware task scheduling. Central to these approaches is the accurate estimation of the power consumption of the various subsystems of a server, particularly, the processor. We distinguish between power consumption measurement techniques and power consumption estimation models. The techniques refer to the art of instrumenting a system to measure its actual power consumption whereas the estimation models deal with indirect evidences (such as information pertaining to CPU utilization or events captured by hardware performance counters) to reason about the power consumption of a system under consideration. The paper provides a comprehensive survey of existing or proposed approaches to estimate the power consumption of single-core as well as multicore processors, virtual machines, and an entire server.**

***Index Terms*—Power consumption models, energy-efficiency, server's power consumption, processor's power consumption, virtual machine's power consumption, power consumption estimation**

I. Introduction

The power consumption of computing devices and systems has always been a research concern. In the past, this mainly concerned battery-operated mobile devices and embedded systems, but at present the prevailing focus is on large-scale Internet servers and data centers. The energy consumption of the information and communication technology (ICT) infrastructure has been globally and steadily increasing. According to Koomey, the amount of power consumed by data centers (servers, storage, communication, and (cooling) infrastructure) worldwide was 70.8 TWh in 2000 and 152.5 TWh in 2005 (a 115% rise in 5 years) [1]. These figures amount, in respective order, to 0.53 and 0.97% of the overall worldwide energy spending. A latest study by the same author estimates the worldwide power consumption between 2005 and 2010 due to data centers to have been between 203.4 TWh and 271.8 TWh (with respect to the overall worldwide energy spending in 2005, this amounts to an increment of 33% to 78%) – the corresponding portion of the total worldwide power consumption likewise is estimated to have been between 1.12% and 1.50% [2].

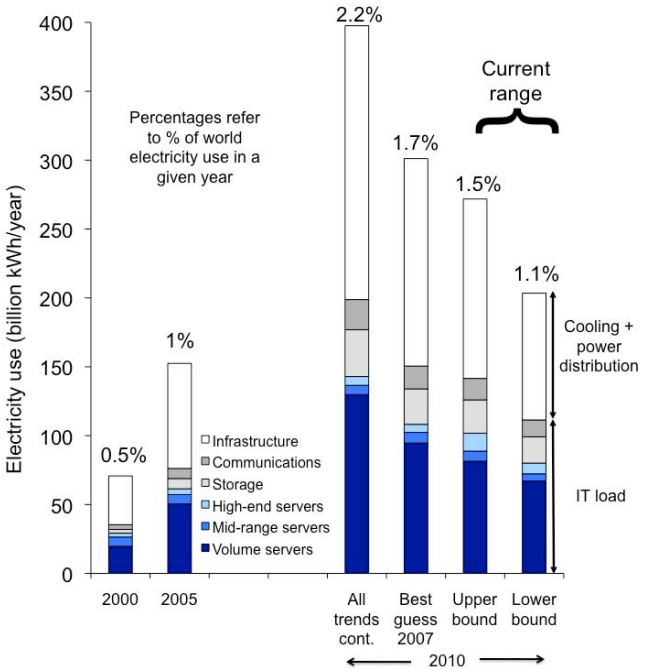

Figure 1. The estimated worldwide energy consumption for data centers between 2000 and 2010 [2].

Figure 1 provides a compact summary of these numbers. As can be seen in the figure, the power consumption of data centers increased year by year. However, beginning from the year 2007, there is a gradual slowdown in the energy spending. One of the reasons given by Koomey is the adoption of resource virtualization by the ICT industry. Even so, there is still a crucial need to improve the energy efficiency of server systems, since high energy consumption does not necessarily correspond to high performance (quality of service) [3] .

The existing or proposed approaches to achieve energy efficiency encompass hardware as well as software solutions. Examining the hardware solutions falls beyond the scope of this paper. The software solutions, however, can be broadly categorized into scaling [4], scheduling [5], [6], and consolidation [7] approaches.

The scaling approaches primarily target the processor subsystem of a server and aim to adapt its operational voltage and

Manuscript was first received on 24 November 2012.
Manuscript revised on 20 April 2013.

C. Möbius is with the Technical University of Dresden, Chair of Computer Networks, 01062, Dresden, Germany. Tel.: +49 351 463 38002. Fax: +49 351 463 38251. E-mail: christoph.moebius@tu-dresden.de

W. Dargie is with the Technical University of Dresden, Chair of Computer Networks, 01062, Dresden, Germany. Tel.: +49 351 463 38352. Fax: +49 351 463 38251. E-mail: waltenegus.dargie@tu-dresden.de

A. Schill is with the Technical University of Dresden, Chair of Computer Networks, 01062, Dresden, Germany. Tel.: +49 351 463 38263. Fax: +49 351 463 3826351. E-mail: alexander.schill@tu-dresden.de

This work has been partially funded by the German Research Foundation (DFG) under agreement SFB 912/1 2011.

frequency according to the task arrival rate at the processor. This approach reduces the idle power consumption of the processor and forces the task scheduler in the operating system to avoid over-provisioning of resources and to execute less time critical jobs at a slower speed.

Scheduling approaches can be low-level or high-level. The low-level approaches refer to energy-efficient schedulers within the operating system whereas the high-level approaches refer to load-balancing (or load management) approaches at the application layer. Energy-efficient schedulers often aim to schedule tasks (jobs) in a way hardware resources are efficiently utilized and unnecessary contention between jobs for hardware resources (such as memory or CPU) are avoided. The load-balancing solutions at the application layer are relevant for a cluster environment (such as a data center) in which multiple servers are used to handle user requests. Hence, an energy-efficient load-balancer aims at selecting m out of n servers, $m \leq n$, that can handle the overall workload of the entire cluster without violating a service-level agreement (SLA). Ideally, m is chosen such that $\forall i, i < n, i \neq m$ it is not possible to improve energy-efficiency and not to violate SLA at the same time.

The last category of approaches, consolidation, particularly concerns virtualized environments. Similar to the load-balancing approaches, the aim is to balance the demand for computing resources with the supply of resources, but this is mainly achieved through service (workload) consolidation and the live migration of virtual machines (services/applications) from one physical machine to another. In this process, idle or underutilized servers are switched off and overloaded servers are relieved of a portion of their load.

All of these approaches share some features to achieve their goal. Two of these features are (1) the prediction of the short term workload of a server, a data center or a cluster and (2) the estimation of the resource requirement and the associated power (energy) consumption of the workload. Without theses two key features it is not possible to carry out energy-efficient adaptation and to objectively quantify the gains of the adaptation.

Measuring the overall AC power consumption of a server is straightforward but does not give sufficient insight into the characteristic of the workload that is being executed. A server may consume a little amount of power and yet may not be able to process additional requests if there is a bottleneck of resources somewhere. Conversely, a server may consume a large amount of power and yet it may have enough resources to accommodate incoming requests (workloads). A server dominated by IO-bound workloads is an example of the first scenario while a server dominated by CPU-bound workloads is an example of the second scenario. Such knowledge is useful to efficiently distribute incoming workloads and to schedule complementary services.

The main focus of this paper is to investigate some of the existing or proposed approaches aiming to estimate and reason about the power consumption of a server and some of its subsystems. The power consumption of a computing system may be measured or estimated. The first refers to the application of a mechanism to measure the actual power consumption of the system. The mechanism can in turn be direct or indirect. In a direct mechanism the electric lines supplying power to the system are intercepted and either power measuring devices [4], [8], [9] or shunt resistors ($< 100m\Omega$) [10] are inserted in between. In an indirect mechanism, the induction law and lossless inductors are employed to measure the electromagnetic fields created by the current that is drawn by the system [11], [12].

Estimation approaches use hardware-provided or OS-provided metrics to reason about the power consumption of a system, assuming that there is a correlation between the values of these metrics and the power consumption. The hardware-provided inputs generally refer to performance monitoring counters (PMC) which are provided by the CPU [13], [14], [15]. While it is possible that other subsystems may provide similar performance monitoring counters, no publication is known to the authors on this subject. The OS-provided inputs refer to metrics which indicate the utilization level of the system. A typical example is the CPU utilization indicator. Such metrics are not directly provided by the hardware, but are rather computed by the operating system [11], [16], [17]. Models that employ both PMC and OS-provided inputs are referred to as hybrid [8], [18], [19].

The measurement and modeling approaches have their merits and demerits. Direct measurements make few assumptions about the architecture of the system or the type of workload being processed by the system, but they are often expensive and obtrusive. Moreover, they should deal with complex electrical circuits. Likewise, the indirect measurement mechanisms can be sensitive to interference. Estimation models, on the other hand, have several amenable features, but they can be inaccurate and their scope and usefulness are mostly determined by the system architecture as well as the type of workload being processed. In the subsequent sections, we will investigate these approaches more closely.

The rest of the paper is organized as follows: In section II, we will give a brief overview of the direct measurement approaches. In section III, we will discuss performance monitoring counters and benchmarks as the essential components of a wide range of existing or proposed power estimation models. In section IV, we will present some of the proposed models to estimate the power consumption of the CPU of a server. In section V, we will present power estimation models that reason about the power consumption of virtual machines. In section VI, we will present power estimation models for an entire server. In section VII, we will give a detailed analysis of the estimation models and the associated estimation errors - the deviation between model output and true values. Finally, in section VIII, we will give concluding remarks.

