Measurement and interpretation of micro benchmark and application energy use on the Cray XC30

Brian Austin, and Nicholas J. Wright [*]

August 29, 2014

Abstract

Understanding patterns of application energy use is key to reaching future HPC efficiency goals. We have measured the sensitivity of energy use to CPU frequency for several microbenchmarks and applications on a Cray XC30. First order fits to the performance and power data are sufficient to describe the energy used by these applications. Examination of the resulting energy model shows that the and application's energy/frequency profiles have minima only if a) the frequency change crosses an architectural balance point that is performance-critical for the particular application or b) a significant fraction of the runtime is spent in off-chip operations or c) there is sufficient static power drawn to motivate a race-to-halt. All three forms of energy minima are represented in our sample of HPC applications. The energy-optimal frequencies on this architecture are: MILC (1.8 GHz), GTC (3.6 GHz) and Mini-DFT (1.6-1.8 GHz).

1 Introduction

The first Top500 list was released in 1993 [1]. Throughout that time, the concomitant increases in power consumption of High Performance Computing (HPC) platforms has grown faster (20% annually) than the rate of monetary inflation (3% annually). Today, most "powerful" system on the most recent Top500 draws most almost 18 MW. At $1M/MWy (a coarse estimate for present electricity prices in the United States), this corresponds to $18 M/year. Thus the power draw and the total energy use are increasingly prominent concerns for operating an HPC facility, especially as it is possible to envision a future where the energy costs over the lifetime of the machine are greater than the capital purchase price. An additional concern is the rate of change in power draw, as

[*]B. Austin and N. Wright are with the National Energy Research Scientific Computing Center, Lawerence Berkeley National Laboratory, Berkeley, CA, 94720 USA e-mail: {baustin,njwright}@lbl.gov.

electricity companies charge more for customers who's demands fluctuate rather than stay fixed.

To computing industry in general has been working on these issues for several years now. Metrics such as PUE[2] have been used to motivate increasing the overall efficiency of a datacenter by decreasing the energy overhead of maintaining the surrounding facility. Also, chip manufacturers and system integrators are pursuing advances in architecture and packaging to reduce the power demands of the system. Among these architectural trends is a shift toward chips with a larger number of (possibly less complex) cores running at slower frequencies. HPC users have little or no control over these design choices, yet it is their use of the system and the behavior of their applications that drive the systems' energy use. In this paper, we aim to understand which patterns of memory access and computation contribute most to application energy use on a current HPC platform, Edison a Cray XC30 at NERSC. We also examine which applications can reduce their energy use by running at lower CPU frequencies. This work was motivated by the recent addition of functionality to the XC30 where i) the CPU frequency could be specified for each instance of an application run and ii) the ability to measure the total energy used on a per node basis.

The principle contributions of this paper are:

- We provide measurements of the sensitivity of performance and power to CPU frequency for elementary computational and memory access microbenchmarks.
- We produce a simple energy model by fitting performance and power data that identifies and explains the presence of three (four if you count f=0) types of minima on the energy vs. frequency curve.
- Our results show that the energy-optimal frequencies on this architecture are different for each of the three applications discussed, MILC (1.8 GHz), GTC (3.6 GHz) and Mini-DFT (1.6-1.8 GHz), which can be explained by an knowledge of their performance characteristics. Our results also indicate that the energy savings by running at these energy-optimal frequencies are less than 10% and always come with the a performance penalty.

The remainder of this paper is organized as follows. Section 2 describes the hardware platform used for our experiments and its interfaces for measuring energy and adjusting power states. The power and performance of several microbenchmarks are measured in section 3. These data are used to motivate the form of the energy model described in section 4. Section 5 measures the energy/performance tradeoffs for three scientific HPC applications (MILC, GTC and MiniDFT) and compares these results to trends in the microbenchmark data. Section 6 frames the contributions of this paper in the context of related work. Our conclusions are summarized in section 7.

2 Experimental Platform

2.1 Computational System

All of our experiments use the "Edison" system, a Cray XC30 system located at NERSC. As of August 2014, Edison is composed of 5576 compute nodes connected in a dragonfly topology by a custom Cray Aries network. Each node has two 12-core Intel "Ivy-Bridge" E5-2695 processors operating at a nominal frequency of 2.4 GHz, offering a theoretical peak flop rate of 460 GF/s/node. The actual CPU frequency varies and may increase up to 3.2 GHz due to Intel's Turbo Boost features.[?] The nodes also provide 64 GB of DDR3-1600 memory, with a theoretical peak bandwidth of 102 GB/s/node.

