Accurate Fine-Grained Processor Power Proxies

Wei Huang[†] Charles Lefurgy[*] William Kuk[**,‡] Alper Buyuktosunoglu[*]
Michael Floyd[**] Karthick Rajamani[*] Malcolm Allen-Ware[*] Bishop Brock[**]

[†]AMD, [*]IBM Research, [‡]Purdue University, [**]IBM System and Technology Group
WeiN.Huang@amd.com, {lefurgy,wwkuk,alperb,mfloyd,karthick,mware,bcbrock}@us.ibm.com [*]

Abstract

There are not yet practical and accurate ways to directly measure core power in a microprocessor. This limits the granularity of measurement and control for computer power management. We overcome this limitation by presenting an accurate runtime per-core power proxy which closely estimates true core power. This enables new fine-grained microprocessor power management techniques at the core level. For example, cloud environments could manage and bill virtual machines for energy consumption associated with the core. The power model underlying our power proxy also enables energy-efficiency controllers to perform what-if analysis, instead of merely reacting to current conditions.

We develop and validate a methodology for accurate power proxy training at both chip and core levels. Our implementation of power proxies uses on-chip logic in a high-performance multi-core processor and associated platform firmware. The power proxies account for full voltage and frequency ranges, as well as chip-to-chip process variations. For fixed clock frequency operation, a mean unsigned error of 1.8% for fine-grained 32ms samples across all workloads was achieved. For an interval of an entire workload, we achieve an average error of -0.2%. Similar results were achieved for voltage-scaling scenarios, too. We also present two sample applications of the power proxy: (1) per-core power billing for cloud computing services; and (2) simultaneous runtime energy saving comparisons among different power management policies without running each policy separately.

1. Introduction

In the last several years, direct measurement of power consumption has been widely deployed in servers [20]. At a system-level, bulk power supply measurement has enabled energy-efficiency optimizations, power capping and shifting, and cost-of-operation analysis. Power measurement of the microprocessor is becoming common as well and allows for more fine-grained power management. For example, Intel Node Manager 2.0, to be introduced this year, will use direct processor and memory power measurements to implement power capping by shifting power allocations between processor and memory subsystems [7]. Core-level management, on the other hand, has languished because there are still no accurate and practical ways to directly measure the power consumption associated with each core.

A viable alternative is to implement core *power proxies*, which are estimates of true core power consumption. They are constructed from real-time measurements of microarchitecture counters and physical sensors. Many previously published power proxy implementations made use of existing processor performance monitoring and analysis signals which were originally put into the hardware to assist in tuning compilers and operating mode settings. While these events are related to the activity of the processor, they typically track performance-sensitive events rather than events that contribute most to power consumption. This can lead to significant error across a wide variety of workloads. More recent work, including ours, leverage activity signals in the microarchitecture that correlate better with power. However, the prior studies have serious limitations. First, they typically model power at fixed voltage and frequencies, which ignores how the power proxy error tracks with dynamic frequency and voltage scaling processors. Second, they focus on active power and give only simplistic treatments of leakage power and do not consider the impact of process variations. This reduces the effectiveness of the proxy since conventional high-performance microprocessors have considerable leakage power consumption and the power consumption of processors of the same type and model can vary widely due to manufacturing-based variation. Without a strategy for covering the entire power of the core, the power proxy cannot be used for applications requiring high accuracy.

The value of accurate power proxies depends on how they are used. For billing applications, a 1% inaccuracy in energy consumption directly translates to an additional 1% cost to either the user or supplier. For power capping applications, reduced accuracy in power estimations means that additional margins added to the requested capping value must be taken to ensure the real power limit is maintained. In our system, we measure that every 1% of Vdd power accuracy translates into 1.2% throughput on the SPECpower workload. For example, the value of power capping with a power proxy that is 1% accurate compared to a power proxy that is 5% accurate is about 4.8% in performance. For energy-efficiency controllers that maximize operations per Watt, the inaccuracy may not matter when optimizing a single component if the power estimation is monotonic with true power. However, when optimizing across many processors that each provide a power estimate, suboptimal decisions could be made if the estimates reverse the true sense of which component draws the most power. Therefore, we believe that continuing to improve accuracy by even small amounts is meaningful. Additionally, shortening the time for estimation enables fine-grain power management. A review of prior work shows our power proxies achieve accuracy comparable to the best known solutions, but do so at a 30x smaller time resolution (32 ms estimations vs. 1 second estimations). Additionally, we validate the power proxies work across full frequency and voltage ranges, and account for process variation.

In this paper, we propose a methodology of constructing highly accurate power proxies, first at the chip level, then at the core level. Our architecture accounts for active power by utilizing specialized activity counters in the chip hardware. Firmware computes the final power proxy by using the measured activity values and incorporating real-time physical sensor measurements. In addition it calculates

[*]This work was done while Wei Huang was a researcher with IBM Research, and William Kuk was an intern with IBM System and Technology Group.

leakage power using manufacturing-time characterization data.

We implement our power proxy architecture on a IBM POWER7+® high-performance system and run many workloads to evaluate its accuracy over full voltage and frequency operating ranges. The power proxies also take into account chip-to-chip variations. We additionally show that the power proxy is flexible enough to account for power even when voltage and frequency pairings are not fixed, but can vary for undervolting and overclocking.

Similar power proxy on-chip circuits in AMD and Intel chips exist. However, implementation details and thorough accuracy evaluations of them have not been published. This paper also significantly extends previous IBM POWER7 power proxy publications by adding accurate models for both clock and leakage power, and for the first time presents a complete full-chip and per-core power proxy development methodology with better accuracy than published work.

After demonstrating an accurate power proxy, we illustrate two use scenarios: billing and predictive management.

First, power has become a precious resource in the data center as it has a growing impact in the cost of server ownership. In response, the idea of billing users for energy use in addition to time-based or MIPS-based accounting is gaining traction. Per-core power proxies open the opportunity for energy-based billing in cloud computing services.

Second, a plethora of runtime energy and power management techniques have been invented and implemented to achieve energy proportionality for servers and data centers to boost energy efficiency and reduce operational cost. Typically a trial-and-error method is used to determine which management policy is the most effective for a given workload. It is impractical to run the same workloads multiple times, each time with a different power management technique or policy enabled. Having power proxies that are accurate across voltage and frequency ranges solves this problem. The power consumption model that underlies the proxy can estimate the instantaneous power consumption for each power management technique simultaneously at run time according to its decision on operating voltage, frequency, temperature, as well as activity counts of the running workload, thus providing a direct comparison across all power management policies. The energy manager can then dynamically and intelligently select a policy in response to changing workload characteristics.

We summarize our contributions as follows:

- We develop a methodology to construct core-level and chip-level power proxies, which are implemented in hardware and system firmware.
- The power proxies take into account full voltage and frequency ranges. It is also adjustable to chip-to-chip process variations.
- We present the first power proxy that works accurately even when chip voltage and frequency settings do not have fixed pairings. This is useful for systems that dynamically undervolt (with fixed frequency) or overclock (with fixed voltage).
- The power proxy values are updated every 32 ms to enable fine-grain energy management which is 30x faster than prior work with comparable accuracy. The power proxies are based on activity counters and sensors for frequency, voltage and temperature that are gathered out-of-band so as not to disturb the running workload.
- We illustrate how to achieve per-core power accounting, which may be incorporated into existing proposals for per-VM power accounting for cloud computing services.
- We discuss the usefulness of power proxies for evaluating different energy management techniques simultaneously at run time.