II. Direct Measurements

Bohrer et al. [20] use the direct measurement approach to quantify the DC power consumption of a web server. They assert that five of the power rails (lines) in the motherboard of the server they considered are sufficient for the task. These are a 3.3V line supplying power to the memory, the video card, and the network interface card (NIC); a 5V line supplying

power to the CPU; a 5V line supplying power to the disk drive's controlling mechanism; a 12V line supplying power to the disk drive's mechanical parts; and a 12V line supplying power to the processor's cooling fan. They inserted a shunt resistor between each line to measure the voltage drop across the shunt resistor. The voltage drop is supplied to a 10KHz low pass filter and amplified by custom hardware. The output of the amplifier is then supplied to a National Instruments PCI-6071E data acquisition card. The authors claim that the measurement error is within 5% range and associate the error with the shunt resistors and the amplifier offset voltage. The same approach is employed by Elnozahy et al. [21].

Similarly, Economou et al. [22] develop custom hardware to measure the power consumption of the subsystems of a blade server. They use four power rails to carry out the measurements: a 12V line that is predominantly used by the processor and memory; a 5V line that supplies power to the disk drive; and a 5V auxiliary line and a 3.3V line supplying power to the remaining components. The 12V line is branched to provide the processor and the memory subsystems with power. The authors report modifying the motherboard's network structure to insert an extra shunt resistor, but do not provide information regarding the modification process. Neither do they describe where exactly the shunt resistor is added. Furthermore, details are missing pertaining to the employed measurement devices and the acquisition board.

Dargie and Schill [4] investigate the power consumption of the various subsystems of a multimedia server (Fujitsu-Siemens D2641) cluster under different load balancing policies. They first investigate how the voltage regulators of the different subsystems are supplied with DC power and observe that the main voltage regulator that generates the processor core voltage predominantly uses a separate 12V line while the voltage regulator of the memory unit predominantly uses a 5V line. Likewise, the voltage regulator of the Southbridge uses a 12V line. All the I/O controllers draw current through the 3.3V line. Unfortunately, some of the power rails supply multiple voltage regulators with power while some of the voltage regulators use multiple rails as their input source complicating the measurement task. For the components that are not accessible (for example, the NIC which is connected to the PCI Express slot), the authors provide a custom made raiser board and intercepted the 3.3V power line of the raiser board to measure the power consumption. The authors analyze the relationship between the probability distribution functions of the workload of the server and the power drawn through the various power rails to estimate the power consumption of the various subsystems.

Bircher et al. [10] measure the power consumption of an IBM x440 server using on-board current-sensing resistors that are originally used by the inherent server management system for overcurrent protection[1]. They identify five such resistors for the CPU, the memory, the Southbridge, the disk drives, and the chipset. Measuring the voltage drop across these resistors enables to compute the power consumption of the respective subsystems. To reduce the effect of noise and to improve the sampling resolution, the authors developed an amplifier board and amplified the voltage drops across the shunt resistors. The amplifier outputs are then quantized by a National Instruments AT MIO-16E-2 card, which is able to sample 8 channels in parallel at 10KHz. Pre-instrumented server hardware such as this simplifies the measurement task. However, not all server architectures are produced with ready-to-use and cleanly separated measurement points.

Measurement techniques, both direct and indirect, reach limits due to several constraints: Firstly, measuring the power consumption of a system as proposed by [20], [22], [4] does not easily scale to clusters or data centers. Secondly, intercepting and instrumenting power rails manually requires effort and may affect the performance of the physical machines. Thirdly, acquiring reliable and precise measurement devices is expensive; the cost becomes pronounced if several subsystems are measured in parallel (for example, modern server boards usually employ a distinct power supply for each processor socket and each power supply provides multiple 12V power lines for safety reason [23]). Fourthly, the approaches require insight into the mainboard layout to ensure that power supply lines are associated with the correct subsystems. Usually such knowledge is not available to the customer, but is rather a protected asset of the board manufacturer. Finally, some of the components whose power consumption should be measured may not be physical at all. This is typically the case in virtualized environments where the power consumption of virtual machines should be estimated. Therefore, developing models using alternative approaches can be a meaningful solution.

[1]Most voltage regulators provide a lossless current sensing inductor to enable overcurrent protection.

III. Estimation Models

A power consumption estimation model takes indirect evidences pertaining to the utilization (activity) of a system under consideration and reasons about its power consumption. There are two essential steps to carry out this task: the selection of the model input parameters and the identification of a tool to train and test the model. In this section, we introduce hardware performance counters as model input parameters and benchmarks as training and testing programs.

A. Model Inputs

Power estimation models rely on information such as the utilization of a subsystem (for example, CPU and memory) or on information delivered by performance monitoring counters (PMC), referred to in the literature also as hardware performance counters (HPC). The utilization information is not a quantity that can be obtained directly from a hardware subsystem itself, instead, it is a quantity computed and provided by the operating system. For example, the Linux kernel computes the CPU (and core) utilization, the disk utilization in terms of read and written bytes, and the network bandwidth utilization. We refer to such metrics as *OS-provided (model) inputs* or metrics. These inputs are relatively easy to obtain

and transferable across heterogeneous platforms, but they can be inaccurate[2].

Unlike the utilization information, performance monitoring counters are provided directly by the CPU. We refer to such metrics as *hardware-provided (model) inputs* or *metrics*. A contemporary CPU may provide one or more model-specific registers (MSR) that can be used to count certain micro-architectural events – *performance monitoring events* (PME). For example, such an event will be generated when the processor retires ("finishes") an instruction or when it accesses the cache. The types of events that should be captured by a PMC is specified by a *performance event selector* (PES), which is also a MSR. The amount of countable events has been increasing with every generation, family, and model of processors. At present, a processor can provide more than 200 events [24]. Repeated investigations demonstrate that accounting for certain events offers detailed insight into the power consumption characteristics of the processor and similar subsystems [25], [26].

The usage of performance monitoring counters is not straightforward, however. To begin with, the number of events that can be counted simultaneously is both limited and different from architecture to architecture. For example, the Intel Core Solo/Duo processors provide at most five counters per core whereas the Intel Sandy-Bridge processor family can provide up to 11 counters per core [24] for core events and 2 counters per C-Box[3] for uncore events. For the AMD Bulldozer family, each core provides six counters for events generated by the core and additional four counters to count the events generated by the Northbridge [27]. Hence, it is important to select the counters that correlate best with the power consumption of the processor.

Secondly, the selection of the specific counters depends on the type of benchmark (or application) – A benchmark that frequently generates integer operations will not use the CPU's floating point unit and, therefore, the events referring to the fetched, retired, or finished floating point operations will not be useful as input parameters to the estimation task. Likewise, the estimation model used for the integer workbench will not be appropriate to determine the power consumption of the processor when a workload that frequently generates floating point operations is employed. In general, models that are trained with benchmark applications are biased to the training set [28], [29].

Thirdly, since events are architecture-specific, the estimation models may not transfer well from one architecture to another architecture and even processors belonging to the same manufacturer may not support all events. For example, Intel distinguishes between architectural and non-architectural events [24]. Accordingly, an event is *architectural* if its characteristics are consistent across different processor families. There are seven architectural events available for Intel processors: *Unhalted Core Cycles*, *Unhalted Reference Cycles*, *Instruction Retired*, *LLC Reference*, *LLC Misses*, *Branch Instruction Retired*, and *Branch Misses Retired*, but the way the events can be programmed and counted may change with extensions to the version of the architectural events monitoring (cf. Architectural Events Monitoring Version 1 to 3 in [24]). Non-architectural events, on the other hand, do not have consistent characteristics across processor families and their availability is not guaranteed at all.

Fourth, some events can be generated more than once per clock cycle (multi-events) and different counters may capture different numbers of multi-events [27]. Finally, the use of the same PME sets may not guaranty the reproduction of the same event count values even when identical instruction sequences are repeatedly executed on the same platform [27].

[2]https://www.kernel.org/doc/Documentation/cpu-load.txt, 2013-04-11

[3]C-Box is Intel's term for the coherence unit between a core and the L3 cache.

B. Training Models with Benchmarks

Most approaches involve benchmarks to train and test the power estimation models. It is therefore appropriate to briefly discuss benchmarks and the reasons why they are used in power consumption estimation models.

Benchmarks are software programs specifically designed to stress some of the subsystems of a server in a comprehensible and repeatable manner. While a benchmark is running, specific sets of performance indicators are sampled and evaluated to establish a relationship between the utilization of the subsystem and its performance or power consumption. Since the stress (utilization) level of the subsystem induced by the benchmark is well defined, results for different instances of the subsystem (such as different CPU models) that run the same benchmark are expected to be comparable. For example, for a CPU benchmark undertaking several rounds of integer multiplications, the results of different CPUs can be compared in terms of the amount of multiplications they can perform in a fixed time interval. Benchmarks that generate a workload for a fixed time interval are called *throughput-based*. The complementary types of benchmarks are *time-based* with which a fixed workload is executed and the time the subsystem takes to process the workload is measured. Benchmarks that combine aspects of throughput-based and time-based workloads are called hybrid [30].

