Application Runtime Variability and Power Optimization for Exascale Computers

Conference Paper · June 2015

DOI: 10.1145/2768405.2768408

Allan K Porterfield
University of North Carolina at Chapel Hill
41 PUBLICATIONS **2,056** CITATIONS
SEE PROFILE

Robert Fowler
UNC
76 PUBLICATIONS **2,021** CITATIONS
SEE PROFILE

Sridutt Bhalachandra
Argonne National Laboratory
18 PUBLICATIONS **184** CITATIONS
SEE PROFILE

Diptorup Deb
University of North Carolina at Chapel Hill
4 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Application Runtime Variability and Power Optimization for Exascale Computers

Allan Porterfield
Renaissance Computing Institute (RENCI)
akp@renci.org

Rob Fowler
Renaissance Computing Institute (RENCI)
rjf@renci.org

Sridutt Bhalachandra
University of North Carolina at Chapel Hill
sriduttb@cs.unc.edu

Barry Rountree
Lawrence Livermore National Laboratory
rountree4@llnl.gov

Diptorup Deb
University of North Carolina at Chapel Hill
dipto@cs.unc.edu

Rob Lewis
University of North Carolina at Chapel Hill
rjlewis@cs.unc.edu

ABSTRACT

Exascale computing will be as much about solving problems with the least power/energy as about solving them quickly. In the future, systems are likely to be over-provisioned with processors and will rely on hardware-enforced power bounds to allow operation within power budget/thermal design limits.

Variability in High Performance Computing (HPC) makes scheduling and application optimization difficult. For three HPC applications - `ADCIRC`, `WRF` and `LQCD`, we show the effects of heterogeneity on run-to-run execution consistency (with and without a power limit applied). A 4% hardware run-to-run variation is seen in case of a perfectly balanced compute-bound application, while up to 6% variation was observed for an application with minor load imbalances.

A simple model based on Dynamic Duty Cycle Modulation (DDCM) is introduced. The model was implemented within the MPI profiling interface for collective operations and used in conjunction with the three applications without source code changes. Energy savings of over 10% are obtained for `ADCIRC` with only a 1-3% slowdown. With a power limit in place, the energy saving is partially translated to performance *improvement*. With a power limit of 50W, one version of the model executes 3% faster while saving 6% in energy, and a second version executes 1% faster while saving over 10% energy.

1. INTRODUCTION

Exascale computing will be power limited [23]. This either limits the number and speed of processors or, more likely, motivates over-provisioning and execution under hardware-enforced power budgets. Exascale computing will be as much about how to solve a problem with the least power and energy as it is about raw execution performance.

ROSS '15, June 16, 2015, Portland, Oregon, USA
Copyright 2015 ACM 978-1-4503-3606-2/15/06 ...$15.00.
http://dx.doi.org/10.1145/2768405.2768408

We will show that currently, nominally homogeneous computing elements exhibit heterogeneous performance and limiting power increases the performance variation. Transistor thresholds and leakage currents vary within and between wafers during fabrication, resulting in processor chips that require different supply voltages to operate at the design frequency and that therefore consume different amounts of power. Furthermore, the amount of cooling available to a chip depends on its position in the system, resulting in temperature differences that cause additional variations in power consumption. Hence, power and thermal constraints will affect each chip differently causing on-chip mechanisms that control operating frequency to also vary. Performance will thus vary between sockets for even perfectly balanced parallel applications.

Heterogeneity and power limits are going to be major factors in the performance of supercomputer applications. Heterogenous performance makes tuning and optimal scheduling of exascale applications difficult. Understanding the run-to-run performance (energy and time) can provide some insight into the types of static and dynamic heuristics that will prove effective.

In this paper, we first examine variations in performance of short (15-30 minute) runs of three full HPC applications. Performance depends on the application type and degree to which the power is limited. `ADCIRC` has noticable variations emanating from both the system and the application. A Lattice Quantum chromodynamics performance test, `t_leapfrog`, is part of the Chroma [10] distribution. By design, it should exhibit perfect load balance in both computation and communication, but it still shows run-to-run variations in energy and execution time. A well-balanced `WRF` test has the least variation.

A simple model is introduced that reduces the effective clock frequency of cores reaching selected MPI collectives significantly early. By controling the effective clock rate of each core separately, Dynamic Duty Cycle Modulation (DDCM) allows applications with minor load imbalances to save significant energy[1]. When power-limited, the hardware does not have to impact the voltage and clock frequency as

[1]DDCM, is core-specific on Intel processors since the Pentium. Dynamic Frequency and Voltage Scaling (DVFS) has been chip-wide until the Intel Haswell architecture. Our test system consists of Intel SandyBridge chips.

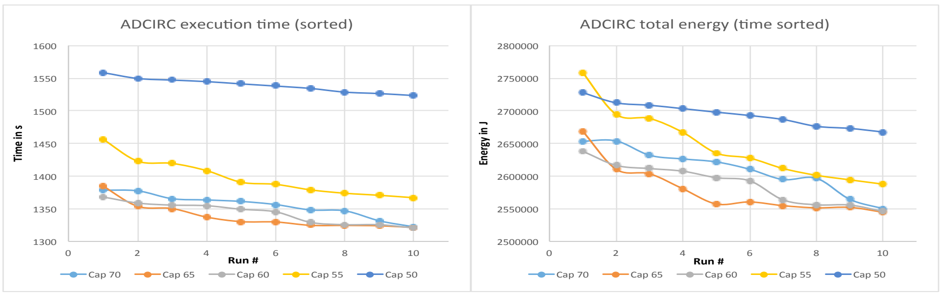

Figure 1: Variation of execution times and energy for `ADCIRC` with different power limits on 16 nodes

