Energy Based Performance Tuning for Large Scale High Performance Computing Systems

James H. Laros III [#1], Kevin T. Pedretti [#], Suzanne M. Kelly [#], Wei Shu [&], Courtenay T. Vaughan [#]
[#] Sandia National Laboratories
[1] jhlaros@sandia.gov
[&] University of New Mexico

Keywords: High Performance Computing (HPC), Power, Energy Efficiency, Frequency Scaling

Abstract

Recognition of the importance of power in the field of High Performance Computing, whether it be as an obstacle, expense or design consideration, has never been greater and more pervasive. In response to this challenge, we exploit the unique power measurement capabilities of the Cray XT architecture to gain an understanding of the power requirements of important DOE/NNSA production scientific computing applications executing at large scale (thousands of nodes). The effect of both CPU frequency and network bandwidth scaling on power usage is characterized in a series of empirical experiments and demonstrates energy savings opportunities of up to 39% with little to no impact on run-time performance. Our results provide strong evidence that next generation large-scale platforms should not only approach CPU frequency scaling differently, but could also benefit from the ability to tune other platform components, such as the network, to achieve energy efficient performance.[1]

1. INTRODUCTION

Power has increasingly been identified as the *tall pole* in the path to Exascale. Ubiquitous in the top three considerations of virtually every report on next generation or Exascale platforms, power has been recognized across the board by government agencies and commercial enterprises alike as possibly the greatest challenge in fielding future HPC platforms. Existing hardware has been successfully leveraged by present day operating systems to conserve energy whenever possible but these approaches have proven ineffective and even detrimental at large scale. While hardware must provide part of the solution, how these solutions are leveraged on large scale platforms requires a new and flexible approach. It is particularly important that any approach taken has a system and node-level, view of these issues.

In response to this challenge we apply a, thus far, unique ability to measure current draw and voltage, in situ, to analyze the trade-offs between performance and energy efficiency of production scientific applications run at large scale (thousands of nodes) while manipulating CPU frequency and network bandwidth. Previous work has been limited to small scale experiments and has predominately used synthetic benchmarks.

Our experiments were conducted on two Cray XT class platforms; *Red Storm* located at Sandia National Laboratories and *Jaguar* hosted by Oak Ridge National Laboratory. The results of our experiments clearly indicate that opportunities exist to save energy by tuning platform components while maintaining application performance. Our goal is to reduce energy consumption of scientific applications run at very large scale while minimizing the impact on run-time performance (wall-clock execution time).

Evaluating acceptable trade-offs between energy efficiency and run-time performance is, of course, somewhat subjective. Our work indicates that the parameters of these trade-offs are application dependent. With annual power costs on track to meet or exceed acquisition costs of next generation large scale platforms, our traditional prioritization of performance above all will be forced to change. It is likely that more emphasis will be placed on energy efficiency metrics like FLOPS/Watt or Energy Delay Product (EDP). Regardless, performance remains a critical parameter of our evaluation.

The main contributions of this work are:

- We demonstrate the ability to perform fine-grained, node-level power monitoring at large scale on Cray XT compute nodes and scalably aggregate the results for analysis.
- We present, to the best of our knowledge, the first in situ empirical experiments showing the energy saving potential for important Department of Energy (DOE) production applications running at large scale (thousands of nodes) on a leadership class architecture.
- We examine the energy savings potential for both CPU frequency scaling and network bandwidth scaling. Prior studies have only examined CPU frequency scaling at small scale. Our results indicate that each of the applications studied has a sweet spot based on its computation and communication requirements.
- We show that a positive trade-off exists between en-

[1]Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

ergy efficiency and performance for many applications, which provides motivation for future exascale platforms to include the ability to tune the power-profile of individual components (e.g., CPU, network, memory) under software control.

The remainder of this paper is organized as follows: Section 2. provides an overview of the power measurement approach used in our experiments. This includes an overview of the Cray XT test systems and their unique power measurement and component tuning capabilities. Our CPU frequency and network bandwidth scaling techniques are then described in Sections 3. and 4., respectively. A brief overview of our application test suite is given in Section 5., followed in Section 6. with results and observations from running the suite using various CPU (Experiment #1) and network configurations (Experiment #2). Related work is summarized in Section 7. Finally, overall conclusions and future work are discussed in Section 8.

2. POWER MEASUREMENT APPROACH

A detailed description of our power measurement approach was previously given in Laros et al.[1]. This section provides a brief summary and additional comments specific to the data collection and analysis framework used in this study.

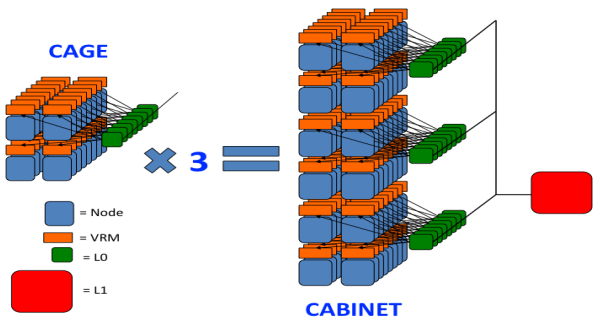


Figure 1. Cray XT Power Collection Infrastructure

Figure 1 illustrates the Reliability Availability and Serviceability (RAS) infrastructure of the Cray XT architecture leveraged for all of the power collection in these experiments. The lines connecting all components in the figure are part of an out-of-band RAS network hierarchy. The blue boxes indicate nodes; there are four nodes per board. The orange boxes represent the voltage regulator modules (VRM). In addition to providing power to each node, the VRMs provide the current and voltage readings when queried by the on board embedded controller (the L0, depicted in green). The L0 on each board collects the current and voltage readings and passes the information to the cabinet level embedded processor (the L1, depicted in red). Four nodes and an L0 make up a board, eight boards combine to form a cage and three cages comprise a cabinet. This represents the power collection hierarchy for a single Cray XT cabinet. We employ many cabinets when collecting data for these experiments. Ultimately, the data is consolidated at the system management workstation (SMW), the top-level node in the RAS hierarchy.

