Iso-energy-efficiency: An approach to power-constrained parallel computation

Shuaiwen Song\textsuperscript{1}, Chun-Yi Su\textsuperscript{1}, Rong Ge\textsuperscript{2}, Abhinav Vishnu\textsuperscript{3}, and Kirk W. Cameron\textsuperscript{1}

\textsuperscript{1}SCAPE Laboratory, Virginia Tech \textsuperscript{2}Marquette University \textsuperscript{3}Pacific Northwest National Laboratory

\texttt{s562673@vt.edu, sonicat@vt.edu, rong.ge@marquette.edu, Abhinav.Vishnu@pnl.gov, cameron@vt.edu}

Abstract

Future large scale high performance supercomputer systems require high energy efficiency to achieve exaflops computational power and beyond. Despite the need to understand energy efficiency in high-performance systems, there are few techniques to evaluate energy efficiency at scale. In this paper, we propose a system-level iso-energy-efficiency model to analyze, evaluate and predict energy-performance of data intensive parallel applications with various execution patterns running on large scale power-aware clusters. Our analytical model can help users explore the effects of machine and application dependent characteristics on system energy efficiency and isolate efficient ways to scale system parameters (e.g. processor count, CPU power/frequency, workload size and network bandwidth) to balance energy use and performance. We derive our iso-energy-efficiency model and apply it to the NAS Parallel Benchmarks on two power-aware clusters. Our results indicate that the model accurately predicts total system energy consumption within 5\% error on average for parallel applications with various execution and communication patterns. We demonstrate effective use of the model for various application contexts and in scalability decision-making.

Keywords: Iso-energy-efficiency, Performance Isoefficiency, Power Consumption, Power-Aware Clusters.

I. INTRODUCTION

As we enter the era of exascale computing, energy consumption of large scale parallel systems and data centers has become one of the most significant hindrances for designing highly scalable data intensive applications and larger parallel systems. For instance, recommendations in a recent report from the US Department of Energy suggest the power consumption of an exaflop machine, capable of a 1000-fold performance increase over current petaflop systems, must be constrained to a 10-fold increase in power consumption [1]. This engineering challenge coupled with the high operational costs and system failure rates associated with many-megawatt computing resources has increased the need to consider power and the entangled effects of performance in emergent exascale systems and applications.

Research [2-5] in high-performance power-aware computing has focused on identifying power saving opportunities in communication phases and applying DVFS [6] (Dynamic Voltage and Frequency Scaling) strategies to these phases to reduce power consumption without sacrificing performance. Fig. 1 depicts the types of controllers used in these techniques to build sophisticated power management software. The focus in previous works has been developing a controller that uses observational data and (in later techniques) predictive data to schedule power states and balance performance.

A key limitation of past approaches is a lack of power-performance policies allowing users to quantitatively bound the effects of power management on the performance of their applications and systems. Existing controllers and predictors use policies fixed by a knowledgeable user to opportunistically save energy and minimize performance impact. While the qualitative effects are often good and the aggressiveness of a controller can be tuned to try to save more or less energy, the quantitative effects of tuning and setting opportunistic policies on performance and power are unknown. In other words, the controller will save energy and minimize performance loss in many cases but we have little understanding of the quantitative effects of controller tuning. This makes setting power-performance policies a manual trial and error process for domain experts and a black art for practitioners. To improve upon past approaches to high-performance power management, we need to quantitatively understand the effects of power and performance at scale.

We use a modeling based approach that captures power-performance tradeoffs system-wide and at scale. Our basic idea is to apply the concept of iso-efficiency [7] for performance, or the ability to maintain constant per-node performance as a system scales, to power-performance management. We want to create techniques that allow us to quantitatively control and maintain power-performance as systems and applications scale; we thus name our approach iso-energy-efficiency. In conducting this work, we found the first essential step toward controlling for iso-energy-efficiency was to create a detailed, sophisticated, accurate model of the effects of performance and power on scaled systems and applications.

The contributions of this work include:
- Development of a fine-grained, analytical iso-energy-efficiency model that incorporates parallel system components and computational overlap at scale.
- Accuracy analysis and verification of the model on two power-scalable clusters.
- Results from a detailed power-performance scalability analysis of EP, FT and CG from the NAS Parallel Benchmarks [8], including use of the iso-energy-efficiency model to bound and maintain system energy efficiency at scale.

To the best of our knowledge, this is the first system-level, scalable, analytical model of both power and performance on real systems and applications. We begin the succeeding discussions with some related work followed by an overview of the model. Next, we show validation and results using the model to perform scalability analysis of the NAS Parallel Benchmarks. Lastly, we show full derivation of the model and its parameters and conclusions.

II. RELATED WORK

A. Isoefficiency

According to Amdahl’s law [9], speedup for parallel systems is limited by the amount of parallelism inherent in the application. This law characterizes the performance impact of parallelism. Though there are several other alternative viewpoints on speedup, the most relevant to our work is that of Grama et al [7] who proposed a formal performance isoefficiency function describing how ideally performance efficiency will remain constant relative to the smallest node configuration.

![Figure 2a. FT performance and energy efficiency.](image)

For a fixed problem size, Fig. 2a and 2b show the performance efficiency curves for FT and CG. FT scales reasonably well while CG drops off at 16 CPUs then recovers relative to the ideal case due to longer networking time caused by the fat-tree networking topology in our SystemG’s Infiniband setup (see section IV-A. All the results and parameters in this paper are obtained from real systems). There are a plethora of performance analysis tools and techniques available to help us interpret and understand an application’s scalability. These analyses may suggest any number of root causes that can be addressed to improve isoefficiency.

