EPPMiner: An Extended Benchmark Suite for Energy, Power and Performance Characterization of Heterogeneous Architecture

Qiang Wang
Department of Computer Science
Hong Kong Baptist University, Hong Kong
qiangwang@comp.hkbu.edu.hk

Pengfei Xu
Department of Computer Science
Hong Kong Baptist University, Hong Kong
pengfeixu@comp.hkbu.edu.hk

Yatao Zhang
Department of Computer Science
Hong Kong Baptist University, Hong Kong
12250686@life.hkbu.edu.hk

Xiaowen Chu
Department of Computer Science
Hong Kong Baptist University, Hong Kong
HKBU Institute of Research and Continuing Education, Shenzhen, China
chxw@comp.hkbu.edu.hk

ABSTRACT

To address the ever-increasing demand for computing capacities, more and more heterogeneous systems have been designed to use both general-purpose and special-purpose processors. On the other hand, the huge energy consumption of these heterogeneous systems raises new environmental concerns and challenges. Besides performance, energy efficiency is now another key factor to be considered by system designers and also consumers. In this paper, we present a benchmark suite EPPMiner for evaluating the performance, power, and energy of different heterogeneous systems. EPPMiner consists of 16 benchmark programs that cover a broad range of application domains, and it shows a great variety in the intensity of utilizing the processors. We have implemented a prototype of EPPMiner that supports OpenMP, CUDA, and OpenCL, and demonstrated its usage by three showcases. Firstly, we use EPPMiner to compare the power efficiency of a set of processors, including two Intel x86 CPUs, two Nvidia GPUs, and one AMD GPU. Secondly, we investigate the impact of multi-threading on the power efficiency of multi-core CPUs. At last, we use EPPMiner to illustrate the effectiveness of GPU Dynamic Voltage and Frequency Scaling (DVFS) on the power efficiency of GPGPU applications. We show that DVFS can improve the energy efficiency by 86% over the default setting on an AMD GPU.

CCS CONCEPTS

• **Hardware → Power and energy**;

KEYWORDS

Benchmarks, Performance Evaluation, Graphics Processing Units, Dynamic Voltage and Frequency Scaling

e-Energy '17, May 16-19, 2017, Shatin, Hong Kong

ACM ISBN 978-1-4503-5036-5/17/05…$15.00
https://doi.org/http://dx.doi.org/10.1145/3077839.3077858

ACM Reference format:
Qiang Wang, Pengfei Xu, Yatao Zhang, and Xiaowen Chu. 2017. EPPMiner: An Extended Benchmark Suite for Energy, Power and Performance Characterization of Heterogeneous Architecture. In *Proceedings of e-Energy '17, Shatin, Hong Kong, May 16-19, 2017,* 11 pages.
https://doi.org/http://dx.doi.org/10.1145/3077839.3077858

1 INTRODUCTION

The increasingly extensive use of high-performance computing (HPC) in applications including big data analysis, bioinformatics, and artificial intelligence driven by deep learning has demonstrated its necessity over the past 40 years [28]. HPC has made remarkable progress in parallel hardware, parallel algorithms, and parallel and distributed programming techniques, primarily in attempts to break the energy wall and memory wall. However, over the past 10 years, technical challenges to overcoming the power wall in aspects including power consumption, energy conservation, and environmental protection have gradually emerged because of the exponential growth of data and the more complex operations that are performed.

Currently, conventional central processing units (CPUs) can barely provide satisfactory performance per watt for the aforementioned applications. Instead, accelerators, such as general purpose graphics processing units (GPGPUs) and Intelą́s many-core Xeon Phi architecture, have gained considerable traction because of not only their higher peak performance but also their energy efficiency. Among the top 10 on the November 2016 TOP500 list of the most powerful commercially available supercomputers, two are equipped with GPGPUs and three are equipped with the Xeon Phi. Among the Green500 list of the most energy-efficient supercomputers, for the same period, the top 10 are all heterogenous systems.

Some studies have explored the energy, power, and performance of these accelerators and revealed they have significant potential energy conservation technologies, such as dynamic voltage and frequency scaling (DVFS) techniques, power- and energy-aware task mapping, and dynamic core assignment. However, few studies have horizontally compared these processors. Moreover, most previously developed widely known benchmark suites focus on computing throughput ability and memory bottlenecks, which may

pertain to the functionality of the processor, but few studies have characterized power and energy.

In this paper, we present a new benchmark suite named EPPMiner[1] for evaluating the energy, power, and performance of heterogeneous systems. Specifically, EPPMiner defines a set of 16 applications that cover a broad range of domains and *dwarves* in parallel computing [2]. We have implemented a prototype of EPPMiner with the following features: First, the benchmark suite can be executed on various types of processors or accelerators, including Intel multi-core and many-core CPUs, which are mainly installed in desktop computers and servers; ARM-based CPUs, which are widely used in mobile equipment and embedded systems; and mainstream GPUs. It can also be executed on Linux and Windows systems. Our current prototype of EPPMiner supports OpenMP, CUDA, and OpenCL, and hence it can support many potential processors/coprocessors. Second, to address the diversity of memory sizes, it includes a set of normal workload for PCs and servers with enough memory, and another set of light workload for devices with less memory. Third, it can be used to more fairly test performance, in particular, for the kernel part of each application on each target processor or accelerator. Each program will automatically repeat the kernel many times, and the time information of each iteration will be recorded. Fourth, it should produce stable power consumption data for the target hardware when running the kernel parts of each application. As many power measurement methods have a low time resolution (e.g., one sample per second), the execution time of each program should be long enough. EPPMiner allows the user to predefine the running time of a program to assure accurate power measurement. Finally, EPPMiner records the processor temperature information when it is possible, considering that temperature has a significant impact on power consumption.

Our study makes several contributions and yields valuable findings, which are listed as follows:

(1) We present a new benchmark suite, EPPMiner, for characterizing the energy, power, and performance, of a wide range of processors and accelerators.
(2) We use EPPMiner to evaluate a set of processors and accelerators, including an ARM CPU, two Intel CPUs, two Nvidia GPUs, and one AMD GPU. We show that although GPUs have a higher power requirement than CPUs, they have a huge advantage over CPUs in energy, performance, and energy-efficiency.
(3) We investigate the impact of multi-threading on the energy-efficiency of multi-core CPUs. We find that when all the CPU cores are being utilized by multi-threading, the energy-efficiency can be improved in general. However, the advantage will diminish when the number of threads is more than that of the physical cores.
(4) We illustrate the effectiveness of DVFS technique on improving the energy-efficiency of GPGPU applications. We first show that frequency scaling alone on GPU core and memory can lead to 55% of difference in terms of energy efficiency on an Nvidia GTX980. We then show that GPU core voltage and frequency scaling can improve the energy efficiency by 86% over the default setting on an AMD RX480.