2.2 Power Measurements

The XC30 architecture includes several features to support power monitoring and management. The Ivy-Bridge processors have Intel Running Average Power Limit (RAPL) counters that estimate the energy used by the chip package, cores or DRAM by counting events that take place on the chip. XC30 nodes include an energy monitoring component that measures the total energy used by the node. The Cray power monitoring (PM) counters sample this energy with a frequency of approximately 10 Hz and write the result to virtual files in `/proc/cray/pm_counters`. A PAPI interface[3] to the Cray PM counters is also provided, which enables sampling experiments to attribute energy and power costs to specific functions without requiring undue source code instrumentation. Sets of four nodes and an Aries router are packaged into a blade, which has its own power monitoring component. Blade energy use is sampled at a frequency of 1 Hz, then summed and logged on a System Environmental Data Collection (SEDC) server.

Our experiments are based on the node-level Cray PM counters because direct access to RAPL counters is not available to unprivileged users on Edison and the SEDC logs do not have sufficient spatial or temporal resolution for our purposes. We created a small library to record the current state of the energy and time counters and then inserted calls to this at the beginning and end of each applications source code, as well as around regions of interest. At the end of the application run one MPI process on each node computes the difference between the final and initial counter values. This data is summed over nodes and the total energy, average wall time and average power use are appended to the application output.

2.2.1 CPU Frequency Control

In addition to the power measurement tools described above, recent versions of the Cray Linux Environment have enabled a degree of user-level power-management. The aprun application launcher provides a `--p-state` option to control, at run-time, the CPU frequency to be used on the compute nodes.

On Edison, the frequency can be adjusted between 1.6 and 2.4 GHz in 0.1 GHz increments. Our experiments use the `--p-state` feature to explore the effect of CPU frequency on power and energy use.

3 Microbenchmarks

We use three simple single-node micro-benchmarks to orient our expectations and analysis of application energy use. Each benchmark represents a limiting case of a computational and memory access patterns.

3.1 STREAM

The STREAM benchmark is commonly used to measure the bandwidth that can be sustained by a node's memory subsystem [4]. We have included it to understand the energy characteristics of memory-bandwidth limited kernels. For our power measurements, we use an OpenMP implementation of the STREAM triad kernel with unit stride and an array size of 4.6 GB, running on all 24 cores. In order to amortize the relatively high cost of reading the energy counters, we perform 300 passes through the array so that the total walltime is on the order of 30 seconds.

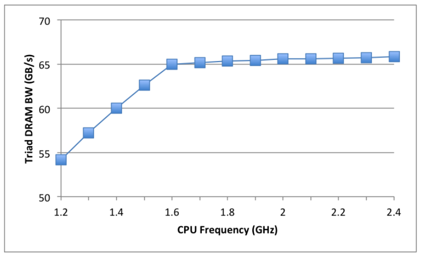
Figure 1 shows the how the STREAM triad bandwidth and power consumption vary with CPU frequency. Both curves are (evidently) piecewise linear with a cusp at 1.6 GHz. The STREAM bandwidth increases quickly with CPU frequency up to 1.6 GHz, but grows very slowly at higher frequencies. At lower frequencies, the bandwidth is throttled by the memory controller, which operates at the CPU frequency. Above the cusp, the memory controller is capable of higher relay rates, but is limited by the speed of the DRAM bus, which, as was mentioned in Section 2 is set to 1600 MHz (1.6 GHz) on Edison.

An alternative explanation could be that the cores' aggregate load/store issue rate increases with the CPU frequency until all available bandwidth from the DRAM bus is used, but this is not consistent with our observation that the cusp does not move to higher frequencies as cores are idled (not shown).[1] The flattening of STREAM performance is reflected more subtly in the power use shown in Fig. 1b. There is a slight decrease in slope above 1.6 GHz, as the dynamic power used by the cores does not increase any further after the controller runs fast enough to saturate the memory bus. Based on this initial observation, we use piecewise fits to the performance and power curves for all kernel and application benchmarks.

3.2 DGEMM

The DGEMM benchmark performs dense matrix-matrix multiplication of large (8000 x 8000) arrays and is representative of routines that exhibit high compu-

[1]A cusp due to saturation of the memory bus does appear when fewer than eight threads are used.

