LLNL-JRNL-698191

A Survey on Software Methods to Improve the Energy Efficiency of Parallel Computing

C. Jin, B. R. de Supinski, D. Abramson, H. Poxon, L. DeRose, M. Dinh, M. Endrel, E. R. Jessup

July 20, 2016

International Journal of High Performance Computing Applications

A survey on software methods to improve the energy efficiency of parallel computing

Chao Jin
The University of Queensland and Monash University
Bronis R. de Supinski
Lawrence Livermore National Laboratory
David Abramson
The University of Queensland
Heidi Poxon
Cray Inc.
Luiz De Rose
Cray Inc.
Minh Dinh
The University of Queensland
Mark Endrei
The University of Queensland
Elizabeth R Jessup
University of Colorado Boulder

Abstract

Energy consumption is one of the top challenges for achieving the next generation of supercomputing. Co-design of hardware and software is critical for improving energy efficiency for future large-scale systems. Many architectural power saving techniques have been developed, and most hardware components are approaching physical limits. Accordingly, parallel computing software, including both applications and systems, should exploit power saving hardware innovations and manage efficient energy use. In addition, new power-aware parallel computing methods are essential to decrease energy usage further. This article surveys software-based methods that aim to improve energy efficiency for parallel computing. It reviews the methods that exploit the characteristics of parallel scientific applications, including load-imbalance and mixed-precision of floating-point calculations, to improve energy efficiency. In addition, this paper summarizes widely used methods to improve power usage at different granularities, such as the whole system and per application. In particular, it describes the most important techniques to measure and to achieve energy efficient usage of various parallel computing facilities, including processors, memories, and networks. Overall, this article reviews the state-of-the-art of energy efficient methods for parallel computing to motivate researchers to achieve optimal parallel computing under a power budget constraint.

Keywords

Parallel computing, high performance computing, power saving, energy efficiency, auto-tuning

Introduction

Energy efficiency has become a primary concern for the design of modern computing systems, from large-scale supercomputers to multi-core laptops. In the context of high performance computing (HPC), power consumption has become a critical concern due to several factors. First, power consumption is a limiting factor that constrains processor frequency and the number of active cores. Second, high power consumption increases the total cost of ownership (TCO) of running a large-scale system, including expensive energy bills and costly cooling systems to keep temperature low. Third, the heat generated by high power consumption compromises the reliability of computing. Finally, high power consumption requires an excessive power supply that is expensive to build and to operate. The exascale supercomputer roadmap, which has identified energy efficiency as one of the top challenges, aims to achieve 10^{18} floating-point operations per second with a 20 MegaWatt power budget [Kogge et al. 2008]. Although many architectural power-saving techniques have been developed [Kaxiras and Martonosi 2008], including low frequency processors and accelerators, the power budget remains a critical challenge to exascale supercomputing [Amarasinghe et al. 2009; ASCAC 2014; Bates et al. 2013; Vaidyanathan et al. 2013]. Accordingly, parallel computing software, including system software and applications, must exploit hardware power saving advances and efficiently manage energy utilization [Dongarra et al. 2011]. In addition, new power-aware methods of parallel computing are essential to decrease energy consumption further. Consequently, understanding state-of-the-art software techniques to save energy for parallel computing is critical for researchers to propose more effective solutions to address the power challenge and to improve energy efficiency.

In this paper, we present a survey of research work that analyzes the relationship between power and performance for parallel computing and that improves the energy efficiency of parallel applications using software methods. However, a review of all research ideas proposed in the related literature is infeasible. We focus on those studies that analyze the overall system utilization and energy efficiency aspects of parallel applications, including power-aware algorithms and tools to optimize energy efficiency. This survey does not cover studies that only improve parallel computing performance although the performance improvement is likely to reduce energy usage.

Presently, hardware innovations provide a rich set of power-saving techniques that software can exploit. For example, heterogeneous systems offer the opportunity to exploit extremely high concurrency with modest energy consumption using accelerators such as GPUs (Graphic Processing Units), and coprocessors, such as the Intel Xeon Phi (also known as the MIC or Many Integrated Core architecture). Supercomputers are already equipped with these sorts of processors and the associated deep memory hierarchy. This paper reviews software methods that utilize several important hardware power-saving techniques, including DVFS (dynamic voltage and frequency scaling), NTV (near threshold voltage), low frequency processors like ARM, accelerators such as GPUs, coprocessors such as the Xeon Phi, and power-aware networks.

The energy consumption of a parallel application is closely correlated to its performance. However, improving performance and efficiently managing power may conflict with each other because faster speeds frequently come from using more resources less efficiently, which may excessively increase power consumption. In addition, the complexity of this relationship is exacerbated by the increasingly large number of CPU cores and new heterogeneous computing facilities, such as GPUs and Xeon Phi coprocessors. Therefore, this paper discusses energy and performance models to analyze parallel application energy efficiency. Most importantly, this paper presents software methods to save energy for the whole system and different applications, such as MPI [MPI Forum 2012], OpenMP [OpenMP ARB 2013], and hybrid programs. In addition, it surveys optimization techniques to tune the energy consumption of scientific applications, including auto-tuning frameworks and approximation-based methods to save energy by improving performance while maintaining the desired accuracy.

The rest of this paper is organized as follows. Section **power consumption and management** discusses the power dissipation of compute components and state-of-the-art hardware power-saving techniques for an HPC system. Section **power and energy measurement** presents the methods of measuring power and energy for parallel computing. Section **energy and performance models of parallel computing** reviews energy models for analyzing the power consumption of parallel computing and metrics to evaluate energy efficiency. Section **taxonomy of energy efficient methods for parallel computing** categorizes the most important software methods to improve energy efficiency for parallel computing. Section **saving**

energy with power-aware resource management presents techniques to improve the power usage for the whole system. Section **parallelism-specific methods of energy efficient parallel computing** presents power-saving techniques applied to a single application at different parallelism granularities, such as processes, threads, and hybridization. Section **communication-oriented power saving** summarizes energy efficient methods applied to the communication layer. Section **saving energy with automatic tuning** describes automatic energy tuning that improves the energy efficiency of parallel applications. Section **saving energy with approximation methods** presents approximate methods to improve the energy efficiency of parallel computing. We provide concluding remarks and future research trends in the **conclusions section**.

Power consumption and management

This section reviews the most important hardware components and corresponding power-saving techniques that can be orchestrated by parallel software to improve energy efficiency. There are two main technical trends coupled with compute component power dissipation: power-bound and expensive data movement.

During the past decade, power constraints have transformed processor performance improvements from frequency increases to increases in the number of cores per chip. However, the future of multi-core CPUs is limited by the dark silicon phenomena [Esmaeilzadeh et al. 2011; Esmaeilzadeh et al. 2012], in which power constraints will eventually prevent the chip from using all of its cores simultaneously. In other words, even if more cores could be added on a chip, the number of inactive cores that must be powered down due to the lack of enough energy increases every process generation. To address this challenge, many different processor architectures have been developed to support extremely high concurrency with modest energy consumption, including NTV processors [Dreslinski et al. 2010; Karpuzcu 2013], ARM big.LITTLE processors [Greenhalgh 2011], GPUs, and the Intel Xeon Phi. Although which processor architecture will dominate in the future is unclear, future supercomputers will contain heterogeneous processors and a deep memory hierarchy.

The power bound affects the compute nodes of future supercomputers as well. Although future supercomputers will have more compute nodes, they potentially may not all be able to run simultaneously with peak performance due to the power limitation, which is imposed by either financial or physical reasons. *Hardware overprovisioning* [Patki et al. 2013] is also proposed for future supercomputers to fully utilize the procured power.

In addition, the computing model is gradually transforming from computationally expensive to data movement expensive [Patterson et al. 2013a]. In particular, moving a word of data, either across a node interconnection or through a deep memory hierarchy, can require orders of magnitude more time and energy than an arithmetic operation [Patterson et al. 2013a].

Power consumption of compute components

The energy consumption of large-scale HPC facilities, such as supercomputers or data centers, mainly consists of two parts: powering cooling systems and running computers. The metric Power Usage Effectiveness (PUE) [Avelar et al. 2014] has been used to measure and to drive the energy efficiency of HPC facilities. PUE is defined as the ratio between the total energy of running an IT facility and the energy specifically used to power its IT equipment, as illustrated in Equation (1). By definition, PUE is at least 1. Many recently built computing centers with advanced energy saving techniques possess a PUE close to 1 [Bates and Patterson 2013].

$$PUE = \frac{Total\ Facility\ Energy}{IT\ Equipment\ Energy} = \frac{IT+Cooling+Power\ Distribution+Lighting+Misc}{IT\ Equipment\ Energy} \qquad (1)$$

In order to measure the HPC equipment's "inside" energy efficiency, ITUE (IT-power usage effectiveness) and TUE (total-power usage effectiveness) were proposed to highlight the ratio of entire energy used to power the compute components [Patterson et al. 2013b]. ITUE and TUE are defined as Equation (2), in which compute components include processors, memory, interconnection, and storage. ITUE mainly measures the energy overhead applied to IT equipment, such as the extra energy consumed by its internal fans, power supplies, and voltage regulators, while TUE is the total energy into the HPC facility divided by the energy used by the compute components inside the equipment. Patterson et al. [Patterson et al. 2013b] analyze the energy efficiency of the Jaguar system, which is a Cray XT5 supercomputer. The result shows that Jaguar's ITUE is 1.49 and its TUE is 1.86. According to current technological trends [Shalf et al. 2010; ASCAC 2014], the processors, memories, and interconnections of future supercomputers are the most significant energy consumption compute components. Therefore, parallel software should attempt to improve the energy efficient use of these compute components.

$$\mathrm{ITUE} = \frac{\text{Total Energy into the IT Equipment}}{\text{Total Energy into the Compute Components}}\ ;\ \mathrm{TUE} = \mathrm{ITUE} \times \mathrm{PUE} \qquad (2)$$

The power dissipation of CMOS components and DVFS

The power consumption of CMOS (Complementary metal–oxide–semiconductor) circuits mainly consists of dynamic power and leakage power, denoted as Equation (3) [Kaxiras and Martonosi 2008]. Leakage power dissipation, also called static power, occurs regardless of switching activity, and is denoted as I_0V in Equation (3), where I_0 is the leakage current and V is the supply voltage.

$$P_{CMOS} = I_0V + CV^2Af \qquad (3)$$

Dynamic power is denoted as CV^2Af in Equation (3), where C is the load capacitance, V is the supply voltage, A is the activity factor, and f is the operating frequency. The aggregated load capacitance (*C*) largely depends on the wire lengths of on-chip structures. Architectural design influences this metric in several ways. For example, smaller processor cores on-chip and independent banks of cache can reduce wire lengths. The activity factor (*A*) is a fraction between 0 and 1 that refers to how often a wire actually transitions from 0 to 1 or 1 to 0. The clock frequency (*f*) not only directly influences power dissipation, but also affects supply voltage (*V*). Normally, a higher clock frequency is supported using a higher supply voltage. Therefore, supply voltage has a cubic impact on power dissipation through the V^2f portion of the dynamic power factor in Equation (3).

Typically, static power represents approximately 20% of overall power dissipation [Kaxiras and Martonosi 2008], while dynamic power dominates the power consumption of CMOS components. Using LULESH, a hydrodynamics application, Leon et al. [Leon et al. 2015] investigate the dynamic and static power consumption of several architectures, including IBM Blue Gene/Q, Intel Ivy Bridge, and AMD Piledriver. The dynamic power consumption for Ivy Bridge and Piledriver is 80% and 87% respectively. However, the static power consumption of Blue Gene/P, which is well known for its low power architecture, is more than 70%.

The dynamic power part of Equation (3) clearly illustrates the opportunity for saving power by adjusting voltage and frequency, also called dynamic voltage and frequency scaling (DVFS). In particular, scaling the supply voltage down offers the potential for a cubic reduction in power dissipation. However, it also linearly degrades performance. Therefore, scaling frequency down to save power using DVFS must recognize periods when lower processor performance is acceptable (e.g., in memory-bound or latency-tolerant regions of code). With OSPM (Operating System-directed configuration and Power Management) compatible operating systems, the performance of CPUs, including Intel and AMD processors, can be adjusted by controlling the performance state (P-State).

Hardware-enforced power bound

Many processor architectures support power capping to save power. For instance, IBM Power 6 and 7 architectures support "soft" power capping that runs the system at a lower power/performance point to save energy. AMD Bulldozer allows the user to specify a thermal design power limit for the processor. In contrast, the Intel Sandy Bridge processors allow a user to specify a time window and an associated maximum average power using the RAPL (Running Average Power Limit) interface [Intel 2011]. The processor guarantees that it will not exceed this average power during the time window. Rountree et al. [Rountree et al. 2012] investigate the variations of processor frequency under different power settings and analyze the potential power saving effect of applying this technique to parallel applications.

Accelerators and coprocessors

Accelerators, such as GPUs, and coprocessors, such as the Intel Xeon Phi, are currently the most power-efficient parallel computing architectures [Li et al. 2014]. Accordingly, heterogeneous systems equipped with NVIDIA or AMD GPUs or Intel Xeon Phi coprocessors offer the opportunity to exploit extremely high concurrency with modest energy consumption. Presently, hybrid machines equipped with GPUs dominate the Green500 list in 2015 [The Green 500]. According to the top 500 supercomputers announced in 2015 [Top 500], more than 20% of top 100 supercomputers, including 4 of the top 10 systems, are heterogeneous systems.

Embedded processors

Low power microprocessors that are used in smart phones and tablets, such as ARM processors, dominate the commodity market. The recently released ARMv8 Instruction Set Architecture (ISA) supports double-precision floating-point (FP-64) and SIMD instructions, which facilitate scientific computing. The cost and power advantage of these mobile processors have attracted the attention of the HPC community [Rajovic et al. 2013] and they have been adopted by data centers to process data-intensive applications [Li et al. 2011b]. The European Mont-Blanc project [Rajovic et al. 2013] seeks to build an energy-efficient supercomputer using ARM processors and GPU accelerators.

Near-threshold Voltage (NTV) Computing

ASIC (Application Specific Integrated Circuit) process scaling allows more transistors to be included on a chip. However, cooling limitations do not expand accordingly. As a result, although more cores can be added to a chip in the future, heat dissipation considerations may lead to some of them being inactive at any given time

[Esmaeilzadeh et al. 2012]. In addition, the gap between what can be integrated onto a chip and what can be operated keeps increasing every process generation.

A promising way to activate more cores is to reduce the supply voltage (V_{DD}). Lowering the supply voltage to slightly above the threshold voltage (V_{th}) can reduce power consumption by more than an order of magnitude. This unconventional operation regime, called near-threshold voltage computing (NTC), enables more cores to operate simultaneously under a given power budget at the cost of performance degradation [Dreslinski et al. 2010]. Moreover, NTC decreases the reliability of the system [Karpuzcu et al. 2013].

However, the increased parallelism of large-scale systems can compensate for the degraded performance of NTC. While the increase in computing errors must be tolerated, NTC is a promising way to reduce the power consumption of parallel computing. Recently, based on NTC, approximate computing, also called significance-based computing, has attracted attention in the HPC research community [Gschwandtner et al. 2014; ASCAC 2014].

Memory

Heterogeneous architectures enable more parallelism, which demands more data movement between the logic and the cores. Moreover, the performance gap between moving a word and applying an arithmetic operation to it will grow exponentially according to current technological trends [Patterson et al. 2013a]. Accordingly, data movement will account for most of the energy consumption of parallel computations. In particular, the energy to move data increases proportionally to bandwidth and transport distance (*energy* = *bitrate* × *distance*^{2} / *cross-section area of interconnect wiring* [Miller and Ozaktas 1997]). Although moving a word of data across a node interconnect consumes more energy than through a deep memory hierarchy, accessing data from memory is more frequent. Actually, even with advanced memory techniques, the amount of energy consumed by the main memory subsystem could be comparable to that of the processors [Shalf et al. 2010]. To decrease the energy consumed by memory in the software layer, applications must increase data locality to avoid unnecessary data movement.

Network

At exascale, the network can consume 10~20% of the total system power [Groves and Grant 2015]. Some experts even estimate that in the future the network may account for 30% of the total power budget of a supercomputer [Dickov et al. 2014]. In order to improve the energy efficiency of the interconnection network, many innovative technologies have been developed, including on-die interconnect fabric, inter-chip network integration, energy proportional network, and power-aware network [Alonso et al. 2006; Nedevschi et al. 2008; Saravanan et al. 2013; Saravanan et al. 2014; Groves and Grant 2015; Miwa et al. 2014; Miwa and Nakamura 2015]. Power-aware networks can save power consumed by the network fabric using techniques such as dynamic link width and frequency, and on/off links. Many experts estimate that power-aware networks will save significant energy for HPC applications [Miwa et al. 2014; Saravanan et al. 2013; Saravanan et al. 2014]. In particular, Energy Efficient Ethernet (EEE) [IEEE 802.3az. 2010] has attracted attention in the HPC community. Although these innovations can decrease energy consumed at the network layer, moving a word of data across a node interconnection can require orders of magnitude more time and energy than an arithmetic operation [Patterson et al. 2013a]. In the application layer, one of the best ways to save network energy consumption is to avoid or to reduce communication [Grigori et al. 2011; Demmel et al. 2013].

Power and energy measurement

Accurate power and energy measurements are critical for the efficient management of parallel application energy use. In particular, fine-grained measurement of power consumption, such as per application, per hardware or software component, or even per instruction, and high sampling frequency are required to provide insights for energy usage optimization. The existing power measurement methods mainly consist of two groups: direct measurement, and model-based. Several projects [Intel 2011; David et al. 2010; Hart et al. 2014; Ge et al. 2009; Venkatesh et al. 2013; Yoshii et al. 2012] have explored APIs that make fine-grained power and energy consumption measurement for various granularities accessible at the application level so that programs can automatically make software engineering decisions to manage their power usage at runtime. Most direct power measurements use on-board sensors or external instruments [Feng et al. 2005; Ge et al. 2009; Laros et al. 2013]. The measurement granularity of these techniques is commonly too large to measure the energy consumed by instructions or basic computing functions. Model based methods, such as RAPL [Intel 2011, David et al. 2010], provide a viable alternative to physical measurements. Some methods, such as ALEA [Mukhanov et al. 2015], refine measurement granularities by improving direct power measurement with probabilistic models.

Power measurement of the whole system

Feng et al. [Feng et al. 2005; Ge et al. 2009] created PowerPack, a software tool that automatically profiles the power consumption of scientific applications running on high-performance distributed systems. PowerPack can measure the power consumption of the major computing components, including CPU, memory, disk, and NIC (Network Interface Controller), on a cluster's computing nodes. In particular, a group of 0.1 Ohm sensor resistors are connected to the node using ATX extension cables. Each of ten digital meters per node collects four samples per second. Using RS232 ports, a data collection computer logs the power samples, which are subsequently analyzed using PowerPack. PowerInsight [Laros et al. 2013] is a similar project to measure power at the component level for a cluster by instrumenting hardware.