The rest of this paper is organized as follows. Section 2 demonstrates the motivation of designing a benchmark suite for heterogeneous systems. Section 3 describes our benchmark suite MPPMiner and the performance metrics. We also discuss how to conduct power and performance measurements on different heterogeneous platforms. Section 4 introduces our experimental platforms and then presents three showcases of EPPMiner. Section 5 discusses some related work on benchmarking and energy-efficient computing. Finally we conclude the paper in Section 6.

2 MOTIVATION

2.1 Development of energy efficient processors

To attain higher peak performance with lower power consumption, different types of energy-efficient accelerators have increasingly been adopted in data centers, PCs, and mobile equipment. GPUs and Intel Xeon Phi are the most representative among them. For example, the latest NVIDIA GTX1080 can provide up to 9 Tflops with a thermal design power (TDP) of only 180 W [25], whereas the AMD RX480 achieves up to 5.8 Tflops with a TDP of 150 W. Intel Xeon Phi 7290 achieves around 3.5 Tflops in double precision with a TDP of 245 W. By contrast, small computers (like mobile phones) equipped with ARM-based CPUs are now more important because of sufficient computational capability with low power consumption. For example, the latest Raspberry Pi 3B+ model features 1.2 GHz frequency and can achieve up to 2 GFlops with no more than 4 Watts. Efforts to overcome the power wall are obviously a design objective of all of the aforementioned productsą́ designers.

2.2 Benchmark suites for power and energy characterizations

Some studies have explored not only the performance but also the power and energy efficiency of GPUs [7, 12, 15, 23] and Intel Xeon Phi cards [16, 18, 29]. However, they have generally considered only one type of device and ran different benchmark programs to reach their findings. Comparing a variety of processors and accelerators would be informative. Finding a sufficiently fair scheme for comparing the performance, power, and energy across different types of computational architectures is difficult. Some widely known benchmark suites, such as Rodinia [5], Parboil [31], SHOC [8], NUPAR [32], are competently designed for measuring the performance of general devices, such as Intel CPUs, NVIDIA, and AMD GPUs, and can even be applied to a whole machine or cluster. However, they do not characterize the power and energy of those devices. To remedy these problems, we develop an extended set of benchmark suites based on them.

2.3 Multi-objectives Task Mapping Design

Developing such benchmark suites is valuable because they can provide sample programs for task mapping strategy testing on a heterogeneous system. Performance is currently not the only factor that should be considered in designing task scheduling strategies. Numerous research papers describe the implementation of power-

[1]The source code and experimental data of EPPMiner can be downloaded from http://eppminer.comp.hkbu.edu.hk.

and energy-aware or even multi-objective task scheduling systems [6, 14, 17, 20, 24]. We believe that an effective benchmark suite that can be used for multi-characteristic explorations could serve as a common platform for performance evaluation and hence help fellow researchers obtain more solid results.

3 THE EPPMINER BENCHMARK

3.1 Description of selected applications and the workload

To select an appropriate set of applications for the benchmark, we need to consider the representability, workload variety, and also development and maintenance cost. As an initial attempt, we follow the concept of dwarves proposed by Asanovic et al. [2] and select a total of 16 applications that cover a broad ranges of domains and dwarves. The summary of the benchmark applications is shown in Table 1.

(1) Breadth-First Search (bfs): traverse the shortest path from a single vertex to each other in a graph of formatted edge matrix.
(2) Distance-Cutoff Coulombic Potential (cutcp): Computes the short-range component of Coulombic potential at each grid point over a 3D grid containing point charges representing an explicit-water biomolecular model.
(3) Histogram (histo): Computes a moderately large, 2-D saturating histogram with a maximum bin count of 255. Input datasets represent a silicon wafer validation application in which the input points are distributed in a roughly 2-D Gaussian pattern.
(4) Dense Matrix-Matrix Multiplication (sgemm): One of the most widely and intensely studied benchmarks, this application performs a dense matrix multiplication using the standard BLAS format.
(5) Sparse-Matrix Dense-Vector Multiplication (spmv): Computes the product of a sparse matrix with a dense vector. The sparse matrix is read from file in coordinate format, converted to JDS format with configurable padding and alignment for different devices.
(6) 3-D Stencil Operation (stencil): An iterative Jacobi stencil operation on a regular 3-D grid.
(7) Magnetic Resonance Imaging - Q (mri-q): Computes a matrix Q, representing the scanner configuration for calibration, used in a 3D magnetic resonance image reconstruction algorithms in non-Cartesian space.
(8) Back propagation (bp): a machine-learning algorithm that trains the weights of connecting nodes on a layered neural network.
(9) HotSpot (hotspot): a widely used tool to estimate processor temperature based on an architectural floorplan and simulated power measurements.
(10) Shortest Path (pathfinder): uses dynamic programming to find a path on a 2-D grid from the bottom row to the top row with the smallest accumulated weights, where each step of the path moves straight ahead or diagonally ahead.
(11) LU Decomposition (lud): an algorithm to calculate the solutions of a set of linear equations.
(12) Needleman-Wunsch (nw): a nonlinear global optimization method for DNA sequence alignments.
(13) K-Means (kmeans): a clustering algorithm used extensively in data-mining and elsewhere, important primarily for its simplicity.
(14) Speckle Reducing Anisotropic Diffusion (srad): a diffusion method for ultrasonic and radar imaging applications based on partial differential equations.
(15) Nearest Neighbor (nn): finds the k-nearest neighbors from an unstructured data set.
(16) CFD Solver (cfd): an unstructured grid finite volume solver for the three-dimensional Euler equations for compressible flow.

Table 1: Summary of benchmark applications

Applications	Dwarves	Domain
bfs	Graph Traversal	Graph Algorithms
cutcp	Unstructured Grid	Medical Imaging
histo	Combinational Logic	Image Processing
sgemm	Dense Linear Algebra	Linear Algebra
spmv	Sparse Linear Algebra	Linear Algebra
stencil	Structured Grids	Fluid Dynamics
mri-q	Dense Linear Algebra	Medical Imaging
bp	Unstructured Grid	Pattern Recognition
hotspot	Structured Grid	Physics Simulation
pathfinder	Dynamic Programming	Grid Traversal
lud	Dense Linear Algebra	Linear Algebra
nw	Dynamic Programming	Bioinformatics
kmeans	Dense Linear Algebra	Data Mining
srad	Structured Grid	Image Processing
nn	N-body	Data Mining
cfd	Unstructured Grid	Fluid Dynamics

Our prototype of EPPMiner currently supports three parallel programming techniques: OpenMP, CUDA, and OpenCL, and hence it can already support many processors and accelerators such as Nvidia GPUs, AMD GPUs, and Intel Xeon Phi. Since many applications have a very short execution time, we allow the user to set the running time of a benchmark program in the configuration file, and then each program will automatically call its computational kernel iteratively and calculate the average kernel execution time and power consumption. There are two advantages to this design. First, collecting the average time consumptions of the kernel parts launched on the targeted device is fairer and more convenient. Second, the programs can run for a long time, which enables stable and sufficient power sampling.