In contrast, just measuring energy use is challenging for non experts. Fig. 2a and 2b show the energy efficiency for FT and CG. Moreover, even though the energy efficiency (or lack thereof) in these applications is obvious as they scale, there are few tools currently available to explain the observed energy efficiency.

![Figure 2b. CG performance and energy efficiency.](image)

Being able to identify the root cause of energy inefficiency would allow us to improve system and application efficiency more in line with the ideal isoefficient case. However, analyzing and potentially predicting energy efficiency is exceedingly difficult since we must identify and isolate the interacting effects of power and performance. For example, changing the power settings on a processor using DVFS affects performance which in turn potentially affects the length of time an application takes to complete which is key to its overall energy usage.

B. Parallel performance models

There has been extensive research conducted on performance speedup and scalability of parallel applications in high performance computing. As mentioned, Amdahl’s law [9] introduced the concept that the speedup is limited by the fraction of the workload that can be computed in parallel. Grama et al [7, 10] formally defined isoefficiency as discussed. The fixed-time speedup model [11], memory-bounded speedup model [12], and other related studies [13, 14] all extend Amdahl’s law in unique ways. However, all of these approaches focus on performance and ignore both energy consumption and the performance effects of power management.

C. Energy efficiency in HPC

Several high-profile efforts such as the TOP500 List [15], the Green500 List [16], the SPECpower benchmark [17], and power-performance evaluation of the HPCC benchmarks [18, 19] have elevated the interest in energy efficiency for high-end systems and servers.
Ge et al proposed the PowerPack [20] framework for measuring correlated power and performance data on large scale systems and we use this framework to collect the results presented. Early work to improve the efficiency of high-end systems [3-5] used various DVFS scheduling strategies to gain significant energy savings under performance constraints. Similarly, Freeh et al [2, 5, 21] studied energy-performance tradeoffs for MPI applications.

Our proposed iso-energy-efficiency model analyzes and predicts the combined effects of performance and power on scalable systems. The policy module highlighted in Fig. 1 is a practical application of improved understanding of the power-performance tradeoffs and contrasts our work with approaches to energy efficiency in HPC which have historically focused on improving controllers and predictors. The iso-energy-efficiency approach will improve our understanding of power-performance to quantitatively bound the impact of power management on performance.

D. Energy modelling

The power-aware speedup model proposed by Ge and Cameron [22] is a generalization of Amdahl’s Law for energy. While this model accurately captures some of the effects of energy management on speedup, it provides little insight to the root cause of poor power-performance scalability. In contrast, the iso-energy-efficiency model generally predicts energy consumption as the system scales up allowing direct analysis and comparison of the tradeoffs between various model parameters.

There have been several related architecture level simulation works such as Wattch [23], SimplePower [24] and SoftWatt [25]. These techniques address a similar problem but typically in the context of computer architecture as opposed to computer systems that include processor, memory, and on-chip and off-chip behaviors of an application. These simulators are also execution-driven using compiled code to execute and evaluate through simulation; whereas our model is strictly analytical (and thus more scalable to large systems) while incorporating online parameters.

The Energy Resource Efficiency (ERE) metric proposed by Jiang et al [26] defines a link between performance and energy variations in a system to clearly highlight the various performance-energy tradeoffs. As with other models that identify energy efficiency, this model analyzes at a very high-level and does not identify causal relationships with poor metric results.

The energy model proposed by Ding et al [27] uses circuit-level simulation to analyze power-performance tradeoffs. While this model shows promise for circuit-level design, it is too unwieldy for use in analyzing existing large-scale power-scalable clusters. The model also makes a number of simplifying assumptions such as homogenous workloads and no computational overlap making it less practical for modeling real systems.

III. ISO-ENERGY-EFFICIENCY MODEL

Here we briefly describe the iso-energy-efficiency model for evaluating the power-performance tradeoffs of parallel applications and systems. The derivation of the model is described in detail in Section VI. Tables 1 and 2 provide a summary of all model parameters.

Let $E_t$ be the total energy consumption of sequential execution and $E_p$ be the total energy consumption of parallel execution for a given application on $p$ parallel processors. Let $E_o$ represent the additional energy overhead required for parallel execution:

$$E_o = E_p - E_t \quad (1)$$

We now define iso-energy-efficiency (EE) as:

$$EE = \frac{E_t}{E_p}$$

<table>
<thead>
<tr>
<th>Table 1 Machine dependent parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
</tr>
<tr>
<td>$W_c$</td>
</tr>
<tr>
<td>$W_m$</td>
</tr>
<tr>
<td>$W_{com}$</td>
</tr>
<tr>
<td>$W_{mem}$</td>
</tr>
<tr>
<td>$M$</td>
</tr>
<tr>
<td>$B$</td>
</tr>
<tr>
<td>$P$</td>
</tr>
<tr>
<td>$N$</td>
</tr>
<tr>
<td>$\alpha$</td>
</tr>
<tr>
<td>$T_o$</td>
</tr>
<tr>
<td>$T_s$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2 Application dependent parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
</tr>
<tr>
<td>$t_c$</td>
</tr>
<tr>
<td>$t_m$</td>
</tr>
<tr>
<td>$t_{msg}$</td>
</tr>
<tr>
<td>$t_{byte}$</td>
</tr>
<tr>
<td>$T_{IO}$</td>
</tr>
<tr>
<td>$P_{cpu}$</td>
</tr>
<tr>
<td>$P_{idle}$</td>
</tr>
<tr>
<td>$\Delta P$</td>
</tr>
<tr>
<td>$P_{mem}$</td>
</tr>
<tr>
<td>$P_{IO}$</td>
</tr>
<tr>
<td>$P_{IO-idle}$</td>
</tr>
<tr>
<td>$P_{other}$</td>
</tr>
<tr>
<td>$P_{total-idle}$</td>
</tr>
<tr>
<td>$f$</td>
</tr>
</tbody>
</table>
\[ EE = \frac{E_s}{E_p} = \frac{E_1}{E_1 + E_2} = \frac{1}{1 + \frac{E_2}{E_1}} \]  