Power monitoring capabilities are available on IBM Blue Gene/P and Blue Gene/Q (BG/Q) systems [Yoshii et al. 2012; Hennecke et al. 2012]. Blue Gene systems have several sensors that monitor the voltage and current of service cards, node boards, bulk power modules, and cooling system boards. The sensor data are collected every 5 minutes and stored in a database. The power consumption of FPU (Floating-point Unit) and memory copy activity can be profiled using Environmental Monitoring (EMON) APIs on BG/Q.

The Cray Power Management Database tool [Hart et al. 2014] is available on Cray XC systems, which supports two ways to access power measurements: the Power Measurement Data Base (PMDB) and power management counters (pm_counters). The database contains comprehensive power readings at a 1 Hz frequency for each node, GPU, blower and network chip, which can be queried per job and per component and which support easy derivation of per job power and energy consumption. In contrast, pm_counters are provided as Sysfs files on Linux for power, accumulated energy, and power_cap of both the CPUs and GPUs. The register files for the pm_counters are updated at a frequency of 10 Hz.

IPMI (Intelligent Platform Management Interface) [Intel 2013] defines a set of low-level interfaces for remotely managing and monitoring the status of computer systems, including power consumption of different components. IPMI is supported in most Intel architectures, and many open source software libraries are available to collect IPMI sensor data. Hackenberg et al. [Hackenberg et al. 2013] find that the power samples collected using IPMI are accurate enough, but its estimation of energy consumption needs improvement for short period jobs.

In order to characterize HPC architectural trends for power consumption, several HPC benchmarks, such as the Top500, the Green500 and the Graph500, accept power metrics for running HPC workloads. The Energy Efficient HPC Working Group (EE HPC WG) conducted a survey on the power consumption listed for supercomputers on the Top500 and the Green500 in 2011. They find that the quality of power measurement varies widely. A three-level power measurement methodology is proposed [Scogland et al. 2014]. Among the proposed three levels, level 3 is the most rigorous measurement and is able to identify energy "hot spots" accurately and precisely. It requires continuous energy measurement of the whole system and each participating sub-system.

CPU power measurement

The Intel Sandy Bridge family of processors is equipped with onboard power meters. RAPL (Running Average Power Limit), a platform-specific power management interface [Intel 2011, David et al. 2010], is provided to allow users to measure energy consumption of processor, DRAM, and uncore devices in a non-intrusive manner. Users are allowed to measure and to control processor power usage using *model-specific registers* (MSRs). On Linux the *msr* kernel module supports reading and writing any MSR on the node using a file interface at /dev/cpu/N/msr. The precision of the power and energy measurements is architecture-specific and is provided by reading the MSR_RAPL_POWER_UNIT register. One limitation of RAPL is MSRs are 64-bit wide and are updated every millisec. Typically, the power values wrap-around every 60 seconds. Venkatesh et al. [Venkatesh et al. 2013] addressed this limitation by extending RAPL and applied the extension to measure energy consumption for MPI operations.

Fine-granularity power measurement

ALEA (Abstract-Level Energy Accounting) [Mukhanov et al. 2015] uses a probabilistic approach to provide fine-grained energy profiling for basic computing blocks in order to overcome the coarse granularity of sampling period of direct power measurement. In particular, ALEA combines the sampling of physical power measurements and a probabilistic model to estimate the energy consumed by basic blocks at any granularity. ALEA assumes that basic blocks execute repeatedly during a program's execution. It samples the program's execution at a predefined rate and then extracts the basic block sampled each time. The latency of each basic block is varied at each iteration, which allows ALEA to build a probabilistic model to estimate the execution time and energy consumption for each block. Evaluated using both sequential and parallel benchmarks, the mean error rates of ALEA are between 1.4% and 3.5%, while the sampling overhead is around 1%.

Energy and performance models of parallel computing

Improving energy efficiency for parallel computing requires pursuing optimal performance with moderate power usage. In the parallel computing domain, optimizing performance and energy consumption may conflict. The simple rule that

Energy = *Power* × *Time*, suggests two general ways to save energy: 1) faster speed, given a constant power; and 2) lower power without increasing run time. Given an ideal embarrassingly parallel application, running it on n cores at frequency *2f* requires the same execution time as running it on $2n$ cores at frequency *f*. However, the latter option consumes less than half of the CPU energy. Unlike this simple example, most parallel applications' execution time and power and energy consumption are interdependent, which complicates the relationship. In particular, interactions between memory, communication, parallelism, and processor frequency affect execution time and energy consumption. Additionally, the characteristics of a parallel application, such as its parallelism portion and its bottlenecks (i.e., either compute-bound or memory-bound), impact its optimal execution time and minimal energy consumption. Generally, increasing the amount of computational resources lowers system utilization and decreases energy efficiency because extra energy is wasted as the number of nodes or CPUs per node increases. Several power models [Bingham and Greenstreet 2008; Woo and Lee 2008; Cho and Melhem 2010; Song et al. 2011; Choi et al. 2013] are proposed to analyze the relationship between power, energy, performance, and parallelism for parallel applications. Most of these power models are extensions of previous performance models, such as Amdahl's Law, iso-efficiency, and the Roofline model. These models focus on providing insights to optimize algorithms or to detect an appropriate system configuration to improve energy efficiency by changing several adjustable parameters, such as processor frequency, parallelism level, and the parallelism portion. Accordingly, various metrics [Weiser et al. 1994; Ge et al. 2005; Bingham and Greenstreet 2008] are proposed to evaluate energy efficiency.

Energy efficiency metrics

Several energy-performance efficiency metrics have been proposed to determine the effect of improving energy efficiency. Weiser et al. [Weiser et al. 1994] propose to use millions-of-instructions-per-joule (MIPJ) to measure CPU energy performance, specifically MIPJ = MIPS/W (millions of instructions per second/watt). Because reducing clock speed causes a linear decrease in performance, but a cubic reduction in energy consumption, lowering frequency generates a better MIPJ. The flops/watt (F/W) metric is successfully used as the de facto standard in measuring the energy efficiency of a computing system [The Green 500]. EDP (Energy-delay product), i.e., *et* and ED2P (Energy-delay-squared product), i.e., et^2, have been widely used to represent the energy-time tradeoff of applying DVFS to parallel applications. In comparison to EDP, ED2P focuses more on the performance effect. Ge et al. [Ge et al. 2005] use ED3P to select the optimal operating point when investigating an appropriate granularity of applying DVFS. Bingham and Greenstreet [Bingham and Greenstreet 2008] propose using et^α, a generic energy complexity metric, to analyze the lower bound of execution time constrained by an energy budget for a number of basic algorithms, including sort, binary addition, and binary multiplication.

Amdahl's law-based energy models

Woo and Lee [Woo and Lee 2008] extend Amdahl's Law to analyze the energy efficiency of multicore processors. In particular, a new variable, k, is added to represent the fraction of power that the processor consumes in the idle state. Equation (4) illustrates energy efficiency in terms of *performance per watt* and *performance per joule* on n processors/cores, where s+p=1, s and p are the sequential and parallel portion of the application respectively. The extended power model was applied to investigate the energy efficiency of both symmetric and asymmetric multicore architectures, and they find that an asymmetric multicore architecture, i.e., a heterogeneous architecture, is more energy efficient.

$$\frac{Perf}{W} = 1/(1 + (n-1) * k * s), \quad \frac{Perf}{J} = \frac{1}{s+\frac{p}{n}} \times \frac{1}{1+(n-1)\times k \times s} \quad (4)$$

Ge and Cameron [Ge and Cameron 2007a] propose a power-aware speedup model by extending Amdahls' Law to account for the effects of parallelism and power-aware techniques on speedup. The model decomposes the workload into on/off-chip characteristics and assumes a constant frequency for the off-chip workload. The model can predict the optimal EDP configuration for a given parallel application.

Similarly motivated by Amdahl's law, Cho and Melhem [Cho and Melhem 2010] propose a theoretical model to analyze the relationship between parallelism, performance, and energy consumption based on the portion of an application that executes in parallel. They analyze how parallelization improves energy consumption. The improvement of dynamic energy for running an application on n processors is illustrated as Equation (5), where the dynamic power consumption of a processor running at frequency f is proportional to $f^{\propto}$.

$$Dynamic\ Energy\ Improvement = 1/(\mathrm{s} + \frac{p}{n^{(\propto-1)/\propto}})^{\propto} \quad (5)$$

Two machine models are examined using the proposed model: M_A, with which individual processors cannot be turned off; and M_B, with which individual processors can be turned off. For M_A, an optimal number of processors exists to minimize energy consumption. Depending on the size of sequential portion, s, and the portion of dynamic energy, adding more processors initially decreases energy consumption, while energy consumption increases after n is larger than a threshold. In contrast, the optimal configuration for M_B either does not exist, or its effect is much less than that for M_A. However, this model ignores the energy consumed to move data across processors.

Other energy models

Korthikanti and Agha [Korthikanti and Agha 2010] propose the use of a linear cost function, $\alpha \times E + T$, to evaluate the energy-time tradeoff for parallel algorithms while varying the number of cores and their frequencies. In particular, the proposed model accounts for several parameters of a parallel application, including its workload, concurrency degree, and communication cost. The energy-time tradeoff of quick sort, FU factorization, and minimum spanning tree algorithms are investigated.

Song et al. [Song et al. 2011] propose an iso-energy-efficiency model to analyze the power-performance tradeoffs of parallel applications by extending the concept of performance iso-efficiency. In particular, an *energy efficiency factor*(*EEF*) is defined as Equation (6), in which E_s is the total energy consumed by a sequential application, E_p is the energy consumed by its parallel execution on p processors, $E_0 = E_p\text{-}E_S$. Overall, a large *EEF* stands for low iso-energy-efficiency, and vice versa. This model can predict the total energy consumption of large-scale parallel applications while varying the parallelism degree and processor frequency.

$$\text{Energy Efficiency Factor } (EEF) = \frac{E_0}{E_s}\text{; Iso-energy-efficiency (EE)} = \frac{\mathrm{E_s}}{\mathrm{E_p}} = \frac{1}{1+\mathrm{EEF}} \quad (6)$$

Choi et al. [Choi et al. 2013] propose an energy roofline model to analyze the time, energy, and power costs for an algorithm from the *energy-balance* point of view. As an analogue to the time-balance proposed in the original performance Roofline model, energy-balance measures the ratio of flops and bytes per unit-energy, i.e., Joule. The model identifies the relationship between *computational intensity*, i.e., flops per memory operation, and consumed energy. As a counterpart of the time-balance, B_τ, in the performance Roofline model, the energy-balance, B_ε, is detected using the energy Roofline model. Real world applications always have $B_\tau > B_\varepsilon$. This implies that race-to-halt strategies are the first-order technology to achieve energy efficiency. With race-to-halt, the system runs at top speed to create long idle intervals, in which certain parts of hardware can be turned off to save energy.

Performance prediction for energy analysis

Adjusting CPU frequency to save energy depends on the accurate estimation of performance degradation. A small prediction error can fail to reduce energy consumption and cause extra power usage [Rountree et al. 2011]. Rountree et al. and Keramidas et al. [Rountree et al. 2011; Keramidas et al. 2010] proposed the use of a new performance counter, *Leading Loads*, to improve the accuracy of performance estimation under DVFS. In particular, a *load* is defined as a non-speculative read that results in a last-level cache miss and the first load is a *leading load*. Assume the performance at frequency f is observed and after adjusting frequency from f to f', the degraded performance can be estimated using Equations (7)~(9), in which *execution time* consists of *CPU time* and *bus time*. With the Leading Load model, the estimation error is limited to within 0.3%.

$$\textit{Predicted Execution Time} \text{ at } f' = \textit{Observed Bus Time} + (f'/f) \times \textit{Observed CPU Time} \tag{7}$$

$$\textit{Observed Bus Time} \text{ at } f = \frac{\textit{LeadingLoadCycles at f}}{f} \tag{8}$$

$$\textit{Observed CPU Time} \text{ at } f = \textit{Observed Execution Time at } f - \textit{Observed Bus Time at } f \tag{9}$$

Insights of energy models

Using the above theoretical models, the following general conclusions are achieved. First, the energy efficiency of parallel computing is both application and platform dependent. Second, energy consumption and performance are strongly correlated. Third, for parallel applications, adjusting processor frequency has less impact on energy efficiency than changing parallelism.

Experiments that evaluate the energy-time trade-off in parallel applications confirm these theory-detected relationships [Freeh et al. 2007b; Minartz et al. 2011; Rahman et al. 2012; Laros et al. 2013; Leon et al. 2015]. Rahman et al. [Rahman et al. 2012] study the potential power-saving effect of applying compiler optimization technologies, including loop and thread affinity optimizations, to multi-threaded applications. They demonstrate that the power-saving space for a multi-threaded application can be up to 28%. Leon et al. [Leon et al. 2015] analyze the effectiveness of code optimizations on the power and energy use of a hydrodynamics application, called LULESH. Several techniques of reducing data movement are investigated, including the loop fusion, data structure transformation, and global allocation optimizations. These optimizations are evaluated for IBM Blue Gene/Q, and for x86 server-class and consumer-class architectures. They find that the effect of same optimization methods depends on architectures. In addition, different code regions of LULESH demand different optimization techniques. However, finding a globally optimal solution by applying different optimization techniques to different regions is challenging.

Freeh et al. [Freeh et al. 2007b] investigate the relationship between energy consumption and performance for the NPB Parallel Benchmarks on a power-scalable cluster with a variety of frequencies and a different number of processors. Given a number of processors, they detect an optimal configuration of processor frequency to minimize energy consumption. They find that energy savings depend on an application's speedup. For the case of good speedup, running the application at a higher frequency on more nodes can save energy. In contrast, for the case of poor speedup, increasing node count and processor frequency may not save much energy. Laros et al. [Laros et al. 2012; Laros et al. 2013] investigate the potential effect of power saving on the Cray XT architecture using processor frequency scaling and network bandwidth scaling. In particular, running a series of empirical experiments demonstrate up to 39% energy savings with little or no negative impact on performance. Minartz et al. [Minartz et al. 2011] investigated the power-saving opportunities of applying DVFS to AMD and Intel clusters for parallel computing. They find 4~8% overall system energy saving with slight increases in execution time. All of the above theoretical analyses and empirical experiments form a foundation to drive a large number of software techniques to improve the energy efficiency of parallel computing.

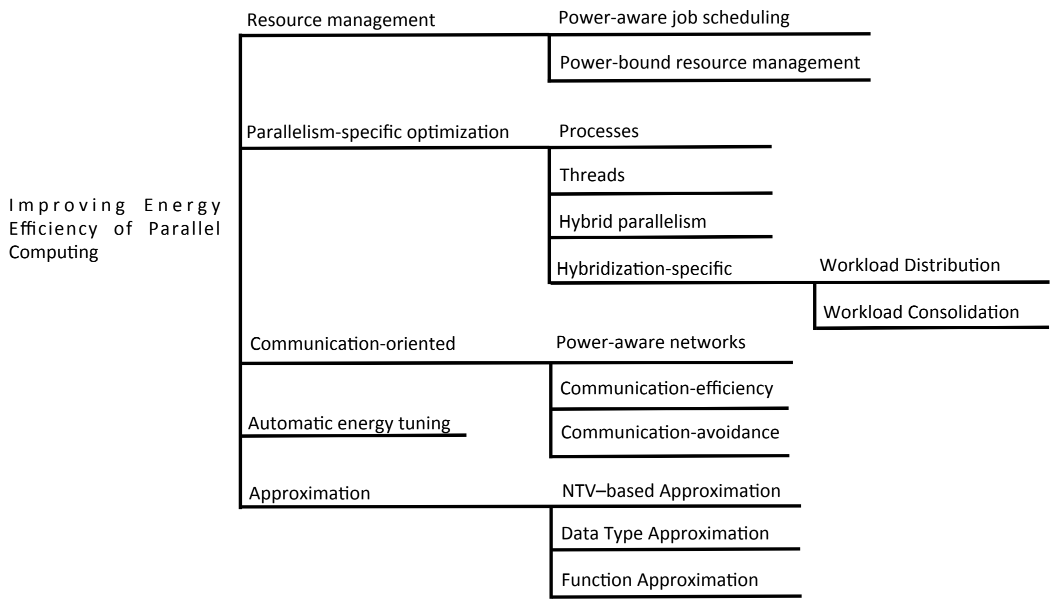

Figure 1. Taxonomy of software methods to improve the energy efficiency of parallel computing.

Taxonomy of energy efficient methods for parallel computing

In order to improve energy efficiency, supercomputers need to take both instantaneous power usage and total energy consumption into account. Power saving can be achieved at different granularities, such as per job, per node and the whole system. Software stacks should utilize various hardware power-saving techniques efficiently by taking advantage of application characteristics. We classify the methods to improve the energy efficiency of parallel computing in the software layer into the following categories, as illustrated in Figure 1.