Because of hardware differences (e.g. on-chip activity counter architecture, data collection rates, voltage rail power measurement availability, frequency range, process variation distribution, etc.), it is nearly impossible for us to conduct a direct comparison with existing methodologies using hardware measurements. However, we do cite and try our best to quantitatively compare with the claimed accuracy from existing work. In addition, we only solve one (probably the most complicated) part of the full-system power model, namely the power consumption of the processor. The methodology needs to be extended with additional techniques to account for power consumption of other system components.

The paper is organized as follows. An overview of power proxies in POWER7+ and discussion of design considerations is covered in Section 2. Next, Section 3 shows how we develop chip power models and incorporate core-based activity measurement with chip-based characterization. We report experimental results to determine their accuracy. Section 4 illustrates use cases for accurate core power estimates. In Section 5 we review related work. Finally, we conclude with Section 6.

2. Background

2.1. Power modeling

The power consumed by a microprocessor can be described by Eqn. (1).

$$\begin{aligned} P_{chip} &= P_{active} + P_{idle} \\ &= P_{active} + P_{clock} + P_{leak} \end{aligned} \tag{1}$$

P_{idle} represents the *idle power* consumed when the processor is on, but not executing instructions and P_{active} is the additional *active power* consumed due to instruction execution. The idle power can be further separated into clock grid power and temperature-dependent leakage power.

Prior work on power proxies has focused mainly on accurate active power estimations and given simplistic treatment of the idle power - often assuming a constant value or one that varied linearly with clock frequency. While this may be valid for steady-state workloads, it is not sufficient for dynamic workloads that induce changing voltages (voltage scaling) and temperatures in the chip since idle power depends strongly on voltage and temperature.

We describe our power model in more detail in Section 3.

2.2. POWER7 chip proxy logic

The POWER7 chip power proxy circuits have been disclosed before in prior publications [4][5][20]. The circuitry in POWER7+ is identical. These circuits are used to estimate core active power and alone cannot accurately account for idle power. Our work complements the prior publications by demonstrating how firmware can be used to accurately account for idle power and for voltage-scaling in the active power component.

The methodology to estimate on-chip active power is to accumulate a weighted sum of activity counters each measurement interval. We use the term *activity proxy* to denote this aggregated activity count. Each chiplet (combination of a single core with its L2 and L3

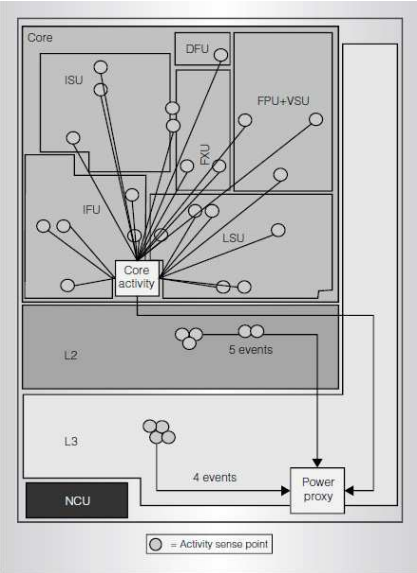

Figure 1: Activity counters in a POWER7 chiplet [4]. POWER7+ chiplet floorplan is slightly different.

caches) in POWER7+ contains activity proxy logic so that per-core active power can be separately accounted. Firmware then adjusts the core activity proxies for the effects of leakage, temperature, process variations and voltage to form the chip and core *power proxies*, which are the final estimations of true power consumption. A discussion of the firmware is in Section 3. The remainder of this section overviews the POWER7 chip power proxy circuit.

Figure 1, reproduced from Floyd et al. [4], shows a diagram of activity proxy event collection in the POWER7 processor chiplet. The activity signals were selected during the initial POWER7 microarchitecture design phase. The top events that caused the majority of the power consumption in each functional unit (e.g. Dispatch, Fixed point, Load-Store) were considered. Example events include: the deactivation of dynamic clock gating, data switching, and register file or array accesses. Relevant signals were added directly from those units to the on-chip proxy logic which adds a per-event programmable weight to an accumulation register whenever that event occurs. Care was taken to avoid redundant counting whenever possible. For example, counting Load-Store issue already covers data cache and D-ERAT (effective-to-real data address translation) reads and Load Miss Queues so both were not included. For some frequent events, such as General Purpose Register File access, a pre-count is performed on the activity before sending a summary signal out of the unit. Other events, such as instruction pipe issue or floating point operation type, are sent as encoded values and a multiplier is applied accordingly based on its anticipated power relative to the other types in that encode group. The L2 and L3 cache units also provide active power events based on cache lookup and types of access. All these activity count accumulators are then scaled and summed to form an aggregate activity proxy per core which can be converted into an estimate of active power consumed over the previous time period.

There are several challenges in such weighted counter-based proxy architecture, including:

- How to choose the minimal set of key activity counts to architect;
- Correctly sizing the counter and weight registers and the final scaling logic; and
- How to decompose the weighted aggregation step so as to minimize hardware complexity and calculation time.

To address these challenges, the methodology relies on systematic, linear-regression based formalism in conjunction with designer intuition and experience. A genetic algorithm (GA) optimization tool further refined the design under hardware constraints. Specifically, the reference (RTL-validated) performance simulator-driven power simulator projects the core power consumption across a carefully selected range of workloads. In each case, hundreds of activity count events were also collected over a pre-selected execution time window. The resulting matrix of several data elements (consisting of power values and activity counts) was fed into the GA-driven regression solver, to deduce the right set of activity counts as well as the size of weights to form the architecture. At the end, activity count events are carefully selected to capture those that correlate maximally with active power consumption as well as those that are the fundamental (pseudo-independent) positive correlators of power. The final design has an affordable number of hardware counters and weight bits which led to an accurate, yet flexible proxy architecture with reasonable cost. The final design measures close to 50 activity counts which include instructions dispatched, instructions completed, execution register file accesses, execution pipeline issue types, instruction fetch unit activity, load-store unit cache activity, load-store unit D-ERAT activity, load-store unit prefetch activity, L2 cache reads/writes and L3 cache reads/writes.

3. Power Proxies

In this section, we demonstrate our methodology and verify its accuracy by creating a proxy that replicates the physical power sensor for the chip Vdd power rail.

3.1. Experimental setup

We have implemented the power proxy in a prototype high-performance POWER7+ server. It has four microprocessors (P0-P3) with 6-cores each, 256 GB of main memory, and runs AIX 7.1. The maximum core clock frequency is 4228MHz. Voltage is controlled independently for each microprocessor with all cores on a chip sharing the same voltage level. A per-core digital phase-lock loop allows clock frequency to be set independently for each core. We purposefully selected the four microprocessors from different process corners and they are distinct in terms of leakage current, nominal supply voltage, and ring-oscillator delay measurements. This allows us to confirm that the power proxy accounts for manufacturing-based variation. We pick P1 as the reference chip because it has a more representative nominal voltage than other three chips. In the following text, for results that are related to a single chip, we show the results on P1.

