Evaluating the Effectiveness of Model-Based Power Characterization

John McCullough, Yuvraj Agarwal, Jaideep Chandrashekhar (Intel), Sathya Kuppuswamy, Alex C. Snoeren, Rajesh Gupta

Computer Science and Engineering, UC San Diego

http://synergy.ucsd.edu  http://variability.org
Motivation

• Computing platforms are ubiquitous
  – Sensors, mobile devices, PCs to data centers
  – Significant consumers of energy, slated to grow significantly

• Reducing energy consumption
  – Battery powered devices: goal of all day computing
  – Mains powered devices: reduce energy costs, carbon footprint
Detailed Power Characterization is Key

• Managing energy consumption within platforms
  – Requires visibility into where energy is being consumed

• Granularity of power characterization matters
  – “Total System Power” or “Individual Subsystem Power”
  – Depends on level of power optimizations desired

• Defining question, from the software stack perspective:
  – How can power consumption be characterized effectively
  – What are the limits: accuracy, granularity, complexity?

• Power characterization has been well studied
  – Need to revisit given the characteristics of modern platforms
Modern Systems - Larger Dynamic Range

• Prior generation of computing platforms:
  – Systems with high base power -> small dynamic range
  – Dynamic component not critical to capture

• Modern platforms:
  – Increasing dynamic fraction
  – Critical to capture dynamic component for accuracy
Power Characterization: Measure or Model

- Two options: Directly measured, or indirectly modeled
  - Modeling preferred because of less hardware complexity
- Many different power models have been proposed
  - Linear regression, learning, stochastic, ..
- Question: how good are these models?
  - Component level as well as system level power predictions
Outline

• Describe power measurement infrastructure
  – Fine grained, per component breakdown
• Present different power models
  – Linear regression (prior work), complex models
• Compare models with real measurements
  – Different workloads (SpecCPU, PARSEC, synthetic)
• Results: Power modeling -> high error
  – Reasons range from complexity, hidden states
  – Modeling errors will only get worse with variability
• Highly instrumented Intel “Calpella” Platform
  – Nehalem core i7, core i5, 50 sense resistors
  – High precision NI DAQs, 16bit / 1.25MS/s, 32 ADCs
Prior Work in Power Modeling

• Total System Power Modeling
  – [Economou MOBS’06] - Regression model, MANTIS
    • AMD blade: < 9% error across benchmarks
    • Itanium server: <21% error
  – [Riviore HotPower ‘08] – Compare regression models
    • Core2Duo/XEON, Itanium, Mobile FileServer, AMD Turion
    • Mean error < 10% across SPEC CPU/JBB benchmarks

• Subsystem Models
  – [Bircher ISPASS ‘07] – linear regression models
    • P4 XEON system: Error < 9% across all subsystems

Prior work: single-threaded workloads, systems with high base power, less complex systems.
• **Counters: CPU + OS/Device counters**
  – For CPU: measure only 4 (programmable) + 2 (fixed)
  – Remove uncorrelated counters, add based on coefficients

• **Benchmarks: “training set” and “testing set”**
  – $k \times 2$-fold cross-validation (do this $n = 10$ times)
  – Removes any bias in choosing training and testing set
Power Consumption Models

• “MANTIS” [Prior Work] – Linear Regression
  – Uses domain knowledge for counter selection
• “Linear-lasso” – Linear Regression
  – Counters selection: “MANTIS” + Lasso/GLMNET
• “nl-poly-lasso” – Non Linear Regression (NLR)
  – Counters selection: “MANTIS” + Lasso/GLMNET
• “nl-poly-exp-lasso” – NLR + Poly term + Exp. Term
  – Counters selection: “MANTIS” + Lasso/GLMNET
• “svm_rbf” – Support Vector Machines
  – Unlike Lasso, SVM does not force model to be sparse.
Benchmarks

