Energy Efficiency in HPC with Machine Learning and Control Theory

Connor Imes, Steven Hofmeyr*, and Henry Hoffmann
University of Chicago, Lawrence Berkeley National Laboratory*
ckimes@cs.uchicago.edu, shofmeyr@lbl.gov, hankhoffmann@cs.uchicago.edu

Abstract

Performance and power management in High Performance Computing (HPC) has historically favored a race-to-idle approach in order to complete applications as quickly as possible, but this is not energy-efficient on modern systems. As we move toward exascale and hardware over-provisioning, power management is becoming more critical than ever for HPC system administrators, opening the door for more balanced approaches to performance and power management. We propose two projects to address balancing application performance and system power consumption in HPC *during application runtime*, using closed loop feedback designs based on the Self-aware Computing Model to *observe*, *decide*, and *act*.

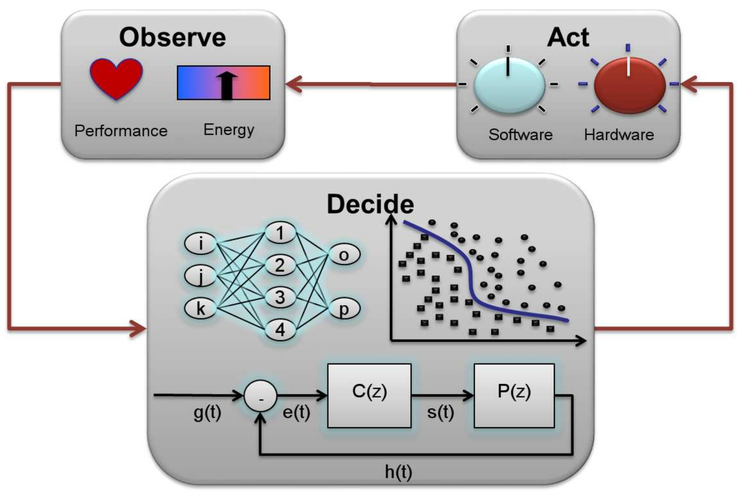

Figure 1: The Self-aware Computing Model.

Machine Learning Classification for Energy Efficiency

Tuning **core allocation (taskset)** and **dynamic voltage and frequency scaling (DVFS)** can save energy by power gating unneeded cores and scaling back active processors, *e.g.,* during I/O or for code not on the critical path. Both are available in HPC job schedulers, though unfortunately today they are typically only configurable when a job is launched.

We propose using **Machine Learning classifiers** to predict the most energy-efficient combination of taskset and DVFS settings *during runtime* to maximize the work-to-energy ratio of an application running on a single node. Low-level *hardware counter metrics*, available through tools like `PAPI` and `PCM`, train and drive our classifiers. At regular time intervals, a classifier predicts the most energy-efficient setting to use based on measured application behavior, then applies the setting to the system. As applications move through *phases*, the predictions change accordingly.

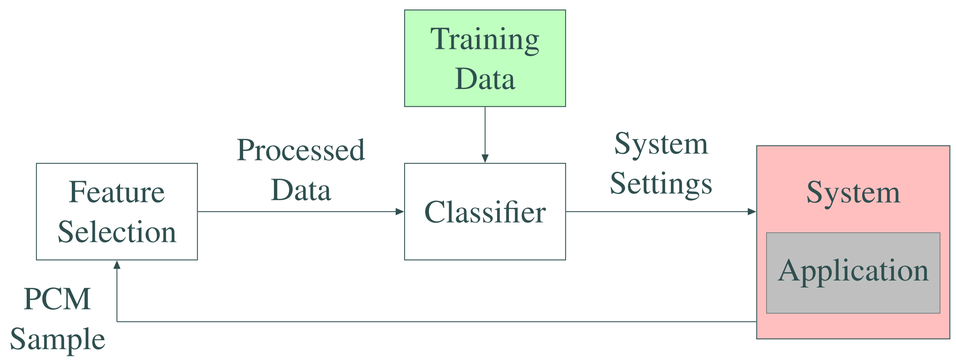

Figure 2: Machine Learning Classifier closed loop feedback control design.

