An iso-energy-efficient approach to scalable system power-performance optimization

Shuaiwen Song*, Matthew Grove* and Kirk W. Cameron*
*SCAPE Laboratory, Virginia Tech
s562673@vt.edu, mat@vt.edu, cameron@vt.edu

***Abstract*—The power consumption of a large scale system ultimately limits its performance. Consuming less energy while preserving performance leads to better system utilization at scale. The iso-energy-efficiency model was proposed as a metric and methodology for explaining power and performance efficiency on scalable systems. For use in practice, we need to determine what parameters should be modified to maintain a desired efficiency. Unfortunately, without extension, the iso-energy-efficiency model cannot be used for this purpose. In this paper we extend the iso-energy-efficiency model to identify appropriate efficiency values for workload and power scaling on clusters. We propose the use of "correlation functions" to quantitatively explain the isolated and interacting effects of these two parameters for three representative applications: LINPACK, row-oriented matrix multiplication, and 3D Fourier transform. We show quantitatively that the iso-energy-efficiency model with correlation functions is effective at maintaining efficiency as system size scales.**

***Keywords*-Iso-energy-efficiency; performance isoefficiency; system utilization; power aware computing;**

I. Introduction

Since 1992, the performance of supercomputers running parallel applications increased 10,000-fold while the performance per watt improved only 300-fold [1]. Designers of high performance computing systems have come to realize that in order to achieve sustained exaflop performance, they must consider both performance and power consumption at scale.

The growing interest in addressing the power wall in large scale systems has led to a number of approaches to improve cluster efficiencies. The majority of this work [2] [3] [4] [5] [6] [7] [8] [9] has focused on scheduling power modes to decrease power consumption while minimizing the impact of power management on performance.

While such approaches can substantially improve energy efficiency, they primarily use observational (and in some cases predictive) data to schedule power states. This ensures only a qualitative bound on the performance impact of power management. Fundamentally, the observational approach provides little insight as to the quantitative effects of power management on performance.

The iso-energy-efficiency model was recently proposed [10] to quantitatively model the interactive effects of power and performance on power-scalable clusters. The basic approach is to capture the key parameters that affect power and performance, determine their isolated and interactive effects on power and performance, and use the model to predict power and performance for algorithm and machine combinations. This approach is well-suited for quantitatively explaining the causal effects of observed power and performance providing an improved understanding of power management on clusters.

The iso-energy-efficiency model enables users to explain an observed efficiency. This can help a system designer identify inefficiencies in either system or algorithm design, determine a root cause and potentially propose an alternative solution. It would be extremely useful to analytically explore the alternative solution space. In other words, it would be even better if we could automate the process of identifying the parameter settings necessary to achieve a given efficiency.

Suggesting parameter settings for a given efficiency requires modeling the detailed interactive effects of the parameter under study at scale. For example, problem size affects efficiency in an application and system specific way at scale. Yet, the iso-energy-efficiency model sees problem size as an isolated, static parameter that can be shown to have causal effects on a measured efficiency. The current model lacks the ability to capture the subtle changes in power and performance behavior that occur as problem size changes in a scalable system. Nonetheless, given a series of potential problem sizes for a given application we would like to additionally model the problem size necessary to maintain a given efficiency or use in automated power management tools.

A more sophisticated representation of the iso-energy-efficiency model would provide the fidelity needed to capture the effects of system level parameters such as problems size and processor power modes on energy efficiency. We could additionally use the approach to suggest models for parameters that will maintain a given efficiency for an application and system combination. This is challenging however since the resulting model must accurately capture the complicated interacting effects of all the parameters of the original model.

In the next section we motivate the need to study efficiency with changing problem sizes and CPU power modes (frequency). Next, we present a summary of the iso-energy-efficiency model. We then propose the concept and use of correlation functions to provide detailed models of these parameters. We use the extended iso-efficiency technique to

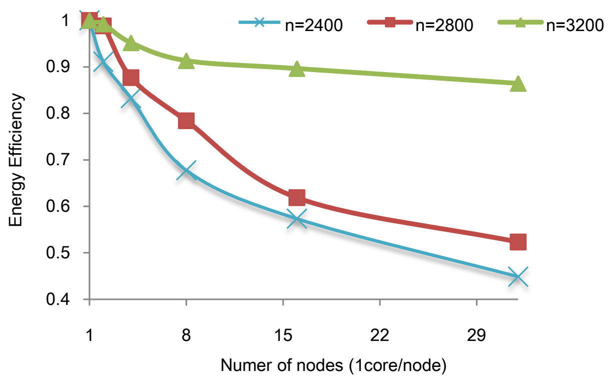

Figure 1. Energy efficiency scaling for Cannon's algorithm with various problem size under fixed frequency.

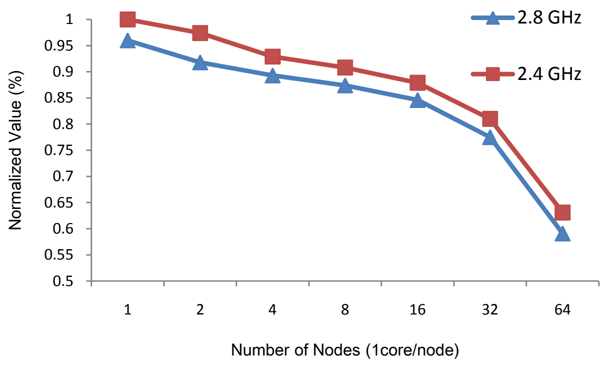

Figure 2. Energy efficiency scaling for Fourier transform under two frequencies.

demonstrate how problem size (and processor power modes) can be managed to maintain user-specified energy efficiency.

II. Motivation

A. Problem size

To illustrate the effect of increasing problem size on system energy efficiency, we ran a simple experiment. For three different sizes of matrix (problem sizes) we applied Cannon's algorithm using an increasing number of nodes (system size). We measured the energy efficiency of the system.

We observe from Figure 1 that overall energy efficiency goes down when increasing system size. However, the energy efficiency improves when increasing the problem size. For example, using 4 nodes with a problem size of 2800 has the same system energy efficiency as 32 nodes with 3200. The graph shows that one potential way to improve or maintain system energy efficiency is to simultaneously increase both system and problem size. Although this is the case for Cannon's algorithm we want to know whether problem size scaling is a generally applicable technique for maintaining system energy efficiency. It may be the case that scaling problem size has a negative effect, or no effect, on some applications.

B. Processor power modes (CPU frequency)

To illustrate the effect of changing frequency on system energy efficiency we performed another experiment. We used Fourier transform with a fixed problem size on an increasing number of nodes (system size) while varying the CPU frequency on all nodes. We measured the energy efficiency of the system.