The power proxies are implemented in two parts in a POWER7+ system. First, activity counters and the calculation of the activity proxies are implemented in hardware logic of the processor. The weights to different activity events are programmable by writing to special on-chip registers. Second, a service coprocessor receives measurements of activity proxies, chip supply voltage, core clock frequencies, and core temperatures from the POWER7+. The measurements are sent over a special out-of-band management interface that does not disrupt running workloads. The chip-level and core-level power proxies are calculated in firmware that runs on the service coprocessor. The firmware performs this computation every 32 milliseconds, which is constrained by the narrow bandwidth of the management interface.

3.1.1. Power delivery path The POWER7+ processor has four independent voltage rails, each fed by a voltage regulator module (VRM). Two of the voltage rails (Vio and Vmem) require fixed voltages, established at chip manufacturing, and are unique for each chip. The other two voltage rails (Vdd and Vcs) have voltages that can be dynamically changed to implement DVFS policies.

The power proxy hardware does not cover circuitry on the Vio and Vmem rails. This design choice was driven by the fact that the current drawn on these rails varies only a small amount and because the load is largely constant. During chip manufacturing, the current associated with Vio and Vmem is measured during a calibration workload and stored in the chip's Vital Product Data (VPD) non-volatile memory along with the associated voltage.

The Vdd and Vcs rails do have circuitry associated with the power proxy hardware. The Vdd rail feeds the cache and core logic and the Vcs rail feeds the L3 cache (embedded DRAM) on the chip. For both rails, chip manufacturing also measures current and voltages, but instead of the single point characterization done for Vio and Vmem, there are four unique points measured for each of the Vdd and Vcs rails. The four unique points cover four standard operating points across the full range of voltages and the associated frequencies that the chip can operate at safely, and these values are written into the chip's VPD. The VPD values are used by product firmware to program the Vdd and Vcs VRMs so that the output voltages from the VRM will be sufficient to establish the characterized chip voltage under worst-case conditions considering load line and other losses in the power delivery system.

The Vdd rail is the most interesting power rail in terms of dynamic power management, as it carries significantly more current than Vcs. The Vdd and Vcs voltages can be varied over a very wide range along with the frequency, with Vcs scaling proportionally to Vdd. At the low end of the voltage range, the minimum Vdd voltage or Vmin is typically only 70% of the maximum Vdd voltage or Vmax used. To go with this, the frequency range varies from a minimum frequency that is only 54.5% of the maximum frequency used.

Roughly 95% of the activity that the activity proxy measures is associated with the Vdd rail. For this reason the paper focuses exclusively around precise characterization of the Vdd rail. The methodology we present in this paper can be easily extended to the Vcs rail, and we leave this as future work.

3.1.2. Sensing In our test system, the power on the input side (12V rail) of the chip Vdd voltage regulator is directly measured with an accuracy of 2%. Therefore, our measurement includes loss due to voltage regulator conversion inefficiency. The firmware reads analog-to-digital converters to measure the average power during each 32 millisecond interval. We use this as the ground truth in all experiments and accuracy claims. The goal of our power proxy is to replicate as closely as possible the measured chip Vdd power for each 32 ms interval. We measure the per-core temperature by averaging the 5 digital thermal sensors located in each core. They provide temperature in units of 1 degree Celsius and are accurate to within 4 degrees of the true temperature.

3.2. Kernels and Benchmarks

We use two sets of workloads for building our runtime power proxies. First, kernels from a variety of available system characterization sources are used for training the activity proxy weights and power model coefficients. Second, testing and validation are done with a separate set of benchmarks.

The majority of our training workloads are kernels constructed from simple array-based loops such as in the popular `lmbench` [14] and STREAM [13] benchmarks. A desired size array is allocated and kernel-specific operations are performed in a sequential or random order over the array elements. By varying the nature of operations, number of distinct arrays and sizes of arrays a variety of workload instruction and storage access patterns are emulated. This gives us a reasonably rich set of power exercisers for the cores, on-chip caches and logic for accessing the different memory layers including off-chip DRAM. We additionally vary the workloads by selecting the multithreading mode of the chip. In all, we use 762 unique workloads for training. This simple kernel-based characterization also helps us set up very steady workloads in terms of activity and power to enable truly representative power measurements to be taken for the training phases. We complement these loops with a set of system stress-test workloads which are targeted at specific components of the system. This set also includes a maximum power workload developed for the POWER7 processor.

For testing and validation we use two sets of popular benchmarks from SPEC—SPECPower_ssj2008 [18] and SPEC CPU2006 [17]. The former provides workloads of different intensities helping us evaluate our models across the full range of system loads and the latter provides a rich variety of processor and memory hierarchy usage examples.

3.3. Activity proxy training

We use a procedure similar to [4] to train weights in the activity proxy calculation. All cores in all chips use the same learned weights. First, we measure the power consumption and temperature of the chip when it is idle. When running the training set of benchmarks, we set the processor (P1) to its nominal frequency and supply voltages. All cores run the same workload. We then sample power and temperature measurements as well as activity counts for each event. Per-core active power is calculated by subtracting the estimated idle power, which includes a temperature-based adjustment (Section 3.4), from total measured power and dividing by the number of cores. For each activity event, we also average across all the cores to reach a per-core count for that event. The IBM SNAP genetic algorithm optimization tool takes per-core active power and activity counts as inputs to derive a linear regression model for active power, in the form of

$$ActivityProxy = \Sigma(W_g \times \Sigma(W_{i_g} \times A_{i_g})) \quad (2)$$

where W_g is an activity group weight, W_{i_g} is the weight for activity event i in group g, and A_{i_g} is the count for event i in group g. The POWER7+ hardware overhead is minimized by splitting the activity weights into an event weight and a group weight. We use a genetic algorithm to optimize the weights rather than a simple linear least-square fit due to the limited scaling ranges and the fact that only W_g is signed. Once the set of weights is determined, the same weights are programmed into all the cores across all the processors. POWER7+ implements the activity proxy, Eqn. (2), in hardware.

Aside from the temperature effects previously mentioned, other sources of correctable errors and biases need to be considered in designing training experiments. One type of potentially correctable error is due to Simultaneous Multi-Threading (SMT) mode, which is effectively the number of active threads per core. We observe systematic variations of up to 5% in estimated power in the training sets based on whether the workload is running with 1, 2 or 4 threads per

core. Consequently, our training kernels are run in all SMT modes.

Even after correctable errors are considered, systematic errors exist that will always bound the maximum accuracy of an active power model based simply on event counting. One important example, to be addressed by future work, is the dependency of processor power on the actual data being processed, as the number of nodes switching at each cycle is data dependent. Cache and register file access power may also vary based on the data stored in the arrays. Using some simple cache-contained integer loops running on a POWER7 processor, we observed variations in total active power of up to 5% depending on the randomness of the data being processed, with power increasing with increasing randomness.

3.4. Chip-level power model

Activity proxy only accounts for active power at the nominal operating point, which is a specific frequency, and associated voltage values set defined in the chip VPD and identified as the default frequency for the processor. In order to achieve a true power proxy, we also must consider other factors, namely the whole supply voltage range and temperature-dependent leakage power. The impact of frequency on power consumption is largely captured by activity counters, since higher frequency results in proportionally more event counts for the same workload.