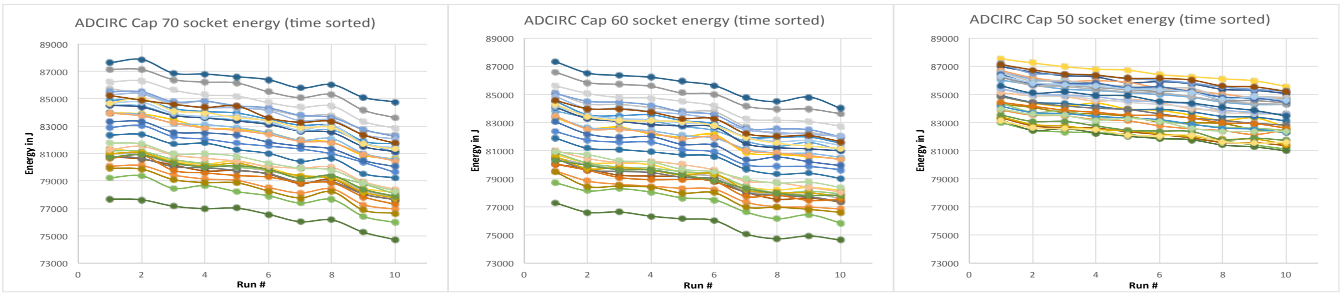

Figure 2: Effect of power cap on socket energy running `ADCIRC` on 16 nodes

much to stay below the power-limit. In some cases, saving energy results in shorter execution times.

Section 2 presents the measured time and energy usage variability for three full HPC applications. Section 3 describes a simple model to save energy for MPI applications that have unbalanced phases. The model is applied to `ADCIRC` in Section 4, resulting in significant energy savings and performance improvement when execution is impacted by a power cap. The last two sections talk about related work and the implications of heterogeneity on Exascale runtime scheduling.

2. APPLICATION RUNTIME VARIABILITY

Run-to-run variation in execution time and energy usage is a serious problem in optimizing systems or application performance. Budgeting time and power will be difficult for Exascale runtimes, workflow managers, and operating systems; when performance is not predictable. To better understand how performance variation differs by application, several short examples of HPC applications that are used in-house were run multiple times on a slice of a local cluster.

2.1 Execution Environment

At RENCI, a sixteen-blade partition of a larger cluster is used for HPC application development and small-scale production runs. This development partition exposes at the user level power and energy instrumentation. It grants the user control of power capping as well as clock frequency and modulation. The partition was dedicated to long-duration experiments required to examine run-to-run variation.

2.1.1 System

All tests use the same 16 Dell M420 nodes within a single bladecenter. Each node has two 8-core Intel Xeon CPU E5-2450 processors for a total of 256 cores with a nominal frequency of 2.1GHz. The blades are air-cooled with air flowing sequentially over the sockets on each blade. Each node runs CentOS 6.5 with a Linux 2.6.32 kernel. Slurm (version 2.6.9) was used for job submission of all the ADCIRC jobs and was upgraded to version 14.11.3 for Weather Research and Forcasting (WRF) and Lattice Quantum Chromo-Dynamics (LQCD) applications. Both Slurms use a modified energy plugin to make the energy RAPL counters available at user level.

2.1.2 Measurement Techniques

Each application was sized to run for between 15-35 minutes This limits any initialization effects and allows the program to compute a significant result. Temperature has a noticeable effect on execution time and energy usage [32]. All tests were run consecutively and are long enough to mitigate the effect of a cool start. Each application was run ten times for each power and software setting. The graphs show all results with the runs sorted by execution time to understand better the average difference between settings.

RAPL All reported energy numbers and power-limit setting are performed with Intel Running Average Power Limit (RAPL) interface. On the SandyBridge architecture, our understanding is that the energy measurements are modeled. Any resulting skew (e.g., difference between measured power and power-limit setting) should be consistent between runs of the same application. The relative difference between runs should be approximately correct, even if there is error in the absolute values.

2.1.3 Control Mechanisms

We use the RAPL power-capping mechanism to control power. Clock modulation is a mechanism that allows the clock duty cycle to be controlled on a per-core basis using a 16-bit mask register that, with k bits set, allows only $k/16$ of each group of 16 clock pulses to reach the core, effectively

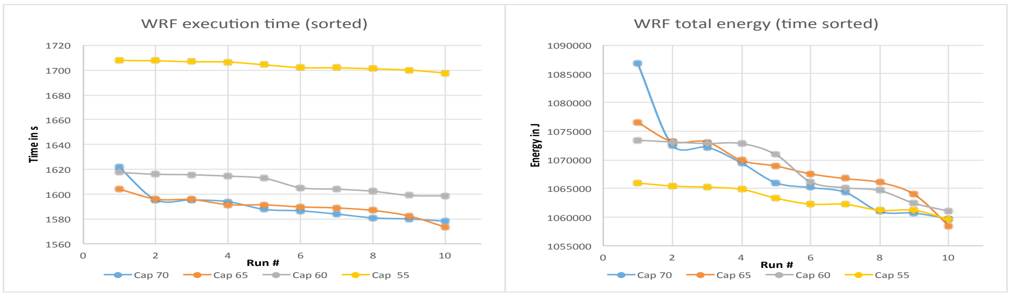

Figure 3: Variation of execution times and energy for `WRF` with different power-limits on 16 nodes

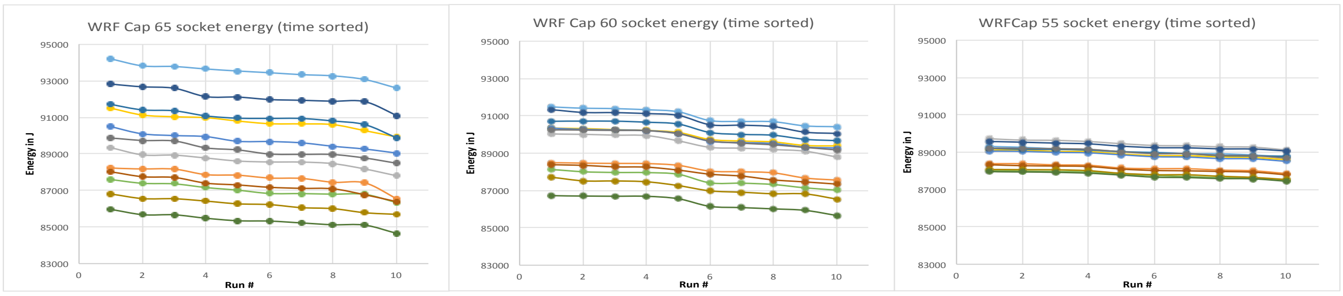

Figure 4: Effect of power cap on socket energy running `WRF` on 16 nodes

reducing clock frequency. Our DDCM mechanism provides a dynamic API to modulation.