In all of the following experiments, current and voltage measurements are collected, simultaneously, from 15 cabinets (1440 nodes), more specifically each node's VRM, at a frequency of one sample per second over the duration of the entire test period to avoid start up and tear down overhead of the collection process. Since we used a range of applications executed at a range of scales, we targeted cabinets in a distributed manner throughout the platform to achieve consistent collection coverage for the applications tested. For example, one of the test applications run on 4096 nodes intersected with 960 of the collection nodes (23% coverage). Similarly, a separate application run on 1536 nodes intersected with 480 of the collection nodes (31% coverage). The number of nodes sampled was not limited by the scalability of the collection mechanism but by the available test time on these large scale platforms. At the scales tested we have seen no indication of scalability issues with the collection mechanism.

Our post processing begins with ensuring all data samples used are from nodes involved in the application run. We then synchronize the data samples with the application execution start and finish timestamps. For each application execution the data is processed and statistically analyzed. For each experiment we calculate the standard deviation, median and mean but rely on the coefficient of variation (CV) to ensure the measurements are dependable since the CV is expressed as a percentage, independent of magnitude. For the purposes of this analysis we focus on the differences between complete data samples or deltas. We have found in this and previous experiments that the deltas provide a reliable foundation for comparison.

2.1. Test Platforms

All experiments were conducted on either Red Storm[2] or Jaguar[3]. Both are variants of the Cray XT architecture. To our knowledge, this is the only platform that exposes the ability to measure current draw, in situ, as described in [1]. The Cray XT architecture contains commodity AMD processors that allow for CPU frequency scaling. These features were leveraged using specific operating systems modifications to the Catamount light-weight kernel[2]. This architecture also affords the ability to tune performance parameters of other components such as network injection rate and network bandwidth.

[2]Red Storm was the first instance of the Cray XT architecture line, and was jointly developed by Cray Inc. and Sandia National Laboratories.

[3]Jaguar is located at Oak Ridge Leadership Computing Facility (OLCF).

Red Storm is currently a heterogeneous architecture containing both dual and quad-core processors. There are 3,360 AMD 64 bit 2.4 GHz dual-core processors (nodes) with 4GB of DDR2 memory on Red Storm. The dual-core nodes were used for the network bandwidth experiments. Additionally, there are 6,240 AMD 2.2 GHz quad-core processors (nodes) with 8GB of DDR2 memory. Jaguar, is a homogeneous architecture comprised entirely of quad-core processors. There are 7,832 2.1GHz quad-core AMD Opteron processors (nodes) with 8GB of DDR2 memory. The CPU frequency scaling studies were conducted on the quad-core processors of both Red Storm and Jaguar. Nodes on both Red Storm and Jaguar are connected via a Seastar 2.1 network interface controller/router (Seastar NIC). The network topology of Red Storm is a modified mesh (mesh in X and Y directions, torus in the Z direction). Jaguar's network topology is a 3D torus. For the purposes of these experiments the network topology differences are not significant. Some applications used in both experiments are export controlled and could not be run on Jaguar. We maximized the used of each platform accordingly.

3. CPU FREQUENCY SCALING

Typical approaches to CPU frequency scaling employed by operating systems, such as Linux, while efficient for single server or laptop implementations, have proven to be detrimental when used at scale causing the equivalent of operating system jitter[3]. For this reason, it is common practice at most sites that deploy medium to large scale clusters to disable frequency scaling. It is clear to us that techniques designed for laptop energy efficiency (solely a node-level approach) are not directly applicable to large scale HPC platforms. We take a more deterministic, full system, approach ensuring all cores participating in an application are executing at the target frequency in lock step.

3.1. Operating System Modifications

To accomplish our goals, we made a small number of targeted modifications to Catamount. We first interrogate the chip architecture capabilities to determine if advanced power management (APM) is supported. Specifically, we are interested in whether hardware P-state and Dynamic Voltage control is available. Changing P-states requires writing to P-state related Memory Status Registers (MSR). If APM is not supported, writing to P-state MSRs will cause the node to fail. Even if APM is enabled, however, only a single P-state is required to be defined. In addition, even if multiple P-states (up to 5) are defined, they may have identical definitions. This is typically not the case but enforces the importance of closely interrogating specific hardware capabilities. From this point forward we assume APM is supported and multiple P-states are defined, at least two of which define different operating frequencies.

Currently, our method of frequency scaling is limited to frequencies defined in the P-state table, although most processors support frequency stepping in 100MHz increments. The impetus of changing P-state (changing P-state changes frequency) is ultimately to lower the input voltage to the processor. Power is proportional to the frequency, capacitance and voltage squared. Consequently, the largest impact on power can be obtained by lowering input voltage. Both the processor and the infrastructure must support dynamic voltage transitions for us to take advantage of this potential power savings. On the platform used in our experiments, all cores are required to be in the same or *higher*[4] P-state before a lower input voltage takes effect. Basically, if one core is operating at a higher frequency (which requires a higher input voltage) the input voltage to the processor remains at the voltage necessary to support the highest active frequency.

At a very early stage in the boot process we collect the default and supported P-states of each core. This information is stored and used by a trap function added to handle a variety of P-state related functionality. Since changing P-state is a privileged operation (writing to MSRs) the ability to change P-states was added in two parts; an operating system trap and a user level library interface. The trap implements query functionality to determine what P-states are available, what P-state the core is presently in and of course the ability to transition from the current P-state to an alternate supported P-state. The trap also reports the final P-state achieved and in debug mode the number of nanoseconds the P-state transition took. The amount of time necessary to transition between P-states is not important for the experiments covered in this paper since we accomplish a single static change prior to application execution. Transition time (effectively overhead) becomes a critical consideration when more dynamic methods of CPU frequency scaling are employed.