Figure 2 shows that using 2.4 GHz improves the energy efficiency by an average of 3.2% compared to 2.8GHz. This shows that one potential way to improve, or maintain system energy efficiency is to scale the CPU frequency. Similarly to the problem size scaling technique, we want to know what the relationship is between CPU frequency scaling and system energy efficiency. It may be the case that CPU frequency scaling is a generally applicable technique for maintaining system energy efficiency. However, for some applications it may be the case that varying the frequency does not result in energy reduction and will not improve system energy efficiency. Whether we can achieve energy reduction by scaling the CPU frequency heavily depends on the execution pattern of the application and available frequency scaling range on the system.

III. Related work

A. Performance modeling, DVFS and energy modeling

Based on the original Amdahl's law [11], speedup tends to saturate or grows sub-linearly when number of processors increase. Also, higher speedup or performance efficiency will increase while increasing the computational problem size (weak scaling). So Grama et al [12] proposed the performance isoefficiency metric to relate problem size with the number of processors required to maintain speedup. Performance isoefficiency focuses on performance scalability and modeling but ignores both performance and system-wide energy effects of power management.

In addition to modeling performance, other work focuses on applying various DVFS scheduling strategies to reduce energy consumption for both HPC and data center use. For HPC, Mishra et al [13] proposed a two-tier feedback control coordinated power management in chip-multiprocessors (CMP) by applying workload based DVFS to sectional core islands in order to save CMPs' energy consumption. [5] [7] [9] also use DVFS control based various strategies and prediction schemes to gain significant energy savings. Freeh et al [6] studied energy-performance tradeoffs for MPI applications. Curtis et al [14] combines both DVFS and concurrency throttling to form a multi-dimensional power saving scheme for CMPs. DVFS control based on system load has also been combined with smart scheduling policies to turn off the idle nodes in order to reduce overall consumption in data centers [15]. However, none of the approaches above, quantitatively bound the impact of power management on performance and predict the combined effects of performance and power on scalable systems.

Energy modeling has been used on both the architecture and system level. Wattch [16], SimplePower [17], SoftWatt [18], SimpleScalar [19], and IPP [20] are all architecture level simulations. All of these approaches are in the context of computer architecture as opposed to computer systems that would include major system components and on-off chip power behaviors of applications. These simulators are often execution-driven using compiled code for instrumentation; whereas our model is strictly analytical and more scalable for large systems. At the system level, Ge and Cameron [21] proposed a power-aware speedup model as a generalization of Amdahl's Law for power and accurately captures some of the effects of energy on speedup. However, it does not provide the root cause for poor energy-performance scalability and strategies to maintain high-energy efficiency. Ding et al [22] proposed a circuit-level simulation model to analyze power-performance tradeoffs; however, this model focuses on circuit level design which makes it less practical for modeling large scale systems. Jiang et al [23] proposed a high level Energy Resource Efficiency metric which highlighted the various energy-performance tradeoffs but does not specifically identify reasons for poor energy scalability.

B. *The Iso-Energy-Efficiency model (I-E-E model)*

In this section we briefly review our I-E-E model and explain how it is used.

We developed the I-E-E model to address two key points. Firstly predict total energy consumption of large-scale systems and secondly model how energy efficiency is affected by altering system parameters such as CPU frequency, problem size, node count and interconnect. For a complete list of parameters used in the model please see Table I.

We define E_1 as the total energy consumption of sequential execution on one processor and E_p as 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 and running extra system components. So we can define the **general form** of I-E-E model as:

$$\theta = EE = \frac{E_1}{E_p} = \frac{E_1}{E_1+E_o} = \frac{1}{1+\frac{E_o}{E_1}} = \frac{1}{(1+\frac{\alpha T_o P_{total-idle}+W_{co}t_c\Delta P_c+W_{mo}t_m\Delta P_m}{\alpha T_1 P_{total-idle}+W_c t_c\Delta P_c+W_m t_m\Delta P_m})} \quad (1)$$

Where: $T_1 = (W_c t_c + W_m t_m)$ *and* $T_o = (W_{co}t_c + W_{mo}t_m + \sum_{i=1}^{p} T_{net_i})(1 \le i \le p)^*$

* *For simplicity,* $\sum_{i=1}^{p} T_{net_i}$ *can be estimated by* $Mt_{msg} + Bt_{Byte}$. *For more complicated communication patterns, communication models such as Pairwise exchange/Hockney [28], LogP [29], LogGP [30], PLogP [31], BSP [32], etc, can be used to replace* $\sum_{i=1}^{p} T_{net_i}$ *according to specific parallel algorithm.*

Table I
SUMMARY OF PARAMETERS USED IN THIS PAPER.

Parameters	**Machine dependent parameters**
W_c	Total on-chip computation workload †
W_m	Total off-chip memory access workload †
W_{co}	Total parallel computation overhead
W_{mo}	Total number of memory access overhead in parallelization
M	Total number of messages packaged in parallelization ⋆
B	Total number of bytes transmitted ⋆
p	Number of nodes (typically one core, single processor)
N	Problem size or total amount of work (in instructions or computations)
α	Corrector factor including components such as overlap among computation, memory access and network transmission, additional cost by code scaling, etc.
T_o	Total overhead time due to parallelism
T_1	Total sequential execution time of an application running on a single processor
Parameters	**Application dependent parameters**
Time related	
t_c	$\frac{CPI_{on}}{f}$ [24] . Average time per on-chip computation instruction (including on-chip caches and registers) †
t_m	Average memory access latency ⊕
t_{msg}	Average start up time to send a message ⋄
t_{byte}	Average time of transmitting a 8-bits word ⋄
t_{IO}	Total IO access time ϒ
Power related	
P_{c-on}	Average CPU power in running state Ω
P_{c-idle}	Average CPU power in idle state Ω
ΔP_c	$P_{c-on} - P_{c-idle}$ Ω
P_{m-on}	Average memory power in running state Ω
P_{m-idle}	Average memory power in idle state Ω
ΔP_m	$P_{m-on} - P_{m-idle}$ Ω
P_{IO-on}	Average IO device power in running state Ω
$P_{IO-idle}$	Average IO device power in idle state Ω
ΔP_{IO}	$P_{IO-on} - P_{IO-idle}$ Ω
P_{other}	Average sum of other devices' power such as motherboard, System/CPU fans, NIC, etc. Ω
$P_{total-idle}$	Average system power on idle state Ω
f	The clock frequency in clock cycles per second
θ	Energy efficiency or user desired energy efficiency. For N scaling, $0 < \theta \le 1$; for f scaling case, $\theta > 0$ and possibly larger than 1 due to normalized to lowest frequency on one processor

Parameters measured by PowerScale using: Perfmomn+libpfm4.0 †; TAU [25] ⋆; LMbench [26] ⊕; MPPTest [27] ⋄; PowerPack 3.0 [4] Ω; /proc/stat [20] ϒ.