1) Resource management: typically, an HPC system is shared by a number of users and it frequently executes multiple jobs simultaneously. The first category consists of methods exploiting energy saving opportunities at the whole system level using a job scheduler or resource management system. By monitoring the energy consumed by a HPC system and the performance achieved by each application, the resource management system can take appropriate actions to save power or to improve energy efficiency according to job specifications [Georgiou et al. 2014; Martin et al. 2015; Pedretti et al. 2015]. In addition, mapping a number of waiting jobs to available nodes should consider the energy characteristics of each application and the power capability of the physical resources [Elnozahy et al. 2003; Lawson and Smirni 2005; Zhou et al. 2014; Mämmelä et al. 2010; Marathe et al. 2015]. As the performance provided by a supercomputer is becoming power-constrained, hardware overprovisioning has attracted signification attention in the HPC community [Etinski et al. 2010; Etinski et al. 2012; Patki et al. 2013; Patki et al. 2015; Sarood et al. 2013; Sarood et al. 2014; Ellsworth et al. 2015].
2) Parallelism-specific optimization: the second category consists of methods exploiting energy saving opportunities enabled at different levels of parallelism, such as processes, threads, and hybridization. Load imbalances between processes and threads generate significant opportunities to decrease energy consumption. DVFS has been applied to improve the energy efficiency of MPI programs by lowering the frequency of processors with light load [Ge et al. 2005; Hsu and Feng 2005; Hsu and Kremer 2003a; Kappiah et al. 2005; Freeh et al. 2005a; Freeh et al. 2005b; Li et al. 2010a; Rountree et al. 2007; Rountree et al. 2009; Rountree et al. 2011]. Dynamic Concurrency Throttling (DCT) is proposed to control the number of active CPU cores for multi-threaded programs, including OpenMP programs, at runtime [Curtis-Maury et al. 2006a; Curtis-Maury et al. 2006b; Curtis-Maury et al. 2007; Freeh et al. 2007a; Grant and Afsahi 2006; Li and Martinez 2006; Li et al. 2010b; Suleman et al. 2008]. The balance between performance and energy consumption is also investigated for hybrid MPI/OpenMP applications [Li et al. 2010b; Bailey et al. 2015]. Further, innovative hardware components, such as accelerators, FPGAs, and coprocessors, that provide vector level parallelism, frequently support improved energy efficiency compared to multicore CPUs [Enos et al. 2010; Wang and Ren 2010; Ghosh et al. 2012; Ma et al. 2012; Huang et al. 2009; Li et al. 2011; Collange et al. 2009; Suda and Ren 2009; Li et al. 2014; Luk et al. 2009; Fowers et al. 2013]. Some of these methods depend on energy improvement of compiler techniques [Hsu and Kremer 2003b; Saputra et al. 2002; Keramidas et al. 2010; Leon et al. 2015].
3) Improving communication energy efficiency consists of three groups of methods: a) saving power for network fabric using power-aware networks [Conner et al. 2007; Alonso et al. 2006; Nedevschi et al. 2008; Saravanan et al. 2013; Saravanan et al. 2014; Groves and Grant 2015; Miwa et al. 2014; Miwa an d Nakamura 2015]; b) improving communication energy efficiency using energy-aware data transfer algorithms [Lim et al. 2006; Kandalla et al. 2010; Venkatesh et al. 2015; Alan et al. 2015]; and c) decreasing network traffic using communication-avoiding algorithms [Demmel et al. 2013; Grigori et al. 2011].
4) Automatic tuning: this category treats a parallel application as a black box and improves its energy efficiency by tuning several energy and performance parameters [Balaprakash et al. 2013; Gschwandtner et al. 2014b; Jordan et al. 2012; Miceli et al. 2012; Rahman et al. 2011; Tiwari et al. 2012].

5) Approximation: the last category is to improve energy efficiency using approximation, such as mixed floating-point precision [Anzt et al. 2010; Dongarra et al. 2012; Lam et al. 2013; Linderman et al. 2010; Rubio-Gonz´alez et al. 2013; Schkufza et al. 2014], significance-based computing using NTV [Dreslinski et al. 2010; Gschwandtner et al. 2014a], and providing approximated computing results for applications that can tolerate inaccurate computation [Baek and Chilimbi 2010; Sampson et al. 2011; Hoffmann et al. 2009; Hoffmann et al. 2011].

Saving energy with power-aware resource management

The resource and job management system, which is also called the job scheduler, distributes waiting jobs to available compute nodes. Traditionally, it only considers improving job performance and maximizing overall system utilization. Because the job scheduler has a global view of the system, including compute resources, running jobs' termination times and waiting jobs' performance requirements, it is the best candidate for monitoring and controlling the energy consumed by parallel applications. To achieve this target, traditional job schedulers must be improved to track energy usage in real time and to predict power requirements. With these two enhancements, the job scheduler can allocate both compute nodes and power to waiting jobs by treating power and energy consumption as job characteristics. Initially, many power-aware job schedulers [Elnozahy et al. 2003; Lawson and Smirni 2005; Zhou et al. 2014; Mämmelä et al. 2010; Marathe et al. 2015] focus on optimizing overall energy usage. Recently, investigating the impact of a power bound imposed on future supercomputers has attracted more attention [Etinski et al. 2010; Etinski et al. 2012; Patki et al. 2013; Patki et al. 2015; Sarood et al. 2013; Sarood et al. 2014; Ellsworth et al. 2015]. The node variability of power consumption in supercomputers and its impact on job scheduling are also investigated recently [Inadomi et al. 2015; Scogland et al. 2015].

Power-aware job scheduling

Compute nodes consume significant energy even when idle. For example, an idle Blue Gene/P rack consumes around 13kW [Zhou et al. 2014]. Turning off idle nodes during low system utilization is a straightforward way to save power. In addition, adjusting CPU frequencies on targeted compute nodes can also save significant energy [Etinski et al. 2010; Etinski et al. 2012; Mämmelä et al. 2010].

SLURM (Simple Linux Utility for Resource Management)'s existing resource utilization collection module is extended to track energy consumption using both RAPL and IPMI and to support energy accounting and control [Georgiou et al. 2014]. With these extensions, SLURM can profile power usage for each job and programmers can control the CPU frequency. Aiming to be deployed on large-scale systems, the overhead of energy monitoring in SLURM is optimized to be lower than 0.6% in energy consumption and less than 0.2% in execution time with less than 2% error rate in most cases.

Cray Advanced Platform Monitoring and Control (CAPMC) [Martin et al. 2015] supports monitoring and controlling of power consumption on Cray XC systems. It reports energy usage both per compute node and per job. In addition, CAPMC supports both CLI and HTTP APIs to allow users to collect energy reports and to control power usage per job. It also supports node-level power capping that enables external software to establish a maximum or a minimum bound on the amount of power consumed by the system or a selected subset of the system [Pedretti et al. 2015]. In addition, external software can modify CPU frequencies and sleep states dynamically, which allows a job scheduler to reallocate power among nodes and to limit the system power consumption within a predefined range.

A power-aware job scheduler designed for IBM Blue Gene/P [Zhou et al. 2014] can reduce energy costs by allocating resources according to variable electricity prices and application power profiles. In particular, this job scheduler prefers allocating jobs demanding high power consumption during the off-peak electricity price period using a 0-1 knapsack model. With the online scheduling algorithm, a number of jobs within the scheduling window are allocated to available resources. The scheduling objective is to maximize system utilization without exceeding the predefined power budget and to make a trade-off between performance and fairness simultaneously. Simulation of the proposed scheduler using both synthetic and real job traces show reductions of energy costs can be up to 25% with slightly lower system utilization.

Different scheduling policies are investigated for power-aware job schedulers [Elnozahy et al. 2003; Lawson and Smirni 2005; Zhou et al. 2014; Etinski et al. 2010; Etinski et al. 2012; Mämmelä et al. 2010; Marathe et al. 2015]. Most of them extend the EASY (Extensible Argonne Scheduling system) backfilling policy and work in an online mode. In particular, the dynamic policies of turning on/off compute nodes according to workload fluctuation were studied [Elnozahy et al. 2003; Lawson and Smirni 2005]. An accurate workload prediction model is the key to the efficiency of these algorithms. In many cases, lowering CPU frequency using DVFS for each compute node based on jobs' power characteristics can allow more jobs to run simultaneously, which can save power and decrease average job waiting time [Etinski et al. 2010; Etinski et al. 2012; Mämmelä et al. 2010]. Elnozahy et al. [Elnozahy et al. 2003] investigate how to adjust CPU frequencies and dynamically turn compute nodes on or off simultaneously. Most of these scheduling algorithms rely on an accurate power and energy prediction model for applications. Conductor [Marathe et al. 2015] is a run-time system that intelligently allocates power, nodes, and cores to applications. Using DVFS and dynamically changing the number of threads, Conductor outperforms other power-constrained schedulers that adopt static power capping per node by up to 30%. Etinski et al. [Etinski et al. 2010; Etinski et al. 2012] investigate how to efficiently utilize the overall system resource given a power budget. The above power-aware schedulers are designed to optimize overall energy efficiency and to maximize system throughput, but do not consider the global power bound imposed on future supercomputers.

Power-bound resource management

The power that can be supplied to a supercomputer will reach a physical bound in the future. In addition, the energy cost of operating supercomputers may also be restricted. Some experts estimate each MW-year costs $1M. Therefore, although future supercomputers will have more compute nodes, they potentially may not all be able to run simultaneously with peak performance. Similarly, the number of active cores on each chip may be restricted (i.e., "dark silicon").

Presently, most supercomputers are designed with *worst-case provisioning*, in which the maximum power draw per node decides the total power allocated to a computer and is designed to make all nodes run at peak power simultaneously. However, many studies show that most supercomputers are under-utilized in terms of power consumption. For example, *Vulcan*, a Blue Gene/Q at LLNL (Lawrence Livermore National Laboratory) consumes only 60% of allocated power on average over a 16-month period [Sarood et al. 2013]. Further, raising power allocated to CPU and memory does not generate a proportional increase in application performance

[Patki et al. 2013; Sarood et al. 2013]. In addition, most scientific applications do not fully utilize the maximum power allocated to each compute node [Sarood et al. 2013; Patki et al. 2015]. In order to utilize the allocated power more efficiently, *hardware overprovisioning* is proposed, initially by Patki et al. [Patki et al. 2013]. By overprovisioning, a supercomputer consists of compute capacity that is more than can be fully powered under the power constraint, but not all system components can run simultaneously at peak power. Instead, the system must be reconfigured dynamically according to the workload's power requirements and characteristics such as scalability and memory intensity. Patki et al. [Patki et al. 2013] show that overprovisioning can be leveraged to improve overall system throughput and to decrease average turnaround time. In particular, overprovisioning outperforms worst-case provisioning by up to 50%.

Sarood et al. [Sarood et al. 2014] proposed an online scheduler for an overprovisioned supercomputer that can constrain the power consumption of each node using RAPL and optimally allocate power and nodes to queued jobs. The goal is to maximize the job throughput for a supercomputer given a power budget. Each time a new job request arrives or currently running jobs terminate, the scheduler re-allocates resources to both running jobs and selected waiting jobs. Mapping nodes and power to jobs is formulated as a resource optimization problem that is solved using an Integer Linear Program (ILP). The scheduling scheme assumes each parallel job is malleable such that the job can shrink or expand across a different number of nodes or CPU cores at runtime. The scheduler also relies on a prediction model to estimate the power and performance characteristics for each job at different scales. The online scheduling uses the performance and power characteristics of each job to make resource allocation decisions that can change the resources allocated to a running job. Using simulation, job throughput is improved up to 5.2X in comparison to power-unaware SLURM. With real experiments on a small-scale cluster, 1.7X improvement of job throughput is obtained.

Patki et al. [Patki et al. 2015] propose a practical and low-overhead resource manager for power-constrained clusters, called RMAP (Resource MAnager for Power). With overprovisioning, RMAP supports power-aware backfilling. It aims to provide faster job turnaround times with increased overall system resource utilization. RMAP predicts the performance and power consumption for each application according to its profiling. Using the estimated power and performance for each application on different configurations, RMAP can allocate idle power to appropriate waiting jobs. But it does not change the configuration of running jobs. Simulation is performed to compare RMAP with traditional scheduling policy by investigating real world scientific applications, and the results show RMAP's new policy increases system power utilization with 18.5% faster average turnaround time.

Ellsworth et al. [Ellsworth et al. 2015] propose a power scheduler (POWsched) to enforce a system-wide power limit. POWsched maintains a system-wide power bound, and implements a dynamic policy to allocate wasted power to more power-intensive applications. POWsched does not predict the power consumption and performance for each application. Instead, it assumes the power consumption of each application is consistent during a short time period. At each round of scheduling, POWshed monitors the power consumption of each application in real time, and detects surplus power allocated to each node. When power is abundant, the surplus power is reallocated to power-scarce nodes. Otherwise, a fair allocation of power across nodes is achieved. The dynamic scheduling policy adjusts power capped for each application and guarantees each node is allocated with enough power to avoid significant performance degradation. Simulation demonstrates that POWshed can decrease overall workload execution time by around 14%.

Parallelism-specific methods of energy efficient parallel computing

The parallelism-specific software methods of improving energy efficiency of parallel computing consist of four groups. The first group applies to process-level parallelism, which exploits imbalanced load distributions. An imbalanced workload across machines or CPU cores causes inefficient resource utilization. In addition, an application frequently consists of memory bound or I/O bound phases during the computation, in which lowering CPU frequency can save power with little or no performance degradation. The second group works at thread-level parallelism to improve the multi-threaded program's energy efficiency by controlling the number of active threads. The third group handles hybrid MPI and OpenMP applications by combining the technologies used to save power for MPI programs and for multi-threaded applications. The last group is for hybridization that exploits an energy efficient hybrid-computing component.

DVFS-based power-saving methods for MPI applications

DVFS has been recognized as one of the most effective ways to reduce processor power dissipation. It has been applied to parallel computing, including MPI applications, to adjust the tradeoff between energy savings and performance dynamically. It works particularly well for applications with load imbalances, including load imbalances between compute nodes and between CPU, memory, and I/O, at a cost of negligible performance degradation.

DVFS can be applied at different levels, such as a whole program or a function call. Using the Wattch CPU energy and performance simulator [Brooks 2000], Hsu and Kremer [Hsu and Kremer 2003a] investigate the opportunity of applying DVFS to save energy for highly optimized scientific codes by taking advantage of memory stalls. In particular, using five SPECfp95 benchmark applications, they demonstrate that energy consumption can be reduced up to 60% with a performance penalty of 9.58% or less. Freeh et al. [Freeh et al. 2005b] investigated the opportunity to save energy for MPI programs in power scalable clusters by reducing processor frequency. However, the power saving depends on application characteristics, including the ratio of computation-to-communication and memory stalls. In the case of perfect speedup, both energy consumption and execution time can be saved using more nodes at a lower CPU frequency. Ge et al. [Ge et al. 2005] investigate the appropriate granularity of applying DVFS to HPC applications on power-aware clusters. They find applying DVFS to a whole program saves less overall energy than applying it to finer levels, because power saved during idle periods is offset by increased execution time of non-idle periods. In addition, they separate the FT application of the NPB Parallel Benchmarks to different regions using communication-to-computation ratio and apply DVFS to each region. This method can decrease up to 36% energy consumption without noticeable performance loss.

Detection of finer granularities. To apply DVFS to parallel applications at finer granularities, it needs to detect regions in the application with different degrees of idleness or load imbalances. Typically, a parallel application is split into a series of phases and a different CPU frequency is used for each phase. Identifying the idle and non-idle periods of a parallel application is typically based on the iterative nature of most scientific computation. Most methods that identify the idle periods of a parallel application are either profile-directed [Hsu and Feng 2005; Freeh et al. 2005a;] or trace-driven [Rountree et al. 2007]. The boundaries between different phases are determined using communication APIs, computing patterns, and memory access patterns. The methods of applying DVFS to a parallel application consist of external methods and internal methods. The external methods monitor the execution of a parallel application to predict its behavior and accordingly adjust the CPU frequency of each machine [Hsu and Feng 2005; Lim et al. 2006]. In contrast, the internal methods allow parallel applications to adjust CPU frequency directly, which are typically achieved using instrumentation [Freeh et al. 2005a; Kappiah et al. 2005]. Finally, it is an NP-complete problem to apply appropriate CPU frequencies to different regions optimally. Frequently, it can be estimated using optimization algorithms, such as linear programming [Rountree et al. 2007] and heuristic-based searching algorithms [Freeh et al. 2005a].

Some methods that identify the different regions of an MPI application not only need to intercept MPI calls, but also rely on compiler support to detect memory or I/O bound phases. Saputra et al. [Saputra et al. 2002] propose energy-conscious compilation based on DVFS. In particular, they adapte many loop-oriented compiler optimizations such as loop permutation, tiling, and loop fusion and distribution [Wolfe 1996] to save energy. Hsu and Kremer [Hsu and Kremer 2003b] present a profile-driven compiler optimization technique to identify program regions using memory stalls. While executing these regions, the CPU frequency is reduced to save energy consumption. The target program is instrumented to detect the boundaries between regions based on analyzing loops and function calls. The profiling phase records the execution time of each region at different CPU frequencies and estimates the corresponding energy consumption.

DVFS-enhanced MPI runtime systems. Many MPI runtime systems [Hsu and Feng 2005; Freeh et al. 2005a; Kappiah et al. 2005; Rountree et al. 2007; Ge et al. 2007b; Rountree et al. 2009] are proposed to adjust CPU frequencies dynamically, most of which are online methods. Profile-driven methods detect workload characteristics for a program, then split the program into different regions by instrumentation and adjust CPU frequencies for each region at runtime. In contrast, non-profile methods monitor the execution of an application to detect idleness or to predict the performance and adjust CPU frequencies for different time intervals. Some methods focus on intra-node load imbalances [Hsu and Feng 2005; Freeh et al. 2005a; Ge et al. 2007b], while other methods [Kappiah et al. 2005; Rountree et al. 2007; Rountree et al. 2009] analyze inter-node load imbalances using a DAG (Directed Acyclic Graph) to represent the dependencies between computation and communication tasks and to lower CPU frequencies for tasks not on the critical path.

Hsu and Feng [Hsu and Feng 2005] propose a power-aware algorithm to schedule the execution of a parallel application running on a cluster by dynamically adjusting the CPU frequency of each compute node. In particular, each compute node asynchronously updates its CPU frequency periodically (every I seconds). Without profiling the targeted application, the preferred CPU frequency is estimated for the next period based on the MIPS rate observed during the current period. The proposed algorithm is evaluated on an AMD Athlon64-base cluster and an Opteron-based cluster respectively using the NAS-MPI benchmarks. On average, 12% CPU energy is saved with 4% performance slowdown on the AMD Athlon64-base cluster; while on the Opteron-based cluster, 8%~25% CPU energy is saved with 3% performance slowdown.

Freeh et al. [Freeh et al. 2005a] propose a method to change CPU frequencies in MPI programs dynamically on a power-scalable cluster. They split an MPI program into a series of phases and assign a preferred frequency to each phase. The program is partitioned according to detected CPU stall periods, during which the CPU waits for memory, disk or communication. A suitable performance-energy point, called a *gear*, is selected for each phase. Phase detection applies to iterative and predictable parallel applications. The boundaries between different phases are determined mainly according to two rules: 1) any MPI operation; and 2) an abrupt *memory pressure* change, which is measured using the metric *operations per miss* (OPM). The detected phases are ordered according to the energy-time tradeoff. The MPI program is profiled using the MPI profiling layer. The proposed method is evaluated on an AMD Athlon-64s system using the NAS Parallel Benchmarks. The results show the proposed method can save up to 16% energy at a cost of 1% performance loss.

Kappiah et al. [Kappiah et al. 2005] propose a system, called Jitter, to save energy consumption in MPI programs by exploiting inter-node slack. Jitter applies to the case that the computational load is not perfectly balanced. In particular, it reduces the frequency on nodes that are assigned less computation. MPI processes that are not on the critical path can arrive at synchronization points early. Therefore, processes on the critical path determine overall execution time. Given an iterative program with stable iterations, early iterations are profiled to predict the behavior of subsequent iterations. Iteration boundaries are detected according to MPI calls. For every iteration loop, each compute node calculates its local *wait time* and *iteration time*. When the ratio of wait-to-iteration time exceeds a predefined threshold, the process identifies itself as a slack node and accordingly decreases its frequency. In contrast, when the ratio is small enough, the process may convert to a bottleneck process and increase its frequency. The predefined switching threshold is hand-tuned for each system. The proposed method is evaluated on an AMD Athlon-64s system for the ASCI Purple benchmarks and NAS Parallel Benchmarks. The results demonstrate that Jitter can save up to 8% energy usage for the ASCI Purple benchmarks at a cost of 2.6% performance loss.