Most benchmarks are standardized to ensure that they are not optimized to favour some specific products. For example, the Standard Performance Evaluation Corporation (SPEC), which is responsible for the SPEC benchmarks, prohibits the submission of benchmarks by member companies, their associates, or a corporation entity. But member companies may freely evaluate submitted benchmarks. SPEC provides a variety of benchmark suites for various evaluation tasks: CPU performance during integer and floating point operations, graphics and workstation performance, performance of server-side and client-side Java execution, file server performance, performance of virtualized hardware and operating systems, etc. Some of the benchmarks have also been extensively used to evaluate the power consumption of processors, virtual machines, and servers. Despite the variety of evaluation opportunities, however, existing benchmarks are mainly CPU- and memory-bound.

There are also non-standardized benchmarks for evaluating

hard disk resources. Examples of these are iometer[4], iozone[5], bonnie++[6] and stream[7]. Likewise, some have developed custom (micro) benchmarks to generate workloads that have predefined stochastic properties (probability density functions) [31], [32], [33], [9].

Understandably, a model trained and tested with a variety of benchmarks may have a larger estimation error than a model trained and tested with a limited number of benchmarks: In the former case, the model has to capture with a limited number of model input parameters a wide range of power consumption characteristics whereas a model trained with a limited number of benchmarks (such as the SPEC suites) will tend to get biased towards these benchmarks [28].

IV. CPU Models

The CPU of a server consumes a considerable amount of power. In most situations, it is in fact the prime power consuming component. Therefore, quantifying its power consumption can be useful for several purposes, such as for developing energy- and thermal-aware task scheduling and low-level dynamic power management policies.

Isci and Martonosi [33] express the power consumption of a Pentium 4 CPU as the sum of the power consumption of the processor's 22 major subunits. These are: the Bus control, L1 and L2 cache, L1 and L2 branch prediction unit (BPU), TLB (instruction lookaside buffer) & fetch, instruction decode, instruction queue 1 and 2, memory order buffer, memory control, data TLB, integer and floating point execution unit, integer and floating point register file, trace cache, microcode ROM, allocation, rename, schedule, and retirement logic. A separate power model is defined for 16 of the subunits and a combined model can be defined for the trace cache, the allocation unit, the rename logic, the schedule logic, and the instruction queues 1 and 2. Altogether, the authors employ 24 different events for their model. Since the processor provides only 18 PMC, some counters are *reconfigured* or *rotated* during the experiment.

The values obtained for the single subunits are weighted and combined to estimate the power consumption of the entire CPU:

$$P(C_i) = a_R(C_i) \times a_S(C_i) \times m_P(C_i) + nGCP(C_i) \quad (1)$$

where C_i is the subunit, a_R is a subunit-dependent metric measuring the access to the respective subunit, a_S is the relative size of the subunit, m_P is the subunit's maximum power consumption, and $nGCP$ accounts for non-linear behavior of some of the subunits (trace cache, rename logic, and retire logic). The CPU's total power consumption is then expressed as the sum of all subunit power values plus the idle power:

$$Power_{CPU} = \sum_{i=1}^{22} Power(C_i) + idlepower \quad (2)$$

[4]http://www.iometer.org, 2013-04-11
[5]http://www.iozone.org, 2013-04-11
[6]http://www.coker.com.au/bonnie++, 2013-04-11
[7]http://www.cs.virginia.edu/stream/, 2013-04-11

A modification to the model is given by Bui et al. [34] in which transistor counts are used as architectural scaling factor.

The idea of Isci and Martonosi is further refined in [28] where a methodology to develop PMC-based models for multicore processors is also proposed. Similar to Isci and Martonosi the authors first determine the architectural components of the processor, identifying more than 25 components and classifying them into three categories, namely, in-order engine, out-of-order engine, and memory. The authors assert that the activity of some of the components in the in-order engine cannot be detected with distinct PMEs. As a result this engine's activity is captured as a whole with the exception of the branch prediction unit. The memory engine is further divided into three parts – L1 cache, L2 cache, and the main memory; the latter includes the front side bus. Likewise, the out-of-order engine is divided into three parts - INT unit, FP unit, and SIMD unit. This way, the authors identify 8 so called power components.

The authors develop a set of 97 micro-benchmarks to stress each of these components in isolation under different scenarios. The aim is to detect those PMEs that reflect the activity level of these components best. Based on this approach, the power consumption of the single-core of the CPU is expressed as:

$$P_{total} = \left(\sum_{i=1}^{i=comps} AR_i \times P_i \right) + P_{static} \quad (3)$$

where AR_i denotes the activity ratio in the i-th component and P_i is the corresponding power consumption of the component so that $AR_i \times P_i$ yields the component's dynamic power consumption. P_{static} is the CPU's idle state power consumption. The activity ratio is given by the value of the designated PMC, normalized by the count of the *cpu_clk_unhalted* event.

For the multi-core scenario the CPU power is expressed as:

$$Ptotal = \sum_{j=1}^{j=cores} \left(\left(\sum_{i=1}^{i=comps} AR_{ij} \times P_i \right) + P_{static} \right) \quad (4)$$

where AR_{ij} refers to the activity ratio of the i-th component of the j-th core. For the single-core model the average estimation error for individual benchmarks is between 0.32% and 6.87%, and 1.89% when all the benchmarks are considered.

For the multi-core model, the average estimation error for the individual benchmarks is between 1.47% and 10.15%; and 4.63% when all the benchmarks are considered. The authors also evaluate the model's accuracy when the processor operates at different frequencies, ranging from 800MHz to 2.533GHz. The results suggest that the estimation error decreases with increasing frequency implying that the model poorly captures nonlinear characteristics due to dynamic voltage and frequency scaling (DVFS) techniques.

Likewise, Bircher et al. [35] propose a simplified linear model for estimating the power consumption of a Pentium 4 CPU. The authors argue that the amount of *fetched μ-ops* (micro-operations) per cycle minimizes the model's estimation error whereas considering the amount of *μ-ops retired*

increases the estimation error; since the number of fetched μ-ops comprises both retired and canceled μ-ops. The estimation model has the following expression:

$$power = 35.7 + 4.31 \cdot (fetched\_\mu - ops/cycle) \quad (5)$$

To train and test the power consumption model, the authors use the SPEC2000 benchmark suite, which they split into ten clusters. Six of these comprise more than one benchmark program. From each cluster one program is used for training the model while the remaining are used to test the model. For clusters comprising only a single benchmark, the same benchmark serves both as a training and as a test program. In addition to the SPEC2000 suite, the authors employ custom-made benchmarks to examine the minimum (with minimum IPC) and the maximum (with maximum IPC) power consumption of the CPU.

Even though the model is trained with five SPEC-INT and five SPEC-FP benchmarks (with an average error of 2.6%), the authors remark that the model performs best with benchmarks that mostly produce integer operations. As a result, the authors refine their model to take advantage of the fact that floating point operations consist of complex microcode instructions [33]. For the Pentium 4 processor, this metric can be obtained with the *uop_queue_writes* event (with the event mask `0x06` [24]). To account for the difference in power consumption between the μ-ops delivered by the trace cache and the μ-ops delivered by the microcode ROM, Bircher et al. assign two counters to the events: one for the event mask `0x02` and the other for the event mask `0x04`. The resulting model expression is as follows:

$$power = 36.7 + 4.24 \cdot deliver - 11.8 \cdot microrom \quad (6)$$

where $deliver$ denotes μ-ops delivered by the trace cache and $microrom$ denotes the μ-ops delivered by the microcode ROM. The average error of the model is reduced from 2.6% to 2.5%.

Singh et al. [29] propose a model to estimate the power consumption of individual cores in an AMD Phenom 9500 multi-core processor. The processor provides 4 PMC to be sampled simultaneously. Thus, the authors identify the four events that best represent the processor's power consumption: *L2_cache_miss:all* (e_1), *retired_uops* (e_2), *retired_mmx_and_fp_instructions:all* (e_3) and *dispatch_stalls* (e_4). e_1 is selected to monitor the usage of the L3 cache (the authors claim that misses in the L2 cache often result in L3 misses as well). e_2 is selected because it provides information about the overall processor performance. e_3 is selected to account for the difference in power consumption between INT and FP units. e_4 is selected because it indicates the usage of out-of-order logic units, which is the case if the core stalls (for example, due to branches or full load/store queues). The stall event can indicate a reduction as well as an increase in power consumption. The latter is true if the core stall is caused by the reorder buffer or by a reservation station. For the proposed model the authors compute the event rate, $r_i = e_i/cycles$ and apply linear weights to estimate the power consumption of an individual core. To obtain a model that is not biased towards certain benchmark applications the authors train the model with microbenchmarks (cf. [28]) and observe different code behaviors for low values of the *L2_cache_miss:all* event count. Therefore, they suggest a two-part model where the particular function, F_1 or F_2, is chosen depending on the *L2_cache_miss:all* event rate, r_1 . The resulting model is given by:

$$\hat{P}_{core} = \begin{cases} F_1(g_1(r_1), \ldots, g_n(r_n)) & \text{if } condition \\ F_2(g_1(r_1), \ldots, g_n(r_n)) & \text{else,} \end{cases} \quad (7)$$

where $F_j = p_0 + p_1 \cdot g_1(r_1) + \cdots + p_n \cdot g_n(r_n) | j = \{1, 2\}$ and $condition$ defines the threshold for r_1.