For the detailed mathematical derivation, validations, and accuracy analysis of (1), please refer to [10]. In Section V, both problem size and frequency scaling approaches have their own derived form from (1) to show different perspectives on improving energy efficiency.

IV. Methodology

In this paper we are using the I-E-E model to show the affects of problem size or frequency scaling on maintaining or improving system-level energy efficiency. To do this we need to solve two correlation functions when system scales:

$$N = F(p,\theta), \text{ under fixed processor frequency} f. \quad (2)$$

$$f = F(p,\theta), \text{ under fixed problem size } N. \quad (3)$$

$F(p,\theta)$ in (2) and (3) is a function of p with user desired energy efficiency θ as a constant. Using the model (1),

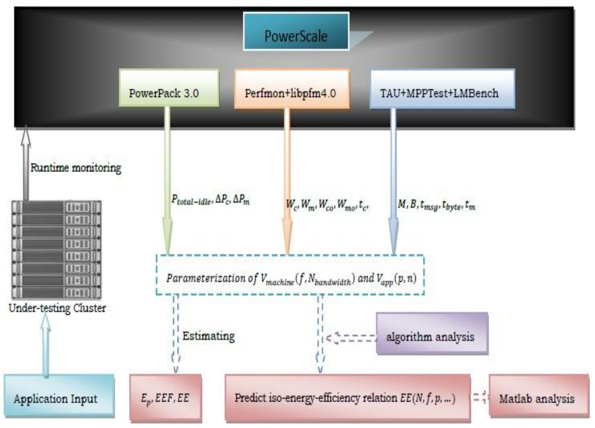
PowerScale
PowerPack 3.0
Perfmon+libpfm4.0
TAU+MPPTest+LMBench
Runtime monitoring
$P_{total-idle}, \Delta P_c, \Delta P_m$
$W_c, W_m, W_{co}, W_{mo}, t_c,$
$M, B, t_{msg}, t_{byte}, t_m$
Under-testing Cluster
Parameterization of $V_{machine}(f, N_{bandwidth})$ and $V_{app}(p, n)$
Estimating
algorithm analysis
Application Input
E_p, EEF, EE
Predict iso-energy-efficiency relation $EE(N, f, p, ...)$
Matlab analysis

Figure 3. Runtime monitoring tool PowerScale software components and data flow diagram.

we find what the effect (improve, degrade or no effect) on system energy efficiency is when scaling problem size or processor frequency while p scales up for a specific application.

When you know effects of problem size or frequency scaling on energy efficiency for a specific application, we then find valid N or f values to maintain or improve energy efficiency θ under specific p. In the following case studies we show how to use (1)(2)(3) to evaluate the energy efficiency of several commonly used parallel applications.

A. Software

Precisely measuring the machine and application parameters such as total on-chip computation workload is the key to building highly accurate model. For a complete list of dependent parameters see Table I. Measuring large numbers of parameters by hand becomes labor intensive and error prone. To solve this problem we implemented a program called PowerScale to automate the process.

Figure 3 shows our semi-automatic runtime monitoring tool PowerScale and its major software components. Table I in Section III-B is annotated to show parameters automatically measured by PowerScale. PowerScale is architecture independent because it will map the architecture specific hardware counters to the correct model parameters. Hardware counters are able to capture effects such as increased memory controller contention caused by a different number CPU cores.

B. The estimating procedure for N and f

As explained in earlier we define N as problem size and f as CPU frequency. In order to maintain system energy efficiency at the user desired level we can estimate N and f for a system size (p).

Figure 4 shows pseudo code for estimating problem size N in order to maintain energy efficiency while p scales up,

```
01: Collect and process the runtime monitoring data and
    machine/application dependent parameters used for
    model.
02: Plug the parameters into the I-E-E model.
03: Pre-define the value of θ ( it can be a loop with a
    group of interested  θ values)
04: for p=1 to (max range of p user wants to evaluate) do
05:   Try to solve N=F(p,θ) under fixed frequency
06:   if !N or N or out of range then
07:       Give value -1 to data structure *A [p]
          and return to the for loop to continue
08:   else *A [p]=1 and *(A [p]-> next)=N
09:   end if
10: end for
11: Find the problem sizes with *A [p]=1 and output their
    *(A [p]->next) with corresponding p
```

Figure 4. Pseudo code for estimating N in order to maintain energy efficiency.

described in (2). (3) can use a similar method to estimate f. However, finding a frequency level to maintain energy efficiency while system scales is sometimes impractical due to the limitations of DVFS [2]. More discussion about the frequency scaling approach can be found in Section V-D.

V. CASE STUDIES AND DISCUSSION

A. Experimental setup

Table II shows the configurations of two power-aware clusters: *SystemG* and *Dori*. Most of the experiments and modeling were conducted on *SystemG*. Some comparison tests in Section V-D are done on *Dori*.

Table II
SYSTEM CONFIGURATION FOR *SystemG* AND *Dori* CLUSTERS.

Cluster	**SystemG**	**Dori**
System size (nodes)	325	8
Processor	2 x Quad-core Xeon	2 x Dual-core Opteron
Memory	8GB	6GB
L1 cache	32KB	64KB
L2 cache	6MB	1MB
Interconnection	40Gbytes/s InfiniBand	1Gbytes/s Ethernet
CPU frequencies	2.8, 2.4 GHz	1.8, 1.6, 1.4, 1.2, 1.0 GHz

B. Problem size scaling

For scaling problem size N to achieve constant energy efficiency, we derive from the general form of the I-E-E model (1):

$$EE(N, p) = \theta = \frac{E_{1,N}}{E_{p,N}} \tag{4}$$

(where $0 < \theta \leq 1, 0 < N \leq$ total allowable memory for p processors, $E_{1,N}$ and $E_{p,N}$ are the system energy consumption for running on one and p processors with problem size N. For simplicity, CPU has fixed frequency.)

In this subsection, we will apply (4) to three parallel applications with different runtime execution patterns and discuss how to solve (2) from Section IV in order to estimate problem size N for specific parallelism p to maintain system-wide energy efficiency θ under fixed frequency. Frequency f is fixed at 2.8 GHz. For the simplicity of modeling network related parameters M and B, we assume a one port communication model and bi-directional communication links. The applications used are:

1) High Performance LINPACK (CPU and Memory intensive with non-ignorable communication);
2) Simple row-oriented matrix multiplication (high computation to communication ratio with a large memory footprint);
3) and 3D Fourier transform (communication intensive).

We use these applications to categorize common parallel applications based on their individual execution pattern and overall increasing rate of N for maintaining energy efficiency while system scales up.