Table 1 lists the supported P-states, corresponding CPU frequencies and input voltages of both Red Storm and Jaguar. The default (baseline) P-states and frequencies differ between Red Storm and Jaguar (P-state 0 at 2.2 GHz on Red Storm, P-state 1 at 2.1 GHz on Jaguar). Testing was conducted using P-states 0, 2, 3 and 4 on Red Storm, and P-states 1, 2, 3 and 4 on Jaguar. While input voltages were consistent on Red Storm we observed two different input voltages for each P-state on Jaguar. The input voltage reported in Table 1 is the voltage observed for the majority of the nodes. This observation had no affect on our analysis since we use voltage measurements taken from each node individually for our analysis.

[4] Additional detailed information specific to the AMD architecture family discussed here can be found in the AMD BIOS and Kernel Developers guide (BKDG).

Table 1. Test Platform P-state Information

P-state	CPU frequency		Input Voltage	
	Red Storm	Jaguar	Red Storm	Jaguar
0	2.2 GHz	2.1 GHz	1.200 V	1.200 V
1	2.0 GHz	2.1 GHz	1.200 V	1.200 V
2	1.7 GHz	1.7 GHz	1.150 V	1.150 V
3	1.4 GHz	1.4 GHz	1.075 V	1.075 V
4	1.1 GHz	1.1 GHz	1.050 V	1.050 V

3.2. Library Interface

Since changing P-states (changing current operating frequency COF) is a privileged operation, the trap is accessed through a variety of functions provided by a user level library. While a single trap function implements all of the functionality, for ease of use and clarity we have implemented a library function interface to exploit each capability separately.

- **cpu_pstates(void)** - Returns detailed processor P-state information
- **cpu_freq_step(P-state)** - Requests a P-state transition (up or down)
- **cpu_freq_default(void)** - Returns default processor P-state

In this experiment, we quantify the affects of *static* CPU frequency modification. Prior to executing the test application we first execute a *control* application. The control application simply changes the CPU's COF by changing the CPU's P-state to the desired level (cpu_freq_step(P-state)). The control application is launched on every core of each node that will be used in the test application. Note, while convenient for our testing, a separate control application would not be required in a production environment. P-state changes are accessible from any portion of the software stack using this library interface. Following execution of the control application we execute the HPC application under test on the same nodes. The HPC application will run at a lower frequency defined by the P-state selected by the control program. During the execution of the HPC application we collect data for both current draw and voltage at one second intervals. Once the HPC application is completed we return all nodes to the default P-state or select a new P-state for a subsequent experiment with cpu_freq_step(P-state).

The trap and library interface was designed to support both static and dynamic frequency scaling. Initially, we anticipated that it would be necessary to change frequency often during application execution to achieve an acceptable trade-off between performance and energy. In testing our modifications, we discovered what we have reported in this paper, that large benefits exist for static frequency scaling. Static frequency scaling has many benefits including simplicity and stability.

4. NETWORK BANDWIDTH SCALING

The goal of this experiment was to determine the affect on run-time performance and energy of production scientific applications run at very large scale while tuning the network bandwidth of an otherwise balanced platform[4]. To accomplish network bandwidth scaling we employed two different tunable characteristics of the Cray XT architecture. First, we tuned the Seastar NIC to reduce the *interconnect* bandwidth in stages to $1/2$ and $1/4^{th}$ of full bandwidth. Next, we used the ability to tune the node *injection* bandwidth, effectively reducing the network bandwidth to $1/8^{th}$. This allowed for the most complete stepwise reduction in overall network bandwidth we were able to achieve using this architecture (see Figure 2).

Modifying the network interconnect bandwidth on the Cray XT requires a fairly simple change to the router configuration file consulted during the routing process of the boot sequence. A full system reboot is required for every alteration of the interconnect bandwidth. Typically, all four rails of the Seastar are enabled. Alternatively, the number of enabled rails can be reduced by specifying a bitmask in the system's router configuration file (e.g., 1111 for four rails, 0011 for two rails). In our experiments, we configured the interconnect bandwidth of the Seastar to effectively tune the network bandwidth to full, 1/2 and $1/4^{th}$.

Since the interconnect bandwidth on the XT architecture is far greater than the injection bandwidth of an individual node, the interconnect bandwidth had to be reduced to $1/2$ before it produced a measurable effect. Multiple nodes may route through an individual Seastar depending on communication patterns and node placement relative to logical network topology. For this reason, we limited our experiments to one application executing at a time. This allowed for the nearest estimation of the impact of network bandwidth tuning on an individual application.

Tuning the node injection bandwidth, to further reduce the network bandwidth, requires a small modification to the Cray XT bootstrap source code. The portion of the code that required modification (*coldstart*) serves an equivalent purpose to the BIOS on a personal computer or server. Early in the power-on sequence, coldstart initializes the HyperTransport link that connects each node to its dedicated SeaStar network interface. The speed of this link is determined by its operating frequency (S) and width in bits (B) as follows: $\mathbf{S}MHz \times 2bits/clock/link \times \mathbf{B}bits/link \times 1Byte/8bits = \mathbf{BW}$.

In normal operation, the injection bandwidth is determined by the maximum negotiated rate between the node and the Seastar. A reboot is required to configure the injection bandwidth to the desired setting. Normally the links operate at S = 800 MHz and utilize the full B = 16 bits of each link resulting in an injection bandwidth of 3.2GB/sec. To achieve $1/8^{th}$ injection bandwidth we configure each link to run at S = 200 MHz with an B = 8 bit per link width reducing the injection bandwidth of each node to 400MB/sec. We selected this injection bandwidth rate since it further reduced the overall network bandwidth beyond what was possible by reducing the interconnect bandwidth of the Seastar.