2.1.4 *Applications*

Several HPC applications are used regularly on the RENCI system - `ADCIRC` for storm-surge modeling research, `WRF` within a larger climate-change research effort, and an optimization effort for `LQCD`. These applications will drive the study of HPC performance variation.

`ADCIRC`.

`ADCIRC+SWAN` [27, 40, 8, 42], a storm surge, tidal and wind-wave model, uses a finite-element method to discretize shallow water equations, while simultaneously facilitating large domains with very high spatial resolution in coasts of interest, without unnecessary resolution in offshore areas. This high-resolution capability requires substantial compute resources [5]. Research and applications with `ADCIRC` include regional and local tidal phenomena [41, 6], inlet and estuarine dynamics [26, 16]; and storm surge and wave hindcasts [2, 8, 24]. `ADCIRC` is approved by FEMA and was used for development of Digital Flood Insurance Rate Maps in TX, LA, MS, AL, DE, VA, NC, SC, GA, and FL [4, 9, 29]. It is also used as the core numerical model in real-time forecasting and prediction systems [12, 5].

Without a power limit, the test case of `ADCIRC` takes, on average, 1339 seconds and uses, on average, 61.1W in each of the 32 sockets. At a power limit of 70W, the cap is rarely reached and the performance is unchanged. Figure 1, on the left side, shows the execution time at various cap levels (sorted by time). Execution times for 70W, 65W, and 60W vary over the same range (1384 to 1320 sec.) Lowering the limit to 55W results in an ≈3% slowdown, and at 50W the slowdown is about 13.5%. On the right side, Figure 1 shows the energy used by each of the runs. At 70, 65, and 60W the run-to-run execution differences are about 4% (4.2, 4.7 and 3.6%). At 55W the variation jumped to 6.5% (2% was one slow run) and at 50W was down to 2.3%.

Within each run there was a 12.7% difference in the energy used between the most power-hungry socket and the most efficient socket (Figure 2). Socket #30 used the most power in each run and Socket #11 used the least. If this graph is viewed in color, one can see that the even (blue/gray) lines tend to be power hungry and the odd (yellow-green) lines require less power. The Dell M420 blades have the cooling air pass over the odd-numbered socket before passing over the even-numbered socket. The odd-numbered sockets are on average 4.4°C cooler and used 3.6% less energy.

In Figure 2, limiting power reduced variation from 12.7% (70W) and 12.9% (60W) to 5.3% (50W). The power for the individual sockets was relatively constant between the runs at a specific power level, but the socket ordering changed. At 50W, Socket #15 uses the most energy and #5 and #11 tie for the least energy. We speculate that transistor-switching costs and leakage-current costs are not perfectly correlated and one becomes more dominant as the power/temperature increases.

`WRF`.

The Weather Research and Forecasting (`WRF`) Model [28] is a much used mesoscale numerical weather prediction system designed to serve both research and forecasting needs. The model serves a wide range of meteorological applications across scales from tens of meters to thousands of kilometers. Our test case is small and forecasts one day of weather for Puerto Rico given different boundary conditions obtained in a larger climate model simulation. It ran on the first six of the 16 available nodes.

Our `WRF` test is very well load-balanced and runs with little run-to-run variation. The example runs in 1590 sec when

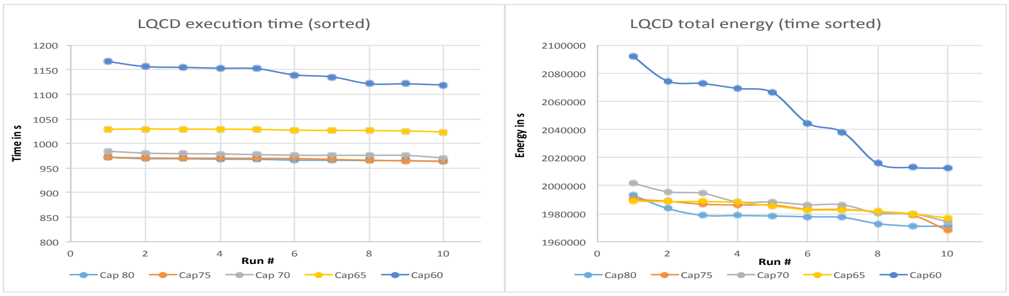

Figure 5: Variation of execution times and energy for `LQCD` with different power-limits on 16 nodes