Chen et al. [31] deal with modeling the performance and power consumption of a multi-core system when running several processes in parallel. In other words, "given k processes running on N cores with some of the cores having multiple processes and some of them being idle" their aim is to estimate "the core and processor power consumption during concurrent execution". Similar to [29], Chen et al. determine the five events that correlated most with power consumption. For an Intel Pentium Dual Core E2220 ("2-core") and an Intel Core2 Q6600 quad core ("4-core"), the events are *L1R* (L1 references), *L2R* (L2 references), *L2M* (L2 misses), *BR* (branch instructions retired), and *FP* (floating point instructions retired). For a single process running on a core the authors find that a linear model for the event counts per second (*PS*) is appropriate:

$$\begin{aligned} P_{core} =& P_{idle} + c_1 \cdot L1RPS + c_2 \cdot L2RPS \\ &+ c_3 \cdot L2MPS + c_4 \cdot BRPS + c5 \cdot FPPS \end{aligned} \quad (8)$$

where c_i are the weighting factors. However, for multiple processes running on a single core, the authors propose a different power model considering event counts per instruction (*PI*) rather than per second. The model is given by the equation: $P_{process} = P_1 + P_2$,

with

$$\begin{aligned} P_1 =& P_{idle} + (c_1 \cdot L1RPI + c_2 \cdot L2RPI + \\ & c_4 \cdot BRPI + c_5 \cdot FPPI)/SPI \end{aligned} \quad (9)$$

and

$$P_2 = c_3 \cdot L2MPR \cdot L2RPI/SPI \quad (10)$$

where SPI stands for *Seconds per Instruction*. P_1 comprises "process properties" that are independent of interaction with other processes, whereas P_2 represents the influence of other processes through cache contention. The core power consumption is then calculated as the weighted sum of the overall process power values for that core:

$$P_{core} = \frac{1}{k} \sum_{i=1}^{k} P_i \quad (11)$$

where k is the number of processes running on the respective core. For a set of cores 1 to N that share the same last level cache, the model is given as:

$$P_{core-set} = \frac{\sum_{p_i \in S_1} \cdots \sum_{p_N \in S_N} P(p_1, \cdots, p_N)}{\prod_{i=1}^{N} |S_i|} \quad (12)$$

where S_i is the set of processes running on core i.

Unlike the previous approaches, Molka et al. [9] model the power consumption of data transfers and arithmetical operations for an AMD Opteron 2435 and an Intel Xeon X5670 processors. They consider `LOAD`, `ADD`, `MUL` for each of packed integer, packed single and packed double data types, and `AND` for packed integer and packed double types. Determining the power consumption of single instructions this way is too fine-grained and may not be applicable for estimating the power consumption of a server.

As a summary, most of the CPU power consumption models require knowledge of the architecture of the CPU and the nature of the benchmark (application) to select the appropriate performance monitoring counters. The predominant mathematical tool employed to establish a relationship between the power consumption of the CPU and the events emitted by the counters is the linear regression. A summary of the CPU power consumption models is given in Table I.

V. VM Models

A virtual machine (VM) is a software abstraction of a real hardware server. It is capable of running a full-fledged operating system thereby freeing an application or a service from the concern of knowing where (i.e., on which physical machine) it is actually running. Encapsulating software entities in a virtual machine has two advantages. First of all, multiple virtual machines can run in parallel on the same physical machine regardless of the operating system or hardware platform they require for their operation. This has the manifest advantage of utilizing computing resources efficiently. Secondly, virtual machines can be moved from one physical machine to another without the need to stop their execution. This facilitates dynamic workload consolidation. Because running virtual machines incurs resource cost (including power), estimating this cost enables to better organize and schedule them.

Bohra and Chaudhary [14] study the use of PMC to model and reason about the power consumption of individual virtual machines in a virtualized environment. Particularly, the authors investigate counters for busy cycles *cpu_clk_unhalted*, memory accesses *dram_accesses*, fetched instructions *instruction_cache_fetches* and fetched data *data_cache_fetches* in an AMD Opteron processor. To gain insight into I/O accesses, the authors utilize the disk monitoring tool *iostat*. The decision for an external disk monitoring tool is motivated by (a) the absence of a PMC that directly account for disk accesses and (b) the restriction on the number of simultaneously readable performance counter values. The authors distinguish between CPU-intensive and I/O-intensive workloads (the latter includes also memory-bound workloads). For these workload types, they propose separate linear models and, then, estimate the system's total power consumption as the weighted sum of both models:

$$\begin{aligned} P_{cpu,cache} &= a_1 + a_2 \cdot p_{cpu} + a_3 \cdot p_{cache} \\ P_{mem,disk} &= a_4 + a_5 \cdot p_{mem} + a_6 \cdot p_{disk} \\ P_{total} &= \alpha P_{cpu,cache} + \beta P_{mem,disk} \end{aligned} \quad (13)$$

where a_1 and a_4 represent the idle power consumption of the different workload types (though it is not clear why these values should be different in both equations). a_2, a_3, a_5 and a_6 are coefficients obtained through linear regression and p_{cpu}, p_{mem} , and p_{cache} are the PMC values for the respective components and p_{disk} is the utilization metric for the disk drive. α and β are determined by observing the mean increase in total power consumption when an idle machine runs a CPU- or I/O-bound workload, respectively.

For a server running several virtual machines the total power consumption is given by

$$P_{server} = P_{idle} + \sum_{k=1}^{N} P_{dom(i)} \quad (14)$$

where $P_{dom(i)}$ is the power consumption of a VM, including $dom0$, the VMM, and is given by:

$$\begin{aligned} P_{dom(i)} = \alpha(a_2 p_{cpu(i)} + a_3 p_{cache(i)}) + \\ \beta(a_5 p_{mem(i)} + a_6 p_{disk(i)}) \end{aligned} \quad (15)$$

The model is evaluated with CPU-bound (BYTEmark, dense matrix multiplication), I/O-bound (Iozone, Bonnie++), and combined benchmarks (NAS NPB, cachebench, gcc). The reported average error is 7% with a median of 6%.

Similarly, Krishnan et al. [15] present a linear model to compute the power consumption of virtual machines. They use PMC as model inputs but point out that the mapping of *inst_retired/s* to CPU power consumption is not straight forward, since memory references that hit in different cache levels consume slightly different amounts of power. Therefore, instructions that execute without cache references or that hit in L1 cache consume less power than instructions with hits in the last level cache. To account for power consumptions caused by hits in different cache levels, the authors define bounds to limit the model's error. A lower bound is defined for instructions that hit at most in L1 cache and an upper bound for instructions that hit in LLC. A second class of bounds reflects the level of memory parallelism. For the system under test used by the authors the power consumption of the memory is between 15 W for maximum memory level parallelism (MLP) and 45 W in cases where MLP cannot be exploited. Combining these boundary cases, the authors define four bounds:

$$B1 = P_{L1} + P_{MLP} \quad (16)$$

$$B2 = P_{LLC} + P_{MLP} \quad (17)$$

$$B3 = P_{L1} + P_{NO\_MLP} \quad (18)$$

$$B4 = P_{LLC} + P_{NO\_MLP} \quad (19)$$

Based on this classification, the authors observe that most of the SPEC2006 benchmarks they used to evaluate their model are close to one of these bounds. For CPU-bound benchmarks with high hit rate in L1 (bzip2, namd, h264ref), the power consumption is expected to be close to either B1 or B3.

Approach	Model Inputs	Model Training Technique	System Under Test	Benchmarks/Applications
Bertran et al. [28]	12 PME in sum (4 PES reconfigurations)	multiple linear regression	Intel Core 2 Duo T9300 (Penryn)	custom μ-benchmarks (training); SPECcpu2006, NAS [36], LMBENCH [37] (evaluation)
Bircher et al. [35]	*Fetched μ-ops*; *delivered μ-ops*	linear regression	Intel Pentium 4	SPEC 2000 suite
Bui et al. [34]	*instructions retired*; *cache accesses*	piecewise linear regression see [33]	Intel Itanium 2 (Madison)	GenIDLEST [38]
Chen et al. [31]	*L1RPS*; *L2RPS*; *L2MPS*; *BRPS*; *FPPS*	linear regression	Intel Core 2 Q6600 (Kentsfield); Intel Pentium Dual Core E2220 (Allendale)	custom μ-benchmarks (training); SPECcpu2000 (evaluation)
Isci & Martonosi [33]	15 PME in sum (4 PES reconfigurations)	piecewise linear regression	Intel Pentium 4	custom μ-benchmarks (training); SPEC2000 suite (evaluation)
Molka et al. [9]	Machine instructions	linear regression	Intel Xeon X5670 (Westmere); AMD Opteron 2435 (Instanbul)	custom μ-benchmarks
Singh et al. [29]	*L2MPS*; *retired μ-ops*; *dispatchStalls*; *retiredMMXandFP instructions*	linear regression with nonlinear parameter transformation	AMD Phenom X4 9500 (Agena)	custom μ-benchmarks (training); SPEC2006, SPEC-OMP, NAS (evaluation)

Table I
A SUMMARY OF THE MODELS PROPOSED TO ESTIMATE THE POWER CONSUMPTION OF A CPU.

