Energy Consumption vs Clock Speed for Various Application Profiles

Spencer Diamond, Arizona State University, SIParCS
Dave Hart, Rory Kelly, Ben Matthews, NCAR

Abstract

High Performance Computing has proven to be a powerful tool for the field of climate research, but its use comes with large costs, not only financial but also environmental costs. One way to reduce these costs is by lowering the energy consumption of HPC systems. However, most HPC optimization is focused on computation time efficiency, with energy efficiency largely being of secondary importance. The goal of this project was to develop a methodology for collecting reliable power data during execution in order to understand how to improve the energy efficiency of HPC systems without significantly impacting performance. In order to streamline the development of this methodology, this project's scope was limited to 64GB nodes on NCAR's Cheyenne computer. Experiments were focused on measuring the effect that CPU clock speed has on energy consumption for single nodes. The experiments were performed on nodes running three common performance benchmarks as well as on idle (sleeping) nodes. Data was collected from the node itself during execution, and from the power supplies feeding the node. Although further exploration is necessary, the results of this project suggest that managing CPU frequency both during execution and while idle could be a viable way to boost energy efficiency and reduce the cost of HPC systems without sacrificing performance.

Methodology

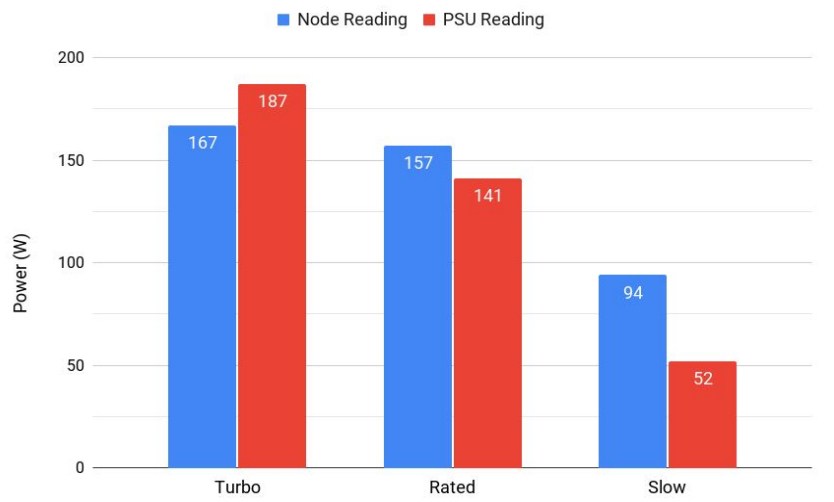
Tests were run using CESM2 benchmark kernels, all modified to extend runtime: WACCM_imp_sol_vector, CESM2_MG2, and CESM2_CLUBB; as well as running sleep. These kernels were chosen because of their ease of use, as well as the fact that they are used as procurement benchmarks at NCAR as they are very representative of the kind of applications being run on NCAR systems.

For collecting data, there are two easily accessible data streams, getting data from the node itself, and getting data from the power supplies that feed the node. Neither data stream is calibratable, so to be confident in the results, data was collected from both and their correlation calculated. The node side data collection becomes far more complicated for applications running on multiple nodes, so we are limited to only single nodes. On Cheyenne, nine power supplies feed a group of 36 nodes, so in order to isolate our single node job, the rest of the nodes in the group need to be empty, and the extra power drawn by the other 35 nodes subtracted out.

In the batch script, the execution node name, timestamps of the start and end of execution, and the node power data is collected. Then, in a separate python script, the node data is read from job output, PSU data is read from database, then both data streams are normalized and averaged, then multiplied by execution time to get total energy. Then, total energy for both data streams and execution time are written to CSV.

Energy Consumption and Execution Time by Job Type

Average Power Readings on Sleeping Nodes



Results

In the upper chart, the value represented by each bar is an average of five separate runs for each job type, with blue being the node side data, red being the power supply data, both indicated in joules on the left, and the green is the execution time indicated in seconds on the right.

The first notable finding displayed by this chart is the agreement between the node readings and the power supply readings. There are some discrepancies, particularly for the slow runs, but these discrepancies are likely do to an error in how the power supply data was processed. Even with these discrepancies, the two data streams have a correlation coefficient of R=0.8615, which is an extremely strong correlation ($P < 0.00001$), so we can be confident in the validity of these results.

The second finding of note is that the energy use is comparable between the rated and slow runs, while the execution time is comparable between rated and turbo runs. Using this data, we see that on average, the rated runs used 45.25% less energy than the turbo runs ($P<0.00001$), with only a 7% ($P<0.02$) increase in execution time. Because of the apparent negative impact on performance, it is not possible to conclude from these results how downclocking to rated would impact costs.

In the lower chart we have the data from sleeping nodes, with each bar representing the average of two runs at each clock speed, the blue being node data and red being PSU data, both being indicated in Watts. From this data, we see that running idle nodes downclocked to slow decreases the average power by 43.5% compared to running them at turbo ($P<0.00001$).

Using these cursory results, we can roughly estimate that by downclocking idle nodes to slow, NCAR-Wyoming's utility bill could be decreased by tens of thousands of dollars annually.

Continuing Research

There are still a lot of unanswered questions brought up in the course of this project, and further research is needed to fully answer these questions. Specifically, testing on other hardware like large memory nodes, GPU nodes, other HCP systems, etc; running actual experimental applications and jobs running on more than one node; and how I/O changes energy usage are all worth investigating. There's also a few questions involving idle nodes, specifically the viability of fulling shutting down portions of the system that will be idle for long stretches of time.

Acknowledgements

Mentors: Dave Hart, Rory Kelly, Ben Matthews
SIParCS & NESSI: AJ Lauer, Virginia Do, Jerry Cyccone, Max Cordes Galbraith, all the other SIParCS and NESSI interns
Thank you to NCAR for hosting this internship and to NSF for providing funding for this research
Special thanks to: Mick Condy, Sidd Ghosh, Richard Valent, Storm Knight, Joseph Mendoza, John Dennis