We also design two sets of workloads. The normal workload is designed for PCs and servers with enough memory, whereas the light workload is designed for devices (such as mobile phones) whose memory size is limited. The details of the workload can be found in 2.

3.2 Design of performance metrics

EPPMiner aims to compare different systems in terms of performance and energy consumption. For a given target platform, EPPMiner will report three performance metrics: mean execution time T, mean energy consumption E, and mean energy efficiency F in terms of operations per second per watt (OPS/W).

Table 2: Input Data Configurations for tested devices

Applications	Light workload	Normal workload
bfs	SF	SF
cutcp	small	large
histo	large	large
sgemm	medium	medium
spmv	large	large
stencil	small	default
mri-q	small	large
bp	65536	65536
hotspot	1024 1024 100 4	1024 1024 100 4
pathfinder	100000 100	100000 100
lud	2400	8000
nw	2048 10 2	2048 10 2
kmeans	kdd_cup	kdd_cup
srad	2048 2048 0 127 0 127	2048 2048 0 127 0 127
nn	10000	131072
cfd	097K	193K

The mean execution time T is calculated as the geometric mean of the set of execution times of each benchmark application:

$$T = \sqrt[n]{\prod_{i=1}^{n} T_i} \tag{1}$$

where n is the number of benchmark programs and T_i is the average kernel execution time of the i-th program.

The mean energy consumption E is calculated as the geometric mean of the set of energy consumptions of each benchmark application:

$$E = \sqrt[n]{\prod_{i=1}^{n} P_i T_i} \tag{2}$$

where P_i is the average power consumption.

We use operations per second per watt (OPS/W) to describe the energy efficiency of running the benchmark programs on each hardware platform. First, we count the total number of operations, including integer operation, floating-point operation, and double-precision floating-point operation, of each benchmark program by software profilers. Each double-precision floating-point operation contributes 2 to the total number of operations. Then we define the energy efficiency F as:

$$F = \sqrt[n]{\prod_{i=1}^{n} O_i/(P_i T_i)} \tag{3}$$

where O_i is the total number of operations of the i-th application.

3.3 Practical issues: Performance and power measurements

Here, we focus on the performance and energy efficiency of tested processors and accelerators rather than the entire host machine. To address performance, we refer to the active kernel execution time, which includes only the part running on the target hardware. We repeat the kernel code running on the tested accelerators for at least 5 minutes and then average the total active kernel execution time to obtain the performance data.

For power measurements, the users need to determine whether they are evaluating the whole system or the processor. For many contemporary processors and accelerators, we can rely on the internal sensors and the corresponding software interface to extract the power information. E.g., for the Intel x86 CPUs, we use power_gov [13], which is a software utility that allows us to monitor CPU power with fine time granularities. For Nvidia GPUs, we use nvidia-smi [27], which is a command line utility, based on top of the Nvidia Management Library (NVML) [26], intended to aid in managing and monitoring Nvidia GPU devices. Typically we can measure the power of the aforementioned devices per 500 milliseconds. For AMD GPUs, we use CodeXLPowerProfiler [1], which is also a command line tool playing the similar role as nvidia-smi but aiming to AMD CPUs and GPUs. We measure the power of our AMD GPUs per 100 milliseconds.

For systems without internal sensors such as Raspberry Pi 3B+ model, we can measure the whole system power consumption by a commercial power meter, such as Watts Up? Pro which takes a power sample every second. The meter has an independent power supply; consequently, it does not considerably affect our system power measurements.

4 SHOWCASES

To illustrate the usage of EPPMiner under different scenarios, we present three showcases in this section.

4.1 Experimental testbed

We test our benchmark suite in terms of energy, power, and performance on a total of six processors and accelerators, as shown in Table 3. For the ARM Cortex-A53 CPU, the Raspberry Pi 3B+ is the host computer and has 1 GB of LPDDR2 memory and uses the Rasbian Linux operating system. For the Intel CPU i7-3820 and three GPUs, we use the same desktop computer with 64 GB of memory. The AMD RX480 uses a new version of the Ubuntu operating system because the latest device driver and OpenCL tools are available only on this version. We use the Leveno System x3650 M5 rack server as the host machine, which has two Intel Xeon E5-2630v3 CPUs; this machine provides 128 GB of memory.

4.1.1 Multicore x86-based CPUs. We test two CPUs that are representative of two usage scenarios, desktop and server. The Intel i7-3820 CPU is typically installed in desktop PCs and is equipped with four cores, each of which supports two hyperthreads. This CPU can provide 30.74 GFLOPS at a frequency of 3.6 GHz with a TDP of 130 W. It features 64 GB of memory support and a 10 MB cache. The other is the Intel E5-2630v3 CPU with 6 physical cores and is representative of the server scenario. It features a 2.8 GHz turbo frequency.

4.1.2 ARM-based CPUs. Raspberry is a widely known and inexpensive single-board computer equipped with an ARM-compatible CPU that can operate on a low power supply. We test the newest model, the Raspberry Pi 3, which was released in 2016. With four cores on an ARM Cortex-A53 CPU running at 1.2 GHz, the Pi 3 can achieve up to 2 GFLOPS on no more than 4 W. It features 1 GB of low-power double data rate 2 (LPDDR2) memory.

4.1.3 General Purpose Graphics Processing Units. We test three GPU platforms: Nvidia's GTX980 and GTX1080, and AMD's RX480. The Nvidia GTX980 is the first full realization of the Maxwell GPU architecture. With 2048 compute unified device architecture (CUDA) cores and 1126 MHz base core clock, the GTX980 can provide 4.27 TFLOPS with a TDP of only 165 W. As a graphics card used in

desktop computers, it has 4 GB of memory with 3500 MHz base memory clock, which provides a bandwidth of up to 224 GB/s. The Nvidia GTX1080 is the most complete implementation of the latest GPU architecture Pascal. Designed for high-performance programming and having a TDP that is only 15 W higher than that of the GTX980, the GTX1080 provides 8.23 TFLOPS with 2560 processor cores and a base core frequency of 1607 MHz. With a 5000 MHz memory clock, the GTX1080 achieves a bandwidth of up to 320 GB/s, which represents an increase of nearly 43%.