1) Case A - High Performance LINPACK (HPL) : High Performance LINPACK is widely used in the HPC community to measure peak system performance . It is also the sole performance benchmark used by the TOP 500 list [33] to rank the worlds fastest supercomputers. HPL is a portable benchmark that solves a (random) dense linear system ($Ax = b$) in double precision (64 bits) arithmetic on distributed-memory computers. The problem size (N) specifies the order of matrix A (therefore A has $N * N$ elements). In order to solve x, we first need to apply LU factorization algorithm to the coefficient matrix [A, b] (shown in Figure 5) and then use backward substitution method to achieve x. Figure 5 shows that the coefficient matrix has been logically divided into block size $NB*NB$ (computational granularity) and then distributed on a $P*Q$ (here $2*4$) grid of processes for LU factorization. HPL system configuration has several parameters affecting overall maximum performance in terms of Gflops, including problem size N, block size NB, rows of process grid P and columns of process grid Q. Compared to these four parameters, other HPL parameters are considered to have a trivial impact on maximum performance [34]. In our modeling process shown in Table III, NB, P and Q are involved in the network performance related parameters M and B. Since the focus of this case is to show how scaling problem size N to maintain or improve system-wide energy efficiency instead of performance maximization, we fix the value of NB to 32 and set P and Q approximately equal, with Q slightly larger than P according to p: $P*Q = p$ and $P \leq Q$. Work such as [34] discuss how to finely tune NB, P and Q to maximize the performance benefit for both data distribution and parallel computation. Detailed discussion of these three parameters is beyond the scope of this paper.

Correlation (2) can help us estimate the value of N according to the number of processes p and desired energy efficiency under fixed frequency. Users can preset θ based on (4) function under an energy efficiency upper bound for

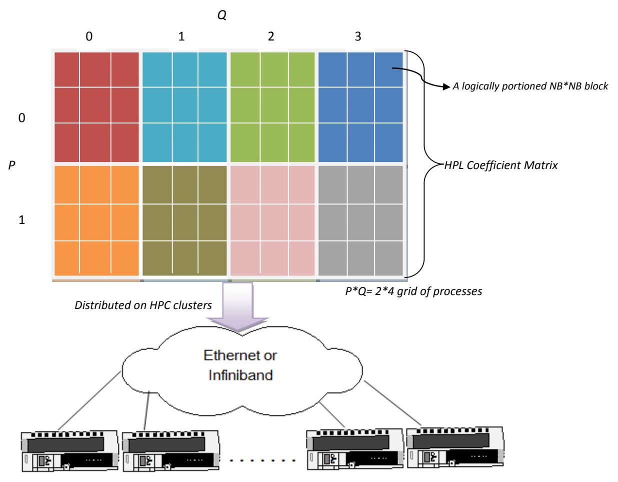
Q
0
1
2
3
0
P
1
A logically portioned NB*NB block
HPL Coefficient Matrix
P*Q= 2*4 grid of processes
Distributed on HPC clusters
Ethernet or Infiniband

Figure 5. Data distribution for HPL on the logical Process Grid.

a specific number of p. In (4), the memory upper bound we set for N in HPL is around 80% of total parallel system memory to avoid memory swapping and TLB misses.

Table III
MACHINE AND APPLICATION DEPENDENT PARAMETER ESTIMATION FOR HPL.

Machine	**Estimation**
t_c	$\frac{4.31}{f} * 10^{-10}$
t_m	$1.12 * 10^{-7}$
t_{msg}	$2.53 * 10^{-5}$
t_{Byte}	$1.82 * 10^{-8}$
$P_{(total-idle)}$	$27.68 * f^{\gamma}$ †
ΔP_c	$2.19 * f^{\gamma}$
ΔP_m	$1.56 * f^{\gamma}$

Application	**Estimation**
α	0.989
W_c	$107927 * N^2$
W_m	$14.6 * N^2$
W_{co}	$3600 * log_2^p N^2$
W_{mo}	$(25.4 * N^2 * log_2^p)/p$
M	$N*((NB+1)*log_2^p+p)/NB$
B	$16 * N^2/p^{1/2}$

† For *SystemG* we fix γ to 2.

Table III shows the machine and application dependent parameters estimated by PowerScale and manual analysis of HPL. Based on the parameter estimation in Table III and (1)(4), we are able to build the I-E-E model for HPL using a fixed frequency of 2.8GHz:

$$EEF_{HPL}(N,p) = \frac{214.6*10^5 T_o + 0.98*\log_2^p N^2 + 3.48*N^2 \log_2^p /p}{214.6*10^5 T_1 + 30.49N^2} \quad (5)$$

(Where $T_1 = 1.82*10^{-5}N^2$ *and* $T_o = 5.54*10^{-7}\log_2^p N^2 + 28.4*10^{-7}\log_2^p /p + 7.9*10^{-7}*N(33\log_2^p + p) + 2.9*10^{-7}N^2/p^{1/2}$*)*

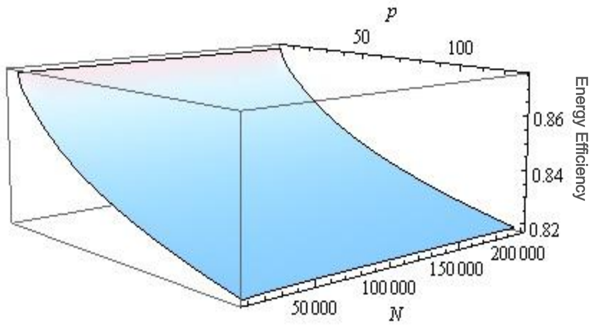

Figure 6. 3D illustration of energy efficiency scaling for HPL while scaling p (from 4 to 128) and problem size N.

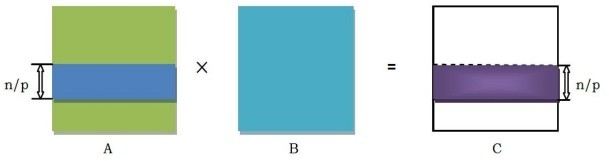

Figure 7. Illustration for ROMM.

We use (5) to plot energy efficiency scaling for HPL under fixed CPU frequency shown in Figure 6.

Figure 6 shows that when we scale up the system size p, the overall system energy efficiency degrades under fixed frequency. After following the procedure in Section IV-B, we discover that there is no possible rate to scale N in order to keep up with increasing p to maintain energy efficiency θ. In other words, overall energy efficiency does not react to more workload while system scales up, so the problem size scaling approach for improving energy efficiency is not feasible for HPL. This observation also shows that, for HPL, energy is more efficiently used for parallel execution if we increase problem size to the upper bound with the fixed p since energy efficiency stays approximately constant while N scales. HPL is a typical weak scaling case stressing both CPU and memory, and its energy efficiency pattern using scaling N will be categorized in Section V-C.

