A Probabilistic Graphical Model-based Approach for Minimizing Energy Under Performance Constraints

Nikita Mishra, Harper Zhang, John Lafferty & Henry Hoffmann
Department of Computer Science, University of Chicago

Abstract
In many deployments, computer systems are underutilized – meaning that applications have performance requirements that demand less than full system capacity. Ideally, we would take advantage of this under-utilization by allocating system resources so that the performance requirements are met and energy is minimized. This optimization problem is complicated by the fact that the performance and power consumption of various system configurations are often application- or even input-depended. Thus, practically, minimizing energy for a performance constraint requires fast, accurate estimations of application-dependent performance and power tradeoffs. This paper investigates machine learning techniques that enable energy savings by learning Pareto-optimal power and performance tradeoffs. Specifically, we propose LEO, a probabilistic graphical model-based learning system that provides accurate online estimates of an application’s power and performance as a function of system configuration.

Why a hard problem?
- Configuration space can be quite large. With brute force it may take a lot of time.
- The behavior of each application is different for different machine configurations.
- The landscape of performance profile for the application vs different configurations can be very complicated and contain multiple local solutions.

Hierarchical Bayesian model to estimate power and performance

Hierarchical Bayesian Model
\[
y_i | \mu_i, \Sigma_i \sim N(\mu_i, \Sigma_i) \\
\mu_i, \Sigma_i \sim N(\mu, \Sigma) \\
\mu, \Sigma \sim \pi, \Psi 
\]

where \( y_i \in \mathbb{R}^n \), \( \mu_i \in \mathbb{R}^n \), \( \mu \in \mathbb{R}^n \), \( \Sigma \in \mathbb{R}^{n \times n} \). It describes that the power (denoted by \( y_i \)) for each of the \( i^{th} \) application, is drawn from multivariate-Gaussian distribution with mean \( \mu_i \) and a diagonal covariance matrix \( \Sigma_i \). Similarly, \( \mu_i \) is from multivariate-Gaussian distribution with mean \( \mu \) and covariance \( \Sigma \). And, \( \Psi \) and \( \pi \) are jointly drawn from normal-inverse-Wishart distribution with parameters \( \nu, \phi, \psi, \rho \). The statistical inference for \( \theta = (\mu, \Sigma, \psi, \rho) \) is carried out using EM algorithm.

LEO (Learning for Energy Optimization)

Experimental Section

Experimental Setup
We have run our experiments on the machine Xeon E5-2690 with 2 8 cores (with hyper-threading), 2 Memory controllers, 1.2-2.9GHz clock speed settings, 75 W hyperpower, giving us 1024 configurations. We have collected the power and performance data for each of the 25 applications from 3 different suites, PARSEC, Mibench, Rodinia and some others. The system is connected to WattsUp meter which gives the total system power measurements at 1 s intervals. And the applications were instrumented to report the heartbeat per seconds. The baseline heuristics that we compare our algorithm against are, Online algorithms - Polynomial multivariate regression over configuration values on the observed dataset, Offline algorithms - Average over the rest of the applications to estimate the power and performance of the given application, Race-to-idle - Allocates all resources to the application and once it is finished the system goes to idle.

Summary of results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Performance accuracy</th>
<th>Power accuracy</th>
<th>Energy savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offline</td>
<td>0.97</td>
<td>0.98</td>
<td>±6%</td>
</tr>
<tr>
<td>Online</td>
<td>0.87</td>
<td>0.85</td>
<td>±24%</td>
</tr>
<tr>
<td>LEO</td>
<td>0.97</td>
<td>0.98</td>
<td>±6%</td>
</tr>
</tbody>
</table>

Reacting to Dynamic Changes
This section shows that LEO can quickly react to changes in application workload. In this section we run fluidanimate, which renders frames, with an input that has two distinct phases. Both phases must be completed in the same time, but the second phase requires significantly less work. In particular, the second phase requires 2/3 of the resources of the first phase. Our goal is to demonstrate that LEO can quickly react to phase changes and maintain near optimal energy consumption.