Rountree et al. [Rountree et al. 2007] develop a system that determines a bound on the energy savings for an MPI application given an acceptable performance loss. Specifically, an MPI program is split into a series of tasks according to MPI calls. Accordingly, a task graph is created to represent the dependency/communication between tasks. Subsequently, heuristic-based linear programming determines an appropriate CPU frequency for each task in terms of the tradeoff between performance and energy. The proposed method is evaluated using 3 scientific applications: Jacobi iteration, a particle simulation, and an unstructured mesh application (UMT2K), in which the particle simulation and UMT2K exhibit load imbalance. With a bound of zero performance loss, Jacobi has no potential energy reduction. In contrast, an energy reduction of up to 15% is available with the particle simulation and the potential reduction for UMT2K is only 3%.

CPU Miser [Ge et al. 2007b] is an online method of applying DVFS to parallel applications on a power-aware cluster. It optimizes the energy consumption for a parallel application according to a given power budget and a user specified performance loss. In particular, CPU Miser splits the runtime into a series of time intervals. It monitors a parallel application's performance in the current time interval and predicts its performance for the next interval using the RELAX algorithm [Ge et al. 2007b]. Evaluation is conducted using the NAS Parallel Benchmarks and showed that CPU Miser can reduce energy consumption up to 20% and constrain performance loss to within 5%.

Rountree et al. [Rountree et al. 2009] develop Adagio to apply DVFS to an MPI program dynamically at runtime. Using MPI calls, Adagio splits an MPI application into a group of tasks that are represented using a DAG (Directed Acyclic Graph). Critical path analysis identifies non-critical tasks for which performance can be degraded to save power. Adagio does not profile the target application prior to its execution. On the contrary, it deploys an online policy to monitor the execution of each task. For a real-world scientific application, Adagio assumes each task's behavior is identical at every invocation. The highest frequency is applied to a task for its first invocation and the lowest frequency is applied to its second invocation. The execution time of both invocations is recorded to estimate the task's performance degradation when applying different frequencies. An appropriate, reduced frequency is applied to the task's subsequent invocations if it is not on the critical path. The performance of Adagio is evaluated using two real-world applications, UMT2K, an unstructured mesh application assembled by LLNL, and ParaDis, a dislocation dynamics simulation. Adagio decrease total energy consumption by 8% and 20% for UMT2K and ParaDis respectively at a cost of less than 1% increase in execution time.

Power-saving for multi-threaded applications

Systems with multiple cores per node can use shared memory to parallelize threaded applications. In general, increasing the number of threads up to at least the number of cores on the node obtains the maximal performance. However, the optimal number of threads for an application depends on the maximum degree of its built-in parallelism and hardware characteristics, such as off-chip bandwidth. *Dynamic concurrency throttling* (DCT) [Curtis-Maury et al. 2006a; Curtis-Maury et al. 2006b; Curtis-Maury et al. 2007; Suleman et al. 2008; Li et al. 2010b] controls the active number of threads and switches off inactive cores to save energy for multi-threaded applications, including OpenMP programs. Grant and Afsahi [Grant and Afsahi 2006] investigate using AMP (asymmetric multiprocessors), the processors of which are not operating at the same frequency, to save energy for multi-threaded programs. After examining the NAS Parallel and SPEC benchmarks on a 4-way SMP server, they find that using an appropriate thread scheduler to apply an optimal frequency to each processor reduces energy consumption by an average 15.6% at a cost of 6.1% performance loss when hyper-threading (HT) is disabled. In contrast for a HT enabled case, 7.1% energy saving is achieved with a 4.8% performance loss.

Curtis-Maury et al. [Curtis-Maury et al. 2006a; Curtis-Maury et al. 2006b; Curtis-Maury et al. 2007] propose a dynamic phase-aware performance prediction (DPAPP) model to provide concurrency throttling for multi-threaded programs. DPAPP can predict application performance under different concurrency levels and thread placement strategies on NUMA nodes. In particular, a multivariate process is required to train the DPAPP model to select hardware events that reflect the scalability of each program phase across different hardware configurations. The proposed method is designed for the iterative execution of scientific applications and can support any application with repetitive behavior as long as the execution properties of each phase between executions remain relatively stable and the concurrency is modifiable. The DPAPP model is trained during early iterations. For subsequent iterations, the trained model steers concurrency throttling at runtime to identify phases in which energy can be saved without sacrificing performance.

The optimal thread count for an application depends on the input set and machine configuration. Suleman et al. [Suleman et al. 2008] investigate the optimal number of threads limited by data-synchronization and off-chip bandwidth. Specifically, when the number of threads exceeds a threshold determined by the contention for shared data or bus bandwidth, additional threads do not improve performance and waste chip power. In particular, Suleman et al. propose *feedback-driven threading* (FDT) to control the number of threads dynamically for applications with iterative loops. In terms of implementation, the first few loops (at most 1% of the total loops) are sampled to estimate the application behavior to adjust the thread count for subsequent loops.

Li and Martinez [Li and Martinez 2006] investigate the application of both DVFS and DCT to optimize the power consumption of a parallel application executing on a many-core CMP. They explore a two-dimensional optimization space for the run-time power-performance tradeoffs: 1) the possible number of active processors; and 2) the different voltage-frequency levels available. In particular, they study how to maximize power saving while delivering a specified level of performance. A heuristic-based search algorithm, a combination of binary search and hill-climbing optimization, is applied to each axis of the search space to converge quickly toward the global optimum. Freeh et al. [Freeh et al. 2007a] investigate the same problem, but they find that DVFS should be used first and then DCT should be considered. In addition, Freeh et al. observe the effectiveness of DCT is application-dependent and the improvement to parallel workloads ranges from small to negligible.

Power-saving for hybrid MPI and OpenMP applications

To handle hybrid MPI and OpenMP applications, the techniques used to save energy for MPI programs and for multi-threaded applications are typically combined together. Li et al. [Li et al. 2010b] propose a power-aware performance prediction model for hybrid MPI/OpenMP applications to support a power-efficient execution algorithm using a combined DCT/DVFS system. Each MPI task is partitioned into a number of OpenMP phases according to the boundaries delineated by MPI operations. DCT is applied to each MPI task with a coordination scheme. In addition, the slack period of non-critical MPI tasks is identified in order to apply DVFS. The effectiveness of the proposed method relies on the prediction of the energy consumption of each OpenMP phase. Using the NAS Parallel Benchmark Multizone suite, they find power saving opportunities increase with MPI task count under weak scaling but diminish under strong scaling. The method reduces energy consumption by 4.2% on average with negligible performance loss or even performance improvement up to 7.2%.

Bailey et al. [Bailey et al. 2015] investigate application performance limitations for power-constrained hybrid MPI and OpenMP applications. The dependencies between the communication and computation tasks of an application are represented using a DAG. A linear programming (LP) formulation is used to optimize the configuration for each task of a DAG. In particular, scheduling sets appropriate DVFS states and OpenMP thread counts for computational tasks between consecutive MPI calls. The proposed solution is evaluated using four fluid and molecular dynamics applications (CoMD, LULESH, and NAS-MZ SP and BT). The conclusion is that algorithms such as LP demonstrate significant opportunities to improve power-constrained performance of current runtime systems, by up to 41.1%.

Hybridization-specific energy saving

Hybrid computers that are equipped with heterogeneous computing components, such as GPUs, FPGAs, and Intel Phi coprocessors, support highly parallel execution for modest energy consumption. Many projects have demonstrated the superior power efficiency of these heterogeneous components. Several research projects [Williams et al. 2010; Fowers et al. 2013] compare GPUs and FPGAs in terms of both performance and energy consumption. Williams et al. [Williams et al. 2010] perform a device characterization analysis in which they find that FPGAs provide better performance for bit operations and 16-bit and 32-bit integer operations, while GPUs' performance is superior in single-precision and double-precision operations. In contrast, FPGAs are more energy efficient than GPUs. Using convolution as a benchmark, Fowers et al. [Fowers et al. 2013] find FPGAs are superior to GPUs in both performance and energy consumption.

Presently, while many hybrid supercomputers have NVIDIA GPUs, Intel's Xeon Phi architecture is emerging as an energy efficient alternative. Li et al. [Li et al. 2014] compare the Intel Xeon Phi 5110P with an old GPU model, an NVIDIA c2050, in terms of both performance and energy consumption using two SHOC kernels. The memory bound Reduction kernel is a little more energy efficient with the Xeon Phi than the GPU. Similarly, when the compute bound GEMM application is evaluated, the Xeon Phi is slightly more efficient for large problem sizes.*Initial investigation.* Huang et al. [Huang et al. 2009] examine the energy efficiency of GPUs for scientific computing. They compare the multi-threaded version of GEM, a bio-molecular program that calculates electrostatic properties of molecules, with its CUDA implementation. An experiment on a compute server equipped with an Intel Core 2 Duo CPU and a NVIDIA GT200 GPU demonstrate that the GPU is several hundred times more energy efficient as measured by energy-delay product.

Enos et al. [Enos et al. 2010] quantify the impact of GPUs on the performance and energy efficiency of parallel computing by executing four HPC applications on an NCSA cluster that is equipped with Opteron dual core processors and NVIDIA Telsa C1060 GPUs. They use performance-per-watt to compare the energy efficiency of CPU-only and GPU accelerated versions. In particular, NAMD, a parallel molecular dynamics simulation package, show only 2.78X more energy efficiency with its GPU implementations. The energy efficiency of MILC, a Quantum Chromodynamics application, is improved 8.1X with the GPU. VMD, a molecular visualization and analysis tool, achieve 10.48X more energy efficiency with its GPU accelerated version. QMCPCK, a set of Quantum Monte Carlo methods to solve the many-body problem of interacting quantum particles, exhibite 22X improvement of energy efficiency with the GPU.

Although GPUs provide an order of magnitude improvement in energy efficiency for parallel computing, each of these devices consumes significant energy. For example, an NVIDIA GTX 280 video card is rated at 236 watts and an NVIDIA Tesla C2050 consumes up to 225 watts [Enos et al. 2010], while the power supply of a typical compute node supports around 500 watts [Huang et al. 2009]. Overall, GPUs in a hybrid system may consume up to 75% of the total energy usage [Enos et al. 2010].

Ghosh et al. [Ghosh et al. 2012] investigate the energy consumption of several parallel scientific kernels on multiple GPUs. Specifically, they examine Matrix-Matrix Multiplication, Fast Fourier Transform, Pseudo-Random Number Generator, and 3D Finite Difference Stencils. Although these applications possess different communication and computation patterns, some common parameters, such as the number of global memory accesses and power consumption to operations per unit time, determine the energy consumption of GPU devices. Collange et al. [Collange et al. 2009] find that memory access patterns and bandwidth have a significant impact on the performance and energy consumption of GPUs. Suda and Ren [Suda and Ren 2009] suggest a way of improving the energy efficiency of GPU devices by maximizing the number of active threads on the accelerator.

Power saving of DVFS. Applying DVFS to accelerators, such as GPU devices, is investigated [Jiao et al. 2010; Abe et al. 2012; Ge et al. 2013; Mei et al. 2013]. Jiao et al. [Jiao et al. 2010] find that the energy efficiency of GPUs is mainly determined by two factors: the rate of issuing instructions and the ratio of global memory transactions to computation instructions. Abe et al. [Abe et al. 2012] investigate the effect of scaling down memory frequency with NVIDIA GeForce GTX 480. Specifically, they find 28% of system energy can be saved for matrix multiplication. Ge et al. [Ge et al. 2013] compare the effect of frequency scaling on both CPU and GPU using three typical parallel applications. They find that scaling GPU frequency higher do not consume more energy. Mei et al. [Mei et al. 2013] investigate the effect of using DVFS to improve the energy efficiency of GPUs over a wide range of benchmark applications. Overall, 19.28% energy reduction on average is saved by scaling down the GPU core voltage and frequency, with up to 4% of performance loss, in comparison to the default configuration. However, the exact effect of energy saving depends on application characteristics, and it is challenging to find the optimal setting of GPU DVFS. Price et al. [Price et al. 2015] investigate how temperature, core clock frequency and voltage affect the energy efficiency of GPUs using a radio astronomy application. They find that the power efficiency of an NVIDIA K20 GPU is improved up to 37–48% over default settings by lowering GPU supply voltage and increasing clock frequency while maintaining a low die temperature.

Workload distribution. A common hybridization method offloads a compute region to the accelerator. For example, OpenMP supports offloading a compute region to GPU devices. However, offloading often actively uses only one device at any time, with the other devices idle during that period. Workload distribution and consolidation are proposed to improve the energy efficiency of hybrid computing.

Instead of offloading a compute region manually, several research projects propose automated mechanisms to distribute workload across heterogeneous multiprocessors to improve the utilization of a hybrid machine. Qilin [Luk et al. 2009] supports an automatic technique to adaptively map computations to heterogeneous processing elements on a hybrid machine equipped with CPUs and GPUs. Qilin supports APIs to allow programmers to edit data parallelism and task parallelism. Each Qilin program is compiled into a number of small tasks that are represented using a DAG. An empirical approach that dynamically determines how to map each task to heterogeneous processing elements consists of two phases: *training run* and *reference run*. The first time that an application is executed is its training run in which the CPU and GPU execution time of each task is recorded. During the application's subsequent execution, the reference run, each task's partition ratio between CPU and GPU is dynamically determined according to the information recorded during its training run. Compared to static mappings, Qilin reduces energy consumption by 20% and improves performance by 25% on average for a set of important computations.

Wang and Ren [Wang and Ren 2010] propose a power-efficient work distribution algorithm to change the ratio of workload distribution and scale processor frequency dynamically to maximize overall system energy efficiency. They adopt a source-to-source compiler and extend the OpenMP language to support workload distribution. A number of important computations are evaluated on an Intel Core I7 CPU and an AMD 4870 GPU using the extended OpenMP language. The results show the proposed algorithm reduce 14% energy consumption on average over static mappings.

Ma et al. [Ma et al. 2012] develop GreenGPU, a holistic energy management framework for GPU-CPU heterogeneous systems. GreenGPU maximizes the overall system energy efficiency with a two-tier method. The first tier distributes the workload across the GPU and CPU dynamically to ensure both finish approximately at the same time, while the second tier applies DVFS to adjust the frequencies of CPU and GPU cores and memories at runtime to improve overall energy efficiency. The proposed method assumes the amount of operations in each loop is similar. The statistics collected during execution of early iterations are used to predict that of subsequent ones. A heuristic method adjusts the percentage of workload assigned to the GPU dynamically. GreenGPU is evaluated on NVIDIA GeForce GPUs and AMD Phenom II CPUs using CUDA benchmarks. The results show that GreenGPU reduces energy consumption on average by 24% with negligible performance loss.

Workload consolidation. Different from workload distribution to improve energy efficiency of a single application, workload consolidation aims to increase the overall system power utilization by placing multiple applications on the same set of hybrid machines. For example, on Cray XC systems, ALPS supports the MPMD mode to launch multiple different binaries simultaneously on the same nodes [Hart et al. 2014]. Li et al. [Li et al. 2011a] develop an energy-aware consolidation framework to consolidate multiple hybrid applications on a machine equipped with an Intel Xeon E5520 quadcore CPU and an NVIDIA Tesla C1060 GPU. They implement a power and performance prediction model to investigate the consolidation's impact on saving energy and improving performance using applications, such as encryption, sorting, and searching. They find during consolidation that any single application's performance and energy consumption can be negatively impacted. However, the overall execution time and energy efficiency of multiple consolidated applications can be improved significantly, between 2 to 20 times on average.

Communication-oriented power saving

High speed and low latency communication is essential for performance critical parallel applications. Besides custom interconnects, InfiniBand [InfiniBand 2002] and Ethernet are the most common commodity networks used to build supercomputers, according to the list of the Top500 released in Nov 2015. InfiniBand provides extremely low latency and high bandwidth communication and supports programmable NICs (Network Interface Cards) that offload protocol processing from the host processor. In contrast, Ethernet provides a cost-effective alternative for HPC interconnection. Recently Benito et al. [Benito et al. 2015] investigate the scalability of Ethernet for building an exascale supercomputer. However, the power consumption of the high performance network dominates a significant fraction of the total system power usage [Groves and Grant 2015]. Actually, the power consumed by the network is mainly used to maintain active links. Given an example of an IBM 8-port InfiniBand 12X switch, the links consume 64% of the total switch power [Dickov et al. 2014].

Most HPC applications are programmed using the model of bulk synchronous parallel (BSP). With the BSP model, all parallel processes are synchronized, and they perform computation or communication together almost at the same time. In addition, most scientific applications perform a large number of iteration, each of which repeats nearly the same pattern of computation and communication. The HPC network requirement commonly demands high bandwidth and low latency, because the communication phases of an HPC application are frequently optimized. As a result, the averaged utilization of the network is low, and may consist of many idle periods that provide a significant opportunity to save energy. To take advantage of this characteristic of network usage, power-aware networks, such as Energy Efficient Ethernet [IEEE 802.3az. 2010], are proposed to save network power using dynamic width, frequency, and on/off links. However, the latency caused by state switching of a power-aware network, such as the time of adjusting network link width and frequency and the delay of disabling and enabling links, must be manipulated to avoid unacceptable performance degradation. Energy Efficient Ethernet is evaluated for its effect on HPC power saving and performance degradation [Saravanan et al. 2013; Saravanan et al. 2014; Miwa et al. 2014]. Overall, the effect of using EEE is promising for future HPC systems.

Besides hardware, the communication stack is also critical for HPC applications. Liu et al. [Liu et al. 2009] evaluate and compare the energy consumption of TCP/IP and RDMA (remote direct memory access) over InfiniBand. They report that using high-speed RDMA adapters consumes a significant amount of power during communication, which may be up to 30% of system idle power. However, RDMA has better energy efficiency in comparison to TCP/IP, especially for communication intensive phases due to much fewer CPU cycles for processing protocols and due to lower traffic across memory and bus.

In the applications layer, there are also significant opportunities to save the energy consumed by data transfer. Typically, there are two main policies: 1) optimizing and reducing data transfer can improve the energy efficiency of network communication, and 2) energy-aware data transfer algorithms are proposed to decrease power consumption.