The RX480, which is equipped with the fourth generation of the Graphics Core Next (GCN) architecture and is also named Polaris, is the most advanced GPU produced by AMD in 2016. With 2304 stream processors and a base clock frequency of 1340 MHz (with OC version of our experimental card), the RX480 provides 6.17 TFLOPS with a TDP of 150 W.

Table 3: The host configurations for each device.

Year	Device	Cores	Memory	OS
2012	Intel CPU i7-3820	4	64 GB	Ubuntu 14.04
2014	Intel CPU E5-2630 v3	12	128 GB	CentOS 7.2
2014	Nvidia GTX 980	2048	4 GB	Ubuntu 14.04
2016	ARM Cortex-A53 CPU	4	1 GB	Rasbian
2016	Nvidia GTX 1080	2560	8 GB	Ubuntu 14.04
2016	AMD RX 480	2304	8 GB	Ubuntu 16.04

4.2 Showcase I: Comparison of different devices

We first use EPPMiner to evaluate the energy, power, and performance of different processors and accelerators. Figure 1 illustrates the power distributions of programs running on different devices with different thread settings. The term **omp** refers to the number of threads with OpenMP implementation. The Raspberry Pi requires only 1.05~1.75 W with **omp** = 1 on all the programs and 1.5~2.55W with **omp** = 4. The Intel i7-3820 CPU's power consumption exhibits almost linear increments when the number of threads changes from one to eight but remains almost the same when it changes from 8 to 16. That is probably because it has four physical cores and supports eight hyperthreads. A similar phenomenon occurs in Intel E5-2630v3 CPUs equipped with a total of 12 physical cores. Three tested GPUs exhibit a wider range of higher power varying from 30~180 W. However, given that GPUs have hundreds of times the number of computation cores compared with CPUs, GPUs could still be much more energy efficient than two x86 CPUs.

One goal of our benchmark suites is to enable comparisons among different types of processors and accelerators in terms of energy, power, and performance. Figures 2, 3 and 4 provide a horizontal comparison of all the devices tested in this study. In this showcase, we present the performance, energy consumption, and energy efficiency of CPUs using a single core only.

We illustrate the performance for each program on all the tested devices with normal workload in Figure 2. GPUs, especially the GTX1080, achieve the optimal performance for all the benchmark programs except *srad*. We show the energy consumptions of all

Figure 1: Runtime power distributions of all the benchmark programs on different devices

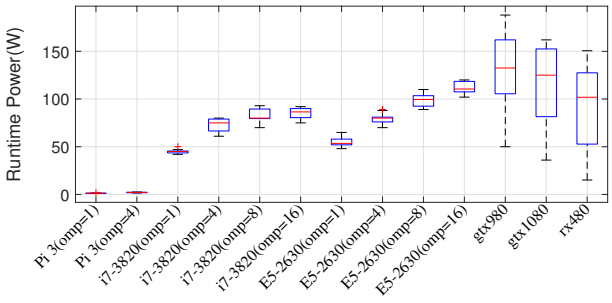


Table 4: The energy efficiency, performance, and energy of different devices

Device	GOPS/W	Time(s)	Energy(J)
Raspberry Pi 3B+	0.06554	2.6433	1.738
Intel CPU i7-3820	0.0462	0.1466	6.562
Nvidia GTX 980	1.758	0.00137	0.1726
Nvidia GTX 1080	2.639	0.00106	0.115
AMD RX 480	1.1456	0.00358	0.265
Intel CPU E5-2630 v3	0.0386	0.1436	7.852

the benchmark programs on each device in Figure 3. Generally GPUs can save more energy compared to CPUs since they can achieve several magnitude of accelerations but with no more than three times of power. In particular, for some applications belonging to Dense Linear Algebra Dwarf including *sgemm*, $mri-q$ and *lud*, GPUs consume only 1% of CPU energy. However, for some irregular applications like $pathfinder$, CPUs and GPUs have nearly the same energy consumption level since GPUs can hardly take full utilization of its cores when maintaining the same power level. The energy efficiencies of all benchmark programs are shown in Figure 4.

We show the overall performance and energy efficiency with our designed metrics of each device by computing mean execution time T, E and F in Table 4. Notice that the results of all CPUs are obtained by using a single CPU core. The results of using multiple cores will be shown in the next showcase. Basically GPUs show nearly 40 times of GOPS/W to those of CPUs. In our experiments, GTX 1080 is the winner in terms of both performance and energy efficiency. We observe that the mean time of GPUs have hundred times of speed up compared to CPUs, which makes GPUs outperform other devices a lot. However, since we do not apply AVX optimization to the CPU implementation of our benchmark programs, there still exists a considerable space to improve the CPUs' energy efficiency.

4.3 Showcase II: Impact of multi-threading on performance/power/energy

Multi-threading programming collaborated with multi-core processing commonly decreases time consumption if the application