• “SpecCPU” – 22 Benchmarks, single-threaded
  – More CPU centric

• “PARSEC” – emerging multi-core workloads
  – Include file-dedup, x264 encoding

• Specific workloads – specific subsystems
  – “Bonnie” – I/O heavy benchmark
  – “Linux Build” – Multi threaded parallel build
  – StressTestApp, CPULoad, memcached
"Calpella" Platform – Power Breakdown

- **Subsystem level power breakdown**
  - PSU power not shown, GPU constant
  - Large dynamic range – 23W (Idle) to 57W (stream)!
Modeling Total System Power

- **Increased Complexity -> Single core to Multi-Core**
  - Modeling error increases significantly
  - Mean Modeling Error < 10%, worse error > 15%

*Error bars indicate max-min per-benchmark mean error*
Modeling Subsystem Power – CPU

• Increased Complexity -> Single core to Multi-Core
  – CPU Power modeling error increases significantly
  – Multicore - Mean Error ~20%, worst case > 150%
  – Simplest case: HT and TurboBoost are Disabled

Error bars indicate max-min per-benchmark mean error
**CPU Power: Single -> Multicore**

**Single-core:**

**Multi-core:**

**CMP inherently increases prediction complexity**
Accurate Power Modeling is Challenging

- **Hidden system states**
  - SSDs: wear leveling, TRIM, delayed writes, erase cycles
  - Processors: aggressive clock gating, “Turbo Boost”

- **Increasing system complexity**
  - Too many states: Nehalem CPU has hundreds of counters
  - Interactions hard to capture: resource contention

- **E.g. consider SSDs vs traditional HDDs**

![Power Prediction Error on SSD is 2X higher than HDD!](image)
Adding Hardware Variability to the Mix

- Variability in hardware is increasing
  - Identical parts, not necessarily identical in power, perf.
  - Can be due to: manufacturing, environment, aging, ...
  - “Model one, apply to other instances” may not hold

- Experiment: Measure CPU power variability
  - Identical dual-core Core i5-540M -- 540M-1, 540M-2
  - Same benchmark, different configurations, 5 runs each
Variability Leads to Higher Modeling Error

- 12% Variability across 540M-1 and 540M-2
  - 20% modeling error + 12% variability \(\Rightarrow\) 34% error!
- Part variability slated to increase in the future
Summary

• Power characterization using modeling
  – Becoming infeasible for complex modern platforms
  – Total power: 1%-5% (single core) to 10%-15% error (multi-core)
  – Per-component model predictions even worse:
    • CPU 20% - 150% error
    • Memory 2% - 10% error, HDD 3% - 22% error, and SSD 5% - 35% error

• Challenge: hidden state and system complexity
• Variability in components makes it even worse

Need low cost instrumentation solutions for accurate power characterization.
Total Power: Single -> Multicore

Single-core:

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>linear-lasso</th>
<th>mantis</th>
<th>nl-poly-lasso</th>
<th>nl-poly-exp-lasso</th>
<th>svm_rbf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>streamcluster</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>canneal.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bodytrack</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>freqmire</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x264</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>blackscholes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StressApp.cpu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LinuxBuild</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>namd</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dedup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StressApp.disk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zeusmp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bonnie</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>memcache.1k</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mcf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StressApp.all</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>memcache.5k</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpuload.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpuload.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpuload.70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpuload.1k</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpuload.90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpuload.100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpuload.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpuload.70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Multi-core:

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>linear-lasso</th>
<th>mantis</th>
<th>nl-poly-lasso</th>
<th>nl-poly-exp-lasso</th>
<th>svm_rbf</th>
</tr>
</thead>
<tbody>
<tr>
<td>freqmire</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>povray</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>streamcluster</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LinuxBuild</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>soplex</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>canneal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>memcache</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>blackscholes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>namd</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bodytrack</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x264</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StressApp.cpu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StressApp.disk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StressApp.100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>memcache.5k</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpuload.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpuload.30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpuload.90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpuload.100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpuload.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpuload.70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Increase in error, sensitivity to individual benchmarks