(a) 

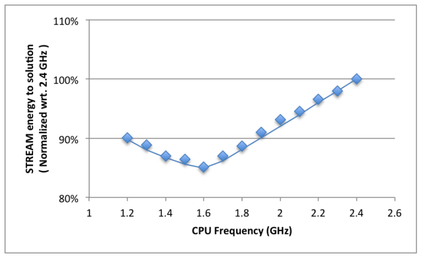
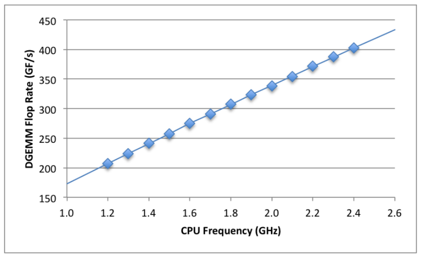
(b) 

Figure 1: Frequency sensitivity of the STREAM triad bandwidth(a) and energy to solution (b). Measured data are marked with points. Lines are fits to the form described in Section 4.

(a) 

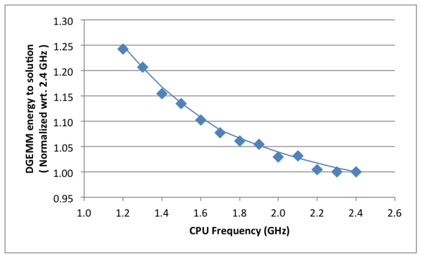
(b) 

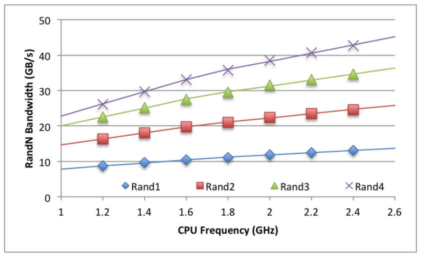
Figure 2: Frequency sensitivity of DGEMM performance (a) and energy to solution (b). Lines are fits of the measured data.

tational intensity, exploit vector operations and have extensive cache re-use. We use the threaded DGEMM routine provided by Intel's MKL library, running on all 24 cores of a node.[2] DGEMM performance and power use are shown in Figure 2; both scale linearly with frequency. The absence of a cusp at 1.6 GHz is in accord with our expectation that DGEMM is insensitive to memory bandwidth and has performance that is proportional to clock speed.

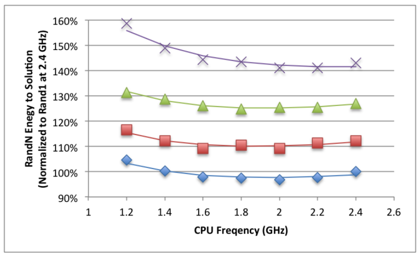
3.3 RandN

We have written a multi-threaded pointer chasing benchmark, RandN, to represent algorithms in which memory latency is a performance limiting factor. During the preliminary phase of the benchmark, an array of `Ndata` elements is is initialized to a random non-cyclic permutation. Each thread then follows `Nstream` sequences of addresses through the array. (Rand1 follows one stream, Rand2 follows two streams, etc.) The core loop of Rand2 is shown in Algorithm 1. Although only one array of addresses is shared among all threads, the

[2]Results obtained from Cray's libsci DGEMM were qualitatively similar to those obtained using MKL.



(a) Performance vs. Frequency



(b) Power vs. Frequency

Figure 3: Frequency sensitivity of the RandN performance(a) and energy to solution(b). RandN bandwidth is calculated assuming one 64-byte cache line is transferred for each address.

streams are effectively independent from each other and are unlikely to share cache lines if `Nstream` $\ll$ `Ndata`.

```
//Rand2
Nstream = 2;
Nthread = omp_get_num_threads();
jstride = Ndata / Nstream / Nthread;
#pragma omp parallel
#pragma omp private(ithread,j0,j1,i)
{
  ithread = omp_get_thread_num();
  j0=(0*Nthread+ithread)*jstride;
  j1=(1*Nthread+ithread)*jstride;
  #pragma omp for
  for( i=0; i<imax; i++ ){
    j0=a[j0];
    j1=a[j1];
}
```

Algorithm 1: Source code for RandN benchmark with `Nstream=2` (Rand2). A third stream can be added by inserting `j2=a[j2];` into the for loop.

The performance and power plots in Figure 3 have the same piecewise linear shapes observed for STREAM, though the change in slope at the cusp is significantly less acute. It is hardly perceptible for Rand1, but becomes gradually more pronounced as the number of simultaneous address streams increases and creates more contention at the memory controller.

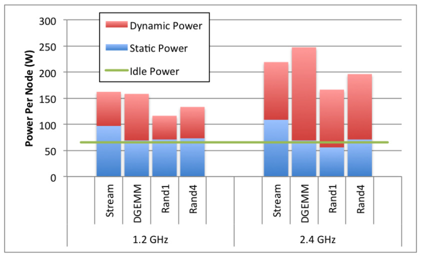


Figure 4: Microbenchmark power comparison. Static and dynamic power are application dependent. The horizontal lines show the total power used by an idle node.

