EPerf: An Application Level Energy Debugger

Priyal Suneja, Tom Anderson
University of Washington

Introduction

- Data centers are responsible for about 1-2% of worldwide electricity consumption
- Total Energy = Energy to operate + Energy to compute
- A lot of work has been done to reduce energy spent to operate data centers
- Next step would be to optimize our code to utilize less energy
- How do we know where energy is being spent when we compute?

Existing Energy Measurement Tools

- Power Monitors
  - Sit in between server and wall plug
  - Report energy consumed by entire server (too coarse)
- Intel RAPL
  - Measures energy of a socket and attached peripherals
  - No way to isolate energy consumed per core
  - Have to run processes in isolation to measure energy consumption
  - Reports energy every millisecond
  - Can be too coarse grained for some applications
  - Has implementation errors with updating counters

EPerf - Overview

- Enables energy measurement without modifying code or running programs in a silo
- Estimates energy consumption of application at process level
- Key Idea: Use microarchitectural events such as cache misses, TLB misses, stalls, instructions etc. to predict energy consumption of applications

EPerf - Design

Building a model

- Use Linux perf to measure performance counters
- Use Intel RAPL to measure ground truth for energy
- Use CVXPY to build a linear model that minimizes the root mean square relative error between ground truth and predicted energy

\[
\sum_{j \in B} \left( \sum_{c \in C} \left( \frac{e_j \cdot w_c}{e_j} - e_j \right)^2 \right)
\]

- B is the set of all benchmarks we use to train the model
- C is the set of all the counters we use as input
- e is the ground truth energy measured by RAPL for each benchmark
- Input to solver: L1 cache misses, L2 cache misses, LLC misses, #instructions, #stalls, TLB misses
- Output from solver: set of weights for all the counters we consider (w)

Using the model

- Use perf to measure micro-architectural events
- Use the counts as input to model to get energy estimates

\[
\sum_{c \in C} (c \cdot w_c) = \text{energy}
\]

Evaluation

- Use SPEC 2017 and GapBS benchmarks to train the model
- Use GapBS to evaluate the model using holdout methodology
- Build and evaluate model on two processors
  - p1 is a Skylake, single socket processor
  - p2 is a Haswell, dual socket processor

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Algorithm</th>
<th>Error</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>bc</td>
<td>betweenness centrality</td>
<td>1.2</td>
<td>21.45</td>
</tr>
<tr>
<td>bfs</td>
<td>Breadth First Search</td>
<td>14.18</td>
<td>20.71</td>
</tr>
<tr>
<td>cc</td>
<td>Connect Components</td>
<td>5.64</td>
<td>17.49</td>
</tr>
<tr>
<td>cc_sv</td>
<td>cc using Shiloach-Vishkin Algorithm</td>
<td>6.01</td>
<td>20.38</td>
</tr>
<tr>
<td>pr</td>
<td>Page Rank</td>
<td>1.59</td>
<td>15.05</td>
</tr>
<tr>
<td>pr_spmv</td>
<td>Page Rank using sparse matrix vector multiplication</td>
<td>4.63</td>
<td>3.7</td>
</tr>
<tr>
<td>sssp</td>
<td>Single source shortest path</td>
<td>6.63</td>
<td>19.51</td>
</tr>
<tr>
<td>tc</td>
<td>Triangle counting</td>
<td>8.11</td>
<td>31.56</td>
</tr>
<tr>
<td>Avg Error</td>
<td></td>
<td>6.06</td>
<td>18.73</td>
</tr>
</tbody>
</table>