Accurate Fine-Grained IBM POWER7+ Power Proxy

Wei Huang¹, Charles Lefurgy², William Kuk³, Alper Buyuktosunoglu², Michael Floyd², Karthick Rajamani², Malcolm Allen-Ware², Bishop Brock²

¹now with AMD; ²IBM; ³while an intern with IBM
Outline

- Why do we need both fine-grained and accurate power proxies?
- POWER7+ power proxy
  - What is new in this work?
  - Chip-level power proxy
  - Core-level power proxy
- An example application of power proxies
Outline

- Why do we need both fine-grained and accurate power proxies?
- POWER7+ power proxy
  - What is new in this work?
  - Chip-level power proxy
  - Core-level power proxy
- An example application of power proxies
Current Situation

- Practical ways to directly measure power consumption of a core in a microprocessor do not exist.
- What-if scenarios
  - Accurate evaluation without actually switching between power management policies.
- Power proxies, especially core-level power proxies, provide a practical solution.
Finer Granularity Power Estimations with Improved Accuracy

**Finer Granularity – Space and Time**

- Core-level power management
  - Per-VM power capping, within or across chip boundary
  - core-level D(V)FS
  - With awareness of power consumption variations among cores
- Finer time granularity for fast response to workload and environment variations.
- Energy-based virtual machine billing/accounting

**Accuracy**

- Inaccuracy can lead to wrong power management decisions.
- Reclaiming excessive margins/guardbanding in power management
  - 1% improvement in power estimation accuracy → ~1% perf improvement (cf. Lefurgy et al. *Cluster Computing*, 2008.)
Outline

- Why do we need both fine-grained and accurate power proxies?
- POWER7+ power proxy
  - What is new in this work?
  - Chip-level power proxy
  - Core-level power proxy
- An example application of power proxies
Overview of POWER7+ Power Proxy Methodology

One-time Offline Training
(counter weights, voltage exponents, temperature coefficient, etc.)

Activity Counters in a P7+ Chiplet (Hardware)

Activity proxy

Per-Chip Nominal Voltage
Sensors (Voltage, Temp, Freq)

Chip-Level Power Proxy (Firmware)

Core-Level Power Proxy (Firmware)
Novelty of POWER7+ Power Proxy

- First published complete and easy-to-follow methodology for both chip and core power proxies
- Competitive accuracy compared to published work
- Fine time granularity (every 32ms in this work)
- Work across chips with significant process variations
- Account for full voltage and frequency range
- Decouple voltage and frequency, instead of modeling for fixed voltage/frequency pair.
- Highly adaptable to future design changes and new features
  - per-core voltage rails
  - on-chip VRMs
- Account for leakage and temperature dependency
- Account for core-to-core variation by on-chip thermal sensors
- Use simple model formula with physical meaning
- Hardware + Firmware implementation = Speed + Flexibility + Practicality
Components of a Power Proxy

- Idle power
  - Clock grid power
  - Leakage power
- Active power
Determine Idle Power Model: *Leakage Power + Clock Grid Power*

On idle chip, sweep voltage and frequency (253 measurement points)

Measure power on Vdd rail, and measure chip temperature

\[ P_{\text{idle}} = \frac{F}{S_0} \left( \frac{V}{V_{\text{nom}0}} \right)^\beta + P_{\text{leak, nom}} \left( \frac{V}{V_{\text{nom}}} \right)^\gamma \left( 1 + m_0(T - T_0) \right) \]

Genetic Algorithm-based Optimization

Find fitting parameters to minimize 
\( (P_{\text{measure}} - P_{\text{idle}}) \) for all measurement points

Clock grid power

Leakage power
Idle Power Results
Processor Core Activity Proxy

- Same activity counters as POWER7 – cf. Michael Floyd et al. *HotChips*-2011
- Architectured ~50 power-related events per chiplet (39 used for training)
- Both core and L2/L3 caches
- Use groups to minimize hardware complexity and calculation time
- 762 in-house kernel workload runs for weights training
- Trained at nominal frequency

\[ ActivityProxy = \sum \left( W_g \times \sum \left( W_{ig} \times A_{ig} \right) \right) \]
Determine Active Power Model: On-Chip Activity Proxies

Run training kernel workload

\[ (V = V_{\text{nom}}, F = F_{\text{nom}}) \]