3.4 Microbenchmark comparison

Figure 4 compares the power used by each microbenchmark at the lowest and highest frequencies tested. Linear fits to the power data from Figures 1-3 were used to separate static power (intercept) from dynamic power (slope). DGEMM was the most power-hungry benchmark– it excercises the FPUs, caches and memory bandwidth resources. At the higher frequency, STREAM's performance is limited by memory bandwidth, so the FPUs are not fully utilized and the dynamic power is somewhat lower than that of DGEMM. The static power used by STREAM is higher than the other benchmarks due to its heavy use of DRAM, which operates at a different frequency. The latencies and data dependencies of the RandN benchmark prevent full utilization of processor or the memory, resultign in low static and dynamic power. In all cases, static power is a large fraction (28-60%) of the power used.

4 Energy Model

The microbenchmark data from the previous section shows that the relationships between power and performance and CPU frequency can be closely fit by first order models except at architectural balance points where the performance bottleneck switches from the processor to another device (e.g. DRAM or network interface) that operates at an independent frequency.

The intercept of the linear power model $P = p_0 + p_1 f$ can be interpreted as the static power caused by leakage current (p_0) while the slope (p_1) is proportional to the sum of the dynamic power due to gates switching states and the short circuit power caused by transient loop closures when gates are in the process of switching. Our experiments cannot distinguish dynamic and short circuit power and we refer to both collectively as dynamic power.

The walltime for each benchmark can be approximated $T = w_0 + w_1/f$. This model partitions the time into a frequency independent term, w_0, due to off-chip operations that cannot be temporally overlapped with other work done on the processor and a frequency dependent term w_1/f that corresponds to operations

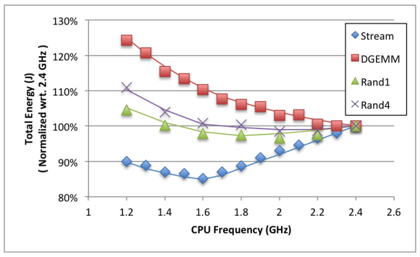


Figure 5: Microbenchmark energy comparison. For each benchmark, the energy-to-solution is normalized with respect to its energy use at 2.4 GHz. Values less than than 100% represent energy savings. Lines are fitted to the measured data.

that take place on the processor (at a rate proportional to the CPU frequency).

The units and values of the p_0, p_1, w_0 and w_1 parameters depend on the details of each benchmark. It would be interesting to relate these parameters to specific events as done by Kestor [5], but this exceeds the needs of our analysis.

For a uniform workload at a constant CPU frequency, the energy can be estimated using

$$E = P \times T = (p_{0,i} + p_{1,i}f)(w_{1,i}/f + w_{0,i}), \tag{1}$$

where the i subscripts correspond to different regions of the piecewise models.

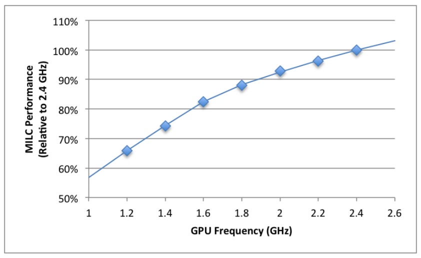
Figure 5 shows that this energy model agrees well with the measured data. (We obtained the values for the p_0, p_1, w_0 and w_1 parameters by a linear fit to the experimentally measured data.) The qualitative differences between the energy/frequency curves in Figure 5 are readily explained by the limiting cases of Equation 1.

The minimum for STREAM at 1.6 GHz coincides with movement of the performance bottleneck from the memory controller to the memory bus. This minimum appears due to the piecewise nature of the energy model; not an analytical minimum. Performance of the DGEMM benchmark is expected to be determined by floating-point performance, so the constant time parameter (w_0, i) is nearly zero, leading to

$$E_{\text{DGEMM}} = (p_{0,i} + p_{1,i}f)p_{1,i}f = p_{0,i}w_{1,i}/f + p_{1,i}w_{1,i} \tag{2}$$

which, like the measured data, decreases monotonically with frequency. DRAM latency is a critical element of RandN performance, so the w_0 parameter is nonzero and an energy minimum appears at $f = \sqrt{p_0 w_1 / p_1 w_0}$. As the number address streams per thread increases from 1 to 4, the out-of-order processor is able to overlap the latency of the load requests, so the w_0 parameter decreases relative to w_1 and the minimum shifts to higher frequencies. It is also interesting to consider the hypothetical case where static power is eliminated $(p_{0,i} = 0)$, giving

$$E_{nostatic} = p_{1,i}f(w_{1,i}/f + w_{0,i}) = p_{1,i}(w_{1,i} + w_{0,i}), \tag{3}$$

