PowerMan: Online Power Capping by Computationally Informed Machine Learning

Johannes Gebert[1]
Daniel Barry[2]
Jack Dongarra[2]

[1] High-Performance Computing Center Stuttgart, University of Stuttgart

[2] Innovative Computing Laboratory, University of Tennessee Knoxville

INTRODUCTION

Appropriately adjusting the power draw of computational hardware plays a crucial role in its efficient use. While vendors have already implemented hardware-controlled power management, additional energy savings are available, depending on the state of the machine. We propose the online classification of such states based on computationally informed machine learning algorithms to adjust the power cap of the next time step. This research highlights that the overall energy consumption can be reduced significantly, often without a prohibitive penalty in the runtime of the applications.

Haidar et al. proved that the power consumption can be reduced by setting the power cap with minimal or no loss of performance when the system runs memory-bound codes. However, automated approaches are required for online power capping in production.

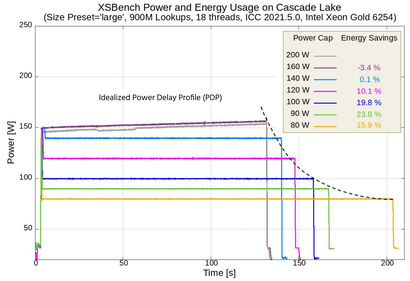

Fig. 1: Power capped energy savings on Intel Cascade Lake.

MEASURING TRAINING DATA FOR THE MACHINE LEARNING CLASSIFIER

Kernels for training	BLAS Level 1 DAXPY	BLAS Level 2 DGEMV	BLAS Level 3 DGEMM
Range and intervals of the package power for training	$80 \le P_p \le 200$[W]	$\Delta P_p \in \{5, 10, 15, 20, 30, 40\}$ [W]	
Hardware counters traced to predict the computer's state	$P_{i,p}$ [W] $P_{i,p,lim}$ [W] $P_{i,m}$ [W] Additional counters for utility functions "Power Limit": B_iW [GiB/s] & DP_i [flop/s]		

Fig. 2: Kernels, wattage ranges, and hardware counters used for the self-supervised training of the sktime *TimeSeriesForestClassifier* with *n_estimators=25.*

UTILITY FUNCTIONS

We assign a label l_i for self-supervised learning to the training instances 'i' to identify whether a system is compute- or memory-bound. Subsequent adjustments of the package power limit of the next instance $P_{i+1,p,lim}$ of eq. 2 depend on the compute power scaled by the ratio of the memory power of the current instance $P_{i,p,a}$ of eq. 1, the known variables and the labels l_i.

$$P_{i,p,a} = \frac{\left(\overline{P_{i,p}} - \overline{P_{i,m}}\right) \cdot P_{i,m,max}}{\overline{P_{i,m}}} + \overline{P_{i,m}} \qquad (1)$$

$$P_{i+1,p,lim} = \begin{cases} \overline{P_{i,p}} & \text{if } l_i = 1 \\ P_{i,p,a} & \text{if } l_i \in \{2,4\} \\ P_{i,p,a} - 5\,\mathrm{W} & \text{if } l_i = 3 \end{cases} \qquad (2)$$

RESULTS

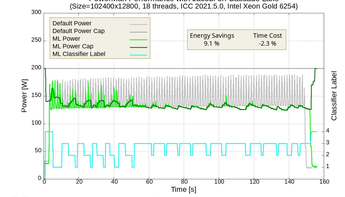

Fig. 5: The energy consumption of software with haircomb- or other power pattern can be reduced significantly by online power capping.

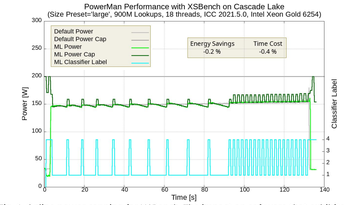

Fig. 6: Online power capping for XSBench. The impact on software that exhibits a steady state power draw is negligible.