Using the B1 bound the power estimation error is 6% on average, for the B3 bound no estimation error is reported, but since the memory power consumption is larger when no memory level parallelism can be used, one can assume the estimation error to be larger. For CPU-bound benchmarks with a high hit rate in LLC (gobmk, sjeng, povray), power consumption is approximated to be close to either B2 or B4. An average estimation error of 10% is observed for the B2 bound, however, similar to the previous case, no estimation error for the B4 bound is reported. For memory-bound workloads with high MLP (libquantum, milc, lbm), the power consumption is estimated to be close to either B1 or B2, and results show that the model's estimation error is 7% on average with the B1 bound. For a memory-bound workload with no MLP (omnetpp), the power consumption is estimated to be close to B3 or B4, and the average estimation error is 1% with the B4 bound. The same model is tested by combining the previous benchmarks (namd + lbm; povray + omnetpp; povray + namd + h264ref + lbm) and an average estimation error of 5% is observed for all the cases.

Dhiman et al. [12] employ CPU utilization and PMC – instructions per cycle (IPC), memory accesses per cycle (MPC), cache transactions per cycle (CTPC) – along with a Gaussian Mixture Model (GMM) to estimate the power consumption of a virtual machine. The input values form a vector $x = (x_{ipc}, x_{mpc}, x_{ctpc}, x_{util}, x_{pwr})$ which are assigned to different classes (or $bins$). Depending on the CPU utilization, bin sizes range from 20% utilization (5 bins) to 100% CPU utilization (1 bin). Within every bin the data vectors are quantized using a Gauss Mixture Vector Quantization (GMVQ) technique [39]. The output of this quantization are multiple Gaussian components g_i that are determined by their mean $m_i = \{m_{ipc}, m_{mtc}, m_{ctpc}, m_{util}, m_{pwr}\}$, covariance $K_i(x_{ipc}, x_{mpc}, x_{ctpc}, x_{util}, x_{pwr})$, and probability p_i. To train the model, the x_{pwr} component is removed from the x vectors and the resulting vectors are compared with those in the training set. A GMVQ algorithm then finds the nearest vector m in the training set to the input vector, and the value for the m_{pwr} component is retrieved. The retrieved value is compared to the original x_{pwr} value from which the accuracy of the model is determined. This part of the training phase is repeated multiple times with different sizes of utilization bins. The bin size resulting in the smallest error is selected as the model parameter.

The same method is applied during the running phase: PMC values are obtained and the vector is compared to the model where the GMVQ algorithm finds the nearest (training) vector and returns the average power value. The approach is tested against several benchmarks from the SPEC2000 suite. Depending on the utilization level, the average error of the model is between 11% (for 50% CPU utilization) and 8% (for 100% CPU utilization). Nevertheless, even at 100% utilization level, the absolute error is between 1% and 17%, depending on the benchmark. The smallest error is obtained with the art and mcf benchmark while the largest error is observed with the gcc benchmark. Furthermore, the model accuracy is affected by the system configuration. For example, for a machine with 8GB memory the model error is between 8% and 11%. For an increased memory capacity (24GB), the error ranges between 8.5% and 9%.

In the same way, Kansal et al. [18] combine PMC and CPU utilization to estimate the power consumption of a virtual machine. While CPU utilization is used to calculate the power consumption of the CPU by a VM, the power consumption of the memory by the same VM is estimated by counting the Last Level Cache Misses (LLCM).

Zhu et al. [40] propose a power-aware workflow scheduling mechanism for a virtualized heterogeneous environment. The authors employ the Kernel Canonical Correlation Analysis (KCCA, [41]) to determine the mapping of the resource utilization of a VM (each VM runs a single task in a workflow) to power consumption and performance (i.e., execution time)

indicator metrics. For every task, the authors record ten time series samples of the CPU, memory, disk, and network utilization and for each of the parameters they obtain a statistical signature comprising the maximum value and the variance. In addition to these, the server's capacity for the respective resources is added into the model. Altogether, the authors consider 52 parameters (*features*) in a feature space X. For the same observation interval, the server's power consumption and execution time for all tasks are recorded, resulting in another two-dimensional feature space Y. Nevertheless, a few features are accountable for most of the data's total variance so that these few features are sufficient to explain the entire data. This advantage is used to reduce the dimension of X without losing too much information. The CCA then finds a linear transformation of X and Y such that the transformation of Y is sufficiently explained by the transformation of X.

To account for nonlinear properties, the CCA is kernelized. A kernel in this context is a function that maps some point in a lower dimensional feature space A to a higher (potentially infinite) dimensional feature space B in a nonlinear fashion. The idea behind is that nonlinear relations in A are linearized in B. This means that X and Y are mapped to a higher dimensional feature space in which (linear) CCA can be applied. Two of the challenges in applying this approach are finding the appropriate kernel function and tuning its parameters properly. As Schölkopf and Smola [41] point out, "the kernel *is* prior knowledge about the problem at hand". In the absence of such knowledge one may choose a Gaussian kernel as Zhu et al. do. The authors report an average estimation error of 3.3%.

As a summary, unlike the CPU power consumption models, the VM power consumption models include the CPU utilization into the estimation model. We find that this is appropriate, since in a virtualized environment a diversity of services (applications) can be hosted by one and the same physical server. Hence, relaying on a selected number of performance monitoring counters alone may not provide an adequate insight into the power consumption characteristic of the CPU. Moreover, the CPU utilization can easily be obtained. Therefore, including this additional metric into the model certainly enriches the model.

Table II summarizes the models for estimating the VM power consumption.

VI. Server Models

Even though the CPU is the predominant power consumer of a server, the main memory, data storage, and networking devices also consume a considerable amount of power that cannot be disregarded. Developing distinct models for all these components is not trivial. The alternative is to estimate the overall power consumption of the entire server or even the entire cluster or the data center. One of the merits of this approach is the possibility of realistically budgeting power and undertaking a high level dynamic power management decisions (for example, selectively switch off underutilized servers and distributing the workload of overloaded servers).

Heath et al. [32] propose an energy model for a heterogeneous server cluster. The power consumption of individual servers is estimated with a linear model that solely employs utilization metrics. The model for each server i is given by:

$$P_i = P_{i_{idle}} + \sum_r M_i^r \cdot \frac{U_i^r}{C_i^r} \tag{20}$$

where $P_{i_{idle}}$ is the idle power consumption, M_i^r is a matrix of maximum power consumption values for named resources r, U_i^r is the utilization of the resource r, and C_i^r is the server's capacity matrix for the resource r. Whereas it is straightforward to determine the utilization level of a resource for a single server (cf. subsection III-A), the intra-cluster communication between the web servers necessitates an individual model. Hence, to determine U_i^r , the resource consumption due to the fraction of requests served locally, the cost of sending requests to other nodes and the cost of serving requests on behalf of other nodes should be summed together:

$$u_i^r = \sum_t \left(F_i^t R_{ii}^t L_i^{rt} + D_i F_i^t \sum_j R_{ij}^t S_{ij}^{rt} + \sum_j D_j F_j^t R_{ji}^t A_{ji}^{rt} \right) \tag{21}$$

$$U_i^r = thruput \cdot u_i^r \tag{22}$$

where i denotes a local server in the cluster, j denotes a remote server, t denotes the request types, and $thruput$ denotes the number of requests being served per second by the cluster. Moreover, the variables have the following meaning:

F	partition matrix of requests into types
L	resource cost matrix for locally served requests
S	resource cost matrix for sending requests
A	resource cost matrix for accepting remote requests
R	request distribution matrix
D	request distribution vector

The values for the C^r, L^r, A^r, M^r, and S^r matrices are determined using microbenchmarks for different types and sizes of requests. The term "request distribution" in R and D should not be confused with probability distribution functions. Instead, both terms represent the solutions to an optimization problem: Requests are distributed such that the utilization of every resource on every server is below its respective capacity and the cluster's overall power consumption is minimized. The authors apply simulated annealing to find near-optimal solutions. The model is evaluated against traces of requests made to real servers hosting the 1998 World Cup events. The model is evaluated as a function of D and R. The estimation error is 1.3% on average, with the maximum error of 2.7%.

Likewise, Economou et al. [22] propose a linear model to estimate the power consumption of a server in a cluster using utilization metrics as inputs. These metrics refer to the utilization of the CPU (u_{cpu}), the main memory accesses (u_{mem}), hard disk I/O rate (u_{disk}), and the network I/O rate (u_{net}). The authors employ custom made benchmarks to stress the components in isolation each time measuring the reference power of the respective subcomponents. A blade server with an AMD Turion processor and a server with four Intel Itanium 2 processors are used to train the model and to obtain the coefficients of the linear model.