2) Case B - Row-Oriented Matrix Multiplication (ROMM) : In this case study, we demonstrate how to use our I-E-E model to estimate N for ROMM in order to maintain constant system-wide energy efficiency. ROMM has a high computation to communication ratio with a large memory footprint. To simplify our analysis, we assume that A, B and C are all $n*n$ matrices and the total number of computations is $N = 2n^3$ (including both additions and multiplications). Figure 7 shows each process is responsible for computing n/p rows of C and needs to refer to n/p rows of A and every element of B.

Table IV shows the machine and application dependent parameters for *SystemG* under a fixed frequency of 2.8GHz:

Table IV
MACHINE AND APPLICATION DEPENDENT VECTORS ESTIMATION FOR ROMM.

Machine	**Estimation**
t_c	$\frac{3.64}{f} * 10^{-10}$
t_m	$1.12 * 10^{-7}$
t_{msg}	$2.53 * 10^{-5}$
t_{Byte}	$1.82 * 10^{-8}$
$P_{(total-idle)}$	$27.68 * f^2$
ΔP_c	$1.43 * f^2$
ΔP_m	$0.99 * f^2$

Application	**Estimation**
α	0.854
W_c	$26.1 * N$
W_m	$4.35 * 10^{-2} N$
W_{co}	$2.5 * 10^{-3} Np$
W_{mo}	$6.17 * 10^{-3} * \frac{N}{(p-1)}$
M	$2.53 * 10^{-5} * (p-1) N^{\frac{2}{3}}$
B	$(p-1)(1 + N^{\frac{2}{3}} + \frac{2}{p} * N^{\frac{2}{3}})$

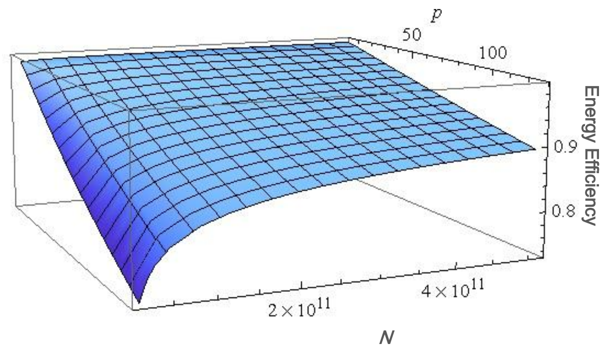

Figure 8. 3D demonstration of energy efficiency trend while scaling number of p and problem size N on *SystemG*.

We then use Table IV and (1)(4) to build the I-E-E model:

$$\begin{aligned} & EEF_{ROMM}(N,p) \\ & = \frac{185.3T_o + 3.64 * 10^{-11} Np + 50.53 * 10^{-10} * \frac{N}{p-1}}{185.3T_1 + 4.17 * 10^{-7} N} \qquad (6) \\ & EE_{ROMM}(N,p) = \frac{1}{1 + EEF_{ROMM}(N,p)} \end{aligned}$$

(Where $T_o = 0.325*10^{-11} Np + 6.479*10^{-10} * \frac{N}{p-1} + 2.53*10^{-5} * (p-1) * N^{2/3} + 1.82*10^{-8} * (p-1)(1 + N^{2/3} + \frac{2}{p} * N^{2/3}$ *and* $T_1 = 3.8498*10^{-8} N$*)*

Figure 8 is the rendered image of (6). We now use the pseudo code from Section IV-B to solve I-E-E correlation $N_{ROMM} = F(p, \theta)$. For example, we set $\theta = 0.85$ and $\theta = 0.98$ with solutions:

$\theta = 0.85 = EE_{ROMM}(N,p) = EE_{ROMM}(1.3824 * 10^{10}, 64) = EE_{ROMM}(1.1128 * 10^{11}, 128)$
$\theta = 0.98 = EE_{ROMM}(1.3824 * 10^{10}, 8) = EE_{ROMM}(1.8432 * 10^{10}, 16) = EE_{ROMM}(3.093 * 10^{10}, 32)$

Using (6) we can approximately estimate the iso-energy-efficiency relationship between N and p: $N_{ROMM} = f^{'}(\theta)\phi(log^p)$, where $f^{'}(\theta)$ is a function of θ and $\phi(x)$ is the set of all functions that have the same growth rate as x. From the equation, we can infer that the total problem size N needs to grow with the number of processors at an overall rate of $\phi(log^p)$ in order to maintain system-wide desired energy efficiency $\theta(N > 0)$. So ROMM's energy efficiency scalability is less than Embarrassingly Parallel which is an approximately ideal case for iso-energy-efficiency but better than linear growth applications. This shows that the problem size scaling approach can be used to help maintain energy efficiency for ROMM. For performance efficiency analysis, we use Michael J. Quinn's [35] performance isoefficiency relation to evaluate ROMM. We find that in order to maintain a constant level of performance efficiency, memory utilization per processor needs to increase as $\phi(p)$. We conclude that in general, system energy efficiency is a combination of the effects of power and performance and so may not be evaluated by only using basic performance efficiency analysis ($\phi(log^p) \neq \phi(p)$).

3) Case C - 3D Fourier Transform (3D-FT) : 3D-FT is one of the eight benchmark suites in the NAS Parallel Benchmark which uses a divide-and-conquer strategy to evaluate a 3D partial differential equation. As a communication intensive application, 3D-FT is composed of several computation and communication phases, and stresses CPU, memory and network during execution. Among all the communication calls, all-to-all communication dominates the overhead. We use PowerScale to measure both machine and application dependent parameters, shown in Table V.

Table V
MACHINE AND APPLICATION DEPENDENT VECTORS ESTIMATION FOR 3D-FT.

Machine	**Estimation**
t_c	$\frac{6.41}{f} * 10^{-10}$
t_m	$1.12 * 10^{-7}$
t_{msg}	$2.53 * 10^{-5}$
t_{Byte}	$1.82 * 10^{-8}$
$P_{(total-idle)}$	$27.68 * f^2$
ΔP_c	$3.4 * f^2$
ΔP_m	$0.76 * f^2$

Application	**Estimation**
α	0.86
W_c	$1.06 * 10^4 N$
W_m	$9.49N$
W_{co}	$4.46 * 10^3 * Nlog_2^p$
W_{mo}	$-0.73 * Nlog_2^p$
M	22
B	$\frac{4N}{4^{(log_2^p - 1)}}$

We can calculate energy efficiency for 3D-FT on *SystemG*:

$$EE_{3D-FT}(N,p) = \frac{1}{1 + \frac{1.97\log_2^p + 2.8p(p-1)(\frac{11500}{N} + \frac{0.376}{4^{(\log_2^p - 2)}})}{226.56}}$$

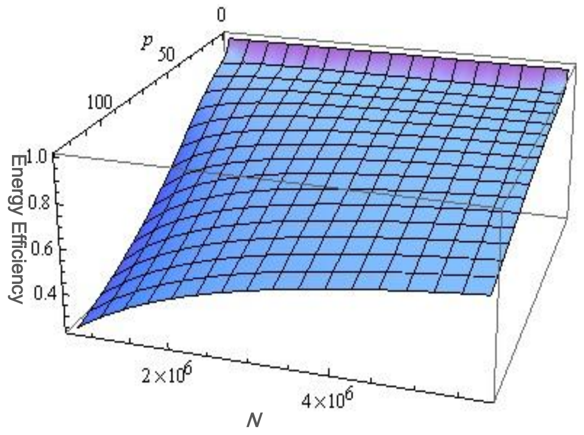

Figure 9. Illustration of 3D-FT's energy efficiency scaling with frequency fixed to 2.8 GHz.

