A Dynamic Power Capping Library for HPC Applications

Sahil Sharma
Illinois Institute of Technology
Chicago, Illinois
ssharma18@hawk.iit.edu

Zhiling Lan
Illinois Institute of Technology
Chicago, Illinois
lan@iit.edu

Xingfu Wu
Argonne National Laboratory
Lemont, Illinois
xingfu.wu@anl.gov

Valerie Taylor
Argonne National Laboratory
Lemont, Illinois
vtaylor@anl.gov

I. Introduction

As HPC systems increase in scale and capability, the cost of supplying power to these systems grows significantly. This introduces the urgent need for energy efficient computing through the management of power consumption. The PowerStack initiative defines a holistic power management framework at three levels, i.e., at the cluster level, the job level, and the node level [1]–[3]. It is expected that a cluster will be given a system-wide power budget (aka an allocated power budget), which can be intelligently managed and distributed at the job and node levels. Our work provides a method for intelligently managing node-level power during execution.

While several solutions provide useful power monitoring or management capabilities, little work has been done to investigate **when** and **how much** power capping should be applied **dynamically** during application execution. For example, UPScavenger is a runtime system targeting uncore power saving and is not designed for package level power saving [4], which is the focus of our work. PoLiMEr provides user-level functions to set and monitor power caps, but these must be set in the application source code manually [5]. In [6], the impact of online performance under power management was investigated. Yet, it does not provide any runtime solution for dynamic power capping. Despite the considerable research on power management, *there is a lack of software tools at the user level to automatically perform application power capping during application execution without offline profiling*.

In this work, we address the need for such software tools by developing *DNPC* (Dynamic Node-level Power Capping library) for energy efficient computing. *Given an application and a use-defined performance degradation limit, DNPC dynamically follows the application's power profile and adjusts the power cap, with the objective to minimize package level power consumption within the performance degradation threshold.* It includes several key techniques for detecting package power phase changes, estimating performance degradation, and correspondingly adjusting the package-level power cap during an application's execution. Both performance degradation and power phase detection are enabled by actively monitoring various performance counters and power readings. More importantly, DNPC automatically adjusts the power cap throughout application execution without any offline profiling.

```
#include "dnpc.h"
int main(int argc, char *argv[])
{
    /* application code */
    MPI_Init(&argc, &argv);
    dnpc_init();
    /* application code */
    dnpc_finalize();
    MPI_Finalize();
    /* application code */
}
```

Listing 1: An example of adding DNPC to a C application.

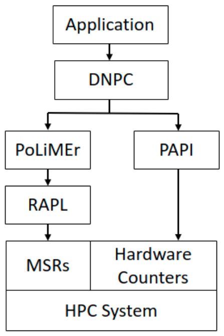
DNPC is an open-source, user-level library for dynamically power capping HPC applications. It is lightweight and has extremely low overhead.

II. DNPC Design

Figure 1 illustrates the high-level overview of DNPC and its state machine. DNPC leverages PAPI and PoLiMEr to execute its dynamic power capping mechanism [5], [7]. Its design is based on our experience with PoLiMEr. Without a power cap, the frequency reading reported by PoLiMEr is equivalent to the maximum frequency on the system (denoted as f_{max}). When the power cap is set below the actual application power curve, the PoLiMEr frequency reading becomes less than f_{max}. According to these observations, DNPC estimates performance degradation caused by a power cap, or $\Delta perf(t)$ as:

$$\Delta perf(t) = \frac{\int_0^t (f_{max} - f(t))\ dt}{\int_0^t f_{max}\ dt} \tag{1}$$

DNPC is implemented in C and supports C, C++ and Fortran applications. To use the library, the user only needs to add a few of DNPC's functions within their application code as shown in Listing 1. At application submission the user provides a performance degradation limit to DNPC through an environment variable. During execution, DNPC automatically monitors the application's power trend, estimates its potential performance degradation, and tunes the power cap correspondingly. All the actions are performed without offline application profiling.