Chip power consumption can be further divided into idle power and active power, both of which are dependent on supply voltage. Idle power in turn can be divided into leakage power and clock grid power. Leakage power is also dependent on temperature. We model chip-level power consumption in the following format:

$$\begin{aligned} P_{chip} &= P_{active} + P_{clock} + P_{leak} \\ &= \frac{AP}{R}\left(\frac{V}{V_{nom}}\right)^{\alpha} + \frac{Freq}{S_0}\left(\frac{V}{V_{nom0}}\right)^{\beta} \\ &\quad + P_{leak\_nom}\left(\frac{V}{V_{nom}}\right)^{\gamma}(1+m_0(T-T_0)) \end{aligned} \tag{3}$$

where AP is the activity proxy; R is the ratio between activity proxy and active power for the reference processor at nominal frequency; V is the supply voltage at the VRM output; V_{nom} is the nominal voltage for each chip at the VRM output; $Freq$ is the chip frequency; S_0 is a constant scaling factor across all chips and is derived during chip characterization; V_{nom0} is the nominal voltage at VRM output for the characterized chip; P_{leak_nom} is the idle leakage power of the chip at nominal voltage; m_0 is a linear scaling factor for temperature-dependency of leakage power; T and T_0 are the actual temperature and characterization temperature of the chip, respectively; α, β and γ are constant exponents derived from characterizing the reference chip.

Among all these parameters in the model, S_0, T_0, m_0, V_{nom0}, α, β and γ are constants, regardless of chip-to-chip variations. R, V_{nom} and P_{leak_nom} are unique per chip, and R can be calculated based on VPD data (more details in Section 3.6). AP, $Freq$, V and T are runtime measurements at the chip level.

It is interesting to note that the active power term in Eqn. (3) does not include chip frequency. This is because changes to core frequency are reflected in the rate at which the activity proxy to accumulate events. Therefore, AP naturally incorporates core frequency.

Through a series of experiments we found the dependence of leakage power on temperature is approximately linear for a fixed voltage. This is the reason we include a linear dependence on temperature in the leakage power term in Eqn. (3). We ran different steady-state workloads such as the kernel loops and the maximum power workload with constant inputs at nominal voltage and frequency, each starting from a cool temperature and gradually reaching a warmer steady temperature. The only factor that causes power change for each workload is temperature-dependent leakage power. We found this power component is linearly proportional to temperature change.

We used the IBM SNAP genetic algorithm optimizer to determine the parameters for Eqn. (3). First, we measured the idle power and temperature of each chip across 22 voltage settings and 22 frequency settings, for a combination of 253 unique voltage-frequency points. Then we programmed SNAP to find the parameters that minimized the error for the idle power across all chips every voltage-frequency points. We obtained $S_0 = 159.634$, $m_0 \approx 0.031 W/^{\circ}C$, $\beta = 1.584$, $\gamma = 4.070$, and a unique P_{leak_nom} value for each chip.

We verified P_{leak_nom} for all the chips by measuring leakage power at nominal Vdd (when the chip is idle and before clock grid is enabled). After that, we repeat the same procedure for different voltages, to verify $\gamma = 4.070$ across all the chips. We verified S_0 by measuring chip idle power (including both leakage power and clock grid power) at nominal frequency, and subtracting leakage power calculated above from idle power.

With the total chip power measurement and the idle power model, we can calculate active power. From multiple training workloads and we find that $R = 2450$ for the reference chip, and $\alpha = 2.2$ for active power. The validation of total chip power is shown in Fig. 5 in Section 3.6, as we also show the total power for each of the four processors with manufacture-based variations.

3.5. Total chip power and idle power estimation results

We now present our main results for the total chip power model and idle power model. Additionally, we show that our models are accurate even when the chips are undervolted.

Fig. 2 shows results for total chip power for training kernels, testing kernels/SPECpower, and SPEC CPU2006. We report two metrics here. The first one is the absolute (i.e. unsigned) average percentage error among 32 ms samples of each workload, see Fig. 2(a)-(c). This metric helps evaluate instantaneous power proxy errors and is useful for what-if scenarios for runtime power management policies that need to make many decisions every second. The second metric is the average percentage error for each entire workload run, see Fig. 2(d)-(f). This is useful for evaluating energy consumption over a relatively longer time, such as a minute or longer.

We achieve 1.8% (std. dev. 2.0%) unsigned percentage errors for 32 ms samples of total chip power across all workloads (the last bar in Fig. 2(a)-(c)). The low standard deviation means the errors from a vast majority of activity proxy samples are close to each other. For mean percentage errors for each entire workload (Fig. 2(d)-(f)), on average, we achieve -0.2% (std. dev. 2.6%). This indicates the set of weights from the training is especially useful for long-term energy estimation. The worst case error across all testing workloads is under 9.5% (vector copy kernel). For SPEC CPU2006, the worst case workload error is 8.1% for *calculix*. This compares well to prior work [6] that achieved a median error of 1-5% (maximum error 7-10.7%) for SPEC CPU2006 workloads across multiple chip architectures, but used 1-second samples for validation.

Our idle power model ($P_{leak} + P_{clock}$) has a maximum error of 2 W across all chips and voltage-frequency points. The maximum

