Full-System Power Analysis and Modeling for Server Environments

D. Economou, S. Rivoire, C. Kozyrakis, P. Ranganathan
Stanford University / HP Labs

Workshop on Modeling, Benchmarking, and Simulation (MoBS)
June 18, 2006
Motivation

- **Costs of power and cooling**
  - Electricity now ~50% of data center costs (*ComputerWorld*, 4/06)
  - Data center cooling consumes ~1W per W consumed by system

- **Power density and compaction**

- **Thermal failures**
  - 10C temperature increase → 50% reliability decrease

- **Environmental issues**
  - EnergyStar Enterprise Server and Data Center Efficiency Initiative, 2006
Goals: Prerequisites to Optimizing Power

• **Understand server power**
  – Across different types of systems
  – Component breakdowns
  – Temporal variation
  – Within and between workloads

• **Develop model for server power**
  – Fast, online model deployable in a data center scheduler
  – Zero hardware cost to the end user
  – Input: accessible OS metrics; Output: “good enough” (within 5-10%) estimate of power
Outline

• Motivation

• Experimental setup

• Power characterization

• Power modeling

• Future work

• Conclusions
Test Machines

- **Power-optimized** blade server
  - Low-power processor states
- **Compute-optimized** Itanium server
  - Zero power-saving technology in processors
  - Resources imbalanced in favor of processors

<table>
<thead>
<tr>
<th></th>
<th>Blade Server</th>
<th>Itanium Server</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>1 * AMD Turion, 2.2 GHz</td>
<td>4 * Itanium 2, 1.5 GHz</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>512 MB SDRAM</td>
<td>1 GB DDR</td>
</tr>
<tr>
<td><strong>Storage</strong></td>
<td>1 HDD, 40 GB, 2.5”</td>
<td>1 HDD, 36 GB, 3.5”</td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td>10/100 Ethernet</td>
<td>10/100 Ethernet</td>
</tr>
</tbody>
</table>
Measurement Infrastructure

Diagram:
- Control and Measurement System
- Component Power DAQ
- System Under Test
- AC Power Meter

Connections:
- Control and Measurement System to Component Power DAQ
- System Under Test to AC Power Meter

• System Under Test: Blade or Itanium server

• Runs **benchmark** + low-overhead **performance monitors** (e.g. sar, caliper) at 1 sample/sec
Measurement Infrastructure

Insert measurement between machine and wall to measure overall power

- Blade server: 1 sample/sec
- Itanium server: Currently 20 sample/sec
Measurement Infrastructure

- We cut into and instrumented the individual power planes of the servers, to capture component-level DC power (~20 samples/sec)
- This is NOT required for our model
Measurement Infrastructure

PC: synchronizes measurements, collects data

- Performance metrics from system under test
- Overall power from AC power meter
- Component power from ADC
• Average DC power of components

• Benchmarks: idle, SPECint, SPECfp, SPECjbb, SPECweb, matrix multiply, streams
• **Disk, net, fan, and misc components**
  
  • Non-negligible contributors to power
  
  • Small variation in average power consumption (occasional spikes)
Blade processor is the single largest consumer of power, although memory is close behind.

High variation in processor power consumption shows that blade is optimized for power.
Power Characterization

**Blade**

- **100 W when idle??**
- Not much variation (30%) between idle and max power in Itanium
- So the 4 processors dominate
- High variation in memory, percentage-wise
Power Characterization Conclusions

• Conventional wisdom
  – After CPU, memory is the next bottleneck
  – Lots of variation in CPU power if chip is optimized for power; otherwise runs near 100% at all times

• More surprising
  – The assorted “misc” components – the arcane circuits on different power planes – really matter (~20% of blade power). Optimizing these may be worthwhile
  – Disk contribution is relatively small
  – Enormous idle power on the Itanium system
Power Modeling

• **Goal:** Develop an online model for use in data center schedulers

• **Model requirements**
  – Full-system
  – Non-intrusive; easy for end user
  – Fast enough for online use
  – Reasonably accurate (within 5-10%)
  – Inexpensive
  – Generic (applicable to different types of systems)
Power Modeling: Past Approaches

• **Simulation-based detailed models**
  – Inexpensive, arbitrarily accurate
  – Not full-system
  – Tailored specifically to particular systems & components

• **Direct hardware measurements**
  – Accurate, fast, easy
  – Expensive (especially over many machines)

• **The Mantis Question**
  – Can high-level combined metrics give a good approximation?
Power Modeling

- Run **one-time** calibration scheme (possibly at vendor)
  - *Inputs*: performance metrics, AC power measurements
  - Workloads that stress individual components: CPU, memory, disk, network
- Fit model parameters to calibration data
  - Linear model for simplicity
- Use model to predict power
  - Inputs: performance metrics (as from sar or caliper) at each point in time
  - Output: estimation of AC power at each point in time
Calibration