Figure 2 depicts the four network bandwidth rates as mea-

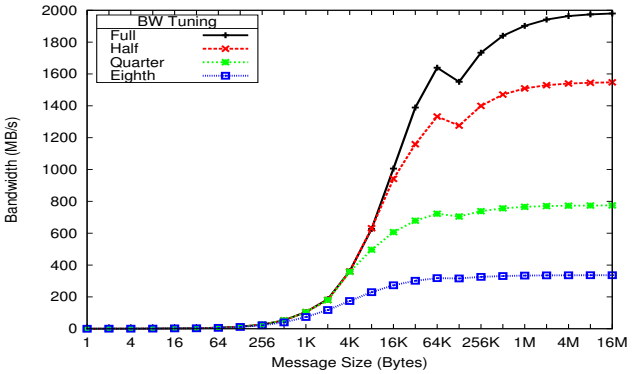


Figure 2. Pallas PingPong bandwidth for all levels of network bandwidth tuning

sured by Pallas (IMB)[5] PingPong between two dual core nodes, one core per node. The maximum bandwidth observed at $1/2$ using these benchmarks is not $1/2$ of full network bandwidth, due to interconnect bandwidth being greater than injection bandwidth on our test platforms. Injection bandwidth was not altered other than to achieve the $1/8^{th}$ bandwidth configuration. Below $1/2$ bandwidth, the steps become regular as seen in Figure 2. Using the configuration techniques available this is the best approximation of a step-wise reduction in network bandwidth that could be achieved on this platform. Experiments were conducted in phases beginning with a baseline full bandwidth run for each application followed by subsequent executions at each reduced bandwidth. For each phase power samples (current draw and voltage) were collected as described in Section 2.

5. APPLICATIONS

The applications used in our experiments were selected based on their importance to the three DOE National Nuclear Security Administration (NNSA) nuclear weapons laboratories (Sandia, Los Alamos and Lawrence Livermore). As part of the procurement of Cielo, (DOE/NNSA's most recent HPC capability platform (2010)) each laboratory in the Tri-Lab complex specified two production scientific computing applications that would be used in the acceptance phase of the procurement of Cielo. These applications are herein referred to as the *6X* applications (due to the requirement they, on average, must perform six times faster on Cielo, not that there are six applications). The 6X applications include; SAGE, CTH, AMG2006, xNOBEL, UMT and Charon. In addition to the 6X applications we used LAMMPS, another production DOE application and two synthetic benchmarks, HPL and Pallas. The following are brief descriptions of the applications along with citations for additional information.

SAGE[6] SAIC's Adaptive Grid Eulerian hydro-code, is a multidimensional, multi-material Eulerian hydrodynamics code. **CTH**[7] is a multi-material, large deformation, strong shock wave, solid mechanics code developed at Sandia National Laboratories. **AMG2006**[8], developed at Lawrence Livermore National Laboratory, is a parallel algebraic multigrid solver for linear systems arising from problems on unstructured grids. **xNOBEL**[9], developed at Los Alamos Laboratories, is a one, two, or three dimensional multi-material Eulerian hydrodynamics code used for solving a variety of high deformation flow of materials problems. **UMT**[10] is a 3D, deterministic, multigroup, photon transport code for unstructured meshes. **Charon**[11], developed at Sandia National Laboratories, is a semiconductor device simulation code. **LAMMPS**[12] is a classical molecular dynamics code, and an acronym for Large scale Atomic/Molecular Massively Parallel Simulator. High Performance Linpack **(HPL)**[13] is the third benchmark in the Linpack Benchmark Report, used as the benchmark for the bi-annual Top500 report.The HPL benchmark is well understood and recognized as a compute intensive application. **Pallas**[5], now called the Intel MPI Benchmark (IMB), successor to Pallas GmbH, is a suite of benchmarks designed to measure the performance of a wide range of important MPI routines. Pallas is communication intensive.

6. RESULTS

In this section, we will individually discuss the results of both the CPU frequency scaling experiment and the network bandwidth tuning experiment. Increases in run-time or energy percentage in Tables 2 and 3 are indicated by positive numbers, negative values are indicated by parenthesized numbers (all relative to the baseline values listed). In all cases, the same set of nodes was used for each individual application execution.

6.1. Experiment #1: CPU Frequency Scaling Results

The frequency scaling experiments were conducted during five dedicated systems times. Four were conducted on Jaguar during eight to twelve hour sessions between March 2010 and December 2010. The final experiments were conducted on Red Storm in March of 2011 during a three day dedicated system time. Over this period we conducted a range of experiments using real production scientific applications and synthetic benchmarks listed and described in Section 5. As can be seen in Table 2 we were able to test some applications in all available P-states. Others exhibited clear results in early testing and did not warrant further experiments and in some cases we were simply unable to obtain results at higher P-states (lower frequencies) due to hardware issues.

Decreasing CPU frequency, in general, slows computation. If applications were solely gated by computation this approach would be entirely detrimental. However, applications exhibit a range of characteristics. In this experiment,

Table 2. Experiment #1 CPU Frequency Scaling: Run-time and CPU Energy %Difference vs. Baseline