(2)

Let \( EEF = \frac{E_2}{E_1} \) be the energy efficiency factor (EEF). EEF is the ratio of parallel energy overhead to the energy of an application running sequentially. An application with a large EEF has low energy efficiency, and vice versa. Effective use of the iso-energy-efficiency model (EE) requires accurate estimation of the EEF. We can more accurately estimate EEF using the following equation:

\[ EEF = \frac{aT_sP_{total-idle} + W_{co}T_c + W_{ma}T_{ma}P_m}{E_1} = \frac{aT_sP_{total-idle} + W_{co}T_c + W_{ma}T_{ma}P_m}{aT_sP_{total-idle} + W_{co}T_c + W_{ma}T_{ma}P_m} \]  

(3)

\[ EE = \frac{1}{1 + \frac{E_2}{E_1}} = \frac{1}{1 + \frac{EEF}{1}} \]

(4)

Equations (3) and (4) form the basis for computing iso-energy-efficiency. The challenge is to capture each of the parameters used in these equations for a given application and system combination.

Tables 1 and 2 show the model parameters used to calculate EEF and EE, which can be classified as either machine-dependent or application-dependent. The machine-dependent variable vector can be described as a function of frequency (i.e. computational speed) and workload bandwidth (i.e. computational throughput) of the hardware:

\[ V_{machine}(f, N_{bandwidth}) = (t_c, t_m, t_{msg}, t_{byte}, P_{total-idle}, \Delta P_c, \Delta P_m) \]

The application-dependent variable vector can be described as a function of the amount of parallelism available and the workload for the application:

\[ V_{app}(p, n) = (a, W_c, W_m, W_{co}, W_{ma}, M, B) \]

Section VI provides details describing and motivating the use of these parameters. The reader may skip to this section to learn more about the iso-energy-efficiency model derivation or continue to the next two sections where we validate the iso-energy-efficiency model and demonstrate its usefulness for evaluating parallel power-performance efficiency.

IV. TEST ENVIRONMENT AND MODEL VALIDATION

A. Test Environment

We use two different power-aware clusters to conduct our experiments: SystemG and Dori. The SystemG 22.8 TFLOPS supercomputer provides a research platform for development of high-performance software tools and applications at scale. It utilizes 325 Mac Pro computer nodes and each node has two 4-core 2.8 GHz Intel Xeon Processors. Each node has an 8 GB RAM and each core has a 6 MB cache. SystemG is equipped with Mellanox 40Gbytes/sec end to end InfiniBand adapters and switches which dramatically increases the transmission bandwidth and reduce the latency. Since G stands for ‘green’, SystemG is a power-scalable system and has over 10,000 power and thermal sensors. DVFS, concurrency throttling and dynamic thermal monitoring are enabled. Intelligent Power Distribution Units (Dominion PX) are attached to adjacent machines so users can dynamically profile power consumption of controlled machines or remotely turn on/off nodes, etc.

The Dori system is composed of 8 nodes and each node contains dual core AMD Opteron Processors. Each node has 6 GB RAM and each core has 1 MB cache. Dori is equipped with 1 Gbytes/sec Ethernet and switches.

PowerPack 2.0 [18, 20], designed and implemented by the SCAPE Laboratory at Virginia Tech, is a framework for power/energy profiling, analysis and prediction of parallel applications and systems. The PowerPack infrastructure is composed of both hardware and software components: the hardware is responsible for accurate and reliable direct measurement of both system-wide and component level power consumption and the software automatically collects processes and synchronizes power data with system load. We used the PowerPack toolkit for all of the power and performance measurements obtained herein on both clusters.

The NAS Parallel Benchmarks consist of 5 kernels and 3 pseudo-applications that mimic the computation and data movement characteristics of large scale CFD applications which are widely used in HPC community. We validate the proposed model on both systems for the NAS Parallel Benchmarks. We conducted scalability studies for three benchmark suites (FT, CG, EP) on SystemG.

B. Model Validation

To validate the iso-energy-efficiency model, we need to verify the correctness of the model on single and parallel processor configurations. We vigorously measure and derive the parameters from Tables 1 and 2; namely the machine and application dependent parameters.

For the machine-dependent parameters, we built a tool using the Perfmon API from UT-Knoxville to automatically measure the average \( t_c \) (time per on-chip computation instruction) derived as \( e_{Pfem} \). We use the \( \text{lat_mem_rd} \) function from the LMbench microbenchmark [29] to estimate memory costs \( t_{m} \) and \( t_{msg} \) and \( t_{byte} \) are obtained by using the MPPTest tool [30] for both the InfiniBand [31] and Ethernet interconnects in the two clusters. In addition, \( P_{total-idle}, \Delta P_c \) and \( \Delta P_m \) can be obtained by using PowerPack [20]. We did not include disk I/O in our estimations for our energy efficiency model because the applications we tested are not disk intensive. We leave this to future work. For completeness, though it is not used in the current study, we were able to estimate \( t_{id} \) by using the Linux pseudo file /proc/stat.
For the application-dependent parameters, we build a workload and overhead model for each parameter by analyzing the algorithm and measuring the actual workload for each application. We use Perfmon to measure each workload parameter, \( W_c, W_m, W_{co}, W_{mo} \) and we use the TAU [32] performance tool from the University of Oregon to measure \( M \) and \( B \). Fig. 3 illustrates the accuracy of the energy model for \( P \) processors. (Note: Specifically, these results are for Equation (15) in the derivation Section VI.)