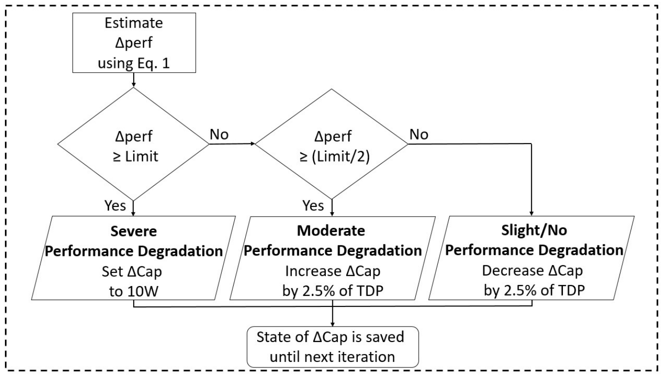

Estimate Δperf using Eq. 1
Δperf ≥ Limit
No
Yes
Δperf ≥ (Limit/2)
No
Yes
Severe Performance Degradation Set ΔCap to 10W
Moderate Performance Degradation Increase ΔCap by 2.5% of TDP
Slight/No Performance Degradation Decrease ΔCap by 2.5% of TDP
State of ΔCap is saved until next iteration

Fig. 1: DNPC overview (left) and the DNPC state machine for estimating performance degradation (right).

III. Case Study

We evaluated DNPC with five NPB and ECP benchmarks on the Cray XC40 machine Theta [8] at Argonne National Laboratory. Each Theta node is deployed with a 64-core Intel Xeon Phi (Knight's Landing) 7230 processor, 192 GiB of DDR4 DRAM, and 16 GiB of high bandwidth MCDRAM. The MCDRAM was configured in the cache-quad setting. Sixty-four OpenMP threads were used for each test, and they were mapped to the cores using a scatter affinity.

For each application, we analyzed its execution time, energy consumption, and rescued energy savings by using DNPC. In a holistic power stack, a cluster-level scheduler would have to allocate a certain power budget to a node. When the cluster-level manager has limited knowledge of application power behavior, it typically reserves the Thermal Design Power (TDP) for the node. If the application sets its own dynamic power cap on the node, the power cap can be sent to the cluster-level power manager, which can then reclaim the difference between the TDP and the dynamic power cap over a interval of time, and allocate it somewhere else. We denote this energy saving as *rescued energy*.

We did several experiments with varying performance degradation limits. Table I shows one experiment where the performance degradation limit is set to 10%. It indicates that DNPC can manage application performance degradation within the given limit. These results are obtained without any offline application profiling. The results also demonstrate that DNPC can save up to 50% in terms of rescued energy. Energy consumption is application dependent.

TABLE I: Results given a 10% performance degradation limit

Application	Performance Degradation (%)	Energy Savings (%)	Rescued Energy Savings (%)
MiniQMC	7	3	25
SW4Lite	7	2	19
XSBench	2	0	42
FT	2	3	0
MG	4	2	2

In summary, our case studies illustrate that given a performance degradation limit, DNPC can dynamically adjust the package power cap during application execution to minimize power usage while maintaining performance degradation within the user-defined limit. Such an automatic node-level power management is essential for the hierarchical power management discussed in the HPC PowerStack initiative.

Acknowledgment

This work is supported in part by US National Science Foundation grants CCF-1618776 and CCF-1801856. We acknowledge Argonne Leadership Computing Facility for use of the Cray XC40 Theta machine.

References

[1] C. Cantalupo, J. Eastep, S. Jana, M. Kondo, M. Maiter, A. Marathe, T. Patki, B. Rountree, R. Sakamoto, M. Schulz, and C. Trinitis, "A strawman for an hpc powerstack," Tech. Rep., 2018.

[2] X. Wu, A. Marathe, S. Jana, O. Vysocky, J. John, A. Bartolini, L. Riha, M. Gerndt, V. Taylor, and S. Bhalachandra, "Toward an end-to-end auto-tuning framework in HPC PowerStack," in *Proceedings of Energy Efficient HPC State of Practice 2020 (EE HPC SOP 20).* Washington, DC, USA: IEEE Computer Society, 2020.

[3] "The hpc powerstack." [Online]. Available: https://hpcpowerstack.github.io/index.html

[4] N. Gholkar, F. Mueller, and B. Rountree, "Uncore power scavenger: A runtime for uncore power conservation on hpc systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3295500.3356150

[5] I. Marincic, V. Vishwanath, and H. Hoffmann, "Polimer: An energy monitoring and power limiting interface for hpc applications," in *Proceedings of the 5th International Workshop on Energy Efficient Supercomputing*, ser. E2SC'17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3149412.3149419

[6] S. Ramesh, S. Perarnau, S. Bhalachandra, A. D. Malony, and P. Beckman, "Understanding the impact of dynamic power capping on application progress," in *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2019, pp. 793–804.

[7] D. Terpstra, H. Jagode, H. You, and J. Dongarra, "Collecting performance data with papi-c," in *Tools for High Performance Computing 2009*, M. S. Müller, M. M. Resch, A. Schulz, and W. E. Nagel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 157–173.

[8] "Theta/thetagpu," 2021. [Online]. Available: https://www.alcf.anl.gov/alcf-resources/theta