		Baseline Frequency		P-2 - 1.7 GHz %Diff		P-3 - 1.4 GHz %Diff		P-4 - 1.1 GHz %Diff	
	Nodes/Cores	Run-time (s)	Energy (J)	Run-time	Energy	Run-time	Energy	Run-time	Energy
HPL	6000/24000	1571	4.49×10^8	21.1	(26.4)				
Pallas	1024/1024	6816	1.72×10^8	2.30	(43.6)				
AMG2006	1536/6144	174	9.49×10^6	7.47	(32.0)	18.4	(57.1)	39.1	(78.0)
LAMMPS	4096/16384	172	2.79×10^7	16.3	(22.9)	36.0	(48.4)	69.8	(72.2)
SAGE	4096/16384	249	4.85×10^7	0.402	(39.5)				
	1024/4096	337	1.51×10^7	3.86	(38.9)	7.72	(49.9)		
CTH	4096/16384	1753	3.60×10^8	14.4	(28.2)	29.0	(38.9)		
xNOBEL	1536/6144	542	4.96×10^7	6.09	(35.5)	11.8	(50.3)		
UMT	4096/16384	1831	3.48×10^8	18.0	(26.5)				
Charon	1024/4096	879	4.47×10^7	19.1	(27.8)				

we altered CPU frequency and measured the impact on CPU energy and run-time (other platform parameters are left unchanged). We begin our analysis with a discussion of the extremes represented by two synthetic benchmarks, HPL and Pallas. Note, for all experiments we contrast both run-time and CPU energy to the baseline runs conducted at P-states 0 or 1 (depending on the platform used) and report the contrast as percent difference. For all runs we record the execution time in seconds (s) and the energy used in Joules (J).

The CPU frequency experiments focus on the effect that CPU frequency modifications had on CPU energy alone. CPU energy is the single largest contributor to total node energy as observed in [14]. On the test platforms, CPU energy ranges from 44-57% of total node power. For this reason we feel measuring CPU energy in isolation is important. Further, the results of this experiment have the potential to be more widely applicable. In a later section, total system is energy analyzed which considers the contribution of additional node components. Both approaches have utility and provide interesting insights.

HPL is a compute intensive application chosen to demonstrate a high impact resulting from CPU frequency reduction. The HPL results were consistent with expectations. In Table 2 we see a change to P-state 2 causes a 21.1% increase in run-time and a 26.4% decrease in energy used. This performance impact would likely be unacceptable for a real application unless the priority was energy savings.

In contrast to HPL, Pallas (IMB) is a communication intensive benchmark. Pallas was chosen to demonstrate an application that should be less affected by reductions in CPU frequency. Again, as expected, Pallas demonstrates only a 2.30% increase in run-time and a 43.6% reduction in energy when run in P-state 2. This would likely be a favorable trade-off. Given the results from these synthetic benchmarks we expect our real applications will fall somewhere between these extremes.

AMG2006 was executed at P-states 1-4 at a scale of 6K cores. At P-state 2 an increase in run-time of 7.47% was observed, accompanied by an energy savings of 32.0%. The trade-off at P-state 3 is not as clearly positive. The run-time impact increases more than the energy is reduced in P-states 3 and 4.

LAMMPS (tested at 16K cores) does not display a clear win when run at lower frequencies. Results for P-state 2 show a 16.3% increase in run-time and a 22.9% decrease in energy. The results for P-states 3 and 4 demonstrate even larger run-time impacts. Note, however, energy savings as a percentage are larger than run-time impacts.

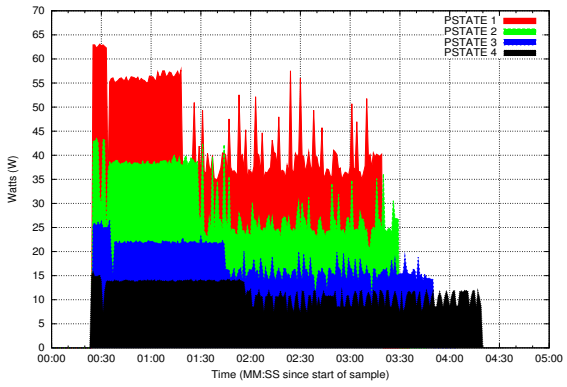
Figures 3a and 3b graphically depict four separate executions of AMG2006 and LAMMPS at P-states 1-4. The shaded area under each curve represents the energy used over the duration of the application. Figure 3a depicts the positive run-time vs. energy trade-off for AMG2006 indicated in Table 2, between P-states 1 and 2. In contrast, more dramatic increases in run-time versus area under the curve can be seen in figure 3b for LAMMPS. These graphs (application signatures) are very useful in identifying computationally intense phases of applications.

Results for SAGE were obtained at two different scales (4K and 16K cores). In both cases, a small increase in run-time (0.402% at larger scale and 3.86% at smaller scale) is observed accompanied by a very significant reduction in energy (39.5% at large scale and 38.9% at small scale). We were able to obtain results for a 4k core run of SAGE at P-state 3. The impact on run-time remains low while additional energy savings were recorded.

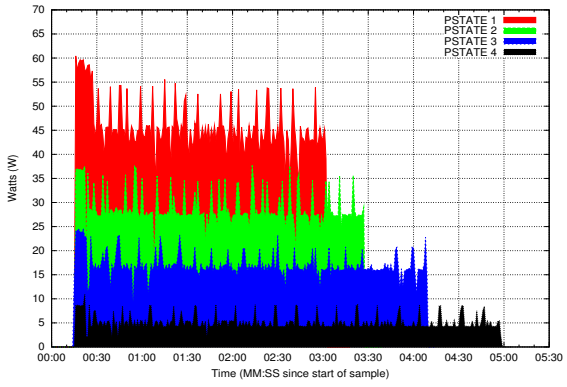
CTH was executed at P-states 0, 2 and 3 at a scale of 16K cores. Similar to LAMMPS, no clear win is observed when the CPU frequency is lowered.

We obtained results for xNOBEL at 6K cores at P-states 0, 2 and 3. Our results indicate that xNOBEL, like AMG2006 and SAGE, are good candidates for CPU frequency reduction.

UMT and Charon behaved in a very similar manner. Since UMT was run at a much larger scale than Charon (16K cores vs. 4K cores) we feel the results obtained for UMT are more meaningful and more accurately represent what we could ex-



(a) AMG



(b) LAMMPS

Figure 3. Application Energy Signatures of AMG2006 and LAMMPS run at P-states 1-4

pect at large scale. Charon may act differently when run at larger scale but these results indicate that both UMT and Charon are sensitive to CPU frequency changes.