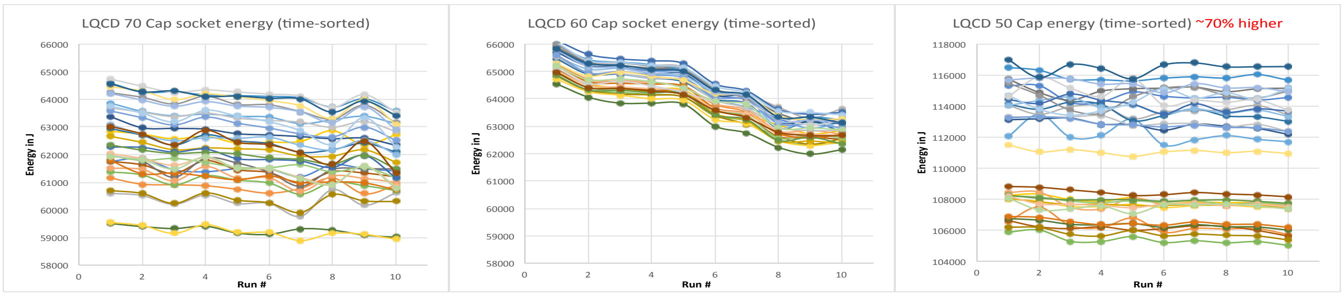

Figure 6: Effect of power cap on socket energy running `LQCD` on 16 nodes

the power-limit is set to 70W or 65W and uses 1.068 million Joules at both settings. The run-to-run variation for both time and energy of between 1.7% and 2.7%. Looking at the left side of Figure 3, the execution time at 60W increases by around 1%; and at 7% at 55W. Not shown is the 60% increase at 50W. The right side of Figure 3 shows the energy consumed, where 50W uses slightly less energy than the higher limits (0.5%). The increased time required at 50W increases energy usage by 31%.

Figure 4 shows the energy used by each node during the ten tests. At 70W (not shown), 65W, and 60W socket #0 used the most energy and socket #11 used the least. At 65W, the sockets differed by 9.5%. At 60W, the difference was down to 5.4%. At 55W the gap between hot and cool sockets is significant, but the overall variation is down to 2.0%. The advantage of better cooling clearly differentiates the sockets running on our cluster when cluster is power-limited.

`LQCD`.

The `t_leapfrog` program, distributed with Chroma [10], is a timing and functional test for a "leapfrog" integration scheme to compute trajectories. The test example is on a $32 \times 32 \times 32 \times 64$ lattice and uses periodic boundary conditions. This example is compute-bound near the limit of useful strong scaling with data fitting in cache as well as with load balance in both computation and communication.

Uncapped, the `t_leapfrog` example takes 16 minutes (967 secs) and uses an average of 63.9W per socket. The tests were run for power limits between 80W and 50W in five-Watt increments. Figure 5 shows some of the results. On average, 80W and 75W differ by only 0.8%, but lowering the limit to 70W increases execution time by 1.4%. This is above the average socket utilization, but the average increase is measurable. We speculate that the program's power usage rises, and the hardware limits slow execution, between computational blocks. The hardware power limit is in effect for only a small percentage of execution time, but the effect is measurable. When the power limit is set low enough that the voltage and clock frequency are lowered for most of the execution (not shown), 55 and 50W, the impact on overall execution time is significant. `t_leapfrog` execution time increases by 97% at 55W and 163% at 50W. The energy penalties (right side of Figure 5) are lower than the execution penalties. At 65W uses only 0.5% more energy is used than in the 80W execution, although it takes 5% longer.

For compute-bound applications, running under a power limit will be a problem. OS scheduling techniques that allow for uneven power distribution maybe beneficial, or the application could allocate a bigger slice of the system than desired and idle some sockets so that the power is moved to the active sockets.

Execution variation for the compute-bound application is lower than for `ADCIRC`. `t_leapfrog` varied between than 0.6% and 1.5% for all of the power limits except 60W. At 60W, the variability was 3.9%. The increased variability as the power limit significantly impacts performance was also observed in `WRF`. This may be because of some sockets run without limits and other sockets limited, resulting in message delays during communication. 60W is the highest, at 3.9%. The execution penalty for running with a power limit of 65W is 6.2%, but the energy penalty is negligible at 0.3%. At 60W the 18% slowdown, results in only a 3.6% increase in energy.

Figure 6 shows the same general pattern. At 70W, sockets #26 and #30 use about 8.5% more energy than sockets #11 and #15. At 60W, socket #22 uses the most power, about 2% more than socket #11. When the power limit degrades performance, the cooler chips can run faster and spend significant time waiting. The node energy graph sepa-

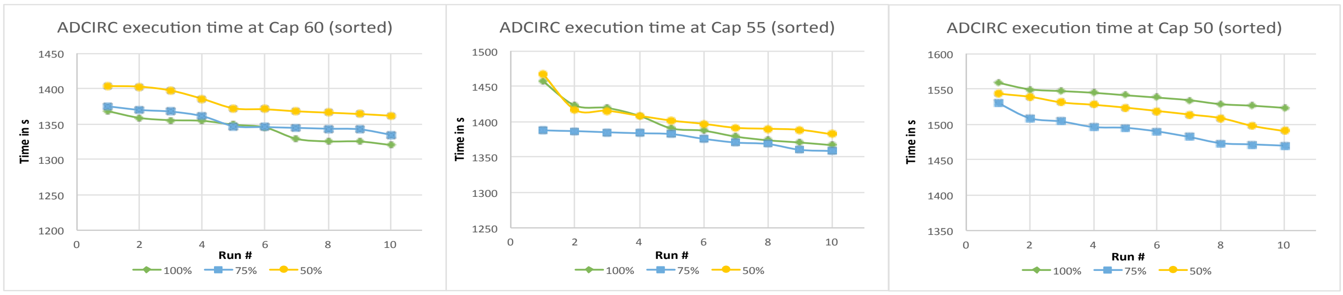

Figure 7: Execution times for ADCIRC with combinations of power cap and minimum threshold duty cycle

