Estimating system-wide power consumption through artificial neural networks

L. F. Cupertino, G. Da Costa and J. M. Pierson

IRIT - Toulouse Institute of Computer Science Research
UPS - University of Toulouse (Paul Sabatier)

GreenDays@Rennes – 02/07/2014
Outline

1. Introduction
2. Energy conversion losses
3. Hardware profiling
4. Limitation of CPU proportional models
5. System-wide power models
6. Conclusions and perspectives
Motivation

- Power consumption varies according to servers’ usage
  - Energy efficient scheduling
  - Power estimation of applications

- Current models limitations
  - Application specific or lack accuracy
  - Hardware dependent

- External power meters
  - Easy to measure
  - Energy providers charge power in AC mode
Estimating system-wide power through neural networks

- Definition of the Training Set
- Performance indicators
- Accurate power measurements
**Artificial Neural Networks**

- Feedforward Multilayer Perceptron Network topology

![Diagram of a feedforward multilayer perceptron network](image)

\[ x_j^{(l)} = \theta \left( \sum_{i=0}^{d^{l-1}} w_{ij}^{(l)} x_i^{(l-1)} \right) \]  

where \( \theta(s) = \tanh(s) \)

- Backpropagation *

\[ \Delta w_{ij}^{(l)} = -\eta x_i^{(l-1)} \delta_j^{(l)} \]  

\[ \delta_i^{(l-1)} = \left( 1 - (x_i^{(l-1)})^2 \right) \sum_{j=1}^{d^{(l)}} w_{ij}^{(l)} \delta_j^{(l)} \]
Environment setup

- **RECS compute box**
  - Processor: Intel Core i7-3615QE
  - Memory: 16GB of RAM
  - NIC: Intel 82579LM Gigabit Ethernet
  - Disk: diskless environment

- **OS**: Scientific Linux release 6.4 (kernel v2.6.32)

- **Power meter**: Plogg

- Modules management and power monitoring is done by an external server to not impact the measurements.

Modeling PSU’s conversion losses

- PSU Input/Output power data provided by PSU’s vendors
- Cubic model: $DC \sim w_0 \times AC + w_1 \times AC^3$
Memory power profile
Network usage

- Benchmark: iperf3
- CPU usage < 5%
Dynamic Voltage and Frequency Scaling

![Graph showing the relationship between processor's frequency (GHz) and average power (W) for Loaded, Idle (C0), and Idle (CX) states.](image-url)

- **Loaded** (Orange squares)
- **Idle (C0)** (Blue circles)
- **Idle (CX)** (Green triangles)

**Processor's frequency (GHz)**

**Average Power (W)**

- 30
- 40
- 50
- 60
- 70

Cupertino, Da Costa, Pierson (IRIT)

Estimating system-wide power through neural networks
Power profile of computation intensive applications

Average Power (W), $\sigma < 0.85$

Number of stressed cores

Cupertino, Da Costa, Pierson (IRIT)

Estimating system-wide power through neural networks
Training set’s power profile

- **Training set**: CPU, memory and network stressed
- **3 frequencies**: min (1.2GHz), max (2.3GHz) and boost (3.1GHz)
Calibrated capacitive model

- Capacitive model:

\[ P \sim \%cpu \ast (cap \ast volt^2 \ast freq) \]  \hspace{1cm} (4)

- Simplified version:

\[ P \sim w_0 + w_1 \ast \%cpu \ast freq \]  \hspace{1cm} (5)

- Calibration: Linear Regression
Calibrated capacitive model

Power Model: $w_0 + w_1 \cdot \text{CPU}_u \cdot \text{CPU}_{af}$

MAPE = 5.42%

$R^2 = 0.7144$

Output $\approx 0.71 \cdot \text{Target} + 12.99$

Cupertino, Da Costa, Pierson (IRIT)
Artificial neural network model

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>cycles</td>
<td>instructions</td>
</tr>
<tr>
<td></td>
<td>cache references</td>
<td>cache misses</td>
</tr>
<tr>
<td></td>
<td>branch instructions</td>
<td>branch misses</td>
</tr>
<tr>
<td></td>
<td>bus cycles</td>
<td>idle cycles frontend</td>
</tr>
<tr>
<td></td>
<td>idle cycles frontend</td>
<td>cpu clock</td>
</tr>
<tr>
<td></td>
<td>task clock</td>
<td>page faults</td>
</tr>
<tr>
<td></td>
<td>context switches</td>
<td>cpu migrations</td>
</tr>
<tr>
<td></td>
<td>minor faults</td>
<td>major faults</td>
</tr>
<tr>
<td></td>
<td>alignment faults</td>
<td>emulation faults</td>
</tr>
<tr>
<td></td>
<td>L1d loads</td>
<td>L1d load misses</td>
</tr>
<tr>
<td></td>
<td>L1d stores</td>
<td>L1d store misses</td>
</tr>
<tr>
<td></td>
<td>L1d prefetch misses</td>
<td>L1i load misses</td>
</tr>
<tr>
<td></td>
<td>LLC loads</td>
<td>LLC load misses</td>
</tr>
<tr>
<td></td>
<td>LLC stores</td>
<td>LLC stores misses</td>
</tr>
<tr>
<td></td>
<td>L1d prefetches</td>
<td>LLC prefetch misses</td>
</tr>
<tr>
<td></td>
<td>dTLB loads</td>
<td>dTLB load misses</td>
</tr>
<tr>
<td></td>
<td>dTLB stores</td>
<td>dTLB store misses</td>
</tr>
<tr>
<td></td>
<td>iTLB loads</td>
<td>iTLB load misses</td>
</tr>
<tr>
<td></td>
<td>branch loads</td>
<td>branch load misses</td>
</tr>
<tr>
<td></td>
<td>node loads</td>
<td>node load misses</td>
</tr>
<tr>
<td></td>
<td>node stores</td>
<td>node store misses</td>
</tr>
<tr>
<td></td>
<td>node prefetches</td>
<td>node prefetch misses</td>
</tr>
<tr>
<td>SYS</td>
<td>cpu usage</td>
<td>resident set size</td>
</tr>
<tr>
<td></td>
<td>received bytes</td>
<td>sent bytes</td>
</tr>
<tr>
<td>MSR</td>
<td>cpu frequency</td>
<td>cpu time in C0</td>
</tr>
<tr>
<td></td>
<td>cpu temperature</td>
<td></td>
</tr>
</tbody>
</table>

ANN setup

- **Topology**
  - Feedforward Multilayer Perceptron
  - 2 hidden layers: 20 5
- **Training set**: 70, 15, 15%
- **Training algorithm**: Levenberg-Marquardt
Artificial neural network model
Conclusions and perspectives

Conclusions:

- When using external power meter, energy conversion losses need to be modeled
- CPU proportional models: calibration and accuracy
- Artificial Neural Networks seems promising for generic power estimators

Future work:

- Reduce the number of variables
- Validation with real usecases
- Validation on distributed applications

Thank you.

fontoura@irit.fr