The CPU is only one component that affects application performance. In the following section we will experiment with tuning network bandwidth and observe the trade-offs between performance vs. total system energy.

6.2. Experiment #2: Network Bandwidth Scaling Results

Total energy in Table 3 includes the measured energy from the nodes CPU, a measured energy from the Seastar and an estimated energy from the memory subsystem. Node CPU energy is calculated by totaling the energy used by all CPUs measured for each experiment divided by the number of nodes measured to produce the average energy used by each node (E_{cpu}). Current draw of the Seastar, measured from the VRM supporting the entire mezzanine (Figure 1), is constant since the SerDes do not throttle up and down based on network traffic or demand. There are four Seastars in a mezzanine, therefore, we multiply the current reading by the input voltage and divide by four to produce the baseline Seastar energy value ($E_{network}$). For $1/2$, $1/4^{th}$ and $1/8^{th}$ network bandwidth calculations we assume a linear reduction in power. To avoid other components having a disproportionate affect on the total calculation, we use an estimated per node memory energy value (20W) for each experiment (E_{memory}). The calculation is as follows (where **E** = Energy): $(\mathbf{E}_{cpu} + \mathbf{E}_{network} + \mathbf{E}_{memory}) \times$ number of nodes = **Total Energy**.

In all calculations 25 W is used for the full network bandwidth value, 12.5 W for $1/2$, 6.25 W for $1/4^{th}$ and 3.125 W for $1/8^{th}$ network bandwidth.

Addressing each application in table order (see Table 3) we see SAGE displays similar characteristics for both input problems tested. Reducing the network bandwidth by $1/2$ has little affect on the run-time while a significant savings in energy is experienced. The impact on run-time is larger when the network bandwidth is reduced to $1/4^{th}$ with little additional energy savings. At $1/8^{th}$ network bandwidth SAGE, for both input problems, experiences significant impacts on run-time accompanied by smaller energy savings. Based on this data, reducing network bandwidth by $1/2$ would be advantageous, if we could reduce the corresponding energy consumption of the network by half.

CTH was affected more by changes in the network bandwidth than any other application we tested. Even at $1/2$ bandwidth, CTH experiences a greater percent increase in run-time (9.81%) than is saved by reducing network energy (7.09% decrease). At $1/4^{th}$ bandwidth, CTH experiences a very large increase in run-time (30.2%) accompanied by an actual increase in energy used of 1.04%. Clearly, reducing network bandwidth further is highly detrimental to both run-time and energy as can be seen from the $1/8^{th}$ network bandwidth results. Even at this moderately large scale CTH requires a high performance network to execute efficiently.

AMG2006 and xNOBEL are insensitive to the network bandwidth changes in terms of run-time, but demonstrate large energy savings opportunities. Reductions down to $1/8^{th}$ network bandwidth cause virtually no impact in run-time for both AMG2006 and xNOBEL while a 25.9% savings in energy can be achieved for both. We do note the savings in energy seems to be flattening by the time we reduce network bandwidth to $1/8^{th}$.

UMT produced similar results to AMG2006 and xNOBEL when the network bandwidth was reduced up to $1/4^{th}$, little to no impact in run-time accompanied by a large energy savings. At $1/8^{th}$ network bandwidth we see different characteristics. UMT experiences a much higher impact on run-time at $1/8^{th}$ network bandwidth (6.32%) than at $1/4^{th}$ (1.07%) with virtually no additional energy savings (21.7% at $1/4^{th}$ and 21.8% at $1/8^{th}$). We seem to have found the limit of network bandwidth tuning that should be applied to UMT at least at this scale. We should note that UMT was run at a smaller scale relative to the other applications. It is possible that at larger

Table 3. Experiment #2 Network Bandwidth: Run-time and Total Energy %Difference vs. Baseline

		Baseline Bandwidth (BW)		1/2 BW %Diff		$1/4^{th}$ BW %Diff		$1/8^{th}$ BW %Diff	
	Nodes/Cores	Run-time (s)	Energy (J)	Run-time	Energy	Run-time	Energy	Run-time	Energy
SAGE_prob1	2048/4096	337	5.79×10^7	(0.593)	(15.3)	8.90	(15.5)	20.2	(11.4)
SAGE_prob2	2048/4096	328	5.64×10^7	0.609	(14.3)	8.23	(15.8)	22.6	(9.63)
CTH	2048/4096	1519	2.58×10^8	9.81	(7.09)	30.2	1.04	40.4	3.50
AMG2006	2048/4096	859	1.45×10^7	(0.815)	(15.8)	(0.116)	(22.7)	0.931	(25.9)
xNOBEL	1536/3072	533	7.01×10^7	(0.938)	(15.4)	(0.375)	(22.2)	(0.375)	(25.9)
UMT	512/1024	838	3.57×10^7	0.357	(14.7)	1.07	(21.7)	6.32	(21.8)
Charon	1024/2048	1162	9.96×10^7	1.55	(13.7)	2.15	(20.8)	2.67	(24.5)

scale our results would differ.

Charon showed small, but increasing, impact on run-time as we reduced network bandwidth. At this scale it is clear that the network bandwidth could be reduced to $1/4^{th}$ with an acceptable impact in run-time (increase of 2.15%) accompanied by a very significant savings in energy (decrease of 20.8%). Moving from $1/4^{th}$ to $1/8^{th}$ network bandwidth shows indications that the energy savings is flattening but results are not conclusive. Experiments with Charon at larger scale are also warranted.

