Run-Time DFS/DCT Optimization for Power-Constrained HPC Systems*

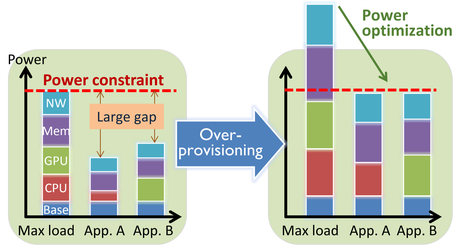
Ikuo Miyoshi[1], Shinobu Miwa[2], Koji Inoue[3] and Masaaki Kondo[4]

[1]Fujitsu Limited, [2]The University of Electro-Communications, [3]Kyushu University, [4]The University of Tokyo

*This work was partially supported by JST under CREST.

Background

- For development of exascale HPC systems, power consumption is one of the major design constraints. HW-overprovisioning is an effective approach to build systems under such constraints.
- Since job throughput of HW-overprovisioned systems is restricted by power limit, "performance-per-watt" has become a more important indicator of efficient system use.



Conventional design — Power-constrained adaptive system

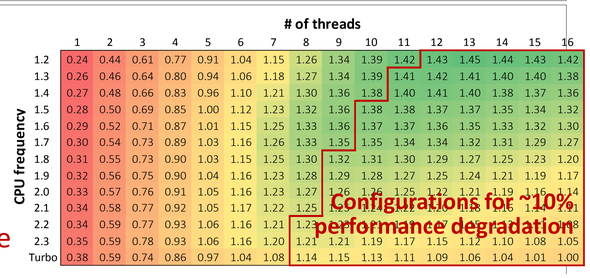
Motivation

- Because application developers are not expected to pay much attention to "performance-per-watt", power optimizations should be done by system providers and system administrators.
- Although optimization techniques for power and energy, such as DFS and DCT, have been studied well, practical use is limited to a DB-based approach, which records and reuses the relationship between a job's execution time and its locked CPU frequency.
- Conducting DFS/DCT optimization during the execution of a job is simple to use, adaptive to the environment of executing systems, and possibly robust in terms of CPU's manufacturing variability.

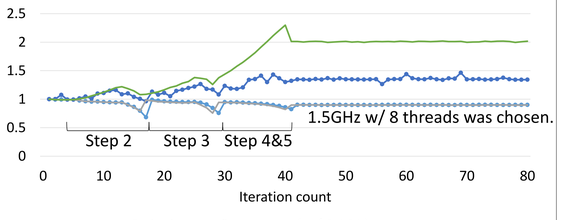
Optimization Algorithm

- Assumptions
  - ✓ Target application does iterative computation, such as time integration and iterative solver, and can be divided into several regions based on computational characteristics.
  - ✓ From the application users' viewpoint, performance fluctuation by ~10% is ignored or acceptable.
- The optimization target is a DFS/DCT configuration of max "performance-per-watt" for each region in the range of acceptable performance degradation.
- Optimization steps
  1. For use as reference performance, observe the performance during initial iteration(s)
  2. Under the "Turbo" frequency, search the # of threads in decreasing order for the best efficiency
  3. Under the base frequency, search the # of threads in the same way as Step 2
  4. With the # of threads chosen by Step 2&3 as the best efficiency in the range of acceptable performance degradation, search CPU frequency in decreasing order for the best efficiency
  5. When performance degradation exceeds the acceptable range, continue the search after adding one more thread
- Calculation of efficiency
  - When measuring by RAPL, efficiency = "measured energy for the reference performance" / "measured energy for a configuration".
  - When estimating without RAPL, "measured energy" is replaced with "estimated energy"; "estimated energy" = "CPU cycles" * (1 + ("# of threads" - 1) * "correction factor").

  The "correction factor", which is 0.06 for Haswell and 0.13 for Sandy Bridge, was derived from the results of the STREAM benchmark by regression analysis.