Power-aware networks

Saving power for parallel computing by shutting down data links has been investigated substantially. The opportunity to save energy for networks is normally due to two factors: 1) the average network utilization is low, and 2) the power consumption is high even when the network is idle. For example, Conner et al. [Conner et al. 2007] examine the opportunities of shutting down links between nodes during collective communications. They use simulation to investigate MPI all-to-all and all-to-one communications on a 3D Torus network, which is similar to the one used in IBM Blue Gene/L. They find approximately 50% of links are unutilized for all-to-all scatter and all-to-one reduce operations. Accordingly, almost 99% of the total network link time can be set to a shutoff state on a 64-node toroidal network. This reduce around 15~28% overall system energy by simulation. Alonso el al. [Alonso et al. 2006] investigate using on/off links to save energy for fat-tree networks by taking advantage of redundant paths between each source/destination pair and inactivity period. Nedevschi et al. [Nedevschi et al. 2008] investigate how to reduce network energy consumption using sleeping and rate-adaptation without adversely affecting performance. They find the effect of energy saving using sleeping and rate-adaptation mainly depends on the power profile of network equipment and the utilization of the network.

Traditionally, the physical layer devices (PHYs) of Ethernet dominate the power usage of network. Links are always powered on even when no data is transferred, because dummy data needs to be transferred to guarantee each link is active. In order to save the power consumed by the idle period, Energy Efficient Ethernet supports power saving mechanisms by shutting down idle links. In particular, a EEE PHY can switch to a low power mode, called Low Power Idle (LPI), with which up to 70% power can be saved for Ethernet [Saravanan et al. 2013]. However, the delay incurred by switching power state must be tolerated in order to guarantee the performance requirement for HPC applications. For example, a 10Gbps EEE link takes 3 μs to sleep and 4 μs to wake [Saravanan et al. 2014]. Using simulation, Miwa et al. [Miwa et al. 2014] find that EEE significantly decreases the system performance and the worst case of performance degradation can be 25%. Saravanan et al. [Saravanan et al. 2014] propose identifying the communication pattern of HPC applications and accordingly detecting the idle period of network usage to save more than 60% energy consumed by network links with 1% performance degradation. Miwa and Nakamura [Miwa and Nakamura 2015] investigate power allocation policies for a supercomputer with EEE. In particular, a power shifting policy can reallocate the power saved by EEE to other devices. The power shifting is evaluated using NAS Parallel Benchmarks, and the results show no performance degradation.

Similar to EEE, InfiniBand introduces power-saving features. For example, Mellanox Host Channel Adapters support Speed Reduction Power Saving (SRPS) and Width Reduction Power Saving (WRPS), with which the bandwidth of each port and the number of active lanes can be adjusted. Manipulating InfiniBand links to save power by taking advantage of the idle period of the HPC network are also investigated [Dickov et al. 2014; Dickov et al. 2015]. After analyzing the trace of typical HPC applications, such as WRF, and NAS Parallel Benchmarks, Dickov et al. [Dickov et al. 2014; Dickov et al. 2015] detect 90% of the total network idle time is inside idle internals typically longer than 200 μs. They propose to use prediction algorithms to detect these large idle periods and accordingly to shut links off during these periods. Using simulation that assumes switching link mode between on and off takes around 10 μs, they show around 20%~30% energy can be saved with the averaged performance loss less than 1%. One drawback of similar methods is that they cannot efficiently handle sudden changes in the network traffic.

Groves and Grant [Groves and Grant 2015] investigate the existing InfiniBand products in terms of power saving. As reported, there is potential for modest power saving with InfiniBand's WRPS. But the present real systems still need to improve in order to leverage these savings fully. In particular, the latency of adjusting link frequency and width of a Qlogic QDR InfiniBand network is around 4s. Disabling the link is as fast as 0.045s, because it has no interaction with the InfiniBand Subnet Manager, while enabling the link can take around 4s. With disabling links, a Mellanox 36 port switch of 4X width saves 46 watts power, which implies a potential for larger power saving on greater widths.

Energy-aware data transfer

Many power saving methods [Lim et al. 2006; Kandalla et al. 2010; Venkatesh et al. 2015] are applied to the communication phase at the application layer. Lim et al. [Lim et al. 2006] develop an MPI runtime system that supports an adaptive method to adjust the CPU frequency transparently during communication phases in MPI programs. Specifically, each MPI call in an application is intercepted using the MPI profiling layer. An adaptive training phase identifies program regions with a high concentration of MPI calls. An appropriate CPU frequency is determined for each detected region based on its CPU load, which is measured using the rate micro-operations/microsecond (or OPS) to achieve the overall energy budget. The proposed method is evaluated on a 10-node AMD Athlon-64 cluster using the NAS Parallel Benchmarks to show an average energy reduction of 12% at a cost of 2.1% longer average execution time.

Kandalla et al. [Kandalla et al. 2010] propose a power-aware collective communication algorithm for multi-core clusters equipped with InfiniBand using DVFS and DCT. A typical multi-core aware collective algorithm consists of 3 phases: the intra-node phase, the network phase, and the inter-node phase. Power-saving techniques are applied to the latter two phases. With the proposed algorithm, during the inter-node phase in which data are transferred across the network, the frequency of each CPU core is reduced down to its minimal level. In addition, the processes on the same CPU socket are grouped and only one group is allowed to process messages actively. Accordingly, the CPU frequencies of processes in the groups that are not actively processing messages are decreased. During the network phase, the CPU frequencies of non-leader processes are decreased. Using the NAS Parallel Benchmarks, the proposed algorithm is evaluated on a 64-core Intel Nehalem cluster connected with InfiniBand. The results show an 8% overall energy reduction with little performance degradation.

Venkatesh et al. [Venkatesh et al. 2015] propose Energy Aware MPI (EAM) that supports an application-oblivious MPI runtime to optimize energy consumed during communication. EAM aims to optimize the energy consumed at a slack period, i.e., the time spent in a single MPI call. Therefore, EAM does not need to profile the targeted application. It predicts the communication time for common MPI primitives, such as point-to-point, collective, progress, and blocking/non-blocking. When EAM detects a slack period is long enough, it applies appropriate power levers, including DVFS and core-idling, at the start of an MPI call to decrease energy consumption. It works effectively when communication times are increased by workload congestion or system noise. In particular, when the predicted communication time exceeds a lever's overhead, the lever is applied. EAM is implemented using MVAPICH2. EAM performance is evaluated against the default MPI performance optimization for ten applications using up to 4,096 processes and energy consumption is reduced by 5-41% with less than 4% performance loss.

Alan et al. [Alan et al. 2015] introduce data transfer algorithms that consider energy efficiency at the end systems. The algorithms model and estimate the energy consumption during data transfers and tune application-layer parameter levels for the required optimization. The parameters tuned include pipelining of the transfer of a large number of small files, parallelism of the number of streams used to transfer a file, and concurrency of the transfer of multiple files over different channels. Benchmarking is performed against various non-energy aware algorithms using a custom GridFTP client on both wide area and local area network test beds. The evaluation find that energy-aware data transfer algorithms can achieve up to 30% overall energy savings with no or minimal degradation in throughput.

Communication-avoidance

Minimizing communication in numerical linear algebra is extensively studied [Ballard et al. 2011; Baboulina et al. 2012]. Ballard et al. [Ballard et al. 2011] investigate the communication lower bound for the computation problem the structure of which resembles three nested loops, such as matrix multiplication, LU factorization, n-body problem, and Fast Fourier transform (FFT). Typically, the lower bound is achieved by compensating memory space, i.e. using extra memory to reduce communication. In terms of matrix (n by n) multiplication, using the 2.5D algorithm [Solomonik and Demmel 2011], the required extra memory space for each node can be up to $({n^2}/{p^{2/3}} - {n^2}/{p})$, where p is the number of nodes. With the extra memory space, the performance gain can be up to 3X in comparison to the case without extra memory [Solomonik and Demmel 2011].

Demmel et al. [Demmel et al. 2013] analyze the energy saving effect of communication-avoiding algorithms. In particular, they find a region of perfect strong scaling exists for communication-avoiding algorithms, in which the execution time decreases as the number of processors increases while no extra energy is consumed. However, the perfect strong scaling region only exists when $p \leqq \left({n^3}/{M^{3/2}}\right)$, where M is the used memory.

Saving energy with automatic tuning

Automatic performance tuning, or auto-tuning [Demmel et al. 2009; Benkner et al. 2014], is used to minimize the execution time of parallel programs, and optimal performance can be obtained with minimal programmer involvement. In the parallel computing domain, traditionally automatic tuning is expressed as a single-objective problem to improve performance only. Specifically, auto-tuning evaluates a set of alternative implementations, searching for the best combination of code transformations and parameter settings that delivers optimal performance on the targeted architecture. Auto-tuning can provide portable optimized performance across different platforms for a domain-specific computing library or a general application.

The general approach of auto-tuning is also suited to optimizing the energy consumption of parallel computing [Tiwari et al. 2012]. Represented as a multiple-objective optimization problem [Balaprakash et al. 2013; Gschwandtner et al. 2014b], auto-tuning can be applied to parallel applications to make an optimal tradeoff between reducing energy consumption and improving execution time.

Auto-tuning tools typically search for the right compiler configurations on the targeted platform within an extremely large space. The exhaustive search of the whole space is too excessively time-consuming to be practical. Therefore, heuristic, non-linear optimization, and machine learning are frequently used to prune the search space at a risk of possibly losing the best combination.

Tiwari et al. [Tiwari et al. 2012] utilize Active Harmony [Tapus et al. 2002], which is a performance auto-tuning system for distributed applications, to tune the energy usage and performance for stencil computation. The auto-tuning is achieved by adjusting software level performance-related tunable parameters, including tiling factors and loop unrolling factors, and processor clock frequency. An offline search method is applied to find the optimal parameter combination with respect to performance, energy, energy-delay product, and energy-delay-squared product respectively. With the proposed method, energy consumption can be reduced 5.4% with only 4% performance loss.

Rahman et al. [Rahman et al. 2011] develop an automated empirical tuning method for scientific applications to balance energy consumption and performance on a Chip Multi-Processor machine. They utilize POET [Rahman et al. 2011], an interpreted program transformation language for parameterizing compiler optimizations, to generate a performance-optimized implementation for an annotated source code. A transformation-aware search engine explores the optimization space to determine an appropriate value for each optimization parameter specified in a POET script. Power consumption statistics collected in real time during the execution of the optimized binary provide feedback to tune performance and energy consumption collectively. Using matrix computation kernels as benchmarks, the proposed method is evaluated on two machines equipped with quad-core AMD Opteron processors and quad-core Intel processors respectively. The results show energy consumption is decreased without sacrificing performance.

Balaprakash et al. [Balaprakash et al. 2013] formulate the relation between performance, power, and energy for HPC kernels using multi-objective optimization. In particular, they use a Pareto front to represent the optimal tradeoff between different optimization criteria: time, power, and energy. They investigate common HPC kernels, including sparse matrix multiplication, quick sort, and TORCH, for making multiple-objective decisions. Additionally, they formulate the conditions in which the multi-objective formulation can benefit real problems and the situations in which the number of objectives can be reduced. The proposed framework is evaluated on an Intel Xeon Phi coprocessor, an Intel Xeon E5530, and an IBM Blue Gene/Q. In the case where optimization objectives are correlated, a single "ideal" decision exists. In contrast, significant tradeoffs are required to address orthogonal multiple objectives.

INSIEME [Jordan et al. 2012] supports finding the optimal Pareto set using an evolutionary algorithm to prune the search space that consists of various combinations of computing environment settings, including both performance and power counters. Gschwandtner et al. [Gschwandtner et al. 2014b] utilize the INSIEME compiler to generate optimized code for scientific computing and to analyze the trade-off between time, energy, and resource usage. In particular, they propose a multi-objective search-based method, called RS_GDE3, to optimize for three conflicting criteria: execution time, energy consumption, and resource usage. The proposed auto-runner is evaluated for applications, including matrix multiplication, an *n*-body problem, 2D Jacobi, and a 3D stencil computation. In comparison to a hierarchical and a random search, RS_GDE3 offers superior quality at a fraction of required time (5%) or energy (8%).

Miceli et al. [Miceli et al. 2012] propose AutoTune to support a plugin-driven approach for tuning parallel applications in an automated manner. Each plugin is for a specific tuning objective, such as MPI runtime and energy consumption optimization via adjusting CPU frequency. AutoTune develop a Periscope Tuning Framework (PTF) that takes source code as input and performs static analysis. The analysis outcome steers the tuning process of selecting a tuning plugin and conducting its actions. All plugins work with PTF to achieve a multi-aspect application tuning. Specifically, each plugin specifies a number of tuning parameters and each code region may be tuned by several plugins. The whole tuning process can repeat several times to generate a tuning report that is used to assist application or production deployment.

Saving energy with approximation methods

Approximation-based methods to save energy are motivated by both software and hardware factors. At the hardware layer, NTC has been taken as an effective way of saving processor power. In order to mitigate the degraded reliability of NTV processors, significance-based computing [Sampson et al. 2011; Karpuzcu et al. 2013; Dreslinski et al. 2010; Gschwandtner et al. 2014a] is proposed, which typically partitions a given program into significant and insignificant parts. The insignificant part of a program is processed on NTV processors to achieve better power-saving and lower energy usage with acceptable accuracy loss. At the software layer, many computations, including web search engines, image processing, video encoding, machine learning, and data mining, exhibit a trade-off between the accuracy of the produced result and the time and energy required to produce the result. Approximation-based energy efficient computing [Hoffmann et al. 2009; Baek and Chilimbi 2010; Sampson et al. 2011] exploits the opportunity of sacrificing computation accuracy to improve performance and to decrease energy usage.

Scientific simulations often use over-provisioned data types, like double-precision floating-points. Using single-precision or mixing floating-point precisions are practical methods to generate code with better performance and energy efficiency [Anzt et al. 2010; Dongarra et al. 2012; Lam et al. 2013; Linderman et al. 2010; Rubio-Gonz´alez et al. 2013; Schkufza et al. 2014]. Moreover, most scientific simulations exploit iterations to converge the results gradually. Sometimes, a large number of extra iterations are performed to pursue a tiny accuracy improvement, the loss of which may be tolerable in many scenarios.

NTV-based Significance-driven Computing

Gschwandtner et al. [Gschwandtner et al. 2014a] propose significance-driven computing for energy-aware parallel computing utilizing NTV processors. They present a significance-driven execution paradigm that selectively uses NTV and algorithmic error tolerance to reduce energy consumption. In particular, they analyze codes in terms of their significance and then apply an adaptive execution scheme to an example of Jacobi with iterative loops, which switch between unreliable execution over NTV cores and reliable execution on a normal core. Iterative solvers, like Jacobi, repeatedly update the solution of a system of equations until the result converges. Iterations executed on NTV processors face the errors caused by unreliable hardware. These errors can be mitigated using an increased number of iterations. The evaluation shows that up to 65% energy saving is achieved for a parallel version of Jacobi without compromising performance and accuracy. However, the energy saving and performance improvement is sensitive to the bit position of errors. Typically, bit errors at lower positions require fewer iterations to recover, which results in overall energy efficient execution. In contrast, bit errors at higher positions require a significant number of extra iterative operations to recover and the corresponding energy saving is lower.

Approximate Data Types

In regard to the energy-accuracy tradeoff, a key challenge is how to isolate the parts of the program that must be precise from those that can be approximated. In order to address this challenge, Sampson et al. [Sampson et al. 2011] propose a model for approximate programming, which is a type system that isolates the precise portion of the program from the approximate portion. In particular, the proposed model allows the programmer to use type qualifiers to declare data that may be subject to approximate computation. In addition, they develop a language, called EnerJ, for principled approximate computing, which extends Java by adding approximate data types. Using these approximate date types, the system can map the approximate data values to low-power storage, use low-power operations, and apply more energy-efficient algorithms. EnerJ supports *endorsements*, programmer-specified points at which approximate-to-precise data flow may occur. To demonstrate the effectiveness of EnerJ, Sampson et al. port a number of applications, including FFT, LU, and a Monte Carlo simulation. For these applications, nearly all of the floating-point operations are approximated, while no, or very little approximate integer operations are exhibited. With EnerJ's annotated approximate type qualifiers, they achieve 10%~15% energy saving with little accuracy loss.

A significant number of scientific applications rely on floating-point operations. Frequently, programmers choose the maximum practical precision for the variables (typically double-precision with most languages). However, for many scenarios, double-precision is overkill, using more precise representation than it is necessary. In comparison to single-precision, using double-precision floating-point data requires twice the memory space to store and twice the bandwidth to transfer. Dongarra et al. [Dongarra et al. 2012] evaluate the performance and energy consumption of LAPACK and PLASMA LU Factorization using single-precision, double-precision, and mixed-precision respectively on Intel Xeon system. The results show that mixing floating-point precisions or selecting an appropriate floating-point precision dynamically at runtime can improve performance and correspondingly save energy. In particular, LU Factorization executed with single-precision saves more than 50% time and 50%

energy in comparison to double-precision. Using mixed-precision, the execution time and energy can be reduced around 25% to 30%. However, with mixed- or single-precision, the accuracy must be guaranteed using a careful design of algorithms.

Anzt et al. [Anzt et al. 2010] investigate the energy saving effect of applying mixed floating-point precision to a linear solver, such as those used in computational fluid dynamics. The idea is to use single-precision in the inner solvers of an iterative refinement method, while updating the solution using double-precision. The mixed-precision linear solver is evaluated on several HPC clusters equipped with Intel Xeon processors with a varied number of machines. On average, energy consumption is reduced more than 50% with the mixed-precision linear solver.

Rather than allowing programmers to control the precision of computation manually, several research projects support automatic methods to adjust precision dynamically. Using compile-time program-driven *static analysis*, Linderman et al. [Linderman et al. 2010] propose an automated method to analyze numerical precision and to conduct program optimization. In particular, they develope Gappa++ to assist automatic analysis of numerical errors in floating-point, linear and non-linear computation using a combination of range arithmetic and algebraic rewriting. The programmer or compiler can use the proposed method to optimize a program's precision. Gappa++ is verified using applications such as Bayesian network, neural prosthetics and Black-Scholes stock option pricing.

Lam et al. [Lam et al. 2013] develop a tool that uses binary instrumentation to adapt a double-precision program to a mixed-precision version without modifying the source code. An automatic breadth-first search algorithm is applied to find all possible operations in which double-precision can be replaced with single-precision. The tool is evaluated using the NAS Parallel Benchmarks. They find 20% of all floating-point operations in these benchmarks can be replaced with single-precision equivalence. In addition, the entire AMG (Algebraic MultiGrid) microkernel can be replaced with single-precision floating-point operands and accordingly achieve a 2X performance improvement.

Rubio-Gonz´alez et al. [Rubio-Gonz´alez et al. 2013] develop a tool, called Precimonius, to tune a program's floating-point precision in an automated way. The programmer can specify an acceptable accuracy criteria for a program and Precimonius searches all floating-point variables to detect an appropriate type configuration using the delta-debugging approach, the average complexity of which is $O(n\log n)$. The generated type configuration maps each floating-point variable to its appropriate precision. The effect of Precimonius is evaluated using the GNU Scientific Library to show 25% performance improvement.

Schkufza et al. [Schkufza et al. 2014] propose to use stochastic search to address floating-point optimization. The proposed method is supported in a program optimization tool, the implementation of which takes advantage of a JIT assembler. Using a technique, called validating optimization, they provide evidence of correctness rather than proving the correctness of the proposed method. Significant performance improvement, up to 6 times, is found when applying the proposed method to the Intel C numerical library and real-world applications, such as S3D. The above tools that optimize floating-point precision can also decrease the power consumption of scientific computing due to the strong correlation between performance and energy efficiency.

Approximate Iterations and Functions

Hoffmann et al. [Hoffmann et al. 2009] develop a technique, called *code perforation*, for automatically augmenting an application to trade off accuracy in return for improved performance and reduced energy consumption. The applied scope of applications include video, audio and image processing, machine learning and information retrieval applications, Monte Carlo simulations, and scientific computations for which an output is produced within an acceptable precision range. The proposed technique is based on SpeedPress, an LLVM-based compiler that exploits code perforation to trade accuracy for performance and energy saving, and SpeedGuard, a runtime system that dynamically monitors the accuracy degradation and accordingly enables code perforation. In particular, programmers are allowed to specify a distortion bound and SpeedPress automatically identifies loop iterations of the computation that can be discarded without violating this bound. Specifically, the compiler uses profiling to detect the tradeoff space of accuracy and performance/energy saving. Overall, the detected tradeoff space can be explored to maximize the performance target given a distortion constraint or to minimize the distortion target given a performance constraint. The proposed system is evaluated using the PARSEC benchmark suite [Bienia et al. 2008]. The results show that the transformed applications run 2~3 times faster with 10% distortion.

Baek and Chilimbi [Baek and Chilimbi 2010] propose a system called Green to support an energy-conscious programming framework using loop and function approximation. Specifically, the programmers specify a maximal Quality of Service (QoS) loss, while Green provides statistical guarantees that the application will meet the targeted QoS. In particular, to support function approximation, programmers must provide a series of versions with different approximate levels for the same function. Green measures the QoS loss by comparing the values returned by both the approximate version and the precise version of the same function. With the loop approximation, fewer loop iterations are conducted and a programmer must provide a function to enable Green to calculate the QoS loss generated from early loop termination. Given an application, using the approximation input specified by programmers, Green constructs a "calibration" program. At runtime, Green observes the QoS loss and determines which approximate version of the program is executed dynamically. The proposed Green system is evaluated using web search, graphic computation, machine learning, signal processing, and financial applications. In particular, with 0.27% QoS degradation, the performance and energy consumption of Bing Search, a commercial web search engine, are improved by 27% and 14% respectively.

Hoffmann et al. [Hoffmann et al. 2011] implement a system, called PowerDial, that adapts the behavior of an application to execute in the presence of load and power fluctuations. Specifically, PowerDial can translate static configuration parameters into *dynamic knobs* that are manipulated by the PowerDial control system to trade the accuracy of the computation dynamically for reductions in computational resources, which improves performance and reduces energy consumption. The proposed system is evaluated using 4 benchmark applications, including the swaptions financial analysis application and the swish++ search engine, in environments with fluctuating load and power characteristics. PowerDial enables applications to adapt effectively as a power cap changes. Specifically, PowerDial can dynamically move the applications between Pareto-optimal points with different computational demands. In addition, PowerDial can dynamically change the number of machines required to service time-varying workloads while guaranteeing a pre-defined QoS level.

Conclusions

Optimizing the performance of parallel applications is already a challenging task. The future of parallel computing must account for energy efficiency, which further increases the complexity of parallel programming. This paper describes the state-of-the-art of power-saving methods adopted in the software layer of parallel computing and provides programmers insights in how to balance energy consumption and optimal performance without increasing programming complexity.

Recent hardware innovations support a rich set of techniques that software can exploit to manipulate power usage and to make appropriate engineering decisions to decrease energy consumption at runtime. This paper surveys the methods to measure power consumption and to analyze energy efficiency for parallel computing on different architectures. In addition, our survey presents a taxonomy for software methods to improve the energy efficiency of parallel applications. Overall, programmers must consider energy efficiency when they design algorithms and when they deploy an application. Mixed floating-point precision and communication-avoiding algorithms are effective methods that can be adopted at the phase of designing an algorithm. Additionally, approximation-based methods achieve improved energy efficiency by refining data types and the number of iterations to avoid wasting memory and computing resources. In contrast, when a parallel application is deployed on a specific platform, several tunable power parameters can be controlled to save energy for different levels of parallelism. DVFS and DCT are suitable for decreasing energy consumption with imbalanced loads. Hybridizing a program exploits energy efficient GPU accelerators and Intel Phi coprocessors. Slowing or shutting down network components can also improve overall system energy efficiency. Significance-driven computing allows parallel applications to exploit the superior energy efficiency of NTC. The overall system energy utilization can be managed efficiently by power-aware job schedulers.

Although many research projects explore techniques to decrease the energy consumption of parallel applications, several open problems remain. It is still unclear how to simplify the programming complexity of balancing performance and power consumption in the application layer. What models are appropriate to provide a transparent power-aware programming interface? Energy auto-tuning improves energy efficiency across platforms in an automated manner. However, making it practical for handling large-scale parallel applications is still challenging. The state-of-the-art methods surveyed in this article identify a wide range of opportunities to improve energy efficiency and motivate researchers to create breakthrough approaches to address the above questions for future parallel computing systems.

Acknowledgements

This work is supported by the Australian Research Council under the Linkage grant scheme, and is supported by Cray Inc.

References

Yuki Abe, Hiroshi Sasaki, Martin Peres, Koji Inoue, Kazuaki Murakami, and Shinpei Kato. 2012. Power and Performance Analysis of GPU-Accelerated Systems. In *Proceedings of the 2012 Workshop on Power-Aware Computing and Systems (HotPower'12).* ACM Press, New York, NY.

Ismail Alan, Engin Arslan, and Tevfik Kosar. Energy-Aware Data Transfer Algorithms. 2015. In *Proceedings of the 2015 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'15).* ACM Press. DOI:http://dx.doi.org/10.1145/2807591.2807628

Marina Alonso, Salvador Coll, Juan-Miguel Martinez, Vicente Santonja, Pedro Lopez, and Jose Duato. 2006. Dynamic Power Saving in Fat-Tree Interconnection Networks using On/Off Link. In *Proceedings* *of the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS'06).* IEEE Press, 299–307. DOI:http://dx.doi.org/10.1109/IPDPS.2006.1639599

Saman Amarasinghe, Dan Campbell, William Carlson, Andrew Chien, William Dally, Elmootazbellah Elnohazy, Mary Hall, Robert Harrison, William Harrod, Kerry Hill, Jon Hiller, Sherman Karp, Charles Koelbel, David Koester, Peter Kogge, John Levesque, Daniel Reed, Vivek Sarkar, Robert Schreiber, Mark Richards, Al Scarpelli, John Shalf, Allan Snavely, and Thomas Sterling. 2009. Exascale Software Study: Software Challenges in Extreme Scale Systems. DARPA Report.

Hartwig Anzt, Björn Rocker, Vincent Heuveline. 2010. Energy Efficiency of Mixed Precision Iterative Refinement Methods using Hybrid Hardware Platforms. *Computer Science - Research and Development*, Vol. 25, Issue 3-4. Springer-Verlag Press, Berlin Heidelberg, 141–148. DOI:http://dx.doi.org/10.1007/s00450-010-0124-2

ASCAC Subcommittee. 2014. The Top Ten Exascale Research Challenges. ASCAC (Advanced Scientific Computing Advisory Committee) Subcommittee Report, Department Of Energy, U.S.

Victor Avelar, Dan Azevedo, and Alan French. 2014. PUE™: A Comprehensive Examination of the Metric. TheGreenGrid White Paper #49.

Marc Baboulina, Simplice Donfacka, Jack Dongarra, Laura Grigoria, Adrien Rémya, and Stanimire Tomovb. 2012. A Class of Communication-Avoiding Algorithms for Solving General Dense Linear Systems on CPU/GPU Parallel Machines. In *Proceedings of the International Conference on Computational Science, (ICCS 2012).* DOI:http://dx.doi.org/10.1016/j.procs.2012.04.003

Woongki Baek and Trishul M. Chilimbi. 2010. Green: A Framework for Supporting Energy-Conscious Programming using Controlled Approximation. In *Proceedings of the 31st ACM SIGPLAN conference on programming language design and implementation (PLDI '10).* ACM Press, New York, NY, 198–209. DOI:http://dx.doi.org/10.1145/1806596.1806620

Peter E. Bailey, Aniruddha Marathe, David K. Lowenthal, Barry Rountree, and Martin Schulz. 2015. Finding the Limits of Power-Constrained Application Performance. In *Proceedings of the 2015 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'15).* ACM Press. DOI:http://dx.doi.org/10.1145/2807591.2807637

Prasanna Balaprakash, Ananta Tiwari, and Stefan M. Wild. 2013. Multi-Objective Optimization of HPC Kernels for Performance, Power, and Energy. In *Proceedings of 4th International Workshop on Performance Modeling, Benchmarking, and Simulation of HPC Systems (PMBS12).* DOI:http://www.mcs.anl.gov/papers/P4069-0413.pdf

Grey Ballard, James Demmel, Olga Holtz, and Oded Schwartz. 2011. Minimizing Communication in Numerical Linear Algebra. SIAM (Society for Industrial and Applied Mathematics) Journal on Matrix Analysis and Applications, vol. 32(3), 866–901.

Natalie J. Bates and Michael K. Patterson. 2013. Achieving the 20MW Target: Mobilizing the HPC Community to Accelerate Energy Efficient Computing. In *Transition of HPC Towards Exascale Computing*, E.H. D'Hollander et. al (Eds.). IOS Press, 37-45. DOI:http://dx.doi.org/10.3233/978-1-61499-324-7-37

Mariano Benito, Enrique Vallejo, and Ramon Beivide. 2015. On the Use of Commodity Ethernet Technology in Exascale HPC Systems. In *Proceedings of 22nd IEEE International Conference on High Performance Computing (HiPC'15).* IEEE Press.

Siegfried Benkner, Franz Franchetti, Hans Michael Gerndt, and Jeffrey K. Hollingsworth. 2014. Automatic Application Tuning for HPC Architectures. Dagstuhl Reports, Vol. 3-9, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 214–244. DOI:http://dx.doi.org/10.4230/DagRep.3.9.214

Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. 2008. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques (PACT'08).* ACM Press, 72–81. DOI:http://dx.doi.org/10.1145/1454115.1454128

Brad D. Bingham and Mark R. Greenstreet. 2008. Computation with Energy-Time Trade-Offs: Models, Algorithms and Lower-Bounds. In *Proceedings of the 2008 International Symposium on Parallel and Distributed Processing with Applications (ISPA '08).* IEEE Press, 143–152. DOI:http://dx.doi.org/10.1109/ISPA.2008.127

David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: A Framework for Architectural Level Power Analysis and Optimizations. In *Proceedings of the 27th International Symposium on Computer Architecture (ISCA'00).* IEEE Press, 83–94.

Sangyeun Cho and Rami G. Melhem. 2010. On the Interplay of Parallelization, Program Performance, and Energy Consumption. *IEEE Transactions on Parallel and Distributed Systems* 21, 3 (Mar. 2010), 342–353. DOI:http://dx.doi.org/10.1109/TPDS.2009.41

Jee Whan Choi, Daniel Bedard, Robert Fowler, and Richard Vuduc. 2013. A Roofline Model of Energy. In *Proceedings of the 27th IEEE International Parallel and Distributed Processing Symposium (IPDPS'13).* IEEE Press, 661–672. DOI:http://dx.doi.org/10.1109/IPDPS.2013.77

Sylvain Collange, David Defour, and Arnaud Tisserand. 2009. Power Consumption of GPUs from a Software Perspective. In *Proceedings of the 9th International Conference on Computational Science (ICCS '09).* Springer Press, Berlin, Heidelberg, 914–923. DOI:http://dx.doi.org/10.1007/978-3-642-

01970-8_92

S. Conner, S. Akioka, M. J. Irwin, and P. Raghavan. 2007. Link Shutdown Opportunities during Collective Communications in 3-D Torus Nets. In *Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS'07).* IEEE Press, 1–8. DOI:http://dx.doi.org/10.1109/IPDPS.2007.370534

Matthew Curtis-Maury, Filip Blagojevic, Christos D. Antonopoulos, and Dimitrios S. Nikolopoulos. 2007. Prediction-Based Power-Performance Adaptation of Multithreaded Scientific Codes. *IEEE Transactions on Parallel and Distributed Systems* 19, 10(Oct. 2008), 1396–1410. DOI:http://dx.doi.org/10.1109/TPDS.2007.70804

Matthew Curtis-Maury, James Dzierwa, Christos D. Antonopoulos, and Dimitrios S. Nikolopoulos. 2006a. Online Strategies for High-Performance Power-Aware Thread Execution on Emerging Multiprocessors. In *Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS'06).* IEEE Press, 298–307. DOI:http://dx.doi.org/10.1109/IPDPS.2006.1639598

Matthew Curtis-Maury, James Dzierwa, Christos D. Antonopoulos, and Dimitrios S. Nikolopoulos. 2006b. Online Power-Performance Adaptation of Multithreaded Programs using Hardware Event-Based Prediction. In *Proceedings of the 20th annual international conference on Supercomputing (ICS '06).* ACM Press, 157–166. DOI:http://dx.doi.org/10.1145/1183401.1183426

Howard David, Eugene Gorbatov, Ulf R. Hanebutte, Rahul Khanna, and Christian Le. 2010. RAPL: Memory Power Estimation and Capping. In *Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design (ISLPED '10).* ACM Press, 189–194. DOI:http://dx.doi.org/10.1145/1840845.1840883

James Demmel, Jack Dongarra, Armando Fox, Sam Williams, Vasily Volkov, and Katherine Yelick. 2009. Accelerating Time-to-Solution for Computational Science and Engineering. SciDAC Review, No. 15.

James Demmel, Andrew Gearhart, Benjamin Lipshitz, and Oded Schwartz. 2013. Perfect Strong Scaling Using No Additional Energy. In *Proceedings of the 27th IEEE International Parallel and Distributed Processing Symposium (IPDPS'13).* IEEE Press, 649–660. DOI:http://dx.doi.org/10.1109/IPDPS.2013.32

Branimir Dickov, Paul M. Carpenter, Miquel Pericas, and Eduard Ayguade. 2015. Self-Tuned Software-Managed Energy Reduction in Infiniband Links. In *Proceedings of the 21st IEEE International Conference on Parallel and Distributed Systems (ICPADS'15).* IEEE Press.

Branimir Dickov, Miquel Pericas, Paul M. Carpenter, Nacho Navarro, and Eduard Ayguade. 2014. Software-Managed Power Reduction in Infiniband Links. In *Proceedings of the 2015 International Conference on Parallel Processing (ICPP '15).* IEEE Press, 311–320. DOI:http://dx.doi.org/10.1109/ICPP.2014.40

Jack Dongarra et al. 2011. The International Exascale Software Project Roadmap. International Journal of High Performance Computing Applications, Vol. 25-1, 3–60. DOI:http://dx.doi.org/10.1177/1094342010391989

Jack Dongarra, Hatem Ltaief, Piotr Luszczek, and Vincent M. Weaver. 2012. Energy Footprint of Advanced Dense Numerical Linear Algebra using Tile Algorithms on Multicore Architecture. In *Proceedings of the 2nd International Conference on Cloud and Green Computing (CGC'12).* IEEE Press, 274–281. DOI:http://dx.doi.org/10.1109/CGC.2012.113

Ronald G. Dreslinski, Michael Wieckowski, David Blaauw, Dennis Sylvester, and Trevor Mudge. 2010. Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits. In *Proceedings of the IEEE*, Vol. 98–2. IEEE Press, 253–266. DOI:http://dx.doi.org/10.1109/JPROC.2009.2034764

Daniel A. Ellsworth, Allen D. Malony, Barry Rountree, and Martin Schulz. 2015. Dynamic Power Sharing for Higher Job Throughput. In *Proceedings of the 2015 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'15).* ACM Press. DOI:http://dx.doi.org/10.1145/2807591.2807643

E.N. Elnozahy, Michael Kistler, and Ramakrishnan Rajamony. Energy-Efficient Server Clusters. 2003. Power-Aware Computer Systems of Lecture Notes in Computer Science, Vol. 2325. Springer Berlin Heidelberg Press, 179–197. DOI:http://dx.doi.org/ 10.1007/3-540-36612-1_12

Jeremy Enos, Craig Steffen, Joshi Fullop, Michael Showerman, Guochun Shi, Kenneth Esler, Volodymyr Kindratenko, John E. Stone, and James C. Phillips. 2010. Quantifying The Impact of GPUs on Performance and Energy Efficiency in HPC Clusters. In *Proceedings of the 2010 International Green Computing Conference.* IEEE Press, 317–324. DOI:http://dx.doi.org/10.1109/GREENCOMP.2010.5598297

Hadi Esmaeilzadeh, Ting Cao, Xi Yang, Stephen M. Blackburn, and Kathryn S. McKinley. 2011. Looking Back on the Language and Hardware Revolutions: Measured Power, Performance, and Scaling. In *Proceedings of the 16th international conference on architectural support for programming languages and operating systems (ASPLOS XVI).* ACM Press, New York, NY, 319–332. DOI:http://dx.doi.org/10.1145/1950365.1950402

Hadi Esmaeilzadeh, Ting Cao, Xi Yang, Stephen M. Blackburn, and Kathryn S. McKinley. 2012. What is

Happening to Power, Performance, and Software? IEEE Micro, vol. 32–3. IEEE Press, 110–121. DOI:http://dx.doi.org/10.1109/MM.2012.20

Maja Etinski, Julita Corbalan, Jesus Labarta, and Mateo Valero. 2010. Optimizing Job Performance Under a Given Power Constraint in HPC Centers. In *Proceedings of the 2010 International Green Computing Conference*. IEEE Press, 257–267. DOI:http://dx.doi.org/10.1109/GREENCOMP.2010.5598303

Maja Etinski, Julita Corbalan, Jesus Labarta, and Mateo Valero. 2012. Parallel Job Scheduling for Power Constrained HPC Systems. Parallel Computing, Vol. 38(12). ELSEVIER Press, 615–630. DOI:http://dx.doi.org/10.1016/j.parco.2012.08.001

Xizhou Feng, Rong Ge, and Kirk W. Cameron. 2005. Power and Energy Profiling of Scientific Applications on Distributed Systems. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*. IEEE Press, 34. DOI:http://dx.doi.org/10.1109/IPDPS.2005.346

Jeremy Fowers, Greg Brown, John Wernsing, and Greg Stitt. 2013. A Performance and Energy Comparison of Convolution on GPUs, FPGAs, and Multicore Processors. ACM Transactions on Architecture and Code Optimization (TACO) - Special Issue on High-Performance Embedded Architectures and Compilers, vol. 9-4. ACM Press, 110–121. DOI:http://dx.doi.org/10.1145/2400682.2400684

V. W. Freeh, T. K. Bletsch, and F. L. Rawson. 2007a. Scaling and Packing on a Chip Multiprocessor. In *Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS'07)*. IEEE Press, 1–8. DOI:http://dx.doi.org/10.1109/IPDPS.2007.370539

Vincent W. Freeh, David K. Lowenthal, Feng Pan, Nandini Kappiah, Rob Springer, Barry L. Rountree, and Mark E. Femal. 2007b. Analyzing the Energy-Time Trade-Off in High-Performance Computing Applications. IEEE Transactions on Parallel and Distributed Systems, vol. 18, 6, 825–848.

Vincent W. Freeh, Feng Pan, Nandini Kappiah, and David K. Lowenthal. 2005a. Using Multiple Energy Gears in MPI Programs on a Power-Scalable Cluster. In *Proceedings of the 10th ACM SIGPLAN symposium on principles and practice of parallel programming (PPoPP 2005)*. ACM Press, New York, NY, 164–173. DOI:http://dx.doi.org/10.1145/1065944.1065967

Vincent W. Freeh, Feng Pan, Nandini Kappiah, and David K. Lowenthal. 2005b. Exploring the Energy-Time Tradeoff in MPI Programs on a Power-Scalable Cluster. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*. IEEE Press, 4a. DOI:http://dx.doi.org/10.1109/IPDPS.2005.346

Rong Ge and Kirk W. Cameron. 2007a. Power-Aware Speedup. In *Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS'07)*. IEEE Press, 1–10. DOI:http://dx.doi.org/10.1109/IPDPS.2007.370246

Rong Ge, Xizhou Feng, and Kirk W. Cameron. 2005. Performance-Constrained Distributed DVS Scheduling for Scientific Applications on Power-aware Clusters. In *Proceedings of the 2005 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'05)*. ACM Press, 34. DOI:http://dx.doi.org/10.1109/SC.2005.57

Rong Ge, Xizhou Feng, Wu-chun Feng, and Kirk W. Cameron. 2007b. CPU MISER: A Performance-Directed, Run-Time System for Power-Aware Clusters. In *Proceedings of the 2007 International Conference on Parallel Processing (ICPP '07)*. IEEE Press, 18. DOI:http://dx.doi.org/10.1109/ICPP.2007.29

Rong Ge, Xizhou Feng, S. Song, H.-C. Chang, D. Li, and K. W. Cameron. 2009. PowerPack: Energy Profiling and Analysis of High-Performance Systems and Applications. IEEE Transactions on Parallel and Distributed Systems, vol. 21, 5, 658–671. DOI:http://dx.doi.org/10.1109/TPDS.2009.76

Rong Ge, Ryan Vogt, Jahangir Majumder, Arif Alam, Martin Burtscher, and Ziliang Zong. 2013. Effects of Dynamic Voltage and Frequency Scaling on a K20 GPU. In *Proceedings of the 42nd International Conference on Parallel Processing (ICPP2013)*. IEEE Press, 826–833. DOI:http://dx.doi.org/10.1109/ICPP.2013.98

Yiannis Georgiou, Thomas Cadeau, David Glesser, Danny Auble, Morris Jette, and Matthieu Hautreux. 2014. Energy Accounting and Control with SLURM Resource and Job Management System. Distributed Computing and Networking of Lecture Notes in Computer Science, Vol. 8314. Springer Berlin Heidelberg Press, pages 96–118. DOI:http://dx.doi.org/10.1007/978-3-642-45249-9_7

Sayan Ghosh, Sunita Chandrasekaran, and Barbara Chapman. 2012. Energy Analysis of Parallel Scientific Kernels on Multiple GPUs. In *Proceedings of the 2012 Symposium on Application Accelerators in High Performance Computing (SAAHPC)*. IEEE Press, 54–63. DOI:http://dx.doi.org/10.1109/SAAHPC.2012.17

Ryan E. Grant and Ahmad Afsahi. 2006. Power-Performance Efficiency of Asymmetric Multiprocessors for Multi-Threaded Scientific Applications. In *Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS'06)*. IEEE Press, 300–308. DOI:http://dx.doi.org/10.1109/IPDPS.2006.1639601

Peter Greenhalgh. 2011. big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7. ARM Whitepaper.

Laura Grigori, James W. Demmel, and Hua Xiang. 2011. CALU: A Communication Optimal LU Factorization Algorithm. SIAM Journal on Matrix Analysis and Applications, vol. 32(4). Society for

Industrial and Applied Mathematics Press, 1317–1350. DOI:http://dx.doi.org/10.1137/100788926

Taylor Groves and Ryan Grant. 2015. Power Aware, Dynamic Provisioning of HPC Networks. Sandia National Laboratories report, SAND2015-8717.

Philipp Gschwandtner, Charalampos Chalios, Dimitrios S. Nikolopoulos, Hans Vandierendonck, and Thomas Fahringer. 2014a. On the Potential of Significance-Driven Execution for Energy-Aware HPC. In Proceedings of *5th Energy-aware High Performance Computing (AnE HPC 2014).* Springer Press, Berlin Heidelberg. DOI:http://dx.doi.org/10.1007/s00450-014-0265-9

Philipp Gschwandtner, Juan J. Durillo, and Thomas Fahringer. 2014b. Multi-Objective Auto-Tuning with Insieme: Optimization and Trade-Off Analysis for Time, Energy and Resource Usage. In Proceedings of the *20th European Conference on Parallel Processing (Euro-Par 2014).* Springer International Publishing, 87–98. DOI:http://dx.doi.org/10.1007/978-3-319-09873-9_8

Daniel Hackenberg, Thomas Ilsche, Robert Schone, Daniel Molka, Maik Schmidt, and Wolfgang E. Nagel. 2013. Power Measurement Techniques on Standard Compute Nodes: A Quantitative Comparison. In *Proceedings of 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS).* IEEE Press, 194–204. DOI:http://dx.doi.org/10.1109/ISPASS.2013.6557170

Alistair Hart, Harvey Richardson, Jens Doleschal, Thomas Ilsche, Mario Bielert, and Matthew Kappel. 2014. User-level Power Monitoring and Application Performance on Cray XC30 Supercomputers. In *Proceedings of the 2014 CUG (Cray User Group) meeting.*

Michael Hennecke, Wolfgang Frings, Willi Homberg, Anke Zitz, Michael Knobloch, and Hans Böttiger. 2012. *Measuring Power Consumption on IBM Blue Gene/P.* Computer Science - Research and Development, Vol. 27(4). Springer-Verlag Press. DOI:http://dx.doi.org/10.1007/s00450-011-0192-y

Henry Hoffmann, Sasa Misailovic, Stelios Sidiroglou, Anant Agarwal, and Martin Rinard. 2009. Using Code Perforation to Improve Performance, Reduce Energy Consumption, and Respond to Failures. Computer Science and Artificial Intelligence Laboratory Technical Report, MIT-CSAIL-TR-2009-042. Computer Science Department, Massachusetts Institute of Technology (MIT), Cambridge, MA.

Henry Hoffmann, Stelios Sidiroglou, Michael Carbin, and Sasa Misailovic. 2011. Dynamic Knobs for Responsive Power-Aware Computing. In *Proceedings of the 16th international conference on architectural support for programming languages and operating systems (ASPLOS XVI).* ACM Press, New York, NY, 199–212. DOI:http://dx.doi.org/10.1145/1950365.1950390

Sunpyo Hong and Hyesoon Kim. 2010. An Integrated GPU Power and Performance Model. In *Proceedings of the 37th annual international symposium on computer architecture (ISCA'10).* ACM Press, New York, NY, 280–289. DOI:http://dx.doi.org/10.1145/1816038.1815998

Chung-Hsing Hsu and Wu-chun Feng. 2005. A Power-Aware Run-Time System for High-Performance Computing. In *Proceedings of the 2005 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'05).* IEEE Press, 1. DOI:http://dx.doi.org/10.1109/SC.2005.57

Chung-Hsing Hsu and Ulrich Kremer. 2003a. Dynamic Voltage and Frequency Scaling for Scientific Applications. Lecture Notes in Computer Science Volume Vol. 2624, 86–99. Springer-Verlag Press. DOI:http://dx.doi.org/10.1007/3-540-35767-X_6

Chung-Hsing Hsu and Ulrich Kremer. 2003b. The Design, Implementation, and Evaluation of a Compiler Algorithm for CPU Energy Reduction. In *Proceedings of the 24th ACM SIGPLAN conference on programming language design and implementation (PLDI'03).* ACM Press, New York, NY, 38–48. DOI:http://dx.doi.org/10.1145/780822.781137

S. Huang, S. Xiao, and W. Feng. 2009. On the Energy Efficiency of Graphics Processing Units for Scientific Computing. In *Proceedings of the 23rd international conference on Parallel and distributed processing (IPDPS'09).* IEEE Press, 1–8. DOI:http://dx.doi.org/10.1109/IPDPS.2009.5160980

IEEE 802.3az. 2010. Active/Idle Toggling with Low Power Idle.

Yuichi Inadomi, Tapasya Patki, Koji Inoue, Mutsumi Aoyagi, Barry Rountree, Martin Schulz, David Lowenthal, Yasutaka Wada, Keiichiro Fukazawa, Masatsugu Ueda, Masaaki Kondo, and Ikuo Miyoshi. 2015. Analyzing and Mitigating the Impact of Manufacturing Variability in Power-Constrained Supercomputing. In *Proceedings of the 2015 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'15).* ACM Press. DOI:http://dx.doi.org/10.1145/2807591.2807639

InfiniBand Trade Association. 2002. InfiniBand Architecture Specification, Release 1.2.

Intel Corp. 2011. System Programming Guide, volume 3B-2 of Intel 64 and IA-32 Architectures Software Developer's Manual.

Intel Corp. 2013. IPMI-Intelligent Platform Management Interface Specification Second Generation. V2.0

Y. Jiao, H. Lin, P. Balaji, and W. Feng. 2010. Power and Performance Characterization of Computational Kernels on the GPU. In *Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing (GREENCOM-CPSCOM'10).* IEEE Press, 221–228. DOI:http://dx.doi.org/10.1109/GreenCom-CPSCom.2010.143

Herbert Jordan, Peter Thoman, Juan J. Durillo, Simone Pellegrini, Philipp Gschwandtner, Thomas Fahringer, and Hans Moritsch. 2012. A Multi-Objective Auto-Tuning Framework for Parallel Codes. In *Proceedings of the 2012 International Conference for High Performance Computing, Networking,* *Storage and Analysis (SC'12)*. ACM Press, 1–12. DOI:http://dx.doi.org/10.1109/SC.2012.7

Krishna Kandalla, Emilio P. Mancini, Sayantan Sur, and Dhabaleswar K. Panda. 2010. Designing Power-Aware Collective Communication Algorithms for InfiniBand Clusters. In *Proceedings of the 39th International Conference on Parallel Processing (ICPP2010)*. IEEE Press, 218–227. DOI:http://dx.doi.org/10.1109/ICPP.2010.78

Nandini Kappiah, Vincent, W. Freeh, and David K. Lowenthal. 2005. Just in Time Dynamic Voltage Scaling: Exploiting Inter-Node Slack to Save Energy in MPI Programs. In *Proceedings of the 2005 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'05)*. ACM Press, 33. DOI:http://dx.doi.org/10.1109/SC.2005.57

Ulya R. Karpuzcu, Nam Sung Kim, and Josep Torrellas. 2013. Coping with Parametric Variation at Near-Threshold Voltages. IEEE Micro, Vol. 33(4). IEEE Press, 6–14. DOI:http://dx.doi.org/10.1109/MM.2013.71

Stefanos Kaxiras and Margaret Martonosi. 2008. Computer Architecture Techniques for Power-Efficiency. (1st. ed.). Morgan and Claypool Publishers.

Georgios Keramidas, Vasileios Spiliopoulos, and Stefanos Kaxiras. 2010. Interval-Based Models for Run-Time DVFS Orchestration in SuperScalar Processors. In *Proceedings of the 7th ACM international conference on Computing frontiers (CF'10)*. ACM Press, 287–296. DOI:http://dx.doi.org/10.1145/1787275.1787338

Peter Kogge, Keren Bergman, Shekhar Borkar, Dan Campbell, William Carlson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Kerry Hill, Jon Hiller, Sherman Karp, Stephen Keckler, Dean Klein, Robert Lucas, Mark Richards, Al Scarpelli, Steven Scott, Allan Snavely, Thomas Sterling, R. Stanley Williams, and Katherine Yelick. 2008. ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems. DARPA Report.

Vijay Anand Korthikanti and Gul Agha. 2010. Avoiding Energy Wastage in Parallel Applications. In *Proceedings of the 2010 International Green Computing Conference*. IEEE Press, 149–163. DOI:http://dx.doi.org/10.1109/GREENCOMP.2010.5598314

Michael O. Lam and, Jeffrey K. Hollingsworth, Bronis R. de Supinski, and Matthew P. LeGendre. 2013. Automatically Adapting Programs for Mixed-Precision Floating-Point Computation. In *Proceedings of the 36th International Conference for High Performance Computing, Networking, Storage and Analysis (SC'13)*. ACM Press, 369–378. DOI:http://dx.doi.org/10.1145/2464996.2465018

James H. Laros III, Kevin T. Pedretti, Suzanne M. Kelly, Wei Shu, Kurt Ferreira, John Van Dyke, and Courtenay T. Vaughan. 2013. *Energy-Efficient High Performance Computing - Measurement and Tuning*. SpringerBriefs in Computer Science. Springer Publications, ISBN 978-1-4471-4491-5

James H. Laros III, Kevin T. Pedretti, Suzanne M. Kelly, Wei Shu, and Courtenay T. Vaughan. 2012. Energy Based Performance Tuning for Large Scale High Performance Computing Systems. In *Proceedings of the 2012 Symposium on High Performance Computing (HPC'12)*. Society for Computer Simulation International Press.

James H Laros, Pavel Pokorny, and David DeBonis. 2013. PowerInsight - A Commodity Power Measurement Capability. In *Proceedings of 2013 International Green Computing Conference (IGCC)*.

Barry Lawson and Evgenia Smirni. 2005. Power-Aware Resource Allocation in High-End Systems via Online Simulation. In *Proceedings of the 19th annual international conference on Supercomputing (ICS '05)*. ACM Press. DOI:http://dx.doi.org/10.1145/1088149.1088179

Edgar A. Leon, Ian Karlin, and Ryan E. Grant. 2015. Optimizing Explicit Hydrodynamics for Power, Energy, and Performance. In *Proceedings of 2015 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE Press, 11–21. DOI:http://dx.doi.org/10.1109/CLUSTER.2015.12

Dong Li, Surendra Byna, and Srimat Chakradhar. 2011a. Energy-Aware Workload Consolidation on GPU. In *Proceedings of the 40th International Conference Parallel Processing Workshops (ICPPW'11)*. IEEE Press, 389–398. DOI:http://dx.doi.org/10.1109/ICPPW.2011.25

Bo Li, Hung-Ching Chang, Shuaiwen Leon Song, Chun-Yi Su, Timmy Meyer, John Mooring, and Kirk Cameron. 2014. The Power-Performance Tradeoffs of the Intel Xeon Phi on HPC Applications. In *Proceedings of the 2014 IEEE International Parallel & Distributed Processing Symposium Workshops (IPDPSW'14)*. IEEE Press, 1448–1456. DOI:http://dx.doi.org/10.1109/IPDPSW.2014.162

Sheng Li, Kevin Lim, Paolo Faraboschi, Jichuan Chang, Parthasarathy Ranganathan, and Norman P. Jouppi. 2011b. System-Level Integrated Server Architectures for Scale-Out Datacenters. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-44)*. ACM Press, 260–271. DOI:http://dx.doi.org/10.1145/2155620.2155651

Jian Li and Jose F. Martinez. 2006. Dynamic Power-Performance Adaptation of Parallel Computation on Chip Multiprocessors. In *Proceedings of the 12th International Symposium on High-Performance Computer Architecture (HPCA'06)*. IEEE Press, 77–87. DOI:http://dx.doi.org/10.1109/HPCA.2006.1598114

Dong Li, Dimitrios S. Nikolopoulos, Kirk W. Cameron, Bronis R. de Supinski, and Martin Schulz. 2010a. Power-Aware MPI Task Aggregation Prediction for High-End Computing Systems. In *Proceedings of the 2010 IEEE International Parallel and Distributed Processing Symposium (IPDPS'10)*. IEEE Press, 1–12. DOI:http://dx.doi.org/10.1109/IPDPS.2010.5470463

Dong Li, Bronis R. de Supinski, Martin Schulz, Kirk Cameron, and Dimitrios S. Nikolopoulos. 2010b. Hybrid MPI/OpenMP Power-Aware Computing. In *Proceedings of the 2010 IEEE International Parallel and Distributed Processing Symposium (IPDPS'10).* IEEE Press, 1–12. DOI:http://dx.doi.org/10.1109/IPDPS.2010.5470463

Min Yeol Lim, Vincent W. Freeh, and David K. Lowenthal. 2006. Adaptive, Transparent Frequency and Voltage Scaling of Communication Phases in MPI Programs. In *Proceedings of the 2006 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'06).* ACM Press, 14. DOI:http://dx.doi.org/10.1109/SC.2006.11

Michael D. Linderman, Matthew Ho, David L. Dill, Teresa H. Meng, and Garry P. Nolan. 2010. Towards Program Optimization through Automated Analysis of Numerical Precision. In *Proceedings of the 8th annual IEEE/ACM international symposium on Code generation and optimization (*CGO '10*).* ACM Press, New York, NY, 230–237. DOI:http://dx.doi.org/10.1145/1772954.1772987

Jiuxing Liu, Dan Poff, and Bulent Abali. 2009. Evaluating High Performance Communication: A Power Perspective. In *Proceedings of the 23rd international conference on supercomputing (ICS '09).* ACM Press, New York, NY, 326–337. DOI:http://dx.doi.org/10.1145/1542275.1542322

Chi-Keung Luk, Sunpyo Hong, and Hyesoon Kim. 2009. Qilin: Exploiting Parallelism on Heterogeneous Multiprocessors with Adaptive Mapping. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 42).* ACM Press, New York, NY. DOI:http://dx.doi.org/10.1145/1669112.1669121