PERFORMANCE OVERHEAD

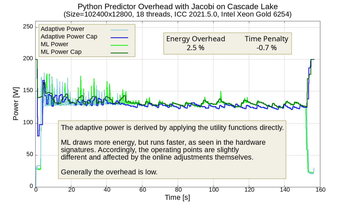

Fig. 7: The Python-based ML prediction induces a small overhead that might be reduced by a hardware implemented classifier firmware.

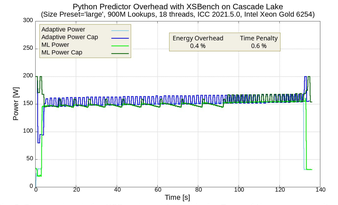

Fig. 8: The overheads for XSBench and Jacobi in fig. 7 are of the same magnitude.

IMPLEMENTATION

- Online power capping
- x86 64 Bit, accelerators pending
- Readily available hardware counters, traced with 5 Hz
- Socket-wide power capping with on-package classification

We observed a latency of power limit enforcement in the regime of $t_{p,lim} = 0.15$ s depending on the power step ΔP. Therefore, the short term power limit "B" is not adjusted. A reasonable compromise of fidelity and prediction overhead was found in the 1.5 s interval of $0.5 \le t_{i,cl} \le 2.0$ [s].

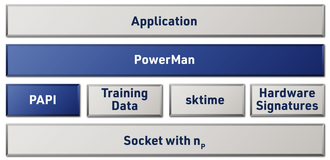

Fig. 3: Components of the online power capping with machine learning.

COMPUTATIONALLY INFORMED, SOFTWARE AGNOSTIC

A processor's state is recognizable by tracing hardware events, hence computationally informed algorithms. State change estimates depend on the operation performed and its problem size, so software-aware approaches will lead to even higher energy savings.

HARDWARE SIGNATURES

Power signatures like fig. 4 show a comparable architectural behavior when in a memory-bound state for a wide range of applications.

The compute power of an instance $P_{i,c}$ is given as $P_{i,p}$ - $P_{i,m}$ and expected to depend on the operational intensity. This connects the compute power with the power of the memory $P_{i,m}$. If the system is memory-bound and the operational intensity is constant, a steady state hardware energy signature like shown in fig. 4 is to be assumed. Other factors the signatures might depend on:

- Type of operations (Single precision, Double precision, Integer, ...)
- Communication
- Temperature of the package
- Core count of the package

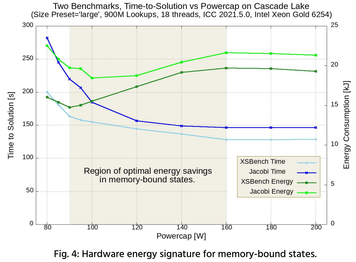

Fig. 4: Hardware energy signature for memory-bound states.

OUTLOOK

We suggest the following use-cases and scenarios for online power capping:

- Different architectures and core counts to be implemented.
- Adjusting the target power of a running program for computers to meet a changing compute-center wide power target.
- Minimizing energy consumption for temporally uncritical applications.
- Wall-time aware energy savings.
- Energy saving on not fully utilized clusters.
- Energy savings by power capping memory.

Further considerations:

- High sampling rates improve the fidelity of power capping.
- Identifying more complex patterns within time series of counters.
- Chiplet- or cluster-wide approaches based on machine learning may enhance hardware signature and load balancing.

LIMITS

- Complex classifiers significantly increase the overhead of prediction.
- Non-x86 architectures to be implemented.
- The approach (currently) is limited to memory-bound states.
- Hardware counters may require root permissions.

CLASSIFIER FIRMWARE

Power capping is very specific to a given configuration of a hardware architecture, but it shows a consistent behavior in memory-bound situations.

Our approach is a software agnostic one and we recommend an on-package classifier firmware like shown in fig. 9 which is based on fig. 2 to minimize the overhead of classifying the time series of the performance counters.

Fig. 9: Integrating the classifier as an on-package feature.

CONCLUSION

Online Power Capping by Computationally Informed Machine Learning can reduce power and energy consumption with Minimal to no loss in runtime.

ACKNOWLEDGEMENTS

This research was supported by the National Science Foundation Office of Advanced Cyberinfrastructure (OAC) CSE Dir. for Comp. & Info Sci. & Eng. under Grant No. 2004541