For the blade server, the power consumption is expressed

Approach	Model Inputs	Model Training Technique	System Under Test	Benchmarks/Applications
Beloglazov et al. [16]	CPU utilization	linear interpolation	n.n.	n.n.
Bohra et al. [14]	*CPU_clk_unhalted*; *DRAM_accesses*; *instruction_cache_fetches*; *data_cache_fetches*	linear regression	AMD Opteron (not specified)	NAS NPB, Iozone, Bonnie++, BYTEMark, Cachebench, Dense Matrix Multiplication, gcc
Dhiman et al. [12]	CPU utilization; *instructions*; *memory accesses*; *cache transactions*	Gaussian Mixture Vector Quantization	Intel Xeon E5507 (Gainestown)	SPEC2000 suite
Kansal et al. [18]	CPU utilization; *LLCM*; Hyper-V-reported metrics	linear regression	Intel Xeon L5570 (Gainestown)	SPEC2006 suite
Krishnan et al. [15]	*instructions retired*; *LLC Misses*	correlation analysis	Intel Core i7 (Nehalem); Intel Core 2 Quad-Core (Yorkfield)	SPEC2006 suite
Zhu et al. [40]	automatically selected	kernelized canonical correlation analysis	AMD Opteron 250 (Sledgehammer); Intel Xeon E5345 (Clovertown)	weather forecasting workflow comprising: gMM5, gWaterLevel, gTurbulentStress, gNetHeatFlux, gGridResolution, gPOM2D, gPOM3D, gVis

Table II
A SUMMARY OF THE MODELS PROPOSED TO ESTIMATE THE POWER CONSUMPTION OF VIRTUAL MACHINES.

as:

$$P_{blade} = 14.45 + 0.236 \cdot u_{cpu} - (4.47E-8) \cdot u_{mem} + 0.00281 \cdot u_{disk} + (3.1E-8) \cdot u_{net} \quad (23)$$

The power consumption for the Itanium server is given by:

$$P_{itanium} = 635.62 + 0.1108 \cdot u_{cpu} + (4.05E-7) \cdot u_{mem} + 0.00405 \cdot u_{disk} + 0 \cdot u_{net} \quad (24)$$

Both models are evaluated with SPECcpu2000, SPECjbb2000, SPECweb2005, matrix multiplication, and *stream* the benchmarks. For the Itanium server, the model error is between 0% and 15% on average and for the Turion blade the error is below 5%.

Fan et al. [17] propose a nonlinear model using CPU utilization as the model input. The model training phase is similar to that of [22], but to account for nonlinear relationship between the performance and the power consumption, they extend the structure of a linear model with an error correction factor. The resulting model has the form:

$$P_{idle} + (P_{busy} - P_{idle})(2u_{cpu} - u_{cpu}^{r}) \quad (25)$$

where P_{busy} denotes the power consumption of a 100% utilized CPU, r is the correction or calibration factor minimizing the squared error of the model. The size of the correction factor depends on the system characteristics and has to be learned during a calibration phase. The authors state that the correction factor minimizes the model error by a value of 1.4 for r. Model evaluation was done using "a few hundred" servers, but it is not clear from the paper which benchmarks are used for this purpose. Regardless, the authors report an average estimation error of below 1%.

Gandhi et al. [42] investigate the optimality of different power management policies of a server farm that trade energy for performance in individual servers as well as in multi-server environments. They model a server as an $M/M/1$ system [43]. Unlike the previous models, the workload of a server is modeled as a random variable having a Poisson arrival rate and an exponentially distributed size. The server can be configured to enter different power modes when it does not process a workload; but when a workload arrives, the system transits to an active state and the transition from these power modes produces a corresponding performance penalty. The operational state of the server is labeled as S_{ON} and the various idle states are labeled as $S_0, \cdots, S_N$, where S_N denotes the off state. Corresponding levels of power consumption are denoted by $P_{ON}, P_{S_0} = P_{IDLE}, P_1, \cdots, P_{S_N} = P_{OFF}$. It is assumed that $P_{ON} > P_{IDLE} > \cdots > P_{OFF} = 0$. The transition from any lower power state S_i to any higher power state S_j requires a transition time T_{S_i}, a duration which is assumed to be independent of the target state. As an upper bound estimation, the power consumed during a transition is P_{ON}. Moreover, $T_{IDLE} < T_{S_1} < \cdots < T_{S_{N-1}} < T_{OFF}$.

For transitions from any higher power state to any lower power state, the transition duration is assumed to be 0 and the same assumption holds for the power consumption of the server during transitions of this kind. A server will transit to a lower power state S_i with a probability p_i as soon as it becomes idle and it will *not* wakeup immediately to S_{ON} as soon as a request (or "job") arrives; instead, it stays in S_i with a probability q_{ij} until j jobs are waiting in the queue. The time between two consecutive *idle* situations is called a *renewal cycle* or simply a cycle. Such cycle comprises three parts:

1) A time spent in sleep state S_i until j jobs arrive in the queue. The expected duration is determined by the Poisson distributed request rate λ and is ,therefore, $\frac{j}{\lambda}$, and the energy consumption during this phase is $\frac{j}{\lambda}P_{S_i}$.
2) A time T_{S_i} spent during a wake-up with an energy consumption of $T_{S_i}P_{ON}$.
3) A time to serve all j requests that are in the queue when starting the wake-up plus the X jobs that queued up during the wake-up phase. The mean response time of

the system for one job is given by $\frac{1}{\mu-\lambda}$ and the expected duration for serving all jobs is $\frac{j+X}{\mu-\lambda}$, and the energy consumption is $\frac{j+X}{\mu-\lambda}P_{ON}$.

The expected power consumption is the expected total energy consumption for all the three parts of the cycle normalized by the expected cycle duration, formally:

$$\mathbf{E}[P] = \frac{\sum_{i=0}^{N} p_i \sum_{j=1}^{\infty} \left[\frac{j}{\lambda}P_{S_i} + T_{S_i}P_{ON} + \frac{j+\lambda T_{S_i}}{\mu-\lambda}P_{ON}\right]}{\sum_{i=0}^{N} p_i \sum_{j=1}^{\infty} q_{ij} \left[\frac{j}{\lambda} + T_{S_i} + \frac{j+\lambda T_{S_i}}{\mu-\lambda}\right]} \tag{26}$$

where λT_{S_i} is the mean value for the Poisson-distributed value X. The authors do not provide an evaluation result of the model.

As a summary, similar to the VM power consumption models, the server power consumption models take a diversity of input parameters for the estimation task. Among these are utilization indicators pertaining to the CPU and the memory subsystems, which indicate that these subsystems are the predominant consumers of the overall power supplied to the server. Table III summarizes the power consumption estimation models for an entire server.

VII. Discussion

In all of the models we considered so far, there are deviations between the estimated power consumption and the actual power consumption obtained by some means (direct or indirect measurement). We refer to this deviation as an *estimation error*. The margin of this error is between 0% (reported in [22]) and 15% (reported in [46], [22]). This section will discuss the influence of different factors on the estimation error, namely, the choice of model input parameters, the model training techniques, the choice of benchmarks or applications for training and evaluation purposes, and the way the reference power of the model is obtained.

A. Model Parameters

The estimation error strongly depends on the parameters chosen as model inputs – all of the models use either OS-provided metrics, PMC, or a mixture of these. In a contemporary CPU, a large amount of hardware related events can be captured with performance monitoring counters, but knowledge of the application/benchmark is necessary to choose the appropriate ones and to establish a meaningful correlation between them and the power consumption of the system under consideration. Whereas there are means to automatically identify the appropriate parameters, only a few of the proposed approaches select model parameters this way: McCullough et al. [47] employ canonical correlation analysis (CCA) to select the most relevant events out of 884 countable events and metrics for an Intel Core i7-820QM processor running a Linux 2.6.37 kernel. The authors include all of the OS-provided metrics because of the flexibility associated with them during event reading. Zhu et al. [40] employ kernelized CCA to select the most relevant OS-provided metrics. Unlike McCullough [47], they do not consider performance monitoring counters, however.

Bertran et al. [28] point out that automatic selection of PME does not necessarily lead to a reduced estimation error. However, conventional benchmarks – such as the SPEC benchmark suites – stress several architectural components of a CPU in parallel but not in equal proportions. An automatic selection of PME will choose those events that represent the most stressed components. Thus, the model gets biased towards the training applications. As a solution they propose to train the model with microbenchmarks that stress CPU components in isolation. Their results shows a comparatively small and predictable estimation error – from <1% to 6%. Their approach is tested using the entire SPECcpu2000 suite. Nevertheless, for the LMBENCH benchmark that is deigned to test the memory bandwidth, their model produces a larger estimation error – up to 20%. This is explained by the presence of active power components in the CPU that are not captured by their model.

Of all the employed PMEs, the one most frequently used and which is highly correlated with the power consumption of a processor is the *instructions retired* event [34], [15], [45], [29]. Nevertheless, Bircher et al. [35] argue that the event does not account for instructions that are aborted even though aborted instructions may have their own share of the CPU power consumption. The authors show that the *instructions fetched* is a more accurate choice in this case.

Interestingly, all of the considered CPU power consumption models employ hardware-provided model inputs (PMCs). In contrast, all except one of the models for server power consumption employ OS-provided metrics as their input parameters. This is most probably due to the fact that OS-provided metrics are much easier to obtain and yet provide sufficient insight into the overall power consumption characteristics of a server. Hence, the quality of a model does not depend on the model input parameters alone, but also on how well the chosen inputs reflect workload variability. Bertran et al. [28] define the term *responsiveness* as the model's ability to capture or reflect changing features in a power consumption. Where little variation is expected, simple models can be sufficient, because workload bias as criticized in [28], [29] has little effect on the estimation error. A clear advantage of OS-provided metrics is their portability. In contrast, availability of PMEs metrics is only guaranteed for a very small set of events [24], even so, consistent semantics is not usually guaranteed across all CPU models and families [27].