Fig. 3 compares the energy consumption predicted by the iso-energy-efficiency model with the actual energy consumption obtained using the PowerPack framework on Dori cluster for \( P=4 \). We repeated all the experiments to reduce measuring errors. The results indicate that the proposed energy model can accurately predict the energy consumption within 5% prediction error. We conducted similar experiments on SystemG for \( P=1, 2, 8, 16, 32, 64, 128 \). Fig. 4 shows the average error rate of EP, FT, and CG applications on SystemG under different levels of parallelism using the InfiniBand interconnect. The results show good accuracy. Upon detailed analysis, the relatively higher errors (8.31%) found with CG were due to: 1) The current CG memory model is based on analysis of the CG program and its memory behavior across different parallel platforms; 2) Our measurements show that the CG memory workload decreases substantially while scaling up on the SystemG system. This appears due to the fact that CG favors both temporal and spatial locality as a system scales. Improving the accuracy for CG is the subject of future work.

Based on the accuracy results for both SystemG and Dori clusters, we conclude that our iso-energy-efficiency model performs well on different network interconnection infrastructures and can predict total system energy consumption with an average of 5% prediction error rate for parallel applications with various execution and communication patterns.

V. EXPERIMENTAL RESULTS

A. Energy consumption and efficiency prediction for large scale systems

Given the accuracy of our modeling techniques as described in the previous section, we use measurements from smaller configurations to predict and analyze power-performance tradeoffs on larger systems. (Note: we build our energy consumption and efficiency models using Equations (13), (15), (18), (21) from Section VI.)

Initially, we obtain machine-dependent variables from the smaller system and use these values and our models to predict values for increasing number of nodes:

\[
V_{machine}(f, N_{bandwidth}) = (t_c, t_m, t_{msg}, t_{byte}, P_{total-\text{idle}}, \Delta P_c, \Delta P_m)
\]

<table>
<thead>
<tr>
<th>Error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00%</td>
</tr>
<tr>
<td>5.00%</td>
</tr>
<tr>
<td>10.00%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SystemG</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
</tr>
<tr>
<td>4.99%</td>
</tr>
<tr>
<td>8.31%</td>
</tr>
</tbody>
</table>

**Figure 4.** The average error rate of EP, FT and CG program. class=B in node number \( p=1,2,4,8,16,32,64,128 \). All variables can be measured as described in the previous section. Frequency-dependent variables can be combined by normalizing measurements obtained through the use of hardware counters, LMBench, MPPTest and Powerpack. For example, \( t_c \) can be described as \( 11.9 \times 10^{-10} \) sec on SystemG. We assume power is proportional to \( f^\gamma (\gamma \geq 1) \).

Next, we model application-dependent variables from the smaller system:

\[
V_{app}(p, n) = (\alpha, W_c, W_m, W_{co}, W_{mo}, M, B)
\]

Except for \( \alpha \), all of these variables in \( V_{app}(p, n) \) depend on a performance model and can be described as a function of problem size, \( n \), and the level of parallelism, \( p \). For example, \( W_{co} \) could be described as \( n \log_2 p \) in one-dimensional, unordered and radix-2 binary exchange Fast Fourier Transform. With all parameters accounted for, we can solve for Equations (3) and (4). (Note: Specifically, we first solve Equations (13), (15), (18), and (21) described in the next section.) We can then project these parameters for larger value of \( p \) to predict the power-performance behavior and tradeoffs of large scale systems.
B. Scalability studies for NAS Parallel Benchmark suites

In this section, we analyze the power-performance characteristics of FT, EP and CG using the iso-energy-efficiency approach. We isolate power-performance efficiency problems and use the model findings to tune parameters such as problem size, n, CPU clock frequency, f, and level of parallelism, p to improve efficiency. In each case, we use the methods described in the previous sections to obtain model parameters and build our model from measurements on a smaller system. Once we’ve identified estimates for $V_{\text{machine}}(f, N_{\text{bandwidth}})$ and $V_{\text{app}}(p, n)$ vectors, we build EE and EEF as described in Equations (3) and (4). In the rest of this section, all the parameterizations are obtained for the SystemG cluster though the same methodology can be applied to other platforms. (Our model uses multiple hardware counters to estimate on-off chip workload pattern and execution time. Underlying architecture level changes are currently hidden in the hardware counter driven model building. So the changes of CPU and Memory level activities will be captured.)

1) FT

FT computes a 3-D partial differential equation solution using Fast Fourier Transforms. The application stresses the CPU, memory and the communication network during various phases. Parallel FT iterates through approximate four phases during the execution: computation phase 1, reduction phase, computation phase 2 and all-to-all communication. The FT benchmark is communication intensive with dominating parallel communication overhead for the all-to-all phase. FT has a large memory footprint compared to the EP (Embarrassingly Parallel) application in the NAS suite.