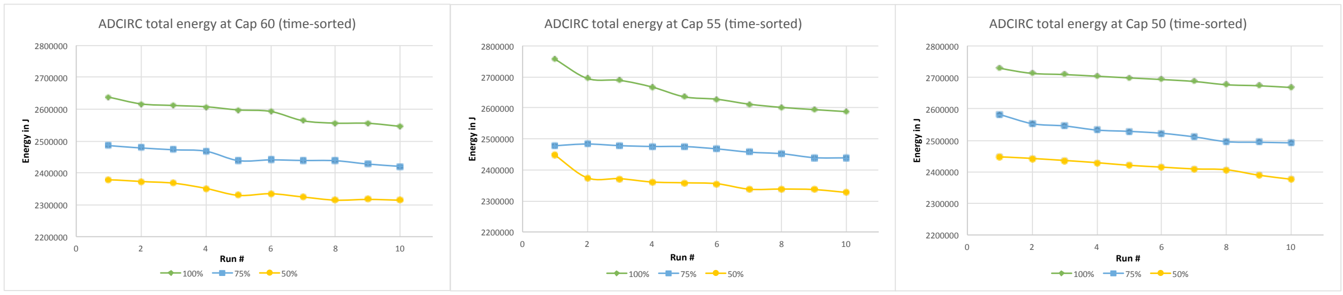

Figure 8: Energy consumed by ADCIRC with combinations of power cap and minimum threshold duty cycle

rates into two regions and the overall variability increases to 10%. In terms of energy usage and variability, `t_leapfrog` has a significant sweet spot around 60W.

3. ENERGY MODEL

A side-effect of current hardware and software tends is to increase the performance variation between threads. Our goal is to detect and to reduce thread imbalance. If a thread is running faster than needed, that thread's clock duty cycle is reduced. The model predicts the lowest effective frequency such that the thread will not be the last to arrive at the next barrier. If the last thread to arrive is running at full speed, the application should experience no slowdown. Reduced clock duty cycles save power. If performance is unchanged, energy is also reduced.

A simple adaptive runtime model constructed in our previous work [3] is applied here to real HPC applications. The model automates the process of picking the clock duty cycle for the next application region by comparing the computing and waiting times of each thread. If a thread reaches a synchronization point early, it is assumed that it will also be early at the next synchronization point. The duty cycle for the next region is calibrated using the compute and waiting times for the previous region. A thread doing more work will run at a higher effective frequency than the one doing less work.

The model uses two rules - one to decrease the clock frequency of a core and the other to increase it. First, we attempt to decrease the clock frequency.

$$L_{down} = \frac{T_{compute}}{T_{total}} * \frac{C_{max}}{C_{current}} \tag{1}$$

- T_* - time
- C_* - clock frequency
- L_* - levels/steps to change clock frequency

The correct clock duty cycle for the next region is a function of the ratio of computing time to total time between barriers and the previous value for duty cycle.

When the previous rule determines that the duty cycle does not need to be reduced, the model then determines whether the duty cycle needs to be increased to prevent the current thread from being the last to arrive at the next barrier, thus slowing the application. Our model incrementally increases the duty cycle rather than jumping directly to the maximum.

The equation to increase the duty cycle level is given by

$$L_{up} = \frac{T_{compute}}{T_{total}} * \frac{C_{min}}{C_{current}} \tag{2}$$

The model estimates the next value for the duty cycle by comparing wait time with how close to the minimum duty cycle the last region was executed.

The two rules return accurate values for $C_{current}$ mathematically, but their direct application to the system does not always produce desirable results. Startup can look very different from the rest of the computation and the model was not used. For low duty-cycle rates, observed performance degradation is higher than predicted. A floors of 75% and 50% duty cycles were used. The duty cycle choosen was rounded up to the next 1/16, because the execution time penalty for running to slow is larger than the potential energy savings.

4. MODEL PERFORMANCE

Section 2 showed a several-percent time variability running the same HPC problem on the same hardware. Performance variations in hardware and software frequently cause some parts of the system to be delayed by their slower counterparts. Section 3 presented a model to lower the effective

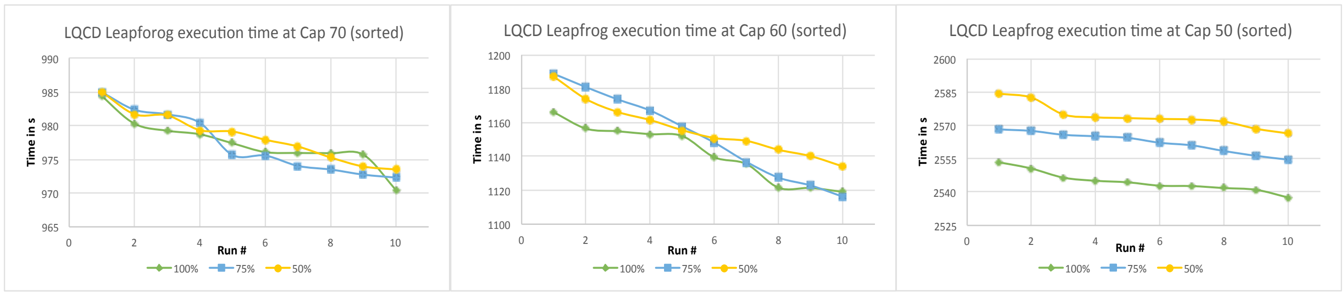

Figure 9: Execution times for `LQCD` with combinations of power cap and minimum threshold duty cycle