We evaluate our approach on a quad-socket (160-logical-core) system with Intel Xeon E7-8870 processors and 512 GB of DRAM using HPC bioinformatic applications, which are often run on such large systems. Training data is collected from characterizations of smaller HPC applications like the NAS Parallel Benchmarks, CoMD, HPGMG-FV, LULESH, and STREAM. Principal Component Analysis determines which features are most relevant to energy efficiency—DRAM power, instructions per nominal CPU cycle (EXEC), L2/L3 cache hit/miss rates, and relative frequency excluding sleep time (AFREQ) are among the most useful. We evaluate 6 common classifiers – Gradient Boosting (GB), K-Nearest Neighbors (KNN), Random Forest (RF), Stochastic Gradient Descent (SGD), Support Vector Machine (SVM), and a Linear SVM.

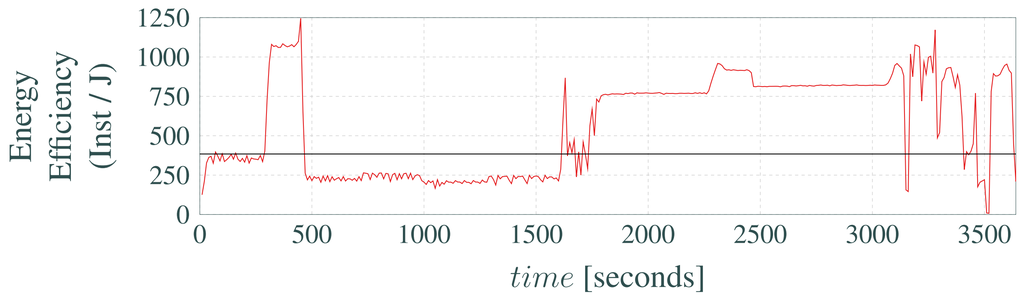

Figure 3: HiPMeraculous genome assembly application (Gradient Boosting classifier).

Figure 3 demonstrates running a Gradient Boosting classifier which uses only 60% power and saves 20% energy over the naive race-to-idle approach, with a 33% increase in runtime. SGD and SVM performed similarly for this application, while KNN, RF, and SVM Linear achieve 55% power, 10% energy savings, and 60% increase in runtime.

Now consider hardware over-provisioned environments subject to power constraints, *e.g.,* DOE's 20 MW goal for exascale. Extrapolating from these results, we could perform nearly *twice as much science* in parallel by better utilizing over-provisioned resources. By trading performance for power savings, maximizing energy efficiency *increases the total throughput of a hardware over-provisioned system by up to 50%.*

MIMO Controller for Power Balancing

In recent years, hardware has been taking more direct control of processor voltage and frequency, making fine-tuned software-management of DVFS obsolete. Fine-grained **power capping** is now the preferred mechanism for managing performance and power tradeoffs, evidenced by the introduction of power capping implementations like **Intel Running Average Power Limit (RAPL)** and the move from Speed Step to Speed Shift.

Nodes in HPC clusters suffer *non-uniformity* in performance and power consumption due to both manufacturing process variation and imbalance in application workloads. We propose a generalized **Multiple Input, Multiple Output (MIMO) Proportional Integral Derivative (PID) controller** to shift power allocations between nodes to eliminate tail idle times in job iterations, thus maximizing application throughput while respecting a global power cap. Unlike heuristic approaches to power management, **control theory** provides *formal guarantees* of *convergence* as well as *robustness* to application, system, and measurement noise. These guarantees hold even across application *phases*.