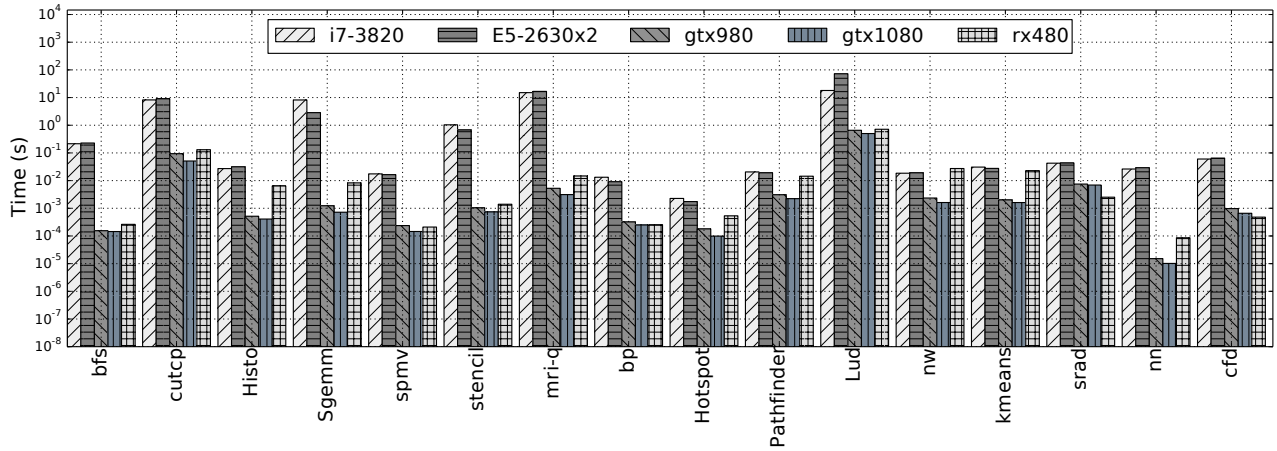


Figure 2: Horizontal comparison of performance

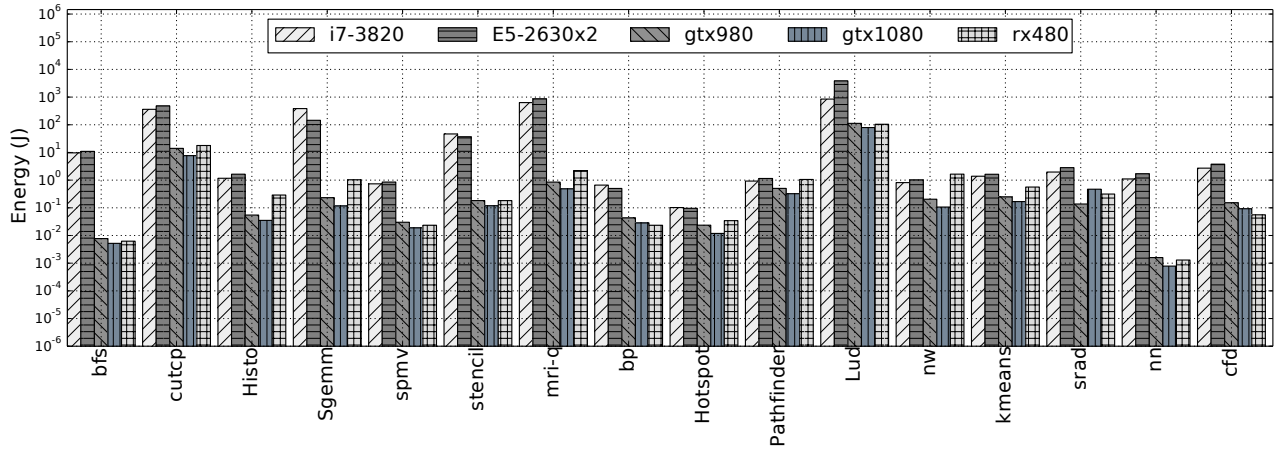


Figure 3: Horizontal comparison of energy consumption

exhibits high parallelism. Furthermore, we analyze the effects of parallelism on energy, power, and performance with different thread settings of each CPU we test.

Figure 5 depicts the effects of both one-thread and four-thread settings on the energy and performance of the Pi 3B+ model. Most programs exhibit speedup of at least three times with the four-thread setting, except for bfs and *histo* because bfs has only small segments of code available for parallelism, whereas *histo* has an excessive number of critical operations when updating the histogram. However, as Figure 5(b) shows, the power level of all the programs can increase by only about 2 times when the thread number changes from one to four. Thus, to apply four-thread setting can conserve up to 50% of energy consumptions for most benchmark applications, which can be inferred from Figure 5(c). Table 5 lists the mean GOPS/W of those two multi-threading settings. Applying four threads further helps exploit 50% more OPS/W compared to just one thread, which also meets our previous analysis.

Because Intel x86 CPUs have more physical cores and a higher frequency than the Pi model, we adopt more thread settings in those experiments. Figure 6 and 7 show the results for the i7-3820 and E5-2630v3, respectively. For the Intel i7-3820 CPU equipped with four cores, the time consumption of each program achieves a minimum with either four threads or eight threads. When the thread number increases to 16, performance decreases. This could be explained by the fact that the i7-3820 has four physical cores and supports eight hyperthreads. By contrast, the power level increases approximately linearly when the thread number changes from one to eight but remains almost the same when the thread number increases from 8 to 16, as illustrated in Figure 6(b). Analogous characteristics are found for the E5-2630v3. Because it has 12 physical cores and supports two hyperthreads in each core, the 16-thread setting usually achieves optimal performance in our benchmark suites while it has also the highest power level. Similar to i7-3820 CPU, E5-2630v3 achieves best energy efficiency when the thread number is closer to its physical number. Table 5 also suggests the fact that applying the same number of threads as that of CPU physical cores helps achieve the best OPS/W.

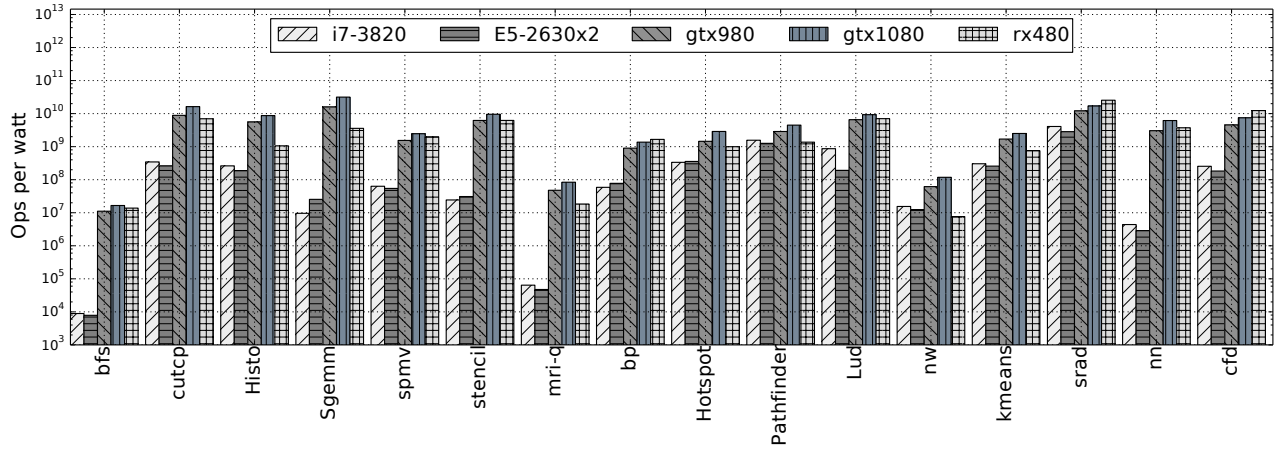
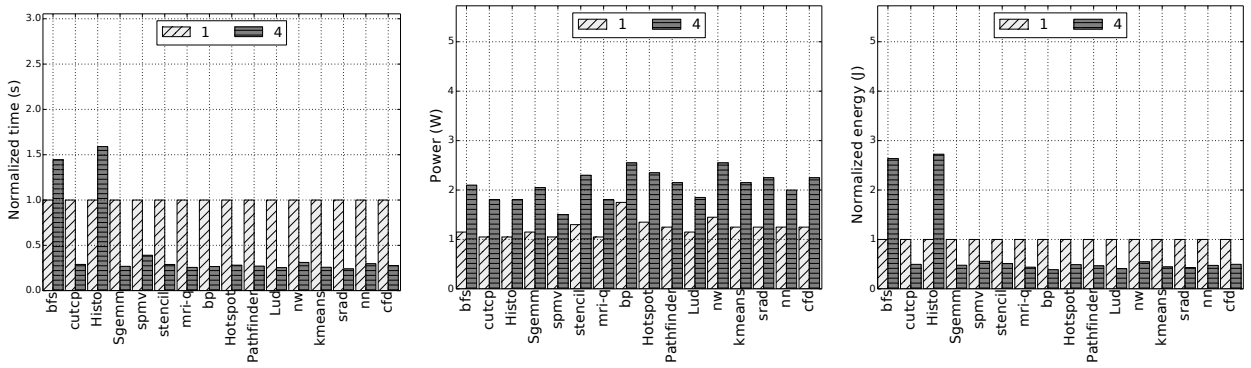