B. Choice of Model Training Techniques

Another factor that influences the estimation error is the statistical tool chosen to establish a relationship between the input parameters and the power consumption. Most of the proposed models apply linear regression to fit model parameters. The minimum average estimation error that is reported for a linear model is 1.5% while the maximum is 10% [15]. For the nonlinear models the minimum estimation error is < 1% [17] and the maximum is 6.67% [13]. It must be remarked that there is a trade-off between the estimation accuracy and the computational complexity. Linear models are simpler to process than nonlinear models. Lewis et al. [48] show that linearity does not hold for all cases and that nonlinear behavior of a

Approach	Model Inputs	Model Training Technique	System Under Test	Benchmarks/Applications
Abbasi et al. [11]	CPU utilization	linear regression	Intel Xeon Dual Core (Yonah)	SPECweb2009
Economou et al. [22]	CPU utilization; memory accesses; disk I/O rate; network I/O rate	linear programming	Intel Itanium 2 (Madison); AMD Turion 64 ML-40 (Lancaster)	SPECint2000, SPECfp2000, SPECweb, SPECjbb, stream, matrix multiplication
Fan et al. [17]	CPU utilization	n.n.	Intel Xeon Netburst (not specified)	Google Web Search, GMail, Mapreduce
Gandhi et al. [42]	request arrival rate λ; state transition times; number of queued jobs	calculation of expected value of power consumption $E[P]$	Simulation; reference values from Intel Xeon E5320 (Gainestown)	n.n.
Heath et al. [32]	CPU utilization, network bandwidth, disk bandwidth, maximum number of concurrently open sockets	linear regression	Intel Pentium 3	micro benchmark (not specified)
Petrucci et al. [44]	CPU utilization, CPU frequency	quadratic relationship between utilization and frequency assumed	Intel Core i7 (Nehalem); Intel Core i5 (Nehalem); AMD Phenom II X4; AMD Athlon 64 X2, Intel Core 2 Duo (not specified)	httperf
Rivoire et al. [45]	CPU utilization; *unhalted clock cycles*; *instructions retired*; *LLC references*; *memory bandwidth*; *floating point instructions*	linear regression	Intel Xeon E5345 (Clovertown); Intel Itanium 2 (Madison); AMD Turion 64 ML-34 (Lancaster)	SPECcpu (year not specified), SPECjbb, stream, ClamAV, Nsort

Table III
A SUMMARY OF THE MODELS PROPOSED TO ESTIMATE THE OVERALL POWER CONSUMPTION OF A SERVER.

time series is inherently chaotic[8]. Therefore, they propose a technique called Chaotic Attractor Prediction, complementary to conventional auto-regressive or moving-average methods. Depending on the benchmark and the system under test, their approach produces an improvement on the average estimation error between 0.1% and 4.8% compared to the linear auto-regressive method, and between 1.4% and 8.3% compared to the auto-regressive moving average (ARMA) method.

In two separate cases linear and nonlinear models are compared [47], [45]. In the first case, Rivoire et al. [45] evaluate three different power consumption models: The first model is a nonlinear model and uses the CPU utilization as a metric for the estimation task (as proposed by [17]), the second is a linear model (as proposed by [32]) using CPU and disk utilization, and the third model is a linear model ("MANTIS") using PMC in addition to CPU and disk utilization (as proposed in [22]). All models are intended to estimate the power consumption of an entire server. The models are tested on several platforms, among others, a high-performance Xeon server (2x Intel Xeon E5345[9]). On each system, five benchmarks are run to evaluate the models: SPECfp, SPECint and SPECjbb (as CPU-intensive workloads), *stream* (as a memory-bound workload); and *ClamAV* virus scanner, *Nsort*, and SPECweb (as an I/O-intensive workload). For the Xeon-based system, MANTIS is reported to have an average estimation error ranging between 1.5% (for the calibration phase and the *stream* benchmarks) and 9.5% (for the SPECint benchmark suite). It is remarkable that for the CPU-intensive SPEC benchmark, MANTIS – which includes PMC –, results in a higher estimation error than a nonlinear model that solely uses CPU utilization [17]: For SPECint benchmark the estimation error of the nonlinear model is 4.25% while it is 2.25% for the SPECfp benchmark. In contrast, the error in MANTIS is 8%. This indicates that PMC-based models are not necessarily more accurate than models based on OS-provided inputs, but that accounting for nonlinear properties reduces the estimation error even when abstract model inputs are employed. However, MANTIS performs better for the *stream* benchmark (1.5% average estimation error) while the nonlinear model of [17] has an average estimation error of 14.75%. This is most likely due to the fact that the CPU utilization is not a good indicator of the memory power consumption.

MANTIS is also evaluated by McCullough et al. in [47]. For best comparison, we consider their results for the multi-core system (Intel Core i7-820QM[10]) and the SPECfp benchmarks (*povray*, *soplex*, *namd*, *zeusmp*, *sphinx3*). The average estimation error is 4.23% which agrees with the observation made in [45]. However, for other benchmarks, MANTIS performs poorly – up to 15% estimation error. McCullough et al. explain that this is due to the existence of "hidden states" in the multi-core CPU producing nonlinear behavior that cannot be captured by the model.

C. Reference Power

The reference power with which a model's output is compared should also be taken into account to understand how the estimation error is quantified. Most of the proposed models take the overall AC power consumption as their reference point [29], [45], [7], [14], [5], [17], [32], [15]). This may work well

[8]Small variations in the input variables lead to very diverse outcomes.

[9]Quad-core CPU from the Clovertown series, based on the Core microarchitecture

[10]Quad-core CPU from the Clarksfield series, based on the Nehalem microarchitecture. Nehalem is the succesor of the Core microarchitecture.

for estimating the overall power consumption of a server, but for the CPU and VM models, this approach disregards the inefficiency of the power supply unit, which amounts to a loss of up to 35% of the AC power drawn by a server [49]. Commercially available power supplies are designed to have an optimal efficiency when operated at full load, which is not always the case with most real world servers. Therefore, the best approach is to take as a reference the power that leaves the voltage regulators supplying the core voltages to the CPUs. This, however, complicates the measurement task and does not scale well for estimating the power consumption of multicore processors.

In view of the three factors we considered, while all the models we presented exhibit competitive qualities, we find that three models stand out in terms of reproducible results and portability. These are, the model of Zhu et al. [40] for estimating the power consumption of virtual machines and the models of Fan et al. [17] and Economou et al. [22] for estimating the power consumption of an entire server. These models have been tested on different server platforms and with different types of benchmarks and proved to be reliable. Moreover, the estimation error for these models reported by different authors have been consistent.

VIII. Conclusion

Understanding the power consumption of processors, virtual machines, and servers is paramount to carry out energy efficient adaptations such as energy-aware scheduling and load balancing, service consolidation, and dynamic frequency and voltage scaling. This paper provided a comprehensive survey of the proposed power estimation models. Most existing models take hardware performance counters as their input and apply regression techniques. The selection of model inputs is strongly dependent on the workload type and the processor architecture. Whereas most of the models are trained and tested with standard benchmarks (for example, SPECcpu2000, SPECcpu2006), in reality, the workload of Internet-based servers is variable both in terms of its type and size. While this is the case, to the best of our knowledge, very few models employ variable workloads. None of them were tested, for example, with the SPECpower2008 benchmark.

Furthermore, the relationship between the input parameters and the power consumption is static. Some models provide alternative expressions for different workloads, but the usefulness of this approach is limited when the workload is dynamic. Instead, a feedback system or a belief revision mechanism is more realistic to implement as is done in probabilistic parameter estimation. For example, recursive estimation and error correction approaches using Bayesian Estimation [50] or minimum Mean-Square Estimation [51] can be very useful. This also means that instead of pursuing elaborate and detailed deterministic approaches only, probabilistic approaches should also be investigated in future.

Another missing aspect in the literature is examining the complexity of proposed models and the latency of estimation. In the end, the scope and usefulness of a model are determined not only by its accuracy but also by its resource consumption and the latency of estimation. Therefore, equal emphasis should be given to these nonfunctional aspects.

References

[1] J. G. Koomey, "Worldwide electricity used in data centers," *Environmental Research Letters*, vol. 3, no. 3, p. 034008, 2008.

[2] J. G. Koomey, "Growth in data center electricity use 2005 to 2010," July 2011.

[3] W. Dargie, A. Strunk, and A. Schill, "Energy-aware service execution," *2011 IEEE 36th Conference on Local Computer Networks*, pp. 1064–1071, Oct. 2011.

[4] W. Dargie and A. Schill, "Analysis of the Power and Hardware Resource Consumption of Servers under Different Load Balancing Policies," in *The 5th International Conference on Cloud Computing (IEEE Cloud 2012)*, (Honolulu), pp. 24–29, IEEE Computer Society, 2012.

[5] G. Chen, W. He, J. Liu, S. Nath, and L. Rigas, "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, no. 1 in NSDI'08, (San Franciso, California), pp. 337—-350, USENIX Association, May 2008.

[6] A. Merkel, J. Stoess, and F. Bellosa, "Resource-conscious scheduling for energy efficiency on multicore processors," in *Proceedings of the 5th European conference on Computer systems*, EuroSys '10, (New York, NY, USA), pp. 153–166, ACM, 2010.

[7] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *Proceedings of the 2008 conference on Power aware computing and systems*, HotPower'08, (Berkeley, CA, USA), pp. 10–10, USENIX Association, 2008.

[8] K. Du Bois, T. Schaeps, S. Polfliet, F. Ryckbosch, and L. Eeckhout, "SWEEP: evaluating computer system energy efficiency using synthetic workloads," in *Proceedings of the 6th International Conference on High Performance and Embedded Architectures and Compilers - HiPEAC '11*, (New York, New York, USA), p. 159, ACM Press, 2011.