We use the Pairwise exchange/Hockney model [33, 34] to estimate the MPI_Alltoall operations required to solve for EE and EEF. (Note: This replaces the general approach to communication estimation described by Equation (17) in section VI.) By analyzing the FT’s Alltoall communication algorithm on the architecture of the SystemG cluster, we found the Pairwise exchange/Hockney model appropriate and accurate in our validation testing. The time duration for this implementation is described as follows:

$$T_{\text{net}} = (p - 1) \ast (\alpha(H) + \beta(H) \ast H).$$

In the equation above, H is the message size, $\alpha(H)$ is message start up time, and $\beta(H) \ast H$ is the transmission time. For details, please refer to the original paper [33]. We use our own measuring tools, MPPTest and the PowerPack framework to obtain the machine dependent parameters:

$$V_{\text{machine}} - \text{FFT}(f, N_{\text{bandwidth}}) = (t_c, t_m, t_{msg}, t_{\text{Byte}}, P_{\text{total-idle}}, \Delta P_c, \Delta P_m)$$

$$= (6.44, 10^{-10}, 1.12 \ast 10^{-7}, 2.53 \ast 10^{-5}, 1.82 \ast 10^{-9}, 24 f^7, 3.4 f^7, 0.76 f^7)$$

In the equation above, for simplicity, we set $\gamma = 2$ based on our Testbed SystemG. We analyze FT and measure the actual workload by observing on-chip executing instructions, L1, L2 cache misses, main memory accesses and total instructions using Perfmon to obtain:

$$V_{\text{app}} - \text{FFT}(p, n) = (\alpha, W_c, W_m, W_{co}, W_{mo}, M, B, \gamma = 2, 1.06 \ast 10^4, 9.49n, 4.46 \ast 10^2 n \log_2 p, -0.73n \log_2 p, 22, \frac{4n}{\log^2 p - 1})$$

We then solve for $EE_{FFT}$:

$$EE_{FFT} = \frac{E_0}{E_1} = \frac{\alpha T_0 (P_{\text{total-idle}} + W_c + W_m + \Delta P_c + \Delta P_m)}{\alpha T_1 (P_{\text{total-idle}} + W_c + W_m + \Delta P_c + \Delta P_m)}$$

$$= \frac{6.87 \log_2 p - 1.75 \log_2 p + (p - 1)}{11500 \frac{0.376}{n} \frac{1}{\log^2 p - 2}}$$

And thus for $EE_{FFT}$ we obtain:

$$EE_{FFT} = \frac{1}{14} \frac{6.87 \log_2 p - 1.75 \log_2 p + (p - 1)}{11500 \frac{0.376}{n} \frac{1}{\log^2 p - 2}}$$

Fig. 5 plots EE with a fixed workload size n. We can see the level of parallelism, p, most affects changes in energy efficiency versus frequency (or DVFS power states). In fact, for this code, frequency f has little impact on energy efficiency. FT is dominated by all-to-all communications and synchronizations which makes it less likely to be influenced by changes in CPU frequency. As the number of processors scales, the effects of CPU clock frequency on on-chip workload diminish eventually while the increasing effects of parallel overhead and memory dominate. Thus, for fixed workloads on FT, increasing p will dramatically decrease the energy efficiency and scaling down the operating frequency will conserve power without reducing the overall system-wide energy efficiency.
Fig. 6 illustrates $EE_{FFT}$ when frequency fixes to 2.8GHz since frequency does not affect energy efficiency. We can see $p$ still dominates the variance of energy efficiency. It is also obvious that increasing the problem size, $n$, does enhance the energy efficiency.

2) $EP$

In parallel computing, an embarrassingly parallel ($EP$) workload has little inter-processor communication between parallel processes. $EP$ in the NPB benchmarks generates pairs of Gaussian random deviates using Marsaglia polar method. It separates tasks with little or no overhead. Results of $EP$ can also be considered as a reference of peak performance of a given machine. We use our measuring tools, MPFTest and the PowerPack framework to obtain the machine dependent parameters:

$$V_{machine-EP}(f,N_{bandwidth}) = \left( t_c, t_{m}, t_{msg}, t_{byte}, T_{total-idle}, \Delta P_c, \Delta P_m \right)$$

$$= (1.19 * 10^{-8}, 1.12 * 10^{-7}, 2.53 * 10^{-5}, 1.82 * 10^{-7}, 189 f^2, 2.67 f^2, 1.52 f^2)$$

After analyzing the parallel $EP$ codes, we have:

$$V_{app-EP}(p, n) = (\alpha, W_c, W_m, W_{co}, W_{mno}, M, B)$$

$$= (0.93, 109.4*n, 1.03*10^{-6} n, 0, 6.7*10^{-7} n, 0, 0, 0)$$

Since communication in embarrassingly parallel is trivial, we simply set $M$ and $B$ to zero in $V_{app-EP}(p, n)$.

Thus, from Equation (19), we have $EE_{EP}$:

$$EE_{EP} = \frac{E_G}{E_1} = \frac{\alpha T_c(\Delta P_{total-idle}) + W_{co} \Delta P_c + W_{mno} \Delta P_m}{\alpha T_c(\Delta P_{total-idle}) + W_{co} \Delta P_c + W_{mno} \Delta P_m}$$

$$= \frac{1.43 (p-1) f^2}{6.3 * 10^6 n + 2.2 f^2}$$

So $EE_{EP}$ becomes:

$$EE_{EP} = \frac{1}{1 + \frac{1.43 (p-1) f^2}{6.3 * 10^6 n + 2.2 f^2}}$$

Fig. 7 illustrates the variation of $EE_{EP}$. This figure indicates that energy efficiency hardly changes with $p$ and $f$. Energy efficiency is close to 1 for different combinations of $p$ and $f$ because only minimum communication overhead is imposed. Since this is nearly ideal iso-energy-efficiency, we cannot improve the energy efficiency by scaling problem size $n$ at all because $E_G$ increases as fast as $E_1$.