(a) 

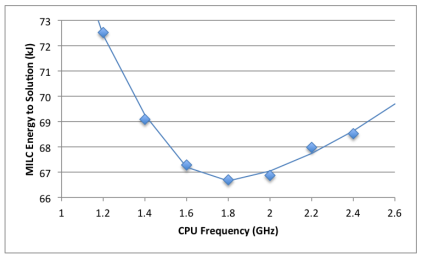
(b) 

Figure 6: Performance/Energy tradeoffs for MILC. (a) Performance vs. CPU Freqency. (b) Energy to solution vs. CPU Frequency. Lines are fits of the energy and walltime models in Equation 1 to the measured data.

which minimizes total energy when f=0. Finally, if both the static power and the constant time parameter are zero, then the total energy is independent of CPU frequency.

5 Application Energy Measurements

5.1 MILC

MILC implements SU3 lattice gauge theory.[6, **?**] Its primary kernel performs a stencil operation on a 4D grid. The benchmark problem run 24 processes on a single node and allocates 8x8x8x8 sites per MPI process.

Earlier performance studies have shown that MILC's performance is sensitive to memory bandwidth,[7, 8] so we expect MILCs energy use to resemble that of STREAM. Indeed, the total energy use is minimized at 1.8 GHz, close to the stream minimum at 1.6 GHz. Energy savings of 2.7% are possible if the performance reduction of 13% is tolerable. Today, this value proposition probably leans towards saving the HPC users time, in the future however this may change.

5.2 GTC

The gyrokinetic toroidal code (GTC) is used to simulate microturbulence in Tokomak fusion reactors.[**?**] GTC has been the subject of much previous benchmarking and optimization research and detailed descriptions of the algorithm are available elsewhere.[9, 10, 11] In short, a gyroaveraged particle-in-cell algorithm is used to propagate particles through a magnetic confining potential. The benchmark problem is based on the 'small' problem from the Trinity-NERSC8 benchmark suite, but has been modified to run on a single Edison node by decreasing `mzetamax` to 24 and increasing `micell` to 266 so that the total number of particles is not changed.

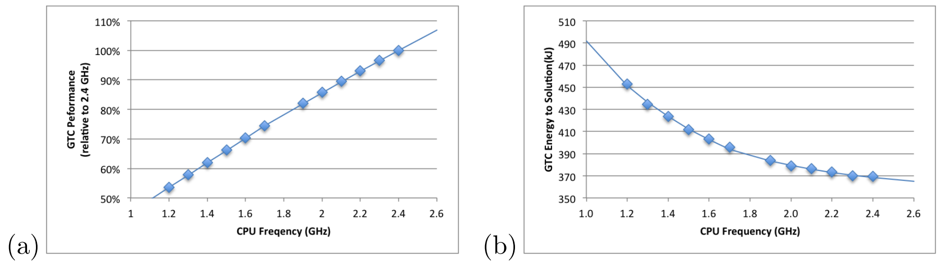


Figure 7: Performance/Energy tradeoffs for GTC. (a) Performance vs. CPU Frequency. (b) Energy to solution vs. CPU Freqency. Lines are fits of the energy and walltime models in Equation 1 to the measured data.

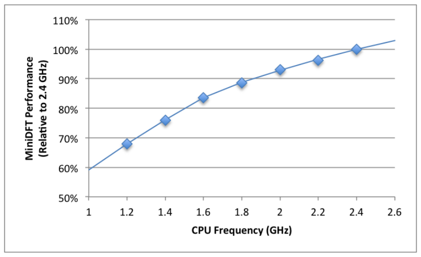
For the benchmark problem used here, 92% of the run time is spent in the push and charge steps of the PIC algorithm. These two steps are responsible for depositing charge from the particles to the cells and interpolating fields from the cells back to the particles. Both steps involve nonuniform memory access patterns into the array of cells as well as indirect addressing. Thus is is not surprising that the energy and time use of the GTC code resemble the Rand4 benchmark. Figure 7 shows that the energy-minimizing frequency for GTC is higher than 2.4 GHz. Also, note that performance is almost a linear function of frequency, at 1.2 GHz the performance is very close to 50%, which, again, is very similar to the behavior of the RAND4 benchmark described in Section 3.