Figure 4: Horizontal comparison of energy efficiency

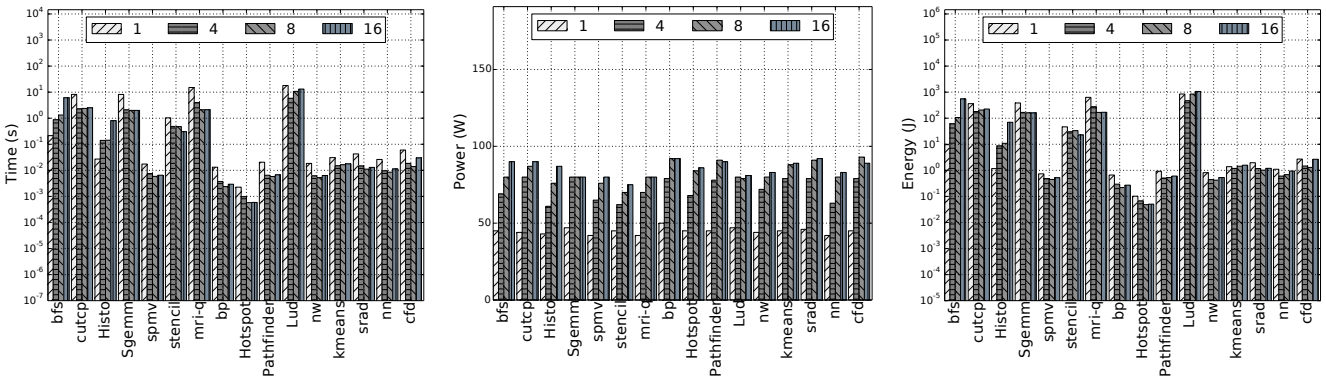


(a) Performance of Pi 3B+ model (normalized with omp=1)

(b) Active Power of Pi 3B+ model

(c) Energy of Pi 3B+ model(normalized with omp=1)

Figure 5: Energy and performance characterization of Pi 3B+ model (normalized with omp=1)

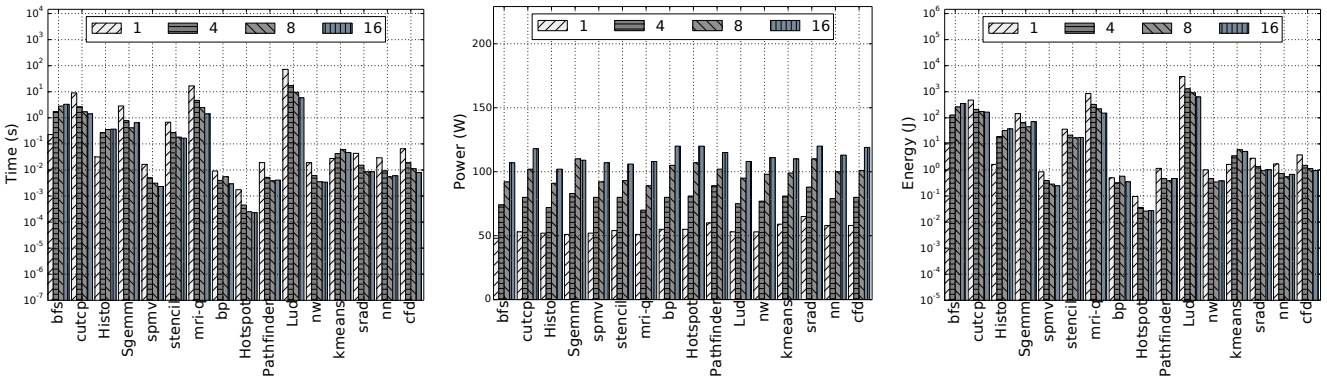


(a) Performance of i7-3820

(b) Active Power of i7-3820

(c) Energy of i7-3820

Figure 6: Energy and performance characterization of Intel i7-3820 CPU



(a) Performance of E5-2630 v3 (b) Active Power of E5-2630 v3 (c) Energy of E5-2630 v3

Figure 7: Energy and performance characterization of Intel E5-2630v3x2 CPUs

Table 5: The energy efficiency of different processors with different number of threads.

Processors	#threads	GOPS/W	Time(s)	Energy(J)
Pi 3B+	1	0.06554	2.643	3.233
	4	0.098	0.915	1.895
i7-3820	1	0.0462	0.1466	6.562
	4	0.0606	0.069	5.006
	8	0.0589	0.062	5.15
	16	0.0417	0.085	7.27
E5-2630 v3	1	0.0386	0.144	7.852
	4	0.0525	0.073	5.782
	8	0.0567	0.054	5.348
	16	0.0573	0.047	5.295

4.4 Showcase III: Impact of DVFS on energy efficiency

Finally, we explore the impact of DVFS on the energy efficiency of GPGPU applications. In this part of experiments, we take Nvidia GTX980 and AMD RX480 as our testbeds. The default voltage and frequency setting for GTX980 is 1130 mV core voltage, 1126 MHz core frequency and 3500 MHz memory frequency, while RX480 has 1150 mV core voltage, 1340 MHz core frequency and 4000 MHz memory frequency. Due to their different DVFS capability, we use GTX980 to investigate the impact of frequency scaling, and RX480 for dynamic voltage and frequency scaling. Specifically, we fix the core voltage of GTX980 to 0.85 V and adjust both the core and memory frequency from 400 MHz to 1000 MHz with a stepsize of 300 MHz, which in total produces nine frequency settings. For RX480, we are able to adjust its voltage from 1000 mV to 1150 mV and scale the core frequency to the highest stable one. We plot the normalized ops per watt of each benchmark application under those DVFS settings in Figure 8 and Figure 9 for GTX980 and RX480 respectively.