(7)

According to (7) and Section IV-B we can solve the correlation $N_{3D-FT} = F(p,\theta)$ while p and N scale simultaneously. For example, we find that $\theta = 0.9$ when $p = 8$ and $N = 0.8 * 10^6$. Looking at the graph, 90% energy efficiency is a reasonable target θ for this case. We need to find N when p scales to 16 and above. We can estimate the workload as:

$\theta = 0.9 = EE_{3D-FT}(0.42*10^6, 8) = EE_{3D-FT}(6.185*10^6, 16)$

In the example above, users can easily find that there is no valid N value that will keep 90% system-wide energy efficiency when p scales to 32 and above. However, users can still choose a value for N that is close to 90% by calculating max(θ) when p is fixed when running the application. By using (7) and (4), we can estimate correlation between N and p under fixed f for 3D-FT on *SystemG*: $N_{3D-FT} = f^{'}(\theta)\phi(\frac{p^2}{log_2^p})$. We can infer that N needs to grow with p at an overall rate of $\phi(\frac{p^2}{log_2^p})$ in order to maintain system-wide desired energy efficiency ($N > 0$). 3D-FT has a larger communication to computation ratio compared to ROMM, thus energy efficiency decreases faster while p scales up due to faster growth of E_p caused by communication overhead. Overall energy efficiency improvement reacts to increasing problem size faster for 3D-FT than ROMM, especially for larger system size (for instance, when $p = 128$). The problem size scaling approach can be effectively used for 3D-FT to reduce the negative impact on energy efficiency caused by increasing p.

C. Discussion and Categorization

Figure 10 shows classification of several applications with individual N's growth rate. Based on the case studies, we can categorize parallel applications by how they are affected by the problem size scaling approach:

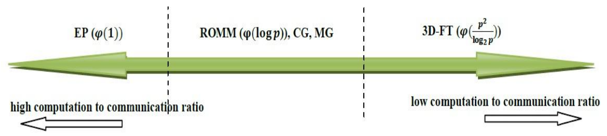
EP ($\varphi(1)$)
ROMM ($\varphi(\log p)$), CG, MG
3D-FT ($\varphi(\frac{p^2}{\log_2 p})$)
high computation to communication ratio
low computation to communication ratio

Figure 10. Categories of applications based on computation to communication ratio.

1) Purely computation intensive with minimal communication (ideal case). These applications do not react to increasing problem size to maintain energy efficiency while scaling up p. Neither p nor N scaling will significantly improve or degrade the overall energy efficiency. For example Embarrassingly Parallel.
2) Both computation and non-negligible communication. Applications such as ROMM, Conjugate Gradient (CG) and Multigrid (MG) from NPB belong in this category. This type of application cannot maintain system energy efficiency as well as the ideal case. The problem size scaling approach is able to help these applications improve their overall energy efficiency.
3) Communication intensive. Both performance and energy efficiency do not scale well. 3D-FT belongs in this category. The problem size scaling approach is able to help these applications improve their overall energy efficiency.
4) Both computation and memory intensive with non-negligible communication. Their system-wide energy efficiency is not sensitive to changing workload. Unlike the ideal case, increasing system size will cause energy efficiency degradation. However, scaling workload will not help improve energy efficiency while p scales. Examples such as HPL and LU from NPB.

D. *Frequency scaling*

Frequency scaling can be another effective approach to maintain or improve overall system energy efficiency while system size scales. For a DVFS-based power aware cluster, we assume each of its compute nodes has s power/performance modes or processor frequencies available $\{ f_1, f_2, f_3, , f_s \}$ satisfying $f_{min} = f_1 < f_2 < ... < f_s = f_{max}$. For the simplicity of discussion, problem size N is fixed in this section. We extend the general model described in (1):

$$EE(f,p) = \theta = \frac{E_{1,f_{min}}}{E_{p,f_\delta}} \tag{8}$$

(Where $1 \le \delta \le s$. We denote total energy consumption for application with fixed workload running on one processor with the minimum frequency as $E_{1,f_{min}}$ and on p processors with frequency f as E_{p,f_δ})

While problem size N can scale continuously, the number of CPU frequencies available are typically very limited and non-congtiguous, so it is not practical to use the method in Section IV-B to find f in order to reach specific desired θ. We focus on studying how to use frequency scaling to improve energy efficiency instead. In this subsection, we use 3D-FT and HPL as case study candidates.

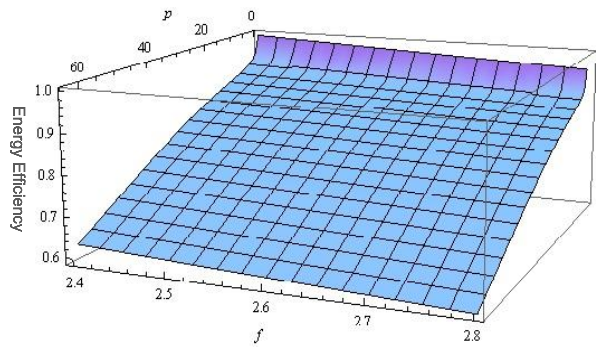

Figure 11. Illustration of 3D-FT under fixed problem size $N = 1.15*10^6$ on *SystemG* while p scales up.

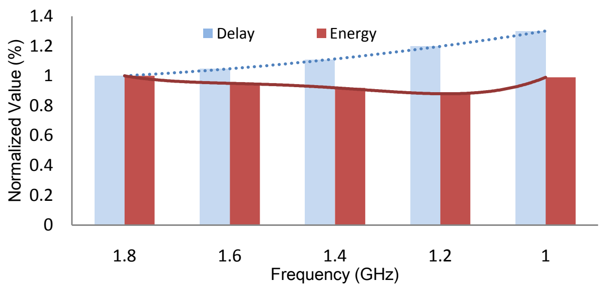

Figure 12. Energy and Delay of 3D-FT running on *Dori* cluster with five available frequencies. Results are normalized with 1.8 GHz situation.