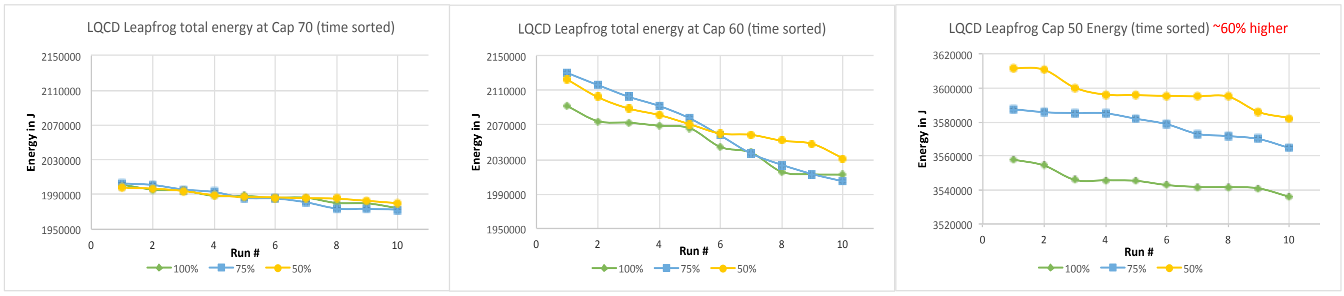

Figure 10: Energy consumed by `LQCD` with combinations of power cap and minimum threshold duty cycle

clock rate of the core to save energy in those cases. The profiling interface of MPI allows a shim library to gain control at the beginning and end of every collective call. The model can be used to control the clock rate of MPI programs without modifying a single line of application code. The only information required by the model are local timestamps, so no additional communication traffic is created. Evaluating the model requires no loops and takes a limited number of instructions. The only danger of significant overhead occurs when the model changes the clock rate frequently. Using DDCM and running as root, such changes can happen quickly. DVFS is inappropriate on our Intel SandyBridge processors because changes affect all of the cores, potentially slowing a critical thread. Relinking the HPC applications to use our shim MPI library means that the instrumentation controlling clock frequency is added quickly and simply.

The danger using the model is at phase changes for one (or more) code regions between collectives a processor will be executing too slowly. For the following tests, minimum speed that the model could set the processor was limited. The SandyBridge architecture allows 16 setting of the duty cycle, the following tests used 3 different minimums; 100% no slowdown allowed; 75% or 12/16 maximum slowdown and 50% or 8/16 maximum slowdown.

`ADCIRC`.

When running `ADCIRC` at 70W (effectively no limit) (Figure 7) the model reduces energy consumption significantly. Running at 3/4 speed, a 6.0% reduction in total energy occurs, and at 1/2 speed it increases to a 11.3% savings. As the power-limit is applied to the application the savings still occur. At 60W and 3/4 speed saves 5.6%, and at 50W, 3/4 at speed saves 6.6%. Examination of the application code reveals that most of the savings occur during IO phases, during which a majority of the work is on a single thread.

HPC systems are expensive, and any energy savings that increases execution time significantly is not cost-effective when you account for computer time. As Figure 8 shows at 60W running at 1/2 and 3/4 speed significant energy is saved, but execution slows slightly. As the power-limit is lowered and starts to impact execution time at 55W and 50W, the energy savings are 7.3% and 6.6%.

Since the power is limited to less than the desired amount, the energy savings can no longer come from a lower average power usage. Saving energy under a power cap means that the execution runs faster. The average increase in execution speed is 1.5% at 55W and 3.1% at 50W. The hardware is intelligent enough to detect that power is being saved in some thread and effectively moves it to the critical threads, IMPROVING performance.

`WRF`.

`WRF` has low run-to-run variation and is well balanced between the threads. At no point does the model detect code regions in which lowering the clock frequency by 6% or greater would not increase execution time. With the model, the execution time, energy, and run-to-run variation are equal to the non-model results presented earlier.

`LQCD`.

`t_leapfrog` is perfectly balanced and has no distinct IO phase. The model detects very few regions in which the clock rate could be reduced (8-15 during a 16 minute run). In Figure 9, the model has minimal impact on execution time. At 60W and 70W, the various settings for the model have no significant impact. At 50W, the model slows the execution by 1% at 3/4 speed and 2% at 1/2 speed.

When energy is examined, Figure 10, at 70W there is no significant difference between the three model settings. At 60W, the total energy used increases slightly (both graphs have the same bounds) but the run-to-run energy variation increases significantly. Overall the increase in execution time at a cap of 50W, greatly increase energy usage. Because the model slows execution speed, the model slightly increases energy usage.

For a compute-bound application, such as `t_leapfrog`, the energy model detects very few regions of code where the imbalance is large enough to change the clock rate. Applying the model has little impact on the execution and energy consumption of compute-bound applications. The application of a power limit can significantly increase execution time and total energy usage.

5. RELATED WORK

Our work has focused on core-specific clock decisions and the interactions between power limits and energy reduction techniques. Most power aware-computing research centered around DVFS uses inter-node [21, 35] or intra-node methods [14, 13] to compute one preferred setting for the chip. Computational workloads have been analyzed to propose ways to save power [11, 19]. Models to amortize the effect of uneven work distribution through slack reclamation have been proposed [22, 21, 20]. Green Queue [39] automates the process of finding phases and optimal frequencies using power models. Automatic tuning of applications based on software performance options and processor clock frequency has also been explored [33]. The empirical software model proposed is similar to the intra-node models, but focuses on individual core (not socket) performance. The model also operates at a finer-granularity than most DVFS work.

Moving beyond DVFS, duty cycle modulation [32], power capping [34] along with similar mechanisms on IBM Power 6 and 7 (capping) and AMD Bulldozer [1] (capping and thermal design power limits) have been explored. Applications have been profiled to determine the best configuration of nodes and power caps for overprovisioned systems [31, 37]. Resource allocation schedulers that use overprovisioning incorporating power-response characteristics of each job along with power cap are being explored [36]. We have shown for several applications energy reduction techniques can reduce execution time in overpovisioned systems saving energy and potentially improving system throughput.

A number of efforts use hardware performance counters [38, 7, 25] to compute optimal off-line settings. Several projects estimate energy usage based on hardware counters with direct correlation to cache access [15], MIPS [17] and CPU stall cycles [18]. Our model only uses the system clock to make lightweight dynamic adjustments to each core's individual clock frequency.

6. CONCLUSIONS

Many exascale applications will have heterogeneous processor load, and with power-limits most exascale systems will have heterogeneous performance. This leads to significant run-to-run variations in the execution time and energy consumed. We introduced a simple model for saving energy by dynamically setting core-specific clock rates. Since the hardware uses less power for cores doing less work, when power-limited it is free to allocate more power to cores running the critical threads. This improves overall performance, suggesting that saving energy will also be a performance optimization.