Excluding CTH, virtually no impact to run-time would be experienced by tuning the network bandwidth to $1/2$ (for the applications tested). The result would be significant energy savings with little to no performance impact. In the case of AMG2006, xNOBEL and UMT the network bandwidth could be reduced to $1/4^{th}$ full bandwidth with little run-time impact, allowing for even larger energy savings. Our observations indicate that a tunable network would be beneficial but they also indicate a high performance network is critical for some applications. The ability to tune the network, similar to how frequency is tunable on a CPU, would be an important characteristic on next generation exascale platforms.

It should be stressed that our data is representative of a single application running at a time. One of the reasons the interconnect bandwidth of the Seastar was designed to be greater than the injection bandwidth of a single node is that the network is a shared resource on the Cray XT architecure, shared by all simultaneously running applications. Often many hops are required for a messages to travel from source to destination, and poor node mappings result in individual network links carrying messages for multiple applications. Having a greater interconnect bandwidth is essential for handling this increased load. Thus, the ability to tune network performance could not be exploited without considering the possible impact on other applications running on the platform, at least for network topologies like meshes and 3D-toruses. Network topologies with fewer hops on average (high radix networks) could benefit more easily from a tunable network.

7. RELATED WORK

Power, as it relates to computers and computation, has been researched from many perspectives. Possibly the largest body of work has been done by Ge, Feng and Cameron et al. The authors use a framework called PowerPack[14][15] to profile and analyze power and run-time effects. Component level measurements are taken on a single node and a technique they call *node remapping* is used to emulate larger scale runs. In [16], the authors expand their collection capability to 16 nodes. The NEMO power aware cluster is comprised of laptops which allow the necessary measurements to be taken using ACPI. The authors provide a thorough evaluation of three different scheduling strategies for Dynamic Voltage Scaling (DVS). Work towards developing a fused metric for energy efficiency is also presented in [16]. Energy Delay Product (EDP), initially proposed by Horowitz [17] and extended to more heavily weight delay by Brooks [18], the authors propose a more dynamic weighted factor method. Finally, in [19] the authors continue their analysis of parallel processing inefficiencies to achieve savings in power with little performance impact. While our motivations and the importance we put on high-frequency component level measurement is clearly shared, our work is focused on production scientific applications executing at large scales. Extending the environment the authors have leveraged is not practical at the node counts we are interested in.

Other researchers have used various modeling and emulation techniques based on node-level or small-scale measurements. In [20], the authors evaluate methods of measuring power using synthetic benchmarks with line meters for single node tests to cabinet level collection on the Cray XT architecture. They provide some modeling analysis and extrapolate to full system scale using their coarse cabinet level collections. Li et al. in [21] model hybrid MPI/OpenMP from a performance and energy perspective examining both dynamic concurrency throttling (DCT) and DVS. In [22] Li uses modeling to investigate task aggregation to reduce energy consumption by reducing the number of nodes. Li uses AMG along with some NPB benchmarks, one of the few efforts that use a real HPC applications for their analysis. The use of performance counters to estimate power efficiency has been researched

from a micro [23][24] and macro [25] perspective.

Kodi et al. [26] discuss the ability to tune network bandwidth using techniques similar to DVS (but for network components) to dynamically reconfigure optical interconnects with the goal of increasing energy efficiency. We found work proposing DVS on network links as early as 2003 (see Shang et al.[27]) but we are not aware of any HPC platform that currently has this capability.

Probably the greatest difference, and contribution, of our work is the sheer scale of our experiments involving a large set of production HPC scientific applications. The scale of our in-situ data measurement techniques is unequaled. Our work is clearly focused on empirical analysis and removes the extrapolation from small-scale to large-scale that other studies have used. Our research involving tuning the network bandwidth and evaluating the impact with these same metrics currently has little related research to compare.

8. CONCLUSIONS AND FUTURE WORK

Our initial approach to this topic assumed that a more dynamic approach to tuning CPU and other platform components would be necessary to achieve a positive trade-off between performance and energy (which is why we designed our interface as described in Section 3.1.). On the path to this goal we found that static changes can produce significant energy savings without sacrificing performance. While we have found this to be the case, we feel there is more work to be accomplished and hopefully more efficiency to be found.

Static tuning has many advantages including stability. Dynamic tuning, at scale, has the potential to be a difficult issue to manage. If not done properly, it could introduce reliability issues. Dynamic frequency scaling, of any component, also requires consideration of how long it takes to accomplish the desired frequency changes. If transitions are too frequent the resulting overhead could negate any potential gain. Regardless, this is an important area of investigation that we are currently pursuing.

Our current efforts are targeting applications that exhibit clearly discernible compute and communication phases. For example, AMG2006 in Figure 3a demonstrates a heavy computation phase early in application execution. If the CPU frequency was kept high during this phase and transitioned to a lower frequency in the latter part of execution would an even more favorable trade-off between run-time and energy be observed? In both Figure 3a and 3b , there appears to be regular spikes in computation. If the CPU frequency could be altered to coincide with these phases could we achieve even better results for AMG2006? Would the results for LAMMPS be positive? We are working to answer these questions for a wide range of applications.

An important component that we have not experimented with is memory, due to our inability to measure this component in isolation, at present. The performance of many scientific applications is bounded by memory bandwidth. For this reason it is important to measure energy use of this component as part of an overall analysis.

Finally, we are investigating the requirements and implementation of a full-system holistic approach to this challenge. There are many issues that must be addressed: hardware sensors, scalable out of band collection and application programming interfaces between operating system, collection mechanisms and the application itself, and others. A set of interfaces and well-defined interactions are required to complete the feedback loop for advanced power management on future platforms. Once created we anticipate that applications could direct hardware, via the interfaces mentioned, when more or less performance is required from specific components to reach maximum efficiency.

ACKNOWLEDGMENTS

We would like to acknowledge the INCITE program for granting us dedicated system time on Jaguar and Robert Ballance and John Noe for securing multiple dedicated test times on Red Storm for these experiments. Dedicated system time, especially for intrusive experiments such as these, require heroic support from the systems administration staff. We acknowledge the wonderful support of Don Maxwell at Oak Ridge and Dick Dimock, Barry Oliphant, Jason Repik and Victor Kuhns at Sandia. Management support from James Ang and Ron Brightwell has also been invaluable. This work was largely funded by the Advanced Simulation and Computing (ASC) program of the NNSA.