1) Studies: Frequency scaling approach on 3D-FT and HPL : Reusing the parameters from Table V, we model 3D-FT under fixed problem size $N = 1.15 * 10^6$, shown in Figure 11. Figure 11 shows that for fixed problem size, scaling down the frequency (from 2.8GHz to 2.4 GHz) will save total energy consumption (normalized to $E_{1,f_{min}}$) and improve overall system energy efficiency. However, scaling frequency does not improve energy efficiency significantly. This is because 3D-FT is a communication intensive application and the effect of frequency change on on-chip workload is diminishing while p scales up. Also, there are only two available frequency levels supported on our test cluster *SystemG*. This limits the possibility of higher energy efficiency improvement by further scaling down frequency. In order to investigate whether continuing to lower the CPU frequency can further reduce the total energy consumption we also ran 3D-FT on another power-aware cluster *Dori* that has more frequency levels (see Table II).

Figure 12 shows that the pattern of energy reduction continued on our second cluster as frequency was lowered until 1.2 Ghz. Between 1.2 and 1Ghz, energy consumption increases because the performance penalty of lowering frequency offsets the benefit of lower CPU power. For the HPC

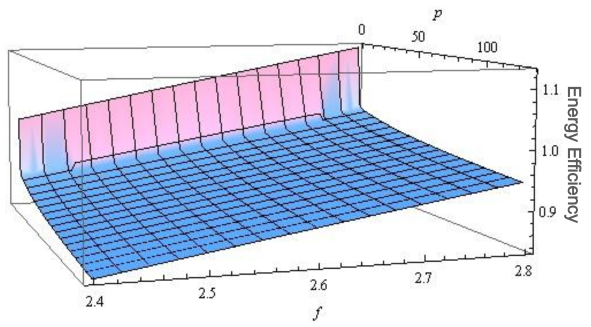

Figure 13. Illustration of HPL under fixed problem size N=10000 on *SystemG* while p scales up.

community, the balancing point in this context is the system configuration that maximizes energy efficiency. Using (3), users can know more about how to scale frequency to achieve better energy efficiency.

For the applications that are not dominated by communication such as HPL, scaling to higher frequency may improve the overall energy efficiency. Using Table III and (3), we model $f_{HPL} = F(p, \theta)$ under fixed workload. Figure 13 shows that HPL exhibits different behavior than the 3D-FT case. Using a higher frequency for processors improves the total energy efficiency. The graph area that is above 1.0 energy efficiency is because we use $E_{1,f_{min}}$ to normalize all the energy efficiency results. For a computation and memory intensive benchmark such as HPL, higher frequency will help achieve better performance and less latency. Also, unlike 3D-FT, the frequency impact on on-chip workload will not dramatically diminish when system scales up due to its higher computation to communication ratio. Since total system energy consumption E is a sum of products of average power and delay over all the computing nodes $E = \sum_{1}^{\#nodes} \overline{Power} * Delay$, performance benefits overcome the increased average power consumption caused by higher frequency and result in a reduced total energy consumption and improved system-wide energy efficiency.

In addition to applying an appropriate frequency to all the operating processors to save energy for a communication intensive case shown in Figure 11, further energy reduction is possible at the application level by applying DVFS to "communication slack". Application level optimizations are beyond the scope of this work.

VI. Conclusions and future work

In this paper, we have modeled the effects on energy efficiency of scaling problem size and CPU frequency. We proposed two correlation functions derived from our I-E-E model and have shown how they can accurately estimate overall energy efficiency. In this section we compare and contrast how effective the approaches are for improving system energy efficiency. Finally we outline our future work.

A. Problem size scaling

We have proposed a methodology that is able to identify whether problem scaling is a feasible technique to help applications maintain or improve system-wide energy efficiency while system size scales up.

The correlation function $N = F(p, \theta)$ can be used to estimate valid problem size N according for specific system size p under fixed user level desired energy efficiency value θ in order to maintain efficiency. Using 3D-FT, HPL and ROMM as examples, we have shown how to calculate application specific rates for increasing N to keep constant energy efficiency. Finally, we classified parallel applications into four categories based on how effective problem size scaling is for maintaining or improving energy efficiency.

We have shown that the problem size scaling technique can be used to effectively improve energy efficiency for some specific applications. In the HPC community, using this approach can help users more effectively use energy, allowing them to solve bigger problems with limited resources.

B. Frequency scaling

We have also shown that frequency scaling can be used as an effective tool to further improve energy efficiency.

Using HPL and 3D-FT as examples we show how to use the I-E-E model and $f = F(p, \theta)$ to locate the optimal frequency level in order to maximize total system energy reduction and energy efficiency. We also discussed the balance of improving energy efficiency whilst limiting the performance impact.

Frequency scaling is used to help users reduce total energy consumption in order to improve energy efficiency. In contrast to problem size scaling, while this technique does not allow for greatly improved system utilization, it does reduce power consumption, which will reduce costs.

C. Summary

Table VI presents the Pros and Cons for the problem size and frequency scaling approaches:

D. Future work

Currently, we have to manually analyze an application to ensure the prediction accuracy of the model. To reduce the amount of work required to apply the model we want to make PowerScale fully automatic. Another way would be to simplify the model at the expense of some flexibility and accuracy.

We would like to explore using I-E-E and the techniques in this paper to model the energy efficiency of heterogeneous architectures such as a GPGPU cluster. We are also interested in modeling energy consumption for the Intel Nehalem architecture due to its automatic power bounded DVFS and overclocking 'Turbo Boost' functionality.

A limitation of the current model is that it does not work for imbalanced parallel workloads. In the future we will

Table VI
PROS AND CONS FOR WORKLOAD AND FREQUENCY SCALING APPROACHES FOR IMPROVING OVERALL SYSTEM-WIDE ENERGY EFFICIENCY.

Scaling	Pros	Cons
f	1) From the view point of saving total energy in order to improve energy efficiency. 2) Easy to apply.	1) Limited levels of frequencies available to scale. This also limits the rate of system energy efficiency improvement. 2) Encounter a certain degree of performance penalty.
N	1) From the view point of more efficiently utilize energy to complete more workload. 2) Has a bigger range to scale and very flexible.	1) Does not fit for the applications that have very limited input data range or underlying systems that have very small memory.

extend the model for this scenario, this will require a more sophisticated methodology.

REFERENCES

[1] W. Feng and K. W. Cameron, "The Green500 List: Encouraging Sustainable Supercomputing," Computer, vol. 40, pp. 50 - 55 Dec. 2007.

[2] J. Li and J. F. Martinez, "Dynamic power-performance adaptation of parallel computation on chip multiprocessors," in The Twelfth International Symposium on High-Performance Computer Architecture, 2006, pp. 77-87.