Research over the last 10-15 years has focused on software techniques to save energy. The techniques save energy but increase execution time. HPC has to date, ignored the results because very high machine depreciation costs make absolute performance critical. With exascale, changes in system design like over-provisioning will make software energy saving techniques relevant to HPC. There needs to be a re-examination of previous energy related research in the context of power-limits and the more flexible DVFS implementations in recent processor architectures.

7. ACKNOWLEDGMENTS

This work is partially supported by the DOE XPress (DE-SC0008704), DoD ATPER (PNNL PO-215099) and SciDAC SUPER (DE-SC0006925). This material is based upon work supported by the U.S. Department of Energy's Lawrence Livermore National Laboratory. Office of Science, under Award number DE-AC52-07NA27344 and supported by Office of Science, Office of Advanced Scientific Computing Research (LLNL-CONF-656877).

8. ADDITIONAL AUTHORS

Additional authors: Brian Blanton (Renaissance Computing Institute (RENCI)), email:`blanton@renci.org`.

9. REFERENCES

[1] BIOS and kernel developer's guide (bkdg) for AMD family 15h models 00h-0fh processors. Technical report, AMD, 2012.

[2] J. Atkinson, T. Wamsley, J. Westerink, M. Cialone, C. Dietrich, K. Dresback, R. Kolar, D. Resio, C. Bender, B. Blanton, et al. Hurricane storm surge and wave modeling in southern Louisiana: A brief overview. In M. Spaulding, editor, *Estuarine and Coastal Modeling X*. ASCE, 2008.

[3] S. Bhalachandra, A. K. Porterfield, and J. F. Prins. Using dynamic duty cycle modulation to improve energy efficiency in high performance computing. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2015*. IEEE, to appear.

[4] B. Blanton. North Carolina Coastal Flood Analysis System: Computational System. Technical Report TR-08-04, Renaissance Computing Institute, The University of North Carolina at Chapel Hill, 2008.

[5] B. Blanton, J. McGee, J. Fleming, C. Kaiser, H. Kaiser, H. Lander, R. Luettich, K. Dresback, and R. Kolar. Urgent computing of storm surge for North Carolina's coast. *Procedia Computer Science*, 9(0):1677 – 1686, 2012. Proceedings of the International Conference on Computational Science, ICCS 2012.

[6] B. Blanton, H. Seim, R. Luettich, D. Lynch, F. Werner, K. Smith, G. Voulgaris, F. Bingham, and F. Way. Barotropic tides in the South Atlantic Bight. *J. Geophys. Res.*, 109(C12024), 2004.

[7] K. Choi, R. Soma, and M. Pedram. Dynamic voltage and frequency scaling based on workload decomposition. In *Proc. of the 2004 Intl. Symposium on Low Power Electronics and Design*, 2004.

[8] J. C. Dietrich, S. Bunya, J. J. Westerink, B. A. Ebersole, J. M. Smith, J. H. Atkinson, R. Jensen, D. T. Resio, R. A. Luettich, C. Dawson, V. J. Cardone, A. T. Cox, M. D. Powell, H. J. Westerink, and H. J. Roberts. A high-resolution coupled riverine flow, tide, wind, wind wave, and storm surge model for southern louisiana and mississippi. Part II: Synoptic description and analysis of hurricanes Katrina and Rita. *Mon. Wea. Rev.*, 138(2):378–404, 2011/10/27 2010.

[9] B. Ebersole, J. Westerink, S. Bunya, J. Dietrich, and M. Cialone. Development of storm surge which led to

flooding in St. Bernard Polder during Hurricane Katrina. *Ocean Eng.*, 37(1, Sp. Iss. SI):91–103, JAN 2010.

[10] R. G. Edwards and B. Joó. The chroma software system for lattice QCD. *Nucl. Phys B1*, 40 (Proc. Suppl.), 2005. arXiv:hep-lat/0409003.

[11] X. Feng, R. Ge, and K. W. Cameron. Power and energy profiling of scientific applications on distributed systems. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, 2005.*, pages 34–34. IEEE, 2005.

[12] J. Fleming, C. Fulcher, R. Luettich, B. Estrade, G. Allen, and H. Winer. A real time storm surge forecasting system using ADCIRC. In M. Spaulding, editor, *Estuarine and Coastal Modeling X*. ASCE, 2008.

[13] V. W. Freeh and D. K. Lowenthal. Using multiple energy gears in MPI programs on a power-scalable cluster. In *PPoPP 2005: Proc. of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2005.

[14] R. Ge, X. Feng, and K. W. Cameron. Performance-constrained distributed DVS scheduling for scientific applications on power-aware clusters. In *SC05: Proc. of the 2005 ACM/IEEE Conference on High Performance Networking and Computing*. IEEE Computer Society, 2005.

[15] R. Ge, X. Feng, W. Feng, and K. Cameron. CPU miser: A performance-directed, run-time system for power-aware clusters. In *ICPP 2007: 36th Intl. Conference on Parallel Processing*. IEEE, 2007.

[16] J. Hench and R. Luettich. Transient tidal circulation and momentum balances at a shallow inlet. *J. Phys. Ocean.*, 33(4):913–932, 2003.

[17] C. Hsu and W. Feng. A power-aware run-time system for high-performance computing. In *SC05: Proc. of the 2005 ACM/IEEE Conference on High Performance Networking and Computing*. IEEE Computer Society, 2005.

[18] S. Huang and W. Feng. Energy-efficient cluster computing via accurate workload characterization. In *CCGrid 2009: Proc. of the 9th IEEE/ACM Intl. Symposium on Cluster Computing and the Grid*. IEEE Computer Society, 2009.

[19] S. Kamil, J. Shalf, and E. Strohmaier. Power efficiency in high performance computing. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–8. IEEE, 2008.