3) $CG$

The NAS $CG$ benchmark evaluates a parallel system’s computation and communication performance. It uses the conjugate gradient method to find out the smallest eigenvalue of a large, sparse matrix. It solves a sparse linear algebra problem which is common to scientific applications on large-scale systems. We first obtain the machine-dependent parameters using the previous methods:

$$V_{machine-CG}(f,N_{bandwidth}) = (t_c, t_m, t_{msg}, t_{byte}, T_{total-idle}, \Delta P_c, \Delta P_m)$$

$$= (1.14 * 10^{-9}, 1.12 * 10^{-7}, 2.53 * 10^{-5}, 1.82 * 10^{-7}, 24 f^2, 4.57 f^2, 1.25 f^2)$$

For the application-dependent parameters we obtain:

$$V_{app-CG}(p, n) = (\alpha, W_c, W_m, W_{co}, W_{mno}, M, B)$$

$$= (0.85, 2.13 * 10^5 n^{1.25}, 0.96 n^{1.75}, 1.86 * 10^6 n^{0.7} * 2^{1.1 (log_2 p-1)}, 4.75 n^{0.5} * 2^{0.14 (log_2 p-1)^2}, 0, 0)$$

Thus, we solve for $EE_{CG}$:
5) Discussion of the effect of the level of parallelism, p
We can rewrite Equation (16) as follows to see the relation between $E_0$ and $p$ when the workload is evenly divided among processors (homogeneous workload):

$$E_0 = E_p - E_i = aT_0(P_{total-ide}) + pw_{co}t_c\Delta P_c + pw_{mo}t_m\Delta P_m = \frac{aT_0(P_{total-ide}) + pw_{co}t_c\Delta P_c + pw_{mo}t_m\Delta P_m}{aT_1(P_{total-ide}) + w_{co}t_c\Delta P_c + w_{mo}t_m\Delta P_m}$$

and then for $EE_{CG}$:

$$EE_{CG} = \frac{1}{1 + \frac{5.3 \times 10^{-2}f n^{0.75} + 1.15 \times 10^{-3}f n^{0.5} + 2.0 \times 10^{14}n^{0.5} + 0.14 \times \log(2p-1)^2}{6.07 \times 10^{-3}f n^{1.25} + 2.33 \times 10^{-6}f n^{1.75}}. \quad (8)$$

Thus, $E_0$ is $O(p^k)$ ($k \geq 1$). Generally speaking, more parallelization will incur lower energy efficiency. In this case, the application’s tasks among all nodes require extra computation, memory accesses and communication efforts to coordinate with each other to complete the job. We observe this phenomenon in FT and CG. In contrast, EP incurs almost no overhead and energy efficiency doesn’t decrease significantly with the increase of the levels of parallelization.

6) Discussion of problem size $n$
Problem size is a dominant factor affecting energy efficiency. The $EE$ for applications FT and CG improve if the problem size scales. However, increasing problem size does not necessarily improve energy efficiency as in the case of energy efficiency for EP.

7) Discussion of frequency, $f$
Decreasing frequency can either increase or decrease energy efficiency. For EP and FT, we observed no energy efficiency improvements for parallel execution when we adjust to low frequency. However, in the case of CG, we found that higher frequencies can improve energy efficiency because the memory overhead $W_{mo}$ value decreases while system scales up.

VI. MODEL DETAILED DERIVATION
In this section we describe the details for deriving the iso-energy-efficiency model first presented in Section III.

A. Performance Model
At the system level, the theoretical sequential execution time for an on-chip/off-chip workload comprises three components [36, 37]: computation time $W_c t_c$ (with on-chip instruction execution frequency), main memory access latency $W_m t_m$, and I/O access time $T_{i0}$ (with off-chip instruction execution frequency). Thus the theoretical execution time can be expressed as:

$$T = W_c t_c + W_m t_m + T_{i0} \quad (5)$$

Since optimization techniques could raise various levels of overlap between components [38], we multiply $T$ by a corrector factor $\alpha$ ($0 \leq \alpha \leq 1$):

$$T' = \alpha T = \alpha (W_c t_c + W_m t_m + T_{i0}) \quad (6)$$

where $T'$ is the actual execution time.
B. Energy Model for one and p parallel processor(s):

When executing a parallel application, total energy consumption can be divided into four parts: computation energy $E_c$, main memory access energy $E_{mem}$, I/O access energy $E_{I/O}$, and other system components energy $E_{other}$, such as motherboard, system and CPU fans, power supply, etc. Thus, we have total energy $E$ [20]:

$$E = E_c + E_{mem} + E_{I/O} + E_{other} \quad (7)$$

The first three parts of this equation can be further separated into two energy states: running state and idle state. For example, $E_c$ can be divided into $E_{c-on}$ and $E_{c-idle}$. Thus, we can deduce total energy $E$ as [18, 20]:

$$E = E_{c-on} + E_{c-idle} + E_{mem-on} + E_{mem-idle} + E_{I/O-on} + E_{I/O-idle} + E_{other} \quad (8).$$

From (6) and (8),

$$E = \alpha TP_{total-idle} + W_t c \Delta P_c + W_m t_m \Delta P_m + T_{I/O} \Delta P_{I/O} \quad (9)$$

Where $W_t c$ is the total computation time; $W_m t_m$ is the total memory access time and $T_{I/O}$ is the total I/O access time.

$$\Delta P_c = P_{c-on} - P_{c-idle}, \quad \Delta P_m = P_{m-on} - P_{m-idle}, \quad \Delta P_{I/O} = P_{I/O-on} - P_{I/O-idle}.$$ 

Equation (9) seems quite cumbersome; however, it is intuitive: $\alpha TP_{total-idle}$ is the total energy consumption of an idle-state system during an application’s execution time. $W_t c \Delta P_c$ is the additional energy used while an application is performing computation. Similarly, $W_m t_m \Delta P_m$ and $T_{I/O} \Delta P_{I/O}$ are the additional energy consumption for conducting main memory and I/O accesses.