REFERENCES

[1] J. H. Laros 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 *IEEE Cluster 2009, International Conference on Cluster Computing*. Sandia National Laboratories, September 2009.

[2] S. M. Kelly and R. B. Brightwell, "Software Architecture of the Light Weight Kernel, Catamount," in *Cray User Group*. CUG, 2005.

[3] F. Petrini, D. Kerbyson, and S. Pakin, "The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage, and Analysis (SC)*. ACM/IEEE, 2003.

[4] R. Brightwell, K. Predretti, K. Underwood, and T. Hudson, "SeaStar Interconnect: Balanced Bandwidth for Scalable Performance," *IEEE Micro*, vol. 26, no. 3, pp. 41–57, 2006.

[5] "PALLAS," Intel. [Online]. Available: http://www.intel.com/cd/software/products/asmo-na/eng/cluster/mpi/219848.htm

[6] R. Weaver and M. Gittings, " Massively Parallel Simulations with DOEś ASCI Supercomputers: An Overview of the Los Alamos Crestone Project ," in *Adaptive Mesh Refinement - Theory and Applications*. Springer Berlin Heidelberg, 2005.

[7] E. S. Hertel, Jr., R. L. Bell, M. G. Elrick, A. V. Farnsworth, G. I. Kerley, J. M. Mcglaun, S. V. Petney, S. A. Silling, P. A. Taylor, and L. Yarrington, "CTH: A Software Family for Multi-Dimensional Shock Physics Analysis," in *Proceedings of the International Symposium on Shock Waves*. NTIS, 1993.

[8] R. D. Falgout, P. S. Vassilevski, Panayot, and S. Vassilevski, "On Generalizing the AMG Framework," in *Society for Industrial and Applied Mathmatics: Journal on Numererical Analysis*. SIAM, 2003.

[9] M. Gittings, R. Weaver, M. Clover, T. Betlach, N. Byrne, R. Coker, E. Dendy, R. Hueckstaedt, K. New, W. R. Oakes, D. Ranta, and R. Stefan, "The RAGE Radiation-Hydrodynamic Code," *Journal of Computational Science & Discovery*, vol. 1, no. 1, p. 015005, 2008.

[10] "UMT2K," Lawrence Livermore National Laboratory. [Online]. Available: https://asc.llnl.gov/computing_resources/purple/arch-ive/benchmarks/umt/umt1.2.readme.html

[11] P. T. Lin, J. N. Shadid, M. Sala, R. S. Tuminaro, G. L. Hennigan, and R. J. Hoekstra, "Performance of a parallel algebraic multilevel preconditioner for stabilized finite element semiconductor device modeling," *Journal of Computational Physics*, vol. 228, pp. 6250–6267, 2009.

[12] S. Plimpton, "Fast Parallel Algorithms for Short-Range Molecular Dynamics," *Journal of Computational Physics*, vol. 117, pp. 1–19, 1995.

[13] J. Dongarra, J. Bunch, C. Moler, and G. W. Stewart, "High Performance Linpack HPL," in *Technical Report CS-89-85*. University of Tennessee, 1989.

[14] X. Feng, R. Ge, and K. W. Cameron, "Power and Energy Profiling on Scientific Applications on Distributed Systems," in *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2005.

[15] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. Cameron, "PowerPack: Energy Profiling and Analysis of High-Performance Systems and Applications," *Transactions on Parallel and Distributed Systems*, vol. 21, no. 5, pp. 658–671, 2010.

[16] R. Ge, X. Feng, and K. W. Cameron, "Performance-Constrained Distributed DVS Scheduling for Scientific Applications on Power-aware Clusters," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage, and Analysis (SC)*. ACM/IEEE, 2005.

[17] M. Horowitz, T. Indermaur, and R. Gonzalez, "Low-Power Digital Design," in *Proceedings of the Symposium on Low Power Electronics*. IEEE, 1994.

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

[19] R. Ge, X. Feng, and K. Cameron, "Improvement of Power-Performance Efficiency for High-End Computing," in *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2005.

[20] S. Kamil, J. Shalf, and E. Strohmaier, "Power Efficiency in High Performance Computing," in *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2008.

[21] D. Li, B. de Supinski, M. Schulz, K. Cameron, and D. Nikolopoulos, "Hybrid MPI/OpenMP Power-Aware Computing," in *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2010.

[22] D. Li, D. Nikolopoulos, K. Cameron, B. de Supinski, and M. Schulz, "Power-Aware MPI Task Aggregation Prediction for High-End Computing Systems," in *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2010.

[23] F. Bellosa, "The Benefits of Event-Driven Energy Accounting in Power-Sensitive Systems." in *SIGOPS, European Workshop*. ACM, 2000.

[24] W. L. Bircher, M. Valluri, J. Law, and L. John, "Runtime Identification of Microprocessor Energy Saving Opportunities." in *Proceedings of the International Symposium on Low Power Electronics and Design, (ISLPED)*. ACM, 2005.

[25] W. L. Bircher and L. K. John, "Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events," in *Proceedings of the International Symposium on Performance Analysis of Systems & Software, (ISPASS)*. IEEE, 2007.

[26] A. Kodi and A. Louri, "Performance Adaptive Power-Aware Reconfigurable Optical Interconnects for High-Performance Computing (HPC) Systems," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage, and Analysis (SC)*. ACM/IEEE, 2007.

[27] L. Shang, L.-S. Peh, and N. K. Jha, "Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks," in *Proceedings of the International Symposium on High-Performance Computer Architecture, (HPCA)*. IEEE, 2003.