[20] J. Kang and S. Ranka. Dynamic slack allocation algorithms for energy minimization on parallel machines. *Journal of Parallel and Distributed Computing*, 70(5), 2010.

[21] N. Kappiah, V. W. Freeh, and D. K. Lowenthal. Just in time dynamic voltage scaling: Exploiting inter-node slack to save energy in mpi programs. In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 33. IEEE Computer Society, 2005.

[22] H. Kimura, M. Sato, Y. Hotta, T. Boku, and D. Takahashi. Emprical study on reducing energy of parallel programs using slack reclamation by DVFS in a power-scalable high performance cluster. In *CLUSTER 2006: Proc. of the 2006 IEEE Intl. Conference on Cluster Computing*. IEEE, 2006.

[23] P. Kogge, K. Bergman, S. Borkar, D. Campbell, W. Carson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, et al. Exascale computing study: Technology challenges in achieving exascale systems. 2008.

[24] N. a. K. E. Lin, J. Smith, and E. Vanmarcke. Risk assessment of hurricane storm surge for New York City. *J. Geophys. Res.*, 115(D18121), 2010.

[25] C. W. Lively, X. Wu, V. E. Taylor, S. Moore, H.-C. Chang, C.-Y. Su, and K. W. Cameron. Power-aware predictive models of hybrid (MPI/OpenMP) scientific applications on multicore systems. *Computer Science - R&D*, 27(4), 2012.

[26] R. Luettich, S. Carr, J. Reynold-Fleming, C. Fulcher, and J. McNinch. Semi-diurnal seiching in a shallow, micro-tidal lagoonal estuary. *Cont. Shelf Res.*, 22:1669–1681, 2002.

[27] R. Luettich, J. Westerink, and N. Scheffner. ADCIRC: an advanced three-dimensional circulation model for shelves coasts and estuaries, Report 1: Theory and methodology of ADCIRC-2DDI and ADCIRC-3DL. Dredging Research Program Technical Report DRP-92-6, USACE/ERDC, Waterways Experiment Station, Vicksburg, MS, 1992.

[28] J. Michalakes, J. Dudhia, D. Gill, T. Henderson, J. Klemp, W. Skamarock, and W. Wang. The weather research and forecast model: software architecture and performance. In *Proceedings of the 11th ECMWF Workshop on the Use of High Performance Computing In Meteorology*, volume 25, page 29. Reading, UK, 2004.

[29] A. Niedoroda, D. Resio, G. Toro, D. Divoky, H. Das, and C. Reed. Analysis of the coastal Mississippi storm surge hazard. *Ocean Eng.*, 37(1):82–90, 2010.

[30] E. Park, J. Cavazos, and M. A. Alvarez. Using graph-based program characterization for predictive modeling. In *CGO*, pages 196–206, 2012.

[31] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski. Exploring hardware overprovisioning in power-constrained, high performance computing. In *Proceedings of the 27th international ACM conference on International conference on supercomputing*, pages 173–182. ACM, 2013.

[32] A. Porterfield, R. Fowler, S. Bhalachandra, and W. Wang. Openmp and mpi application energy measurement variation. In *Proceedings of the 1st International Workshop on Energy Efficient Supercomputing*, page 7. ACM, 2013.

[33] S. M. F. Rahman, J. Guo, A. Bhat, C. Garcia, M. H. Sujon, Q. Yi, C. Liao, and D. Quinlan. Studying the impact of application-level optimizations on the power consumption of multi-core architectures. In *Proceedings of the 9th conference on Computing Frontiers*, CF '12, pages 123–132, 2012.

[34] B. Rountree, D. H. Ahn, B. de Supinski, D. K. Lowenthal, and M. Schulz. Beyond DVFS: A first look at performance under a hardware-enforced power bound. In *HP-PAC 2012: Proc. of the 8th Workshop on High Performance, Power-Aware Computing*, May 2012.

[35] B. Rountree, D. K. Lowenthal, B. R. de Supinski, M. Schulz, V. W. Freeh, and T. K. Bletsch. Adagio: Making DVS practical for complex HPC applications. In *ICS '09: Proc. of the 23rd Intl. Conference on Supercomputing*, 2009.

[36] O. Sarood, A. Langer, A. Gupta, and L. Kale. Maximizing throughput of overprovisioned hpc data centers under a strict power budget. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, ser. SC*, volume 14.

[37] O. Sarood, A. Langer, L. Kalé, B. Rountree, and B. De Supinski. Optimizing power allocation to cpu and memory subsystems in overprovisioned hpc systems. In *Cluster Computing (CLUSTER), 2013 IEEE International Conference on*, pages 1–8. IEEE, 2013.

[38] D. C. Snowdon, E. L. Sueur, S. M. Petters, and G. Heiser. Koala: a platform for OS-level power management. In *Proc. of the 2009 EuroSys Conference*, 2009.

[39] A. Tiwari, M. Laurenzano, J. Peraza, L. Carrington, and A. Snavely. Green queue: Customized large-scale clock frequency scaling. In *CGC '12: Proc. of the 2nd Intl. Conference on Cloud and Green Computing*, Nov. 2012.

[40] J. Westerink, R. Luettich, J. Feyen, J. Atkinson, C. Dawson, H. Roberts, M. Powell, J. Dunion, E. Kubatko, and H. Pourtaheri. A basin- to channel-scale unstructured grid hurricane storm surge model applied to Southern Louisiana. *Mon. Weather Rev.*, 136:833–864, 2008.

[41] J. Westerink, R. Luettich, and J. Muccino. Modeling tides in the western North Atlantic using unstructured graded grids. *Tellus*, 46a:125–152, 1994.

[42] M. Zijlema. Computation of wind-wave spectra in coastal waters with SWAN on unstructured grids. *Coastal Eng.*, 57(3):267–277, 2010.