5.3 Mini-DFT

Mini-DFT performs plane wave density functional theory calculations to simulate the electronic structure of materials.It is a mini-application extracted from the broadly used Quantum Expresso package.[12] The benchmark problem simulates a 2x2x2 supercell of titanium dioxide using the PBE functional and a 100 Ry plane-wave cutoff. MiniDFT's diagonalization function requires a square number of processes; other processes wait at a barrier for the duration of this function. For the purpose of load balancing, we run with 144 processes (the smallest square integer that can be evenly divided among Edison's 24-core nodes.) All runs used four-way task-group parallelism.[3]

To solve the Kohn-Sham equations, MiniDFT iterates between two computational phases. In PW-DFT, wave functions are stored on a uniform rhombohedral grid and the FFT phase transforms wave functions between real space (where the potential is readily evaluated) and reciprocal space (where the kinetic energy is evaluated). This phase is dominated all-to-all communication pattern associated with the transpose steps typical of parallel 3D FFTs. The second phase is composed of various linear algebra functions that construct and diagonalize the Fock matrix. We have profiled Mini-DFT at 2.4 GHz using

[3]This corresponds to the Mini-DFT run-time options -ntg 4 and-ndiag 144.

(a) 

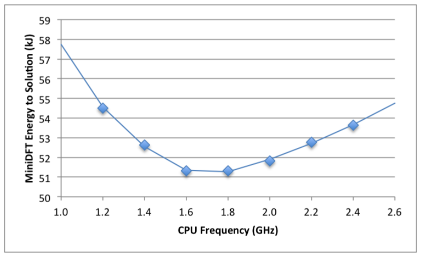
(b) 

Figure 8: Performance/Energy tradeoffs for Mini-DFT. (a) Performance vs. CPU Frequency. (b) Energy to solution vs. CPU Freqency. Lines are fits of the energy and walltime models in Equation 1 to the measured data.

HPC-Toolkit[13] and determined that 43% of the runtime is spent in the FFT phase and 40% is spent in ZGEMM calls from the linear algebra phase.

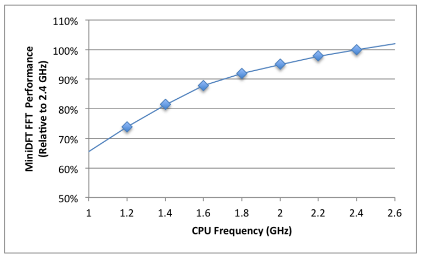
The total energy used by the MiniDFT benchmark is shown in Figure 8. The benchmark calculation is most energy efficient when run between 1.6 and 1.8 GHz. Up to 4.3% energy savings are possible in exchange for a 13% increase in wall time.

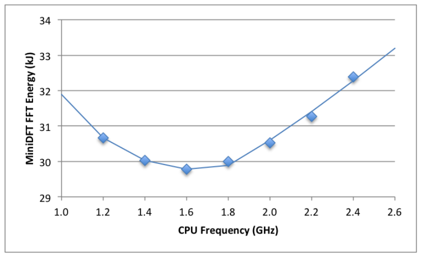
To interpret the MiniDFT energy curve vv. the microbenchmarks, MiniDFT energy usage was measured separately for the 3D-FFT routines and the remainder of the code. These results are shown in Figure 9. The energy-minimum for the 3D-FFTs moves to 1.6 GHz, (Figure 9b) slightly to the left of where it was for the full application. This coincides with the energy minimum of the STREAM benchmark, which and reflects the bandwidth sensitivity of transpose operations. The performance plot in Figure 9a curves more softly than STREAM's, likely due to additional network bandwidth constraints.

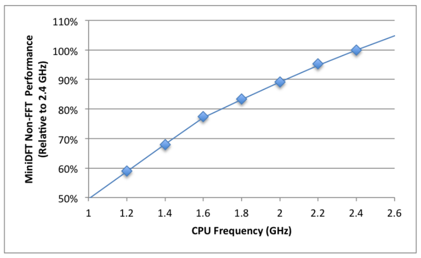
The remainder of the code, contains 70% ZGEMM and numerous other small routines. In this case the characteristics do not completely match that of ZGEMM. Distinctly different performance regimes are evident above and below 1.6 GHz. A number of streaming data calculations and large MPI reductions that are bandwidth sensitive and may be limited by the memory controller below this frequency. Above this threshold, performance continues to increase significantly and the energy shifts to 2.2 GHz, which is loosely consistent with the heavy use of ZGEMM libraries. A frequency dependent profiling study is needed to confirm this interpretation.