• Stress each system component in isolation to develop a model

• Used *gamut* program (J. Moore, 2005) to stress CPU, memory, disk, network at varying degrees of utilization
  – Could use any program that can selectively stress components
  – *Gamut* can’t always stress each component to the absolute maximum

    ▪ *Runs as a user program on top of the OS, so incomplete control of the hardware*
    ▪ *Getting CPU power to the absolute max. may require architectural knowledge*
    ▪ *Overheads (program and OS) prevent it from maxing out subsystems*
Model Creation

- **GOAL:** Predict instantaneous power within 10% using a simple, fast model
  - Inputs: OS-level utilization metrics + AC power for calibration suite
  - Output: An equation which relates power to these metrics

- **INPUT:** Utilization metrics
  - $u_{cpu} = \text{CPU utilization (\%)}$
  - $u_{mem} = \text{Off-chip memory access count}$
  - $u_{disk} = \text{Hard disk I/O rate}$
  - $u_{net} = \text{Network I/O rate}$

- **OUTPUT:** For linear model, an equation of form
  \[ p_{pred,i} = A + B \cdot u_{cpu,i} + C \cdot u_{mem,i} + D \cdot u_{disk,i} + E \cdot u_{net,i} \]
Model Inputs

- Input is a matrix $M$, e.g.:

\[
\begin{array}{cccccc}
\text{idle} & u_{\text{cpu}} & u_{\text{mem}} & u_{\text{disk}} & u_{\text{net}} \\
1 & u_{\text{cpu},t=0} & u_{\text{mem},t=0} & u_{\text{disk},t=0} & u_{\text{net},t=0} \\
1 & u_{\text{cpu},t=1} & u_{\text{mem},t=1} & u_{\text{disk},t=1} & u_{\text{net},t=1} \\
1 & u_{\text{cpu},t=2} & u_{\text{mem},t=2} & u_{\text{disk},t=2} & u_{\text{net},t=2} \\
\ldots
\end{array}
\]

- And a vector $p_{\text{meas}}$, e.g.:

\[
\begin{align*}
p_{\text{meas},t=0} \\
p_{\text{meas},t=1} \\
p_{\text{meas},t=2} \\
\ldots
\end{align*}
\]
Model Creation

- **LP solution**: a vector of weights for each utilization metric

\[ \mathbf{p}_{\text{pred}} = \mathbf{M}\mathbf{s} \]

- **Errors**

\[ \varepsilon_i = \frac{p_{\text{pred},i} - p_{\text{meas},i}}{p_{\text{meas},i}} \]

- **Objective**: minimize absolute error of models over all calibration programs

\[ \min \sum_{n=1}^{N} (t^+_n - t^-_n) \]
Models Developed

Power prediction equation:

\[ p_{\text{pred},i} = A + B \cdot u_{\text{cpu},i} + C \cdot u_{\text{mem},i} + D \cdot u_{\text{disk},i} + E \cdot u_{\text{net},i} \]

<table>
<thead>
<tr>
<th></th>
<th>A (const)</th>
<th>B (cpu)</th>
<th>C (mem)</th>
<th>D (disk)</th>
<th>E (net)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blade</td>
<td>14.45</td>
<td>0.236</td>
<td>4.47*10^-8</td>
<td>0.00281</td>
<td>3.1*10^-8</td>
</tr>
<tr>
<td>Itanium</td>
<td>635.62</td>
<td>0.1108</td>
<td>4.05*10^-7</td>
<td>0.00405</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Evaluation

Mean % Error

90\textsuperscript{th} Percentile Absolute Error
Evaluation

Generic model works (within 10%) on 2 very different systems over a varied set of benchmarks
Applications and Future Work

• **Improving models**
  – Component-level modeling and validation
  – Exploring nonlinear models
  – Adding/replacing CPU utilization % with a generic measurement of ILP

• **Data center resource provisioning**
  – Estimate power costs at different granularities (server, enclosure, rack…)
  – Power-aware scheduling and mapping

• **Data center thermal optimizations**
  – Replace expensive external thermal sensors with Mantis estimates
  – Generate data center thermal map

• **Fan control**
  – Dynamically set fan speed in response to estimated power
  – With component-level models, turn on fans aimed at high-power components
Conclusions

• **Goals:**
  – Understand server power consumption
  – Develop power model that can be used online in data centers

• **Understanding server power**
  – Quantitative component/temporal power breakdown
  – Confirming conventional wisdom: CPU is biggest consumer, memory is next
  – Need cooperation of software for low power
  – “Misc” component is worth paying attention to

• **Developing a power model**
  – High-level metrics give a reasonable approximation of power

• **Future work**
  – Improve model (ILP metrics, non-linear models…)
  – Use model in a data center scheduler