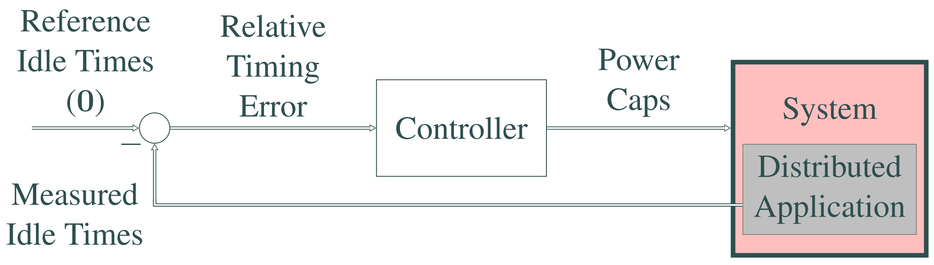

Figure 4: MIMO PID controller closed loop feedback control design.

Given a cluster with n nodes and global power cap Γ, the PID controller computes a new power signal vector $\mathbf{u}$ of size n in each iteration t:

$$\mathbf{u}(t) = \mathbf{u}(t-1) + K_I \cdot \mathbf{e}(t) \tag{1}$$

K_I is the ratio of control change and $\mathbf{e}(t)$ are the node idle times, divided by their mean and normalized around 0. This formulation ensures that the entire global power cap is re-allocated in each iteration. Formally:

$$\sum_{i=1}^{n} e_i(t) = 0 \implies \sum_{i=1}^{n} u_i(t) = \sum_{i=1}^{n} u_i(t-1) = \Gamma \tag{2}$$

Figure 5 simulates meeting a global power cap of 180 Watts on a theoretically imbalanced 4-node cluster ($\Gamma = 180$, $n = 4$) using a MPI+OpenMP application. It begins with evenly distributed power caps, *i.e.,* $u_i(0) = \frac{\Gamma}{n}$ for each node i. We then run 10 time steps:

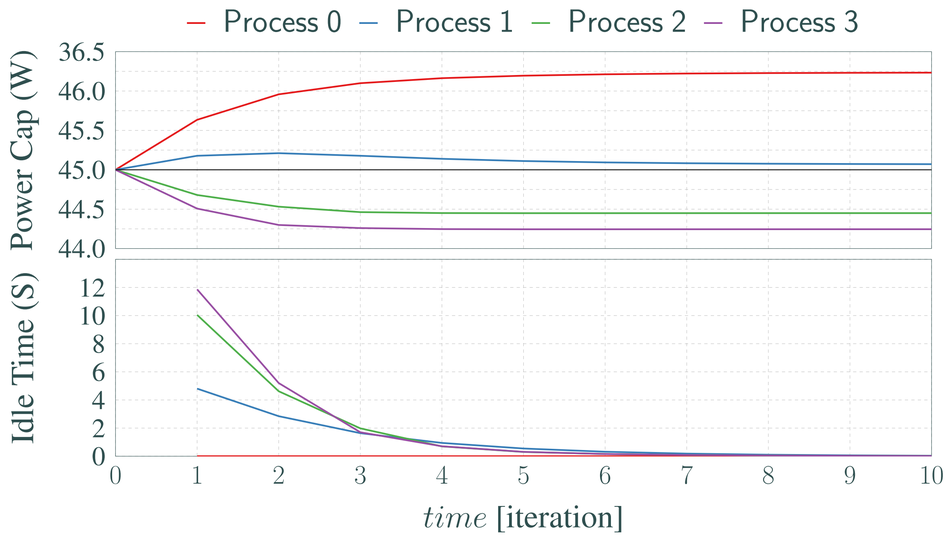

Figure 5: Simulation of changing node power caps between iterations.

The controller quickly re-balances power caps so that nodes finish their jobs with near-zero tail idle times, thus improving the total application performance.

Acknowledgements

The effort on this project is funded by the U.S. Government under the DARPA BRASS program, by the Dept. of Energy under DOE DE-AC02-06CH11357, by the NSF under CCF 1439156, and by a DOE Early Career Award.
Part of this research was conducted while Connor Imes was a Computing Sciences Summer Student at Lawrence Berkeley National Laboratory.

https://people.cs.uchicago.edu/~ckimes/