$$E = \alpha TP_{total-idle} + W_t c \Delta P_c + W_m t_m \Delta P_m + T_{I/O} \Delta P_{I/O} \quad (10)$$

With energy model:

$$E = \alpha TP_{total-idle} + W_t c \Delta P_c + W_m t_m \Delta P_m + \Delta P_{I/O}(P_{net-on} - P_{net-idle}) \quad (11)$$

In our experiments (on both the Dori system with Ethernet and SystemG with InfiniBand), the difference between $P_{net-on}$ and $P_{net-idle}$ is not significant so we simply ignore the effect $\Delta P_{I/O}(P_{net-on} - P_{net-idle})$ in (11):

$$E = \alpha TP_{total-idle} + W_t c \Delta P_c + W_m t_m \Delta P_m \quad (12)$$

C. Energy Model for A Single Processor

Equations (10) and (12) are the kernel components of the performance model and iso-energy-efficiency model in this paper. Let us apply these to $E_i$ which we discussed in Section III. When an application executes on a single processor, there are no messages exchanged. This means no $T_{net}$ in (10). Thus, $E_i$ becomes:

$$E_i = \alpha T_i (P_{total-idle}) + W_t c \Delta P_c + W_m t_m \Delta P_m \quad (13)$$

where $T_i = (W_t c + W_m t_m)$

D. Energy Model for p Parallel Processors

Similarly, to get $E_p$, we define the energy model in ith $(1 \leq i \leq p)$ processor among $p$ parallel processors:

$$E_{p,i} = \alpha T_{p,i}(P_{total-idle}) + (w_{c,i} + w_{m,i})t_c \Delta P_c + (w_{m,i} + w_{mo,i})t_m \Delta P_m \quad (14)$$

where $T_{p,i} = (w_{c,i} + w_{m,i})t_c + (w_{m,i} + w_{mo,i})t_m + T_{net,i}$

In (14), $w_{co,i}$ and $w_{mo,i}$ are computation and memory access overheads for the ith of p processors in terms of parallelism. Thus, we have $E_p$ representing total energy consumption for all processors:
\[ E_p = \sum_{i=1}^{P} E_{p,i} = \alpha T_p (P_{\text{total-idle}}) + (W_c + W_{co}) t_c \Delta P_c + (W_m + W_{mo}) t_m \Delta P_m \]  

(15)

where, \( T_p = \sum_{i=1}^{P} T_{p,i} = [(W_c + W_{co}) t_c + (W_m + W_{mo}) t_m + \sum_{i=1}^{P} T_{\text{net},i}] \) and capital “W” represents summation of all workload in all processors.

From (1) we can calculate the energy overhead \( E_o \):

\[ E_o = E_p - E_0 = \alpha T_o (P_{\text{total-idle}}) + W_{co} t_c \Delta P_c + W_{mo} t_m \Delta P_m \]  

(16)

where \( T_o = (W_{co} t_c + W_{mo} t_m + \sum_{i=1}^{P} T_{\text{net},i}) \)

(17)

In (15) and (16), \( W_{co} \) is the total parallel computation overhead \( W_{co} = \sum_{i=1}^{P} w_{co,i} \) and \( W_{mo} \) represents the total parallel memory access overhead \( W_{mo} = \sum_{i=1}^{P} w_{mo,i} \). \( T_{\text{net},i} \) stands for accumulated networking time. \( T_{\text{net},i} \) can be further divided into two parts: message start up time and data transmitting time [33]. Communication overhead modeling varies depending on application and network infrastructure. Equation (17) is a general approach and specific parameterization for network modeling is applied for each application (see FT example in Section V).

\[ \sum_{i=1}^{P} T_{\text{net},i} = M t_{msg} + B t_{\text{Byte}} \]  

(18)

So that \( E_o \) can be expressed as:

\[ E_o = \alpha T_o (P_{\text{total-idle}}) + W_{co} t_c \Delta P_c + W_{mo} t_m \Delta P_m \]  

(19)

where \( T_o = (W_{co} t_c + W_{mo} t_m + M t_{msg} + B t_{\text{Byte}}) \)

\[ E_{o_2} = \frac{E_o}{E_1} = \frac{\alpha T_o (P_{\text{total-idle}}) + W_{co} c t_c \Delta P_c + W_{mo} t_m \Delta P_m}{\alpha T_1 (P_{\text{total-idle}}) + W_{co} c t_c \Delta P_c + W_{mo} t_m \Delta P_m} \]  

(20)

where \( T_o = (W_{co} c t_c + W_{mo} t_m + M t_{msg} + B t_{\text{Byte}}) \)

E. Energy Efficiency Factor (EEF)

Using the Equations (13) and (18), we can formulate the Energy Efficiency Factor (EEF) more accurately.

\[ EEF = \frac{E_{o_2}}{E_1} = \frac{\alpha T_o (P_{\text{total-idle}}) + W_{co} c t_c \Delta P_c + W_{mo} t_m \Delta P_m}{\alpha T_1 (P_{\text{total-idle}}) + W_{co} c t_c \Delta P_c + W_{mo} t_m \Delta P_m} \]  

Where \( T_o = (W_{co} c t_c + W_{mo} t_m + M t_{msg} + B t_{\text{Byte}}) \)

\[ T_1 = (W_{co} c t_c + W_{mo} t_m) \]  

(21)

Equation (19) contains two categories of parameters which directly impact performance and energy consumption: 1) machine dependent variables \( t_c, t_m, t_{msg}, t_{\text{Byte}}, P_{\text{total-idle}}, \Delta P_c, \Delta P_m \), and 2) application dependent variables: \( \alpha, W_c, W_{co}, W_{mo}, M, B \). For the application dependent vector, \( V_{\text{app}} \), the processor number \( p \) and problem size \( n \) are two main factors affecting these parameters.