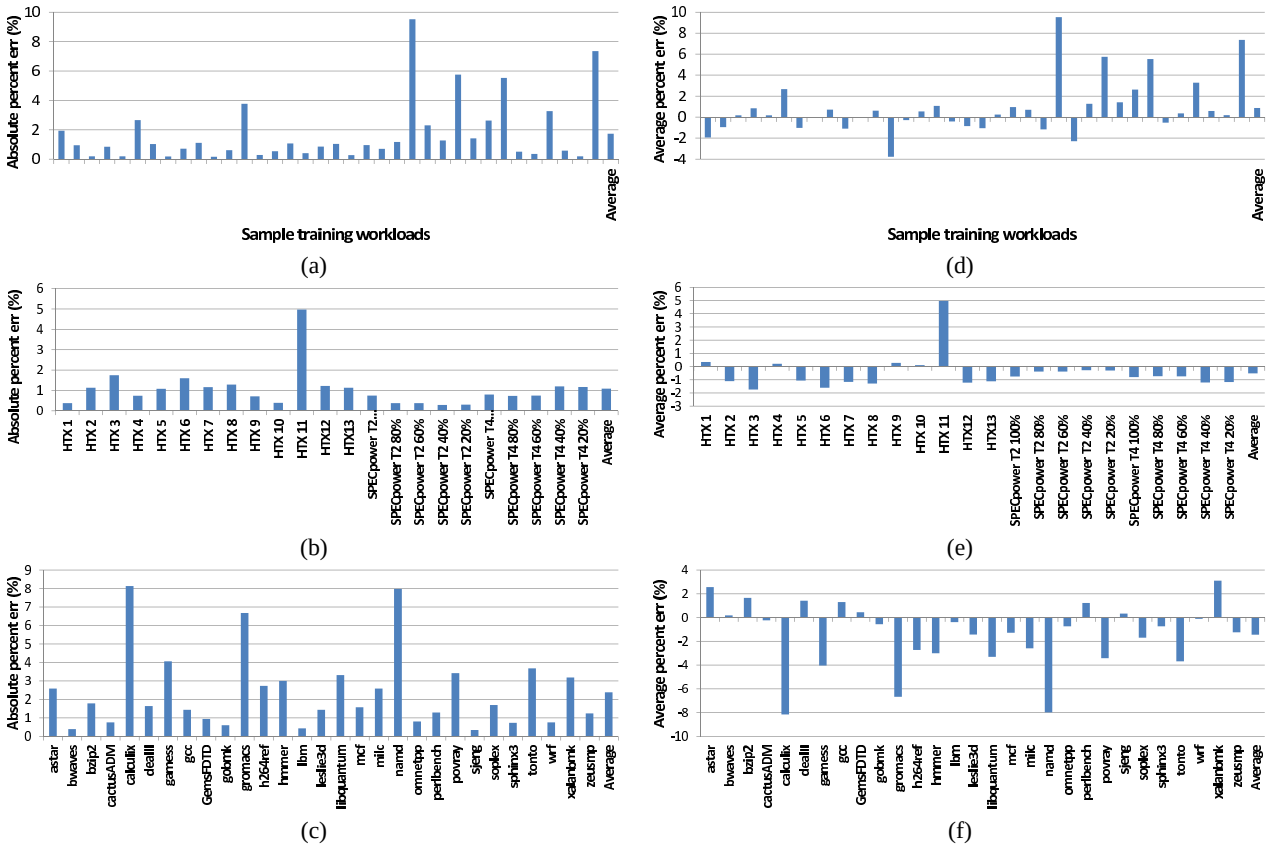

Figure 2: (a)-(c): Chip power (at nominal frequency) absolute (unsigned) percentage errors across all samples for training kernels, testing kernels/SPECpower, and SPEC CPU2006, respectively. (d)-(f): Chip power (at nominal frequency) average relative errors for each workload run.

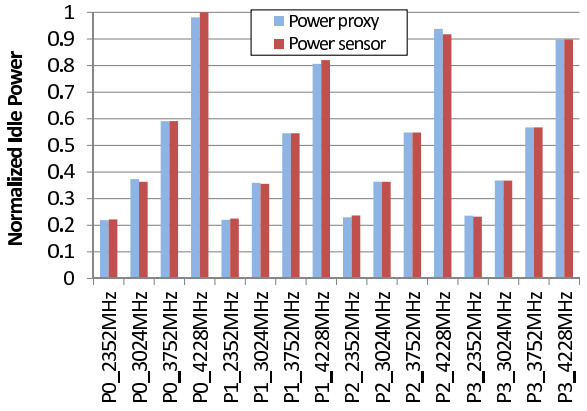

Figure 3: Idle power model validation at four different frequencies.

percent error was 3% which occurs near the lowest idle power measured. Fig. 3 shows the normalized idle power (i.e. leakage power and clock power) at different voltage-frequency pairs for all four chips.

So far, we have derived and verified the chip-level power proxy for a wide range of voltage-frequency pairs. An interesting experiment would be to test the power proxy's accuracy when voltage and frequency pairs are not fixed. Recently, Lefurgy et al. [12] proposed a method of safely undervolting a microprocessor while maintaining the same frequency. Voltage is dynamically selected to maintain a preset guardband level as chip activity changes, resulting in reduced chip power without performance loss. We apply this technique when running the maximum power workload, and allow undervolting up to 112.5 mV (about 9.5% of supply voltage) at the fixed frequency of 4228MHz. The first half of Fig. 4 is for the case where frequency is fixed at 4228 MHz and voltage is set to the traditional corresponding value, whereas the second half of the figure shows the case where dynamic undervolting is enabled such as frequency is fixed while voltage changes with available timing margin. This results in about 25% power reduction for *dealII* in the second half. Comparing the two halves, we see that for decoupled voltages and frequencies, our chip-level power proxy still achieves about the same level of accuracy when frequency and voltage are decoupled from each other. That is, 9.5% variations of supply voltage do not lead to worse power proxy estimation.

3.6. Chip-to-chip variations

The chip-level power proxy (or model) presented above is characterized from a single reference chip and does not consider chip-to-chip variations due to uncertainties in the manufacturing process. Therefore, Eqn. (3) must be adjusted to take variations into account. As mentioned before, to maximize the process variations among the tested chip, we intentionally evaluate four chips from distinct process corners. For example, when considering Performance Sort Ring Oscillator (PSRO) measurements, one chip is 2.6 standard deviation slow, one chip is 1.1 standard deviation fast and the other two chips

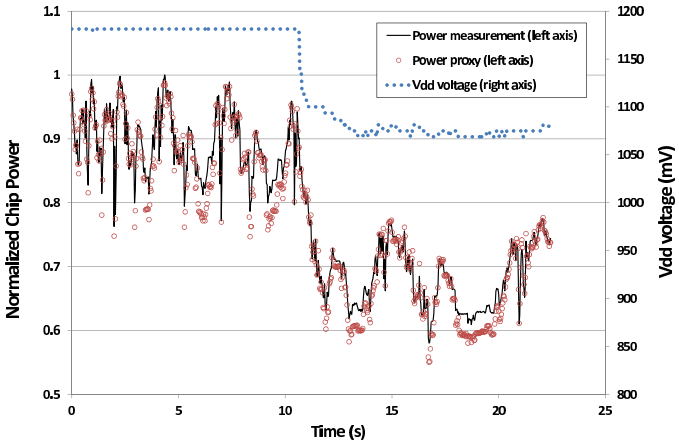

Figure 4: Chip power proxy validation for *dealII* with decoupled voltage and frequency.

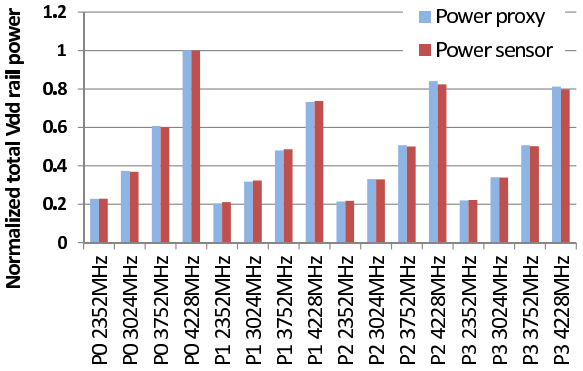

Figure 5: Chip power model validation at four different frequencies for the maximum power workload. Chip-to-chip variation is also accurately accounted for.

are within 0.5 standard deviation of the mean PSRO value. More simply, the PSRO spread of our 4 chips covers 88% of all chips from a sample size of thousands of chips passing module test. Data in the paper also show significant leakage power variations.

For clock power (P_{clock}), $Freq$ can be measured at run time, S_0 and β are constant across all chips. V_{nom0} is also a constant, which is the nominal Vdd for the reference chip. The only variance in clock power from one chip to another is from its supply voltage V that can be measured at run time. For the same operating frequency, different chips have their different characterized supply voltages. Therefore, variations among chips for the clock power component can be accounted for by the different settings of supply voltages.

For the leakage power component (P_{leak}), although each chip's leakage power at nominal voltage (the characterized voltage for this chip to run at nominal frequency) can be measured during system start up, it is also possible to calculate it by subtracting the clock power from the measured idle power for each chip. Both methods achieve almost the same accuracy. The variation among leakage power can be easily calculated from measurements. The assumptions of a constant γ and a constant m_0 for all chips are also valid according to Fig. 3.

For active power, there are two ways to adjust for chip-to-chip variations. The first approach is to utilize the characterized variation information in the VPD data. As mentioned in Section 3.1.1, each chip has VPD data that is established during chip manufacturing test that is unique to that chip. The uniqueness addresses manufacturing variability, and system specific information for the target system it is going into including load line effects between the VRM and the chip package and the losses associated with the unique package being used for that die in the system.

In Eqn. (3), for P_{active}, we can derive a chip-specific value for R based on VPD, load-line equation, and VRM efficiency. Specifically, as mentioned before, $R = (AP)_{nom}/P_{active\_nom}$. We denote R_0 for the reference chip, and R_1 for an un-characterized chip that we want to adjust the active power. If both chips run the same workload, at the same frequency, in the same environment, we know that $(AP)_{nom} = R_0 P_{active\_nom0} = R_1 P_{active\_nom1}$ because both chips have the same amount of activities. Therefore,

$$R_1 = R_0 \left(\frac{P_{active\_nom0}}{P_{active\_nom1}} \right) \tag{4}$$

where

$$P_{\text{active}\_i} = P_{\text{VRM}\_i} - P_{\text{idle}\_i} \quad ...(i = 0, 1) \tag{5}$$

We are able to calculate R_1 for any processors that are not characterized since 1) R_0 is known, 2) P_{idle_i} can be calculated for each chip, and 3) P_{VRM_i} can be calculated from the VPD data of chip i together with knowledge of the load-line equation and VRM efficiency.

However, VRM efficiency is quite sensitive to the load current and the number of phases, and can vary as much as 10%, causing a noticeable error in the resulting R_i calculation. Instead, we found that modifying the active power to the following format results in better accuracy for all the chips:

$$P_{active} = \frac{AP}{R_0} \left(\frac{V}{V_{nom0}} \right)^{\alpha} \tag{6}$$

Where R_0 is a constant value from the reference chip, and V_{nom0} is the nominal Vdd for the reference chip, too. The underlying reason is similar to the clock power component—chip-to-chip variation is largely captured by the different characterized supply voltage settings for different chips at the same operating frequency.

Therefore, the final format of the chip-level power proxy adjusted for chip-to-chip variations is:

$$\begin{aligned} P_{chip} &= P_{active} + P_{clock} + P_{leak} \\ &= \frac{AP}{R_0} \left(\frac{V}{V_{nom0}} \right)^{\alpha} + \frac{Freq}{S_0} \left(\frac{V}{V_{nom0}} \right)^{\beta} \\ &\quad + P_{leak\_nom} \left(\frac{V}{V_{nom}} \right)^{\gamma} (1 + m_0 (T - T_0)) \end{aligned} \tag{7}$$

Fig. 5 compares the modeled total power and measured total power for the maximum power workload for different voltage-frequency pairs, across all four chips. It shows that our proposed chip-level power proxy works accurately despite the large manufacture process variations among the chips. Similar results are achieved for other workloads, too.

3.7. Core-level power proxy

For POWER7+, in order to reach power consumption estimation associated with each core, we take the following steps.

- Active power: We begin with the per-core activity proxy value AP_i calculated by the core. R and V are the same for all the cores in one processor, since all cores share the same voltage rails in POWER7+. Specifically,

$$P_{\text{active\_core\_}i} = \frac{AP_i}{R_0}\left(\frac{V}{V_{nom0}}\right)^{\alpha} \tag{8}$$

- Clock grid power: first use average frequency across all the cores to calculate chip-level clock power. Then divide it to each core's contribution by proportionally scaling to the core's frequency. Specifically,

$$\begin{aligned} P_{\text{clock\_core\_}i} &= \frac{Freq_i}{N_{cores}Freq_{avg}}\frac{Freq_{avg}}{S_0}\left(\frac{V}{V_{nom0}}\right)^{\beta} \\ &= \frac{Freq_i}{S_0 \cdot N_{cores}}\left(\frac{V}{V_{nom0}}\right)^{\beta} \end{aligned} \tag{9}$$

- Leakage power: first use Eqn 3 to calculate chip-level leakage power. Then divide it to each core's contribution by proportionally scaling to the core's temperature change. Specifically,

$$P_{\text{leak\_core\_}i} = \frac{P_{leak\_nom}}{N_{cores}}\left(\frac{V}{V_{nom}}\right)^{\gamma}(1+m_0(T_i - T_{i0})) \tag{10}$$

  where T_i is the core temperature at run time, and T_{i0} is the core temperature during characterization.

- Final adjustment: each core's power proxy now becomes $Proxy_i = P_{\text{active\_core\_}i} + P_{\text{clock\_core\_}i} + P_{\text{leak\_core\_}i}$. To account for the difference between power proxies and power measurement, we can adjust the sum of all power proxies from all the cores to be equal to total measured chip power, by multiplying with scaling factor that equals to $P_{measured}/\Sigma Proxy_i$.

Although POWER7+ does not have individual voltage rails at the granularity of cores, the above approach can be easily extended to such situations by using V from each core. Use of core-level power proxies is shown in Section 4.1.

4. Use Cases of Power Proxies

The chip-level and core-level power proxies that incorporate voltage, frequency and process variations allow the implementation of many novel ideas that are otherwise impractical or impossible. One interesting usage scenario of per-core power proxies is in a power capping environment with a power constraint at the processor socket level. Our per-core power proxies enable a better judgment for balancing power among cores. Prior work [11] shows that every 1% improvement in power estimation accuracy can lead to roughly 1% performance improvement in a power-capped scenario, due to less guardbanding.

In general, this work provides the means of using core-level power proxies for both power-based accounting/billing of virtual machines and more fine-grained power management. This section provides two such example applications.

4.1. Fine-grained power accounting

Power proxies, especially per-core power proxies enable power-based billing for cloud computing services. Our power proxy will work with different assignments of cores to virtual machine and with the virtual machines operating the cores using different partition-level power management techniques. For virtual machines at the sub-core level (i.e. a subset of threads out of a SMT core), further extensions to per-thread power proxies are necessary, which is beyond the scope of this paper.

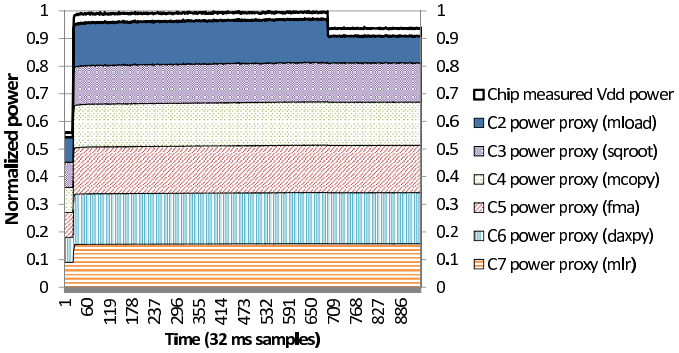

Figure 6: Stacked core power proxy for an estimate of total chip power for a multiprogram workload. Top solid line is the measured chip power.

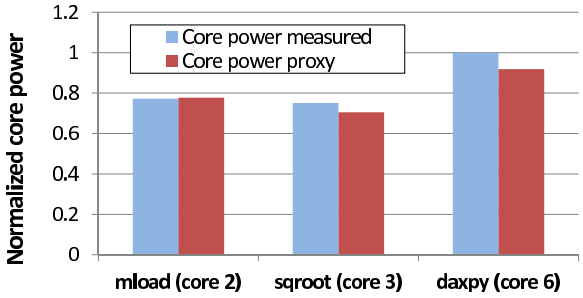

Figure 7: Validation of core power proxy for three kernel benchmarks running on individual cores.

We construct a case to show core-level power estimations on a 6-core processor. Each core runs an independent workload from our kernel benchmarks and each workload has been configured with different memory footprints and different number of threads (1, 2 or 4). The variations among workloads cause power consumption difference among cores. Fig. 6 shows the normalized power over time. We stack the six core power proxies together to get an estimated total chip power, each core power proxy is represented by one pattern in the plot. The top-most solid line is the measured chip Vdd power. As we can see, all the workloads start at the same time. But the kernel on core 2 ends earlier than the others. The sum of the core power proxies is about 3.0% less than the measured chip power.

In order to validate each core power proxy's accuracy, we pick three of the kernels that have memory footprint contained within per-core L2 cache from previous experiment, and run each alone on its associated core with all other cores idle. We calculate the "measured" core power by

$$P_{core\_measured} = \frac{P_{idle}}{N_{cores}} + (P_{chip} - P_{idle}) \tag{11}$$

and compare it with the temperature adjusted core power proxy for those cores in Fig. 6. The results in Fig. 7 show the core power proxies are quite accurate (0.6%, -6.2% and -8.2%, respectively).

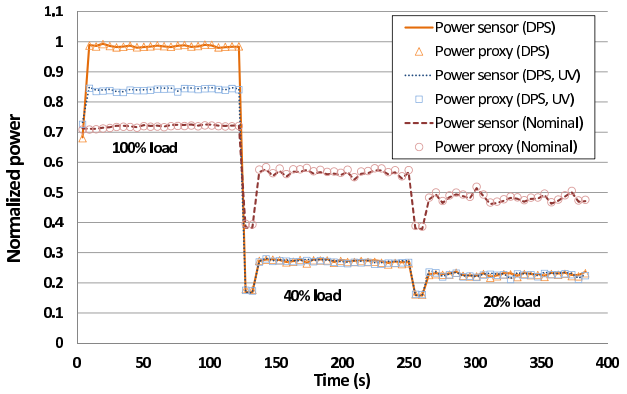

Figure 8: Chip power proxy comparison among three power management policies: Fixed nominal frequency, Dynamic Power Saving (DPS), Dynamic Power Saving with undervolting (DPS, UV).

4.2. Run-time power saving estimation

In this section, we evaluate different power management policies for a workload. Fig. 8 shows runtime power comparison of a SPECpower run under three power management policies. SPECpower requires each load level run for fixed amounts of time, regardless of operating mode. That is why the total run times in all cases are the same in the figure. Additionally, Fig. 8 shows DPS and DPS/UV have better "power proportionality" to load levels than Nominal. This results in significantly improved SPECpower scores for DPS and DPS/UV.

We show three calibration phases, one 100% load level, one 40% level and one 20% load level. The Nominal policy uses a fixed frequency and a fixed voltage throughout the run; The Dynamic Power Saving (DPS) policy dynamically adjusts voltage and frequency to keep processor at a relatively constant high utilization level. The DPS/UV (undervolting) policy allows dynamically lowering voltage for high load levels without changing frequencies, in order to reduce static margins and achieve higher power efficiency. Both chip power measurements and chip power proxy are shown. As can be seen, the chip power proxies match well with the power measurements in all cases.

It is also interesting to compare the three policies. The fixed frequency mode has relatively flat chip power consumption, despite the significant change in load levels and processor utilization levels. The DPS and DPS/UV modes achieve higher frequency, hence higher performance at high load levels, and significantly reduced power consumption at lower load levels and idle state. Thus these modes are more “power proportional” in that they respond to performance demand. In addition, the benefit of DPS/UV is evident that it consumes 15% less power at full load levels, while keeping the same peak performance.

We expect that a chip-level power proxy allows on-the-fly and accurate evaluation of such power management policies. Traditionally, each power management technique is implemented separately in hardware, firmware or software. The same set of workloads must be executed multiple times, once for each technique. Great care must be taken to ensure each run has the same architectural, environmental and initial conditions. This process is time consuming and not rigorous. With an accurate chip-level power proxy, processor power consumptions of different power management policies can be calculated simultaneously for different voltage, frequency and temperature scenarios.

5. Related Work

Bellosa [1] developed some of the first microprocessor power models based on performance counter measurement.

Contreras and Martonosi [2] augment the performance counter-based linear regression model of CPU power for a Intel PXA255 unicore processor to account for voltage scaling by using different weights for the performance counters for each voltage and frequency pairs. In modern server chips, voltages are tuned for each chip so that chips running at the same frequency in the same system likely use different voltages. As undervolting and overclocking become common place in commercial computers, the voltage and frequency pairs are not stable at run time and instead depend on the running workload and the electrical guardbands within the processor. Our methodology can account for both different chip voltages as well as dynamic undervolting and overclocking.

Intel’s Common Activity-based Model for Power (CAMP) [15] uses just 9 micro-architectural events in the processor to create models of activity factor for 180 physical structures in the processor. The activity factors are used to generate per-structure power models that can be combined to provide core-level dynamic power estimates at run time. They show an 8% average error for the core-level dynamic power and a maximum error of up to 12% for entire workloads. The comparison is against a detailed power simulator which itself is estimated to be 5% to 10% accurate. Additionally, the authors provide an excellent summary of prior work in the area. A limitation of the work is that it does not provide a methodology for dealing with run-time voltage scaling or manufacturing variation. Our work is distinguished from Intel’s CAMP work in that 1) we implement our solution in a real chip (not a simulator), 2) we account for manufacturing variation, and 3) variation in voltage and frequency. For example, two cores at different clock frequencies (due to utilization-based frequency selection), but operating on a shared voltage rail set for the higher frequency.

Jacobson et al. [8] improves on Powell et al. by providing a methodology for selecting the best architectural events. The authors use abstracted microarchitectural scaling models that are useful in early-stage power modeling of future generation designs.

Goel et al. [6] derives performance-counter based core-power models and tunes them using real system power measurement using linear regression. One difference from prior work is the inclusion of core-level temperature sensors. Another novel feature is using linear, inverse, exponential, logarithmic, or square-root transformations to scale the performance counters before the linear regression step to correlate better with power consumption. They achieve median errors per benchmark suite of 1-5% across six different CPU models, which demonstrates a portable methodology. Error in average power for individual workloads was measured up to 11%. This work has some of the best error reporting of prior work and an extensive review of prior work.

There are two recent papers that account system power to virtual machines (VMs). They both attempt to account CPU, memory, and device power to VMs running on the system. We limit discussion to the CPU power, which is the sole focus of our work.

Stoess et al. [19] account for processor power by recording CPU performance counters at VM context switches, weighting and accumulating the counters to form a power proxy, and assigning the power to the associated VM. Power during idle periods is equally di-

vided among the running VMs. A limitation of the work is its implementation on a uniprocessor system running a single core. The work does not consider how power could be allocated to virtual machines running on each core, or how voltage scaling or CPU temperature affect power consumption. We address these limitations in our work by architecting a power proxy for each processor core and accounting for voltage and temperature variation across cores and chips in the system.