[9] D. Molka, D. Hackenberg, R. Schöne, and M. S. Müller, "Characterizing the energy consumption of data transfers and arithmetic operations on x86-64 processors," in *International Conference on Green Computing*, pp. 123–133, IEEE, Aug. 2010.

[10] W. L. Bircher and L. K. John, "Measurement Based Power Phase Analysis of a Commercial Workload," in *Unique Chips and Systems* (E. John and J. Rubio, eds.), ch. 8, pp. 217–237, CRC Press, 2007.

[11] Z. Abbasi, G. Varsamopoulos, and S. K. S. Gupta, "Thermal aware server provisioning and workload distribution for internet data centers," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing - HPDC '10*, (New York, New York, USA), p. 130, ACM Press, 2010.

[12] G. Dhiman, K. Mihic, and T. Rosing, "A system for online power prediction in virtualized environments using Gaussian mixture models," in *Proceedings of the 47th Design Automation Conference, DAC '10*, no. 3, (New York, New York, USA), p. 807, ACM Press, 2010.

[13] W. L. Bircher and L. K. John, "Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events," in *2007 IEEE International Symposium on Performance Analysis of Systems & Software*, pp. 158–168, IEEE, Apr. 2007.

[14] A. E. Husain Bohra and V. Chaudhary, "VMeter: Power modelling for virtualized clouds," *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pp. 1–8, Apr. 2010.

[15] B. Krishnan, H. Amur, A. Gavrilovska, and K. Schwan, "VM power metering," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, p. 56, Jan. 2011.

[16] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," in *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science - MGC '10*, (New York, New York, USA), pp. 1–6, ACM Press, 2010.

[17] X. Fan, W.-d. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *ACM SIGARCH Computer Architecture News*, vol. 35, p. 13, June 2007.

[18] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. a. Bhattacharya, "Virtual machine power metering and provisioning," *Proceedings of the 1st ACM symposium on Cloud computing - SoCC '10*, p. 39, 2010.

[19] S. Rivoire, *Models and Metrics for Energy-Efficient Computer Systems*. Doctoral dissertation, Stanford University, 2008.

[20] P. Bohrer, E. Elnozahy, and T. Keller, "The case for power management in web servers," in *Power-Aware Computing* (R. Graybill and R. Melhem, eds.), pp. 261–289, Kluwer Academic Publishers, 2002.

[21] E. N. Elnozahy, M. Kistler, and R. Rajamony, "Energy-efficient server clusters," in *Proceedings of the 2nd international conference on Power-aware computer systems*, (Cambridge, MA, USA), pp. 179–197, Springer-Verlag, 2002.

[22] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, "Full-System Power Analysis and Modeling for Server Environments," in *Proceedings of the 2nd Workshop on Modeling, Benchmarking, and Simulation*, MoBS, (Boston, MA, USA), pp. 70–77, 2006.

[23] SSI, *EPS12V Power Supply Design Guide*, 2.92 ed., 2004.

[24] Intel Corporation, "Intel® 64 and IA-32 Architectures Software Developer's Manual," specification, 2012.

[25] F. Bellosa, "The benefits of event: driven energy accounting in power-sensitive systems," in *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*, EW 9, (New York, NY, USA), pp. 37–42, ACM, 2000.

[26] A. Merkel and F. Bellosa, "Balancing power consumption in multiprocessor systems," *SIGOPS Oper. Syst. Rev.*, vol. 40, pp. 403–414, Apr. 2006.

[27] AMD Corporation, "BIOS and Kernel Developer's Guide for AMD Family 15h Models 00h-0Fh Processors (PUB)," specification, 2012.

[28] R. Bertran, Y. Becerra, D. Carrera, V. Beltran, M. Gonzàlez, X. Martorell, N. Navarro, J. Torres, and E. Ayguadé, "Energy accounting for shared virtualized environments under DVFS using PMC-based power models," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 457 – 468, 2012.

[29] K. Singh, M. Bhadauria, and S. a. McKee, "Real time power estimation and thread scheduling via performance counters," *ACM SIGARCH Computer Architecture News*, vol. 37, p. 46, July 2009.

[30] Standard Performance and Evaluation Corporation (SPEC), *SPEC Power and Performance Benchmark Methodolody V2.1*, 2011.

[31] X. Chen, C. Xu, R. P. Dick, and Z. M. Mao, "Performance and power modeling in a multi-programmed multi-core environment," *Proceedings of the 47th Design Automation Conference on - DAC '10*, p. 813, 2010.

[32] T. Heath, B. Diniz, E. V. Carrera, W. M. Jr., and R. Bianchini, "Energy conservation in heterogeneous server clusters," in *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming - PPoPP '05*, (New York, New York, USA), p. 186, ACM Press, 2005.

[33] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: methodology and empirical data," *22nd Digital Avionics Systems Conference. Proceedings (Cat. No.03CH37449)*, pp. 93–104, 2003.

[34]

[35] 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, 2005.*, pp. 275–280, IEEE, 2005.

[36] D. Bailey, E. Barszcz, J. Barton, and L. Dagum, "The nas parallel benchmarks," *The International Journal of Supercomputer Applications*, 1991.

[37] L. McVoy and C. Staelin, "lmbench: portable tools for performance analysis," in *Proceedings of the 1996 annual conference on USENIX Annual Technical Conference*, ATEC '96, (Berkeley, CA, USA), pp. 23–23, USENIX Association, 1996.

[38] D. Tafti, "GenIDLEST: A Parallel High Performance Computational Infrastructure for Simulating Complex Turbulent Flow and Heat Transfer," in *American Physical Society, Division of Fluid Dymanics 55th Annual Meeting*, p. 9, 2002.

[39] R. Gray, "Gauss mixture vector quantization," *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, vol. 3, pp. 1769–1772, 2001.

[40] Q. Zhu, J. Zhu, and G. Agrawal, "Power-aware consolidation of scientific workflows in virtualized environments," technical report, Ohio State University, 2010.

[41] B. Schölkopf and A. J. Smola, *Learning with kernels - Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[42] A. Gandhi, V. Gupta, M. Harchol-Balter, and M. a. Kozuch, "Optimality analysis of energy-performance trade-off for server farm management," *Performance Evaluation*, vol. 67, pp. 1155–1171, Nov. 2010.

[43] D. G. Kendall, "Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain," *The Annals of Mathematical Statistics*, vol. 24, pp. 338–354, Sept. 1953.

[44] V. Petrucci, E. V. Carrera, O. Loques, J. C. Leite, and D. Mossé, "Optimized Management of Power and Performance for Virtualized Heterogeneous Server Clusters," *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 23–32, May 2011.

[45] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A Comparison of High-Level Full-System Power Models," in *Proceedings of the 2008 conference on Power aware computing and systems*, HotPower'08, (San Diego, California, USA), USENIX Association, 2008.

[46]

[47] J. C. Mccullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, R. K. Gupta, and I. Labs, "Evaluating the Effectiveness of Model-Based Power Characterization," in *Proceedings of the 2011 USENIX conference on USENIX annual technical conference*, USENIXATC'11, (Berkeley, CA, USA), USENIX Association, 2011.

[48] A. Lewis, J. Simon, and N.-F. Tzeng, "Chaotic attractor prediction for server run-time energy consumption," in *Proceedings of the 2010 international conference on Power aware computing and systems*, pp. 1–16, USENIX Association, 2010.

[49] SSI, *Power Supply Design Guideline for 2008 Dual-Socket Servers and Workstations*, 1.2.1 ed., 2012.

[50] A. Papoulis and S. U. Pillai, *Probability, random variables, and stochastic processes*. McGraw Hill, 4 ed., 2002.

[51] Y. Ephraim and D. Malah, "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 6, pp. 1109—-1121, 1984.

Christoph Möbius received a Diploma in Computer Science from the Technische Universität Dresden, Germany in 2011. He is currently a doctoral candidate at the Faculty for Informatics at the same university. His research is supported by a funding obtained from the German Research Foundation (DFG) in the context of CRC 912. His research interest includes applying stochastic models to characterize and predict workloads and to characterize services according to their resource and power consumption history. His aim is to achieve adaptive and energy-aware service execution in large-scale Internet servers and data centers.

Waltenegus Dargie (SM'12) is an Associate Professor at the Technische Universität Dresden. He obtained a PhD in Computer Engineering from the Technische Universität Dresden in 2006 and holds MSc degree (2002) from the Technical University of Kaiserslautern (Germany) and BSc degree (1997) from the Nazareth Technical College (Ethiopia), both in Electrical Engineering. Dr. Dargie is a senior member of the IEEE and a member of the editorial board of the Journal of Network and Computer Applications. His research interest is related to wireless sensor networks, stochastic processes, and energy-efficient computing.

Alexander Schill is a professor of Computer Networks and Distributed Systems at Technische Universität Dresden, Germany. He holds a M.Sc. and a Ph.D. in Computer Science from the University of Karlsruhe, Germany, and received a Dr.h.c. from Universidad Nacional de Asuncion, Paraguay. From 2000 to 2006, he acted as dean of the Faculty for Informatics at his university in Dresden. Prof. Schill also worked with the IBM Thomas J. Watson Research Center, Yorktown Heights, New York. His major research interests are service-oriented computing, quality of service, and cloud computing.