The values of \( W_c, W_m, W_{co}, W_{mo} \) can be obtained by the combination of analyzing an application’s algorithm and directly measuring the specific performance counters to estimate the on-off chip workload. Also, \( M \) and \( B \) can be acquired by using PMPI in MPICH2 [30] or TAU[39]. The corrector factor \( \alpha \) can be estimated using:

\[ \alpha = \frac{\text{actual execution time}}{w_{c} t_{c} + w_{m} t_{m} + t_{\text{net}}} \]  

(22)

The machine dependent vector can be represented as:

\[ V_{\text{machine}}(f, N_{\text{bandwidth}}) = (t_c, t_m, t_{msg}, t_{\text{Byte}}, P_{\text{total-idle}}, \Delta P_c, \Delta P_m) \]  

For machine dependent variables, machine frequency \( f \) and the network bandwidth, \( N_{\text{bandwidth}} \), are the main factors affecting these parameters. For the time parameters, \( t_c \) and \( t_m \) and \( t_{msg} \) can be also described as functions of \( f \).

Only \( t_{\text{Byte}} \) is related with the network bandwidth. From Kim et al [6, 35]:

\[ f_{\text{max}} \propto \frac{(V_{dd} - V_{th})^{\beta}}{V_{dd}} \]  

\[ P = P_{\text{dyn}} + P_{\text{leak}} = ACV_{dd}^2 f + I_{\text{leak}} V_{dd} \]  

(20)

We can assume \( P_{\text{total-idle}}, \Delta P_c, \Delta P_m, \) are also functions of \( f \). Here we assume power is proportional to \( f^{\gamma} (\gamma \geq 1) \). We use the correlation between power and frequency in our energy model to predict total energy consumption and energy efficiency of large scale parallel system.

From Equations (2) and (19), the iso-energy-efficiency model for parallel applications can be defined as:

\[ EE = \frac{1}{1+EEF} = \frac{1}{1+\frac{\alpha T_o (P_{\text{total-idle}}) + W_{co} c t_c \Delta P_c + W_{mo} t_m \Delta P_m}{\alpha T_1 (P_{\text{total-idle}}) + W_{co} c t_c \Delta P_c + W_{mo} t_m \Delta P_m}} \]  

(21)

In equation (21), \( EEF \) is a combination of machine and application dependent parameters. To maximize the system energy efficiency, we need to keep \( EEF \) as small as possible by scaling characteristics such as degree of parallelism, workload, processor frequencies and network bandwidth.

F. Corrector factor

Accurately capturing performance characteristics is critical to a model of iso-energy-efficiency. Early on in our attempt to create an iso-energy-efficiency model we realized computational overlap, or the ability to conduct computations while waiting on memory or communication delays, could not be ignored since they can reduce execution time dramatically [38]. The amount of overlap varies with an application, the underlying machine architecture, and compiler settings. For example, an application code may have computation time \( O(n^2) \), memory access time \( O(n) \) and network transmitting time \( O(n) \). Without optimization, the theoretical total execution time is \( O(n^2 + n + n) \); however, the actual time is smaller.

In addition to the overlaps described above, extra costs such as parallel overhead data transmission between nodes,
synchronization overhead due to load imbalance and serialization, and the unexpected extra computation overhead caused by parallel scheme and task assignment also contribute as parts of inaccuracy of modeling performance. Thus we propose a comprehensive corrector factor, $\alpha$, to capture the effects above and help adjust the time obtained by model to actual execution time. For simplicity, we estimate $\alpha$ as:

\[
\text{Theoretical Execution Time} = \text{Computation Time} + \text{Memory Access Time} + \text{Network Transmission Time}
\]

\[
\alpha = \frac{\text{Actual Execution Time}}{\text{Theoretical Execution Time}}
\]

For parallel applications, we found empirically for the applications studied that an application using the same compiler settings has the same $\alpha$ value under different levels of parallelism. However, different applications could have different $\alpha$ due to different execution patterns. In addition, same applications running on different machines also have different $\alpha$ values because of diverse underlying architectures.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we present a system level energy efficiency model for various parallel applications and large scale parallel system architectures. We extend the concept of performance isoefficiency to iso-energy-efficiency and show how to build an accurate system level energy efficiency model step by step. Then we apply our analytical model to real scientific applications from NAS Parallel Benchmark suites and illustrate how to derive essential model parameters to predict total system energy consumption and efficiency for large scaling parallel systems. After a thorough and detailed investigation of machine and application dependent parameters which have nontrivial impact on system energy efficiency, we apply the model to three scientific benchmarks representing different execution patterns to study what the influential factors are for system energy efficiency and how to scale them to maintain efficiency. The results conducted on two power-aware clusters show that our model can predict total system energy consumption within average 5% prediction error rate for parallel applications with various execution and communication patterns. And also, in the case study experiments, the results clearly show what the most influential factors are and how these factors can be tuned to maintain energy efficiency. Currently, all of the system parameters in the model are achieved automatically. However, some of the algorithmic/application parameters require direct observation and investigation. With the aid of software tool such as Prophesy toolkit [40] from Texas A&M, we plan to combine current individual components into a single component and more user-friendly tool. Also, we are currently applying this model to GPGPU-based heterogeneous cluster [41] as the subject of a future paper.

VIII. ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation under Grant No. 0910784 and 0905187.

REFERENCES