First, we explore how core and memory frequency scaling affects the energy efficiency of GTX980. As Figure 8 illustrates, each legend item represents the core and memory frequency setting (e.g., 400MHz/700MHz indicates that the core frequency is 400 MHz and the memory frequency is 700 MHz). The OPS/W values are normalized with the value at the 1000MHz/1000MHz setting. The results show that the optimal frequency setting varies from application to application. Generally, the highest frequency in our experiments is not necessarily the optimal choice for achieving the optimal energy efficiency. Lower core and memory frequencies can save substantial energy for applications that do not fully utilize GPU core resources, such as bfs. For some applications that mainly execute memory transactions, such as $k-means$, raising the memory frequency and lowering the core frequency help improve electricity utilization. However, some notable cases reveal irregular energy characterization. Some applications, such as *bp*, *histo*, *cutcp*, achieve optimal energy efficiency under immediate frequency settings. This may be because when we increase the frequency to some extent, the performance cannot be significantly improved, but power consumption may be larger.

As for RX480, we show the experimental results in Figure 9 where different core voltages are being compared. In general RX480 can achieve better energy efficiency when reducing the core voltage and scaling down the core frequency. We can infer the possible reasons from Table 7. Notice that the average time consumption has no obvious change when increasing core voltage and core frequency. On contrast, the applications digest more energy since the power with those aggressive settings is higher. Thus, scaling down the core voltage and frequency helps promote the energy efficiency when sacrificing very little performance. To summarize, our benchmark suite can help explore various scaling behavior of different voltage and frequency settings of GPUs among different types of applications.

Table 6 and 7 summarize the energy efficiency under different voltage and frequency settings of GTX980 and RX480 respectively. Compared to 1.758 OPS/W of GTX980, scaling down the core and memory frequency helps increase up to 55% OPS/W. Notice that we lower down the core voltage from 1130 mV to 850 mV and core frequency from 1126 MHz to 1000 MHz under 1000MHz/1000MHz setting, which helps conserve power remarkably. As for RX480, the lowest voltage and core frequency in our experimental settings

outperform nearly 86% OPS/W than that of default setting, which indicates great potentials of RX480 on energy conservation with dynamic voltage and frequency scaling techniques.

Table 6: The energy efficiency of GTX980 with frequency scaling.

F_{core}(MHz)	F_{mem}(MHz)	GOPS/W	Time(ms)	Energy(J)
400	400	2.087	3.895	0.145
400	700	2.445	3.07	0.124
400	1000	2.574	2.813	0.118
700	400	2.273	3.326	0.134
700	700	2.818	2.457	0.108
700	1000	3.105	2.116	0.098
1000	400	2.215	2.977	0.137
1000	700	2.795	2.185	0.109
1000	1000	3.078	1.865	0.099

Table 7: The energy efficiency of RX480 with frequency scaling.

V_{core}(mV)	F_{core}(MHz)	GOPS/W	Time(ms)	Energy(J)
1000	1160	2.131	2.322	0.1424
1040	1266	2.003	2.242	0.1515
1080	1300	1.892	2.201	0.1604
1120	1340	1.726	2.206	0.1759
1150	1390	1.531	2.404	0.1983

5 RELATED WORK

Benchmarking has been playing a key role during the evolution of computing technologies by allowing comparison among different architecture designs and/or system implementations. LINPACK benchmarks are designed to compare the execution rate of floating-point operations by solving a dense system of linear equations, which is popular in engineering applications and scientific computing [9]. Its parallel extension High Performance Linpack (HPL) has also been used to generate the TOP500 list twice per year [11]. As LINPACK and HPL focus on solving dense linear systems which is very computing extensive, they may not reflect the performance of many other real applications. To this end, High Performance Conjugate Gradient (HPCG) benchmark has been recently proposed to include the memory subsystem and interconnect of the supercomputers into consideration [10]. All these three benchmarks use a single number to represent the performance of a tested system.

In the industry, System Performance Evaluation Cooperative (SPEC) has been developing a set of benchmarks to evaluate different computer systems. E.g., SPEC CPU2006 benchmark suite includes a set of integer benchmarks and a set of floating-point benchmarks to compare the performance of different processors. The execution time of each benchmark program is first normalized with a reference processor, and then the geometric mean of the normalized execution time is reported as the single performance metric. SPECpower_ssj2008 is designed to evaluate the power and performance of server computers. It records the power consumption as well as performance (in terms of workload operations per second) of the tested server at different workloads, ranging from 0% to 100% with a stepsize of 10%, and then reports a single performance metric named overall ssj_ops per watt, which is calculated as dividing the total workload operations over the total power consumption [30].

With the ever-growing popularity of hardware accelerators such as GPUs, MICs, FPGAs, new benchmarks have been designed to investigate the performance and power efficiency of such heterogeneous systems. Parboil is a set of benchmark applications which support multiple types of processors and different programming languages to stimulate high throughput computation [31]. Rodinia benchmark suite is also developed for evaluating the performance of heterogeneous computing [5].

Jared et al. explored energy, power and performance characterizations of 34 selected GPGPU benchmark programs by varying the GPU frequency and input data size [7]. They revealed some relationships between those factors and energy efficiency. Joao et al. illustrated that GPU DVFS can affect energy consumption of different types of GPU applications [12]. Mishra did a literature survey and provide thorough analysis of various schemes on DVFS techniques during last decade and concluded that DVFS can work together with other energy conservation technique like load balancing and task mapping where various hardware platform and applications can have significant impact on performance and power efficiency [23]. Mei et al. also investigated the impact of GPU DVFS on energy saving by conducting real experiments on a set of benchmark applications [21, 22]. They also developed a set of microbenchmark programs to dissect the GPU memory hierarchy [19]. Bridges also launched an investigation about GPU power and energy characterizations of the previous work and concluded that performance counters have shown great statistical correlations with GPU power and energy characterizations [3]. Furthermore, modeling and simulation work based on data mining on counters indicates the potential for optimization in programming and even future hardware design. Burtscher pointed out some irregular behaviors when measuring power with the on-board power sensors of the GPUs and proposed a methodology which can precisely compute the power and energy with the sampling sensor data [4].

As for Intel Xeon Phi cards, such many-core systems have been hailed as an important step towards greater energy efficiency. Different applications will suffer from different performance degradation when the voltage and frequency of the processor have changed. Lorenzo [18] studied the power and energy usage of a series of benchmarks on the Intel Xeon Phi for different threads settings. Shao [29] proposed some models to improve energy efficiency at core or instruction level. However, few codes are developed to meet the scalability of thread number on the Xeon Phi. If scalability is limited, using all the resources on the Xeon Phi seems to be not rational for energy conservation. Bo [16] presented a detailed study of the performance-energy tradeoffs of the Xeon Phi architecture and supported the view that limited scalability might introduce worse energy efficiency in some applications on the Xeon Phi.