Kansal et al. [10] take another approach to account for CPU power by tracking the logical CPU utilization of each VM. A simple, linear relationship is used to relate the utilization to a power consumption. The system-level power accuracy is measured to be within 5%. However, such models cannot accurately account power at a core-level due to manufacturing variation between cores, workload variation, or temperature variation. Our modeling uses fabrication-time testing to account for manufacturing variation and run-time sensing to account for workload variation and temperature.

Much of the prior work does not provide error estimates based on measured CPU power. Often system power measurement is used and discounted by power measurements for various system devices to arrive at the CPU power measurement. We base our error estimates on a highly accurate Vdd-rail current sensor for the CPU socket.

Do, Rawshdeh, and Shi [3] propose an application-level programming interface to allow processes to monitor their energy consumption. Their CPU power model relies on assigning a fixed power consumption to each processor frequency state and a fixed energy to frequency transitions. It does not deal with nuances of how instructions actually use the processor or manufacturing variation. Their reported results for an estimated system power of a laptop appear to have over 11% mean error. Our work could be used as a replacement (with higher accuracy) for their CPU power modeling.

AMD includes power-monitoring circuit in its processor cores [9]. 95 activity signals per core are monitored and weighted to form a dynamic power estimate that is considered to be 2% accurate. Since the purpose is for long-term thermal and power control, the circuitry is optimized to eliminate high-speed routing by not sampling every signal every cycle. It takes hundreds of time-based samples to achieve accurate dynamic power estimations.

Intel's Tukwila chip, an Itanium family processor, tracks approximately 120 architectural event per core to estimate switched capacitance every 8 microseconds and compare this to threshold values to select a maximum voltage-frequency pair to stay within a power envelope [16]. The application of power proxy sensors in Tukwila is guardbanding worst-case power, not attempting to replicate real power on a voltage rail. Since all processors must select the same frequency for identical instruction sequences despite manufacturing variation (leakage and circuit speed), it is not actual power that the sensors are responding to, but an estimation of power in a worst-case chip. The accuracy of these sensors compared to real power measurement is unpublished.

Our work complements the work of AMD and Intel in that we differentiate our power model across processors to calculate chip power as accurately as possible for charge-back purposes. Our power proxy has additional novel properties addressing real-world implementation challenges. First, it deals with significant chip-to-chip variations in an accurate yet concise way. Second, it is accurate for undervolting (UV) where voltage adjustment is independent of frequency. Prior work only evaluates with a fixed workset of voltage-frequency pairs. In addition, prior work on real systems [6] estimates power at 1-second intervals. Our estimates are for 32-millisecond intervals, which is more relevant for dynamic power capping and energy-efficiency controllers.

6. Conclusion

In this paper, we present accurate chip-level and core-level power proxies for the IBM POWER7+ processor. We validate the power proxies by accurately replicating an existing Vdd power rail sensor. For a fixed frequency run, we achieve a mean unsigned error of 1.8% for fine-grained 32 ms samples across all workloads. For an interval of an entire workload, we achieve a mean error of -0.2%. The worst-case workload error was under 9.5%. This accuracy is similar to the prior work with the highest accuracy, but is attained at a 30x smaller timescale which is more appropriate for fine-grain power management applications. We also show that the power proxies hold their accuracy across a range of frequency and voltage settings. Additionally, we demonstrate the first power proxies that work on a system that undervolts processors, whereas prior studies only show results for conventional voltage-and-frequency scaling with fixed voltage-frequency pairs.

Our demonstration in a real system shows the technique is sound for deployment in commercial multi-core servers. The power proxies account for full voltage and frequency ranges and also for chip-to-chip manufacturing variations. Such proxies are useful for a number of applications, such as power-based billing strategy for cloud-based services. They also enable powerful runtime what-if evaluations of different power management techniques.

Acknowledgement

We thank Jason F. Cantin for providing the IBM SNAP genetic algorithm optimizer used in this work.

References

[1] F. Bellosa, "The benefits of event-driven energy accounting in power-sensitive systems," in *Proceedings of the 9th Workshop on ACM SIGOPS European Workshop*, 2000.

[2] G. Contreras and M. Martonosi, "Power prediction for intel XScale processors using performance monitoring unit events," in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, 2005.

[3] T. Do, S. Rawshdeh, and W. Shi, "pTop: A process-level power profiling tool," in *Proceedings of the Workshop on Power Aware Computing and Systems, HotPower*, 2009.

[4] M. Floyd *et al.*, "Introducing the adaptive energy management features of the POWER7 chip," *Micro, IEEE*, vol. 31, no. 2, pp. 60–75, March/April 2011.

[5] M. Floyd *et al.*, "Adaptive energy-management features of the IBM POWER7 chip," *IBM Journal of Research and Development*, vol. 55, no. 3, pp. 8:1–8:18, May/June 2011.

[6] B. Goel *et al.*, "Portable, scalable, per-core power estimation for intelligent resource management," in *Proceedings of International Green Computing Conference (IGCC)*, 2010.

[7] Intel Corporation, *Intel Server Board S1200BT*, February 2012.

[8] H. Jacobson *et al.*, "Abstraction and microarchitecture scaling in early-stage power modeling," in *Proceedings of International Symposium on High Performance Computer Architecture (HPCA)*, 2011.

[9] R. Jotwani *et al.*, "An x86-64 core implemented in 32nm SOI CMOS," in *Proceedings of International Solid-State Circuits Conference(ISSCC)*, 2010.

[10] A. Kansal *et al.*, "Virtual machine power metering and provisioning," in *Proceedings of the ACM Symposium on Cloud Computing (SOCC)*, 2010.

[11] C. Lefurgy, X. Wang, and M. Ware, "Power capping: A prelude to power shifting," *Cluster Computing*, vol. 11, no. 2, pp. 183–195, June 2008.

[12] C. R. Lefurgy *et al.*, "Active management of timing guardband to save energy in POWER7," in *Proceedings of International Symposium on Microarchitecture (MICRO)*, 2011.
[13] J. McCalpin, "The STREAM2 Home Page," http://www.cs.virginia.edu/stream/stream2.
[14] L. W. McVoy and C. Staelin, "lmbench: Portable tools for performance analysis," in *Proceedings of the USENIX Annual Technical Conference (USENIX)*, 1996.
[15] M. Powell *et al.*, "CAMP: A technique to estimate per-structure power at run-time using a few simple parameters," in *Proceedings of International Symposium on High Performance Computer Architecture (HPCA)*, 2009.
[16] B. Stackhouse *et al.*, "A 65 nm 2-billion transistor quad-core Itanium processor," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 18–31, January 2009.
[17] "SPEC CPU2006," http://www.spec.org/cpu2006.
[18] "SPECpower_ssj2008," http://www.spec.org/power_ssj2008.
[19] J. Stoess, C. Lang, and F. Bellosa, "Energy management for hypervisor-based virtual machines," in *Proceedings of the USENIX Annual Technical Conference (USENIX)*, 2007.
[20] M. Ware *et al.*, "Architecting for power management: The IBM POWER7 approach," in *Proceedings of International Symposium on High Performance Computer Architecture (HPCA)*, 2010.