6 Related Work

The need for energy efficiency in both HPC and mobile computing has motivated extensive prior research focused on measuring, modeling and minimizing energy use in computing. A number of measurement tools (including PowerPack[14], Power Insight[15] and PowerMon2[16]) have been developed to physically instrument compute nodes and measure the power used by individual system

(a)


(b)


(c)


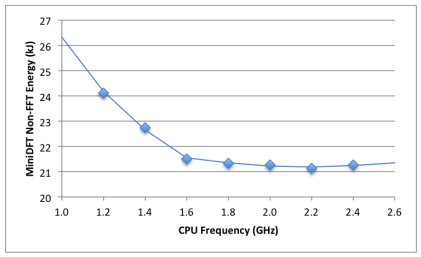
(d)


Figure 9: Energy/Performance tradeoffs for CPU Frequency scaling within MiniDFT regions. (a) FFT Performance, (b) FFT Energy to solution, (c) Non-FFT Performance. (d) Non-FFT Energy to solution, Lines are fits to the measured data.

components and aggregate and analyze these results on a central server. Other power measurement efforts have foregone additional hardware and emphasized gathering node-level power measurements in a way that can be scaled to production supercomputers[17].

Using these types of measurement tools, various researchers have explored the energy/performance tradeoffs that can be achieved by dynamic frequency and voltage scaling (DVFS)[18, 19, 14], dynamic concurrency throttling (DCT)[20], or interconnect bandwidth scaling[18]. For some applications, energy use can be substantially reduced with marginal impact on performance[19]. The relative value of energy savings and performance is subjective and metrics such as the energy-delay squared product help to formalize the evaluation of these tradeoffs[21, 22, 19].

Kestor et al. have used a carefully designed suite of benchmarks to quantify energy costs for data movement through different levels of the memory hierarchy and constructed a counter-based energy model.[5] This model is more predictive than our curve fitting approach and was validated using several HPC applications, but this approach does not provide a simple way of understanding the sensitivity of the model to parameters such as CPU frequency.

The roofline model of energy proposed by Vuduc[23] examines energy use as a function of and algorithm's operational intensity. This model is geared toward identifying balance points that would steer architecture design

Our work is quite similar to earlier DVFS studies, but adds an empirical energy model to reason about which applications are candidates for energy savings. A similar style of empirical fitting was used by De Vogeleer [24], who used their model to propose an energy/frequency convexity rule. Their study targeted mobile computing patterns, examined a different range of computational patterns, and used a different performance model.

7 Conclusion

We have measured the energy/performance trade-offs with respect to frequency scaling for three HPC applications on a Cray XC30. Our measurements for these codes have minima at 1.8 GHz (MILC), 3.6 GHz (GTC) and 1.6-1.8 GHz (Mini-DFT). By comparing the energy usage profiles to those for three micro-benchmarks and applying our knowledge of the applications we have been able to qualitatively explain the energy usage characteristics of the applications. Also, our results show that the applications exhibit much more complicated behavior than that exhibited by the micro-benchmarks, as might be expected.

Frequency scaling, as used in this paper, is not a cost effective approach to energy conservation on current systems. The largest efficiency gain was observed for Mini-DFT, which reduced it's total energy use by 4% relative to the nominal frequency of 2.4 GHz. Using reasonable estimates for energy prices ($1/MWy) and system lifetime (5 years), peak system power (50 kW/cabinet) and capital costs ($1M/cabinet), the maximum savings due to frequency reduction are $10k/cabinet. Purchasing a larger system to compensate for the 13% performance penalty would substantially more expensive (roughly $130k/cabinet).

Several indicators point to an increased value for frequency scaling on future systems. Static power is likely to decrease as manufacturers develop subthreshold technologies to reduce leakage current and enable fine-grained frequency domains to minimize the power drawn by idle CPU components. A renewed emphasis on strong scaling will decrease the amount of work per core (w_{-1} in Equation 1). On-node concurrency is growing faster than improvements in memory (or network) latency and bandwidth, which increases the frequency-independent workload (w_0). Deepening memory hierarchies will increase the number of balance points responsible for local minima on the energy/frequency profile.

In future work we plan to examine the energy usage of the network and disk I/O components of the machine as well as perform more detailed experiments to gain a deeper understanding of the relationship between instruction mix and energy usage.

Acknowledgment

This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

[1] J. Dongarra and P. Luszczek, “Top500,” in *Encyclopedia of Parallel Computing*, 2011, pp. 2055–2057.

[2] D. Azevedo, D. A. French, and E. N. Power, “Pue: A comprehensive examination of the metric,” 2012.

[3] V. M. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore, “Measuring energy and power with papi,” in *ICPP Workshops*, 2012, pp. 262–268.

[4] J. D. McCalpin, “Memory bandwidth and machine balance in current high performance computers,” *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, pp. 19–25, Dec. 1995.

[5] G. Kestor, R. Gioiosa, D. J. Kerbyson, and A. Hoisie, “Quantifying the energy cost of data movement in scientific applications,” in *IISWC*, 2013, pp. 56–65.