Relative "performance-per-watt" of HPCG on Haswell

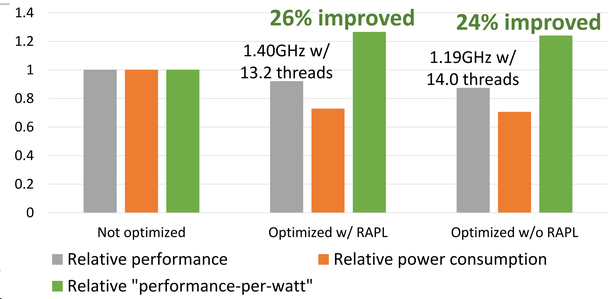


Optimization behavior for ComputeSPMV region of HPCG on Haswell

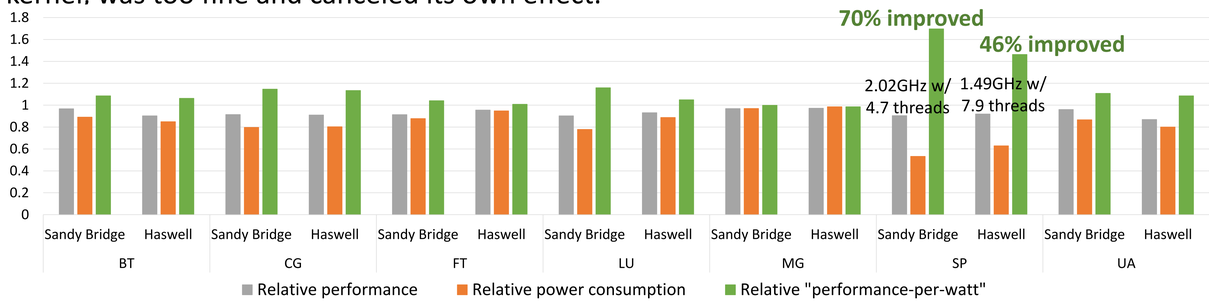
Measurement environments: Xeon E5-2680 2.7GHz 8 cores (Sandy Bridge), Xeon E5-2698v3 2.3GHz 16 cores (Haswell)

Evaluation

- For the HPCG benchmark on Haswell, the optimization chose 1.19GHz with 14.0 threads on average without the RAPL measurement, which improved the "performance-per-watt" by 24%.
- Among 7 iterative kernels in the NAS Parallel Benchmarks, 6 kernels showed improvement in "performance-per-watt" up to 70%. However, the optimization for MG, which is a memory-intensive kernel, was too fine and canceled its own effect.

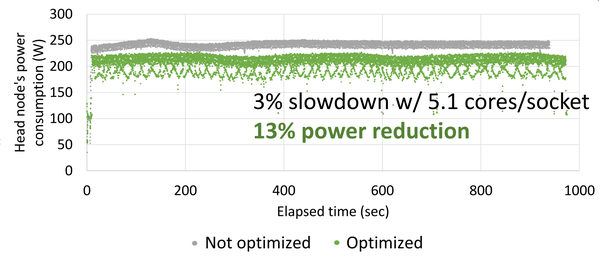


Optimization results for HPCG on Haswell



Optimization results for NAS Parallel Benchmarks class C by OpenMP on Sandy Bridge and Haswell

- Results on the 10 nodes of Sandy Bridge running NICAM-DC-MINI, which is a mini app for Post-K development, indicated the possibility to increase headroom for power-shifting between jobs.



DCT optimization result for NICAM-DC-MINI on 10 nodes of Sandy Bridge x2

Related Work

- READEX[1] proposes a methodology for automatic tuning to improve energy efficiency. In contrast to our approach, pre-execution of applications with a representative dataset is necessary for this methodology.
- GEOPM[2] is an open source run-time framework for researching energy management solutions. By changing the RAPL settings, its power-balancing plug-in improves a job's execution time, whose performance was originally degraded by constant power-capping by RAPL.

Summary and Future Work

- DFS and DCT techniques were integrated into a run-time "performance-per-watt" optimization and its "on-the-fly" optimization demonstrated its effectiveness.
- Further studies are planned as follows: confirming the applicability to new generation Xeons, studying performance controllability by the degree of acceptable performance degradation, drafting of a concrete scenario of power-shifting between jobs, and so on.

References

1. Oleynik, Yury, et al. "Run-time exploitation of application dynamism for energy-efficient exascale computing (READEX)." *Computational Science and Engineering (CSE), 2015 IEEE 18th International Conference on*. IEEE, 2015.
2. Eastep, Jonathan, et al. "Global Extensible Open Power Manager: A Vehicle for HPC Community Collaboration on Co-Designed Energy Management Solutions." *International Supercomputing Conference.* Springer, Cham, 2017.