[3] S. Song, R. Ge, X. Feng, and K. W. Cameron, "Energy Profiling and Analysis of the HPC Challenge Benchmarks," International Journal of High Performance Computing Applications, vol. 23, pp. 265-276, 2009.

[4] R. Ge, X. Feng, S. Song, H. Chang, D. Li and K. W. Cameron, "PowerPack: Energy Profiling and Analysis of High-Performance Systems and Applications," IEEE Transactions on Parallel and Distributed Systems, vol. 99, pp. 658-671, 2009.

[5] V. W. Freeh and D. K. Lowenthal, "Using multiple energy gears in MPI programs on a power-scalable cluster," presented at the Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming, Chicago, IL, USA, 2005.

[6] V. W. Freeh, F. Pan, N. Kappiah, D. K. Lowenthal, and R. Springer, "Exploring the Energy-Time Tradeoff in MPI Programs on a Power-Scalable Cluster," presented at the Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers - Volume 01, 2005.

[7] R. Ge, X. Feng, and K. W. Cameron, "Performance-constrained Distributed DVS Scheduling for Scientific Applications on Power-aware Clusters," presented at the Proceedings of the 2005 ACM/IEEE conference on Supercomputing, 2005.

[8] R. Ge, X. Feng, and K. W. Cameron, "Improvement of Power-Performance Efficiency for High-End Computing," presented at the Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 11 - Volume 12, 2005.

[9] R. Ge, X. Feng, W.-c. Feng, and K. W. Cameron, "CPU MISER: A Performance-Directed, Run-Time System for Power-Aware Clusters," in International Conference on Parallel Processing, ICPP, Xi'an, China, 2007, p. 18.

[10] S. Song, C.-Y. Su, R. Ge, A. Vishnu, and K. W. Cameron, "Iso-energy-efficiency: An approach to power-constrained parallel computation," in accepted to appear in 25th IEEE International Parallel & Distributed Processing Symposium, Anchorage (Alaska) USA, 2011.

[11] G. Amdahl, "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities," in AFIPS Conference Proceedings 1967, pp. 483-485.

[12] A. Y. Grama, A. Gupta, and V. Kumar, "Isoefficiency: measuring the scalability of parallel algorithms and architectures," in Multiprocessor performance measurement and evaluation, 1995, pp. 103-112.

[13] A. K. Mishra, S. Srikantaiah, M. Kandemir, and C. R. Das, "CPM in CMPs: Coordinated Power Management in Chip-Multiprocessors," in Supercomputing 2010, SC10, New Orleans, Louisiana, USA, NOV, 2010.

[14] M. Curtis-Maury, A. Shah, F. Blagojevic, D. S. Nikolopoulos, B. R. d. Supinski, and M. Schulz, "Prediction models for multi-dimensional power-performance optimization on many iso-energy-efficiency ," in Proceedings of the 17th international conference on Parallel architectures and compilation techniques, New York, NY,US, 2008, pp. 250-259.

[15] . Goiri, J. Fit, F. Juli, R. Nou, J. Berral, J. Guitart and J. Torres, "Multifaceted Resource Management for Dealing with Heterogeneous Workloads in Virtualized Data Centers", in 11th IEEE/ACM International Conference on Grid Computing (GRID), 25-28 Oct. 2010.

[16] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in ISCA '00: Proceedings of the 27th annual international symposium on Computer architecture, New York, NY, USA, 2000, pp. 83-94.

[17] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "The design and use of simplepower: A cycle-accurate energy estimation tool," pp. 340-345, 2000.

[18] S. Gurumurthi, A. Sivasubramaniam, M. J. Irwin, N. Vijaykrishnan, and M. Kandemir, "Using Complete Machine Simulation for Software Power Estimation: The SoftWatt Approach," in Proceedings of Eighth International Symposium on High-Performance Computer Architecture (HPCA'02), Boston, Massachusettes, 2002.

[19] D. C. Burger and T. M. Austin, "The SimpleScalar Toolset, Version 2.0," Computer Architecture News, vol. 25, pp. 13-25, 1997.

[20] S. Hong and H. Kim, "An Integrated GPU Power and Performance Model," in ISCA' 10, Saint-Malo, France, June 2010.

[21] R. Ge and K. W. Cameron, "Power-Aware Speedup," in proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium, 2007, pp. 56-56.

[22] Y. Ding, K. Malkowski, P. Raghavan, and M. Kandemir, "Towards Energy Efficient Scaling of Scientific Codes," in IEEE International Symposium on Parallel and Distributed Processing, 2008, pp. 1 - 8

[23] N. Jiang, J. Pisharath, and A. Choudhary, "Characterizing and improving energy-delay tradeoffs in heterogeneous communication systems," Signals, Circuits and Systems, 2003. SCS 2003. International Symposium vol. 2, pp. 409-412, 2003.

[24] D. A. Patterson and J.L.Hennessy, Computer Architecture: A quantitative approach, 3rd ed. San Francisco, CA: Morgan Kaufmann Publishers, 2003.

[25] S. S. Shende and A. D. Malony, "The TAU Parallel Performance System," International Journal of High Performance Computing Applications, vol. 20, pp. 287-311, 2006.

[26] (2010). LMbench - Tools for Performance Analysis. Available: http://www.bitmover.com/lmbench/

[27] (2010). MVAPICH2: MPI over InfiniBand, 10GigE/iWARP and RoCE. Available: http://mvapich.cse.ohio-state.edu/overview/mvapich2/

[28] J. Pjeivac-Grbovi, T. Angskun, G. Bosilca, G. E. Fagg, E. Gabriel, and J. J. Dongarra, "Performance analysis of MPI collective operations," Cluster Computing, vol. 10, pp. 127-143, 2007.

[29] D. E. Culler, R. Karp, D. A. Patterson, A. Sahay, E. Santos, K.Schauser, et al., "LogP: A Practical Model of Parallel Computation," Communications of the ACM, vol. 39, pp. 78-85, 1996.

[30] A. Alexandrov, M. F. Ionescu, K. Schauser, and C. Scheiman, "LogGP: Incorporating Long Messages into the LogP model," in Proceedings of Seventh Annual Symposium on Parallel Algorithms and Architecture, Santa Barbara, CA, 1995, pp. 95-105.

[31] T. Kielmann and H. E. Bal, "Fast Measurement of LogP Parameters for Message Passing Platforms.," in Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing, 2000.

[32] L. G. Valiant, "A Bridging Model for Parallel Computation," Communications of the ACM, vol. 33, pp. 103-111, 1990.

[33] "TOP500 Supercomputing Project". Available: http://www.top500.org

[34] The NAS Parallel Benchmarks. Available: http://www.nas.nasa.gov/Resources/Software

[35] M. J. Quinn, Parallel Programming in C with MPI and OpenMP: TATA McGraw-Hill 2003.