Measure power on Vdd rail.
Measure activity counters

\[ P_{\text{active}} = P_{\text{measure}} - P_{\text{idle}}(V, F, T) \]

Genetic Algorithm-based Optimization
Find weights to minimize

\[ P_{\text{active}} - \frac{\text{ActivityProxy}}{R} \]

Active power model

\[ P_{\text{active}} = \frac{\text{ActivityProxy}}{R_0} \left( \frac{V}{V_{\text{nom0}}} \right)^\alpha \]
Total Power Results at Nominal Frequency (Active Power + Idle Power)

- Training set: kernel workloads only.
- Test set: other kernels, SPECpower and SPEC CPU
- Absolute (unsigned) % error
  - Good for fast run-time power management implementation.
  - Average 1.8% with 2.0% std. dev. across all tested workloads.
  - Errors of 32ms samples are close to each other.
- Average (signed) % error for entire workload
  - Good for long-term energy estimation.
  - -0.2% with 2.6% std. dev.
- Compares well with published prior work, but with 30x faster samples
- Only SPEC CPU2006 results are shown here.
Decoupled Voltage and Frequency

- Fixed frequency run of dealII, while under-volting up to 112.5mV without timing violation.
Chip-to-Chip Variations: Maximum Chip Power Workload

- Variations are mostly captured by operating voltage
- Each chip has a characterized set of supply voltages

![Graph showing normalized total Vdd rail power for different chips and frequencies, comparing power proxy and power sensor readings.](image-url)
Per-Core Power Proxies

- Active Power

\[ P_{active\_core\_i} = \frac{A_P_i}{R_0} \left( \frac{V}{V_{nom0}} \right)^\alpha \]

- Clock Grid Power

\[ P_{clock\_core\_i} = \frac{Freq_i}{S_0 \cdot N_{cores}} \left( \frac{V}{V_{nom0}} \right)^\beta \]

- Leakage Power

\[ P_{leak\_core\_i} = \frac{P_{leak\_nom}}{N_{cores}} \left( \frac{V}{V_{nom}} \right)^\gamma \left( 1 + m_0(T_i - T_{i0}) \right) \]

- Scale each core power to match measured chip power (optional)
Results: Per-Core Power Proxy

\[ P_{\text{core\_"measured"}} = \frac{P_{\text{idle}}}{N_{\text{cores}}} + (P_{\text{chip}} - P_{\text{idle}}) \]
Results: Per-Core Power Proxy
Outline

- Why do we need both fine-grained and accurate power proxies?
- POWER7+ power proxy
  - What is new in this work?
  - Chip-level power proxy
  - Core-level power proxy
- An example application of power proxies
Exploring What-if Scenarios with Power Proxies

- Example: Evaluating the following power management policies
  - **Nominal**: Always run at fixed nominal frequency.
  - **DPS** (Dynamic Power Saving): Adjust (V,F) pairs according to processor utilization level.
  - **DPS-UV** (DPS with under-volting): Adjust frequency according to utilization level + use lowest achievable voltage for each desired frequency level

- Traditional approach
  - Run each policy separately (~3x total run time)
  - Control identical runtime environment (initial temperature, ambient temperature, OS state, etc.)
  - Effort to sync start/stop of workload

- With power proxies
  - Possible to evaluate all policies simultaneously
  - Sync’d start/stop
    - specially suitable for workloads with fixed run durations (e.g. SPECpower_ssj)
  - Efforts are needed to also model temperature for different policies in firmware
SPECpower_ssj2008: DPS, DPS-UV, Nominal modes

- **Power sensor (DPS)**
- **Power proxy (DPS)**
- **Power sensor (DPS, UV)**
- **Power proxy (DPS, UV)**
- **Power sensor (Nominal)**
- **Power proxy (Nominal)**

**Normalized power**

**Time (s)**

- 100% load
- 40% load
- 20% load
Summary

- POWER7+ power proxy
  - Chip level and core level
  - Accurate with fast sample rates
  - Account for variabilities, full voltage and frequency range, decoupled voltage and frequency
  - Practical: easy-to-understand formula, low overhead, highly adaptable to future changes

- Opens opportunities to novel usage scenarios
  - Fine-grained power management
  - Per-VM based power or energy accounting
  - Etc.