[6] C. Bernard, M. C. Ogilvie, T. A. DeGrand, C. E. DeTar, S. A. Gottlieb, A. Krasnitz, R. Sugar, and D. Toussaint, “Studying quarks and gluons on mimd parallel computers,” *International Journal of High Performance Computing Applications*, vol. 5, no. 4, pp. 61–70, 1991.

[7] G. Bauer, S. Gottlieb, and T. Hoefler, “Performance Modeling and Comparative Analysis of the MILC Lattice QCD Application su3 rmd,” in *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012).* IEEE Computer Society, May 2012, pp. 652–659.

[8] J. Carter, Y. He, J. Shalf, H. Shan, E. Strohmaier, and H. Wasserman, “The performance effect of multi-core on scientific applications,” *Lawrence Berkeley National Laboratory*, 2007.

[9] K. Madduri, K. Z. Ibrahim, S. Williams, E.-J. Im, S. Ethier, J. Shalf, and L. Oliker, “Gyrokinetic toroidal simulations on leading multi-and manycore hpc systems,” in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis.* ACM, 2011, p. 23.

[10] S. Ethier, W. M. Tang, R. Walkup, and L. Oliker, “Large-scale gyrokinetic particle simulation of microturbulence in magnetically confined fusion plasmas,” *IBM Journal of Research and Development*, vol. 52, no. 1.2, pp. 105–115, 2008.

[11] L. Oliker, A. Canning, J. Carter, J. Shalf, and S. Ethier, "Scientific computations on modern parallel vector systems," in *Proceedings of the 2004 ACM/IEEE conference on Supercomputing.* IEEE Computer Society, 2004, p. 10.

[12] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari, and R. M. Wentzcovitch, "Quantum espresso: a modular and open-source software project for quantum simulations of materials," *Journal of Physics: Condensed Matter*, vol. 21, no. 39, p. 395502 (19pp), 2009. [Online]. Available: http://www.quantum-espresso.org

[13] L. Adhianto, S. Banerjee, M. W. Fagan, M. Krentel, G. Marin, J. M. Mellor-Crummey, and N. R. Tallent, "Hpctoolkit: tools for performance analysis of optimized parallel programs," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 6, pp. 685–701, 2010.

[14] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. W. Cameron, "Powerpack: Energy profiling and analysis of high-performance systems and applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 5, pp. 658–671, 2010.

[15] J. H. L. III, P. Pokorny, and D. Debonis, "Powerinsight - a commodity power measurement capability," in *IGCC*, 2013, pp. 1–6.

[16] D. Bedard, R. Fowler, M. Linn, and A. Porterfield, "Powermon 2: Fine-grained, integrated power measurement," *Renaissance Computing Institute, Tech. Rep. TR-09-04*, 2009.

[17] J. H. L. III, K. T. Pedretti, S. M. Kelly, J. P. Vandyke, K. B. Ferreira, C. T. Vaughan, and M. Swan, "Topics on measuring real power usage on high performance computing platforms," in *CLUSTER*, 2009, pp. 1–8.

[18] J. H. Laros III, K. T. Pedretti, S. M. Kelly, W. Shu, and C. T. Vaughan, "Energy based performance tuning for large scale high performance computing systems," in *Proceedings of the 2012 Symposium on High Performance Computing.* Society for Computer Simulation International, 2012, p. 6.

[19] R. Ge, X. Feng, and K. W. Cameron, "Improvement of power-performance efficiency for high-end computing," in *IPDPS*, 2005.

[20] D. Li, B. R. de Supinski, M. Schulz, K. W. Cameron, and D. S. Nikolopoulos, "Hybrid mpi/openmp power-aware computing," in *IPDPS*, 2010, pp. 1–12.

[21] M. Horowitz, T. Indermaur, and R. Gonzalez, “Low-power digital design,” in *Low Power Electronics, 1994. Digest of Technical Papers., IEEE Symposium.* IEEE, 1994, pp. 8–11.

[22] D. Brooks, P. Bose, S. Schuster, H. M. Jacobson, P. Kudva, A. Buyuktosunoglu, J.-D. Wellman, V. V. Zyuban, M. Gupta, and P. W. Cook, “Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors,” *IEEE Micro*, vol. 20, no. 6, pp. 26–44, 2000.

[23] J. Choi, D. Bedard, R. J. Fowler, and R. W. Vuduc, “A roofline model of energy,” in *IPDPS*, 2013, pp. 661–672.

[24] K. D. Vogeleer, G. Memmi, P. Jouvelot, and F. Coelho, “The energy/frequency convexity rule: Modeling and experimental validation on mobile devices,” *CoRR*, vol. abs/1401.4655, 2014.