Kai Ma, Xue Li, Wei Chen, Chi Zhang, and Xiaorui Wang. 2012. GreenGPU: A Holistic Approach to Energy Efficiency in GPU-CPU Heterogeneous Architectures. In *Proceedings of the 41st International Conference on Parallel Processing (ICPP'12).* IEEE Press, 48–57. DOI:http://dx.doi.org/10.1109/ICPP.2012.31

Olli Mämmelä, Mikko Majanen, Robert Basmadjian, Hermann De Meer, André Giesler, and Willi Homberg. 2012. Energy-Aware Job Scheduler for High-Performance Computing. Computer Science - Research and Development, Vol. 27(4). Springer-Verlag Press, 265–275. DOI:http://dx.doi.org/10.1007/s00450-011-0189-6

Aniruddha Marathe, Peter E. Bailey, David K. Lowenthal, Barry Rountree, Martin Schulz, Bronis R. de Supinski. 2015. A Run-Time System for Power-Constrained HPC Applications. High Performance Computing of Lecture Notes in Computer Science, Vol. 9137. Springer International Publishing, 394–408. DOI:http://dx.doi.org/10.1007/978-3-319-20119-1_28

Steven J. Martin, David Rush, and Matthew Kappel. 2015. Cray Advanced Platform Monitoring and Control (CAPMC). In *Proceedings of the 2015 CUG (Cray User Group) meeting.*

Xinxin Mei, Ling Sing Yung, Kaiyong Zhao, and Xiaowen Chu. 2013. A Measurement Study of GPU DVFS on Energy Conservation. In *Proceedings of the 2013 Workshop on Power-Aware Computing and Systems (HotPower'13).* ACM Press, New York, NY. DOI:http://dx.doi.org/10.1145/2525526.2525847

Renato Miceli, Gilles Civario, Anna Sikora, Eduardo C´esar, Michael Gerndt, Houssam Haitof, Carmen Navarrete, Siegfried Benkner, Martin Sandrieser, Laurent Morin, and Fran¸cois Bodin. 2012. AutoTune: A Plugin-Driven Approach to the Automatic Tuning of Parallel Applications. In Proceedings of the 11th international conference on Applied Parallel and Scientific Computing (PARA'12), 328–342. Springer-Verlag, Berlin, Heidelberg. DOI:http://dx.doi.org/10.1007/978-3-642-36803-5_24

D.A.B. Miller and H.M. Ozaktas. 1997. Limit to the Bit-Rate Capacity of Electrical Interconnects from the Aspect Ratio of the System Architecture. Journal of Parallel and Distributed Computing – Special issue on parallel computing with optical interconnects, Vol. 41(1). Springer-Verlag Berlin Press, 42–52. DOI:http://dx.doi.org/10.1006/jpdc.1996.1285

Timo Minartz, Thomas Ludwig, Michael Knobloch, and Bernd Mohr. 2011. Managing Hardware Power Saving Modes for High Performance Computing. In *Proceedings of 2011 International on Green Computing Conference and Workshops (IGCC'11).* IEEE Press, 1–8. DOI:http://dx.doi.org/10.1109/IGCC.2011.6008581

Shinobu Miwa, Sho Aita, and Hiroshi Nakamura. 2014. Performance Estimation of High Performance Computing Systems with Energy Efficient Ethernet Technology. Computer Science-Research and Development, Vol. 29(3). Springer Verlag Press, 161–169. DOI:http://dx.doi.org/10.1145/2464996.2465009.

Shinobu Miwa and Hiroshi Nakamura. 2015. Profile-Based Power Shifting in Interconnection Networks with On/Off Links. In *Proceedings of the 2015 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'15).* ACM Press. DOI:http://dx.doi.org/10.1145/2807591.2807639

MPI Forum. 2012. MPI: A Message-Passing Interface Standard. Version 3.0. www.mpi-forum.org

Lev Mukhanov, Dimitrios S. Nikolopoulos, and Bronis R. de Supinski. 2015. ALEA: Fine-grain Energy Profiling with Basic Block Sampling. In *Proceedings of the 24th International Conference on Parallel Architectures and Compilation Techniques (PACT-2015).* ACM Press.

Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. 2008.

Reducing Network Energy Consumption via Sleeping and Rate-Adaptation. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI'08).* USENIX Association Press, Berkeley, CA. 323–336.

OpenMP ARB. 2013. OpenMP Application Program Interface, v. 4.0.

Tapysa Patki, David K. Lowenthal, Barry Rountree, Martin Schulz and Bronis R. de Supinski. 2013. Exploring Hardware Overprovisioning in Power-Constrained, High Performance Computing. In *Proceedings of the* 27th International Conference on Supercomputing (ICS 2013). ACM Press, 173–182. DOI:http://dx.doi.org/10.1145/2464996.2465009.

Tapasya Patki, Anjana Sasidharan, Matthias Maiterth, David K. Lowenthal, Barry Rountree, Martin Schulz and Bronis R. de Supinski. 2015. Practical Resource Management in Power-Constrained, High Performance Computing, In *Proceedings of the* 24th *IEEE International Symposium on High Performance Distributed Computing (HPDC 2015).* ACM Press, 121–132. DOI:http://dx.doi.org/10.1145/2749246.2749262.

David Patterson, Dennis Gannon, and Michael Wrinn. 2013a. The Berkeley Par Lab: Progress in the Parallel Computing Landscape. 2013. Microsoft Corporation Press.

Michael K Patterson, Stephen W Poole, Chung-Hsing Hsu, Don Maxwell, William Tschudi, Henry Coles, David J Martinez, and Natalie Bates. 2013b. TUE, a New Energy-Efficiency Metric Applied at ORNL's Jaguar. Supercomputing Lecture Notes in Computer Science Vol. 7905. Springer Berlin Heidelberg Press, 372–382. DOI:http://dx.doi.org/10.1007/978-3-642-38750-0_28

Kevin Pedretti, Stephen L. Olivier, Kurt B. Ferreira, Galen Shipman, and Wei Shu. 2015. Early Experiences with Node-Level Power Capping on the Cray XC40 Platform. In *Proceedings of the 3rd International Workshop on Energy Efficient Supercomputing.* ACM Press. DOI:http://dx.doi.org/10.1145/2834800.2834801

D. C. Price, M. A. Clark, B. R. Barsdell, R. Babich, and L. J. Greenhill. 2015. Optimizing Performance-per-Watt on GPUs in High Performance Computing. *Computer Science - Research and Development,* Special issue paper. Springer-Verlag Press, Berlin Heidelberg, 1–9. DOI:http://dx.doi.org/10.1007/s00450-015-0300-5

Shah Mohammad Faizur Rahman, Jichi Guo, Akshatha Bhat, Carlos Garcia, Majedul Haque Sujon, Qing Yi, Chunhua Liao, and Daniel Quinlan. 2012. Studying the Impact of Application-Level Optimizations on the Power Consumption of Multi-Core Architectures. In *Proceedings of the 9th conference on Computing Frontiers(CF'12).* ACM Press, 123–132. DOI:http://dx.doi.org/10.1145/2212908.2212927

Shah Faizur Rahman, Jichi Guo, and Qing Yi. 2011. Automated Empirical Tuning of Scientific Codes for Performance and Power Consumption. In *Proceedings of Proceedings of the 6th International Conference on High Performance and Embedded Architectures and Compilers (HiPEAC'11).* ACM Press, 107–116. DOI:http://dx.doi.org/10.1145/1944862.1944880

Nikola Rajovic, Pall Carpenter, Isaac Gelado, Nikola Puzovic, and Alex Ramirez. 2013. Supercomputing with Commodity CPUs: Are Mobile SoCs Ready for HPC? In *Proceedings of the 2013 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'13).* IEEE Press, 1–12. DOI:http://dx.doi.org/10.1145/2503210.2503281

Barry Rountree, Dong H. Ahn, Bronis R. de Supinski, David K. Lowenthal, and Martin Schulz. 2012. Beyond DVFS: A First Look at Performance Under a Hardware-Enforced Power Bound. In *Proceedings of 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW'12).* IEEE Press, 947–953. DOI:http://dx.doi.org/10.1109/IPDPSW.2012.116

Barry Rountree, David K. Lowenthal, S. Funk, Vincent W. Freeh, Bronis R. de Supinski, and Martin Schulz. 2007. Bounding Energy Consumption in Large-Scale MPI Programs. In *Proceedings of the 2007 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'07).* IEEE Press, 1–9. DOI:http://dx.doi.org/10.1145/1362622.1362688

Barry Rountree, David K. Lowenthal, Martin Schulz, and Bronis R. de Supinski. 2011. Practical Performance Prediction Under Dynamic Voltage Frequency Scaling. In *Proceedings of the 2nd International Green Computing Conference (IGCC11).* IEEE Press, 1–8. DOI:http://dx.doi.org/10.1109/IGCC.2011.6008553

Barry Rountree, David K. Lowenthal, Bronis R. de Supinski, Martin Schulz, Vincent W. Freeh, and Tyler Bletsch. 2009. Adagio: Making DVS Practical for Complex HPC Applications. In *Proceedings of the 23rd International Conference on Supercomputing (ICS'09).* ACM Press, 460–469. DOI:http://dx.doi.org/10.1145/1542275.1542340

Cindy Rubio-Gonz´alez, Cuong Nguyen, Hong Diep Nguyen, James Demmel, William Kahan, Koushik Sen, David H. Bailey, Costin Iancu, and David Hough. 2013. Precimonious: Tuning Assistant for Floating-Point Precision. In *Proceedings of the 2013 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'13).* IEEE Press. DOI:http://dx.doi.org/ 10.1145/2510000/2503296

Adrian Sampson, Werner Dietl, Emily Fortuna, Danushen Gnanapragasam, Luis Ceze, and Dan Grossman. 2011. EnerJ: Approximate Data Types for Safe and General Low-Power Computation. In *Proceedings of the 32nd ACM SIGPLAN conference on programming language design and implementation (PLDI'11).* ACM Press, New York, NY, 164–174.

DOI:http://dx.doi.org/10.1145/1993316.1993518

H. Saputra, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, J. S. Hu, C-H. Hsu, and U. Kremer. 2002. Energy-Conscious Compilation Based on Voltage Scaling. In *Proceedings of the joint conference on languages, compilers and tools for embedded systems: software and compilers for embedded systems (*LCTES/SCOPES'02*)*. ACM Press, New York, NY, 2–11. DOI:http://dx.doi.org/10.1145/1816038.1815998

Karthikeyan P. Saravanan, Paul M. Carpenter, and Alex Ramirez. 2013. Power/Performance Evaluation of Energy Efficient Ethernet (EEE) for High Performance Computing. In *Proceedings of 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS).* IEEE Press, 205–214. DOI:http://dx.doi.org/10.1109/ISPASS.2013.6557171

Karthikeyan P. Saravanan, Paul M. Carpenter, and Alex Ramirez. 2014. A Performance Perspective on Energy Efficient HPC Links. In *Proceedings of the 28th International Conference on Supercomputing (ICS'14).* ACM Press, 313–322. DOI:http://dx.doi.org/10.1145/2597652.2597671

Osman Sarood, Akhil Langer, Abhishek Gupta and Laxmikant Kale. 2014. Maximizing Throughput of Overprovisioned HPC Data Centers Under a Strict Power Budget. In *Proceedings of the 2014 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'14).* ACM Press, 807–818. DOI:http://dx.doi.org/10.1109/SC.2014.71

Osman Sarood, Akhil Langer, Laxmikant Kale, Barry Rountree, and Bronis de Supinski. 2013. Optimizing Power Allocation to CPU and Memory Subsystems in Overprovisioned HPC Systems. In *Proceedings of 2013 IEEE International Conference on Cluster Computing (CLUSTER).* IEEE Press, 1–8. DOI:http://dx.doi.org/10.1109/CLUSTER.2013.6702684

Eric Schkufza, Rahul Sharma, and Alex Aiken. 2014. Stochastic Optimization of Floating-Point Programs with Tunable Precision. In *Proceedings of the 35th ACM SIGPLAN conference on programming language design and implementation (PLDI'14).* ACM Press, New York, NY, 53–64. DOI:http://dx.doi.org/10.1145/2594291.2594302

Thomas Scogland, Jonathan Azose, David Rohr, Suzanne Rivoire, Natalie Bates, and Daniel Hackenberg. 2015. Node Variability in Large-Scale Power Measurements: Perspectives from the Green500, Top500 and EEHPCWG. In *Proceedings of the 2015 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'15).* ACM Press. DOI:http://dx.doi.org/10.1145/2807591.2807658

Thomas Scogland, Craig Steffen, Torsten Wilde, Florent Parent, Susan Coghlan, Natalie Bates, Wu-chun Feng, and Erich Strohmaier. 2014. A Power-Measurement Methodology for Large-Scale, High-Performance Computing. In *Proceedings of the 5th ACM/SPEC international conference on Performance engineering (ICPE '14).* ACM Press, 149–159. DOI:http://dx.doi.org/10.1145/2568088.2576795

John Shalf, Sudip Dosanjh, and John Morrison. 2010. Exascale Computing Technology Challenges. In *Proceedings of the 9th international conference on High performance computing for computational science (VECPAR'10).* Springer-Verlag Berlin Press, Heidelberg, 1–25.

Edgar Solomonik and James Demmel. 2011. Communication-Optimal Parallel 2.5 D Matrix Multiplication LU Factorization Algorithms. In *Proceedings of the 18th European Conference on Parallel Processing (Euro-Par 2011).* Springer International Publishing, 90–109. DOI:http://dx.doi.org/10.1007/978-3-642-23397-5_10

Shuaiwen Song, Matthew Grove, and Kirk W. Cameron. 2011. An Iso-Energy-Efficient Approach to Scalable System Power-Performance Optimization. In *Proceedings of 2011 IEEE International Conference on Cluster Computing (CLUSTER).* IEEE Press, 262–271. DOI:http://dx.doi.org/10.1109/CLUSTER.2011.37

Reiji Suda and Da Qi Ren. 2009. Accurate Measurements and Precise Modeling of Power Dissipation of CUDA Kernels toward Power Optimized High Performance CPU-GPU Computing. In *Proceedings of 2009 International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'08).* IEEE Press, 432–438. DOI:http://dx.doi.org/10.1109/PDCAT.2009.65

M. Aater Suleman, Moinuddin K. Qureshi, and Yale N. Patt. 2008. Feedback-Driven Threading: Power-Efficient and High-Performance Execution of Multi-threaded Workloads on CMPs. In *Proceedings of the 13th international conference on architectural support for programming languages and operating systems (ASPLOS XIII).* ACM Press, New York, NY, 277–286. DOI:http://dx.doi.org/10.1145/1346281.1346317

Cristian Tapus, I-Hsin Chung, and Jeffrey K. Hollingsworth. 2002. Active Harmony: Towards Automated Performance Tuning. In *Proceedings of the 2005 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'02).* ACM Press, 1–11.

The Green 500. 2015. http://www.green500.org/lists/green201511

Ananta Tiwari, Michael A. Laurenzano, Laura Carrington, and Allan Snavely. 2012. Auto-Tuning for Energy Usage in Scientific Applications. *Euro-Par 2011: Parallel Processing Workshops*. Lecture Notes in Computer Science Vol. 7156, 2012. Springer International Publishing, 178–187. DOI:http://dx.doi.org/10.1007/978-3-642-29740-3_21

Top500. 2015. http://www.top500.org/lists/2015/11/

Karthikeyan Vaidyanathan, Sasikanth Avancha, and Sunil SherlekarOn. 2013. Exascale Computing & Beyond: Meeting the Challenges. In *Transition of HPC Towards Exascale Computing*, E.H. D'Hollander et. al (Eds.). IOS Press, 24-34. DOI:http://dx.doi.org/10.3233/978-1-61499-324-7-24

Akshay Venkatesh, Krishna Kandalla, and Dhabaleswar K. Panda. 2013. Evaluation of Energy Characteristics of MPI Communication Primitives with RAPL. In *Proceedings of the 27th IEEE International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW).* IEEE Press, 938–945. DOI:http://dx.doi.org/10.1109/IPDPSW.2013.243

Akshay Venkatesh, Abhinav Vishnu, Khaled Hamidouche, Nathan Tallent, Dhabaleswar Panda, Darren Kerbyson, and Adolfy Hoisie. 2015. A Case for Application-Oblivious Energy-Efficient MPI Runtime. In *Proceedings of the 2015 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'15).* ACM Press. DOI:http://dx.doi.org/10.1145/2807591.2807658

Guibin Wang and Xiaoguang Ren. 2010. Power-Efficient Work Distribution Method for CPU-GPU Heterogeneous System. In *Proceedings of 2010 International Symposium on Parallel and Distributed Processing with Applications (ISPA'10).* IEEE Press, 386–393. DOI:http://dx.doi.org/10.1109/ISPA.2010.22

Mark Weiser, Brent Welch, Alan Demers, and Scott Shenker. 1994. Scheduling for Reduced CPU Energy. In *Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation (OSDI '94).* ACM Press.

Jason Williams, Chris Massie, Alan D. George, Justin Richardson, Kunal Gosrani, and Herman Lam. 2010. Characterization of Fixed and Reconfigurable Multi-Core Devices for Application Acceleration. ACM Transactions on Reconfigurable Technology and Systems (TRETS), vol.(3)-4. ACM Press. DOI:http://dx.doi.org/10.1145/1862648.1862649

Michael Wolfe. 1996. High Performance Compilers for Parallel Computing. Addison-Wesley Publishing Company.

Dong Hyuk Woo and Hsien-Hsin S. Lee. 2008. Extending Amdahl's Law for Energy-Efficient Computing in the Many-Core Era. Computer, Vol. 41(12). IEEE Press, 24–31. DOI:http://dx.doi.org/10.1109/MC.2008.494

Kazutomo Yoshii, Kamil Iskra, Rinku Gupta, Pete Beckman, Venkatram Vishwanath, Chenjie Yu, and Susan Coghlan. 2012. Evaluating Power Monitoring Capabilities on IBM Blue Gene/P and Blue Gene/Q. In *Proceedings of 2012 IEEE International Conference on Cluster Computing (CLUSTER).* IEEE Press, 36–44. DOI:http://dx.doi.org/10.1109/CLUSTER.2012.62

Zhou Zhou, Zhiling Lan, Wei Tang, and Narayan Desai. 2014. Reducing Energy Costs for IBM Blue Gene/P Power-Aware Job Scheduling. Job Scheduling Strategies for Parallel Processing of Lecture Notes in Computer Science, Vol. 8429. Springer Berlin Heidelberg Press, 96–115. DOI:http://dx.doi.org/10.1007/978-3-662-43779-7_6.