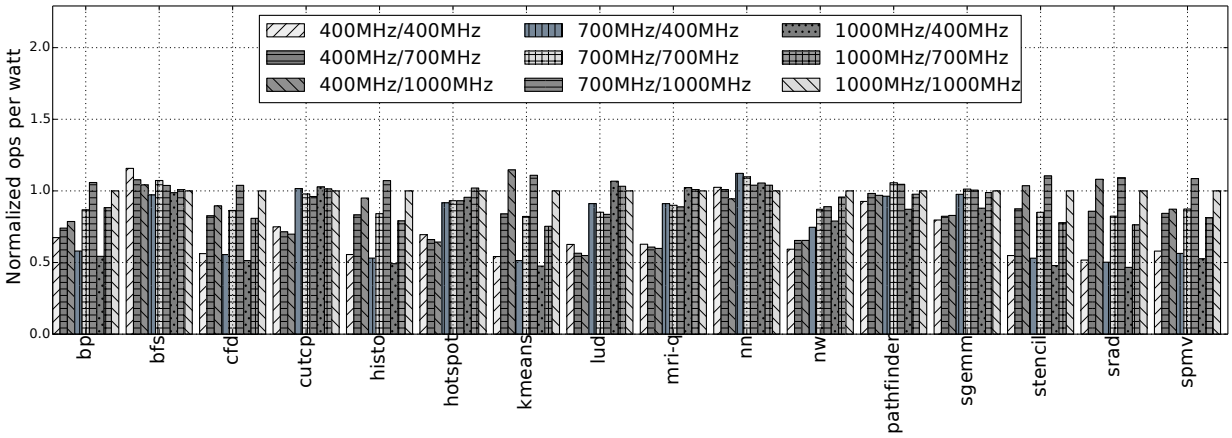


Figure 8: Impact of Nvidia GTX 980 core/memory frequency scaling on ops per watt of each benchmark program(normalized with 1000MHz/1000MHz)

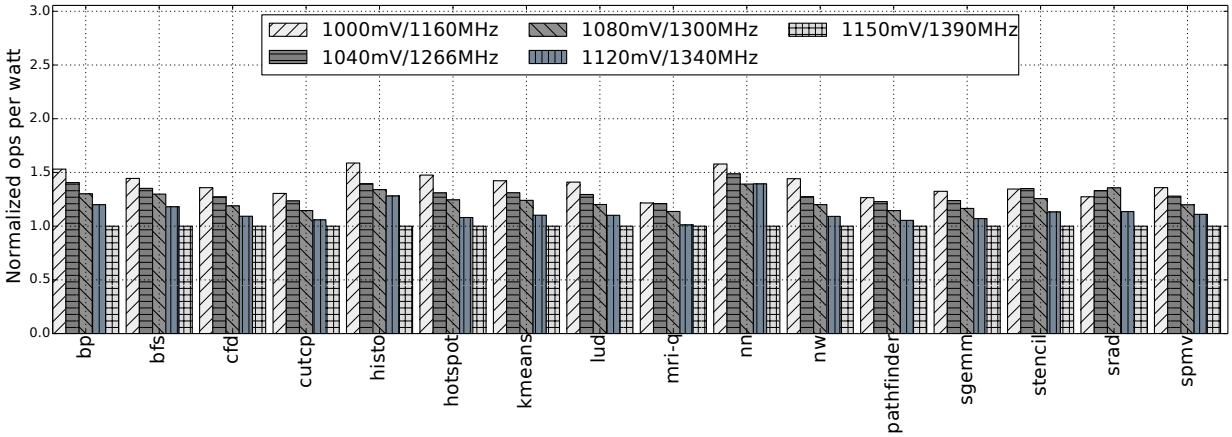


Figure 9: Impact of AMD RX480 dynamic voltage/core frequency scaling on ops per watt of each benchmark program(normalized with 1150mV/1390MHz)

6 CONCLUSIONS

In this paper, we have introduced a new benchmark suite EPPMiner for evaluating and comparing the energy, power, and performance of various heterogeneous systems, which includes a set of 16 programs that support OpenMP, CUDA, and OpenCL. It is designed to help researchers conduct energy-related studies by recoding the detailed time, power, and temperature information when possible. To illustrate the effectiveness of EPPMiner, we have also presented three showcases that cover a broad range of processors and accelerators. In the first showcase, we have compared a set of CPUs and GPUs and shown that GPUs have a huge advantage over CPUs in energy, performance, and energy-efficiency, despite their high power consumption. In the second showcase, we have investigated the impact of multi-threading on the energy efficiency of multi-core CPUs. We have found that when all the CPU cores are being utilized by multi-threading, the energy-efficiency can be improved in general. In the last showcase, we have illustrated the effectiveness of DVFS technique on improving the energy efficiency of GPGPU applications. We have shown that DVFS can improve the energy efficiency by 86% over the default setting on an AMD RX480.

The current prototype of EPPMiner can be further improved in the following direction. First, we plan to optimize the CPU implementation by using Intel AVX instructions. Second, we plan to port EPPMiner to Android system so that mobile equipment can also be tested. Third, the current workload is designed for a single machine. We plan to include the support of MPI such that EPPMiner can also be used by computer clusters.

ACKNOWLEDGMENTS

The authors would like to thank all the reviewers for their insightful comments and valuable suggestions. This work is partially supported by research grant HKBU FRG2/14-15/059 and Shenzhen Basic Research Grant SCI-2015-SZTIC-002.

REFERENCES

[1] AMD. 2016. AMD CodeXL (CodeXL). [Online] http://gpuopen.com/compute-product/codexl/. (2016).

[2] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, and others. 2006. *The landscape of parallel computing research: A view from berkeley*. Technical Report. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley.

[3] Robert A Bridges, Neena Imam, and Tiffany M Mintz. 2016. Understanding GPU Power: A Survey of Profiling, Modeling, and Simulation Methods. *ACM Computing Surveys (CSUR)* 49, 3 (2016), 41.

[4] Martin Burtscher, Ivan Zecena, and Ziliang Zong. 2014. Measuring GPU power with the K20 built-in sensor. In *Proceedings of Workshop on General Purpose Processing Using GPUs*. ACM, 28.

[5] Shuai Che, Michael Boyer, Jiayuan Meng, David Tarjan, Jeremy W Sheaffer, Sang-Ha Lee, and Kevin Skadron. 2009. Rodinia: A benchmark suite for heterogeneous computing. In *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*. IEEE, 44–54.

[6] Vincent Chow, Xiaowen Chu, Hai Liu, and Yiu-Wing Leung. 2017. Energy Efficient Job Scheduling with $DVFS$ for $CPU-GPU$ Heterogeneous Systems. In *e-Energy*. ACM.

[7] Jared Coplin and Martin Burtscher. 2016. *Energy, Power, and Performance Characterization of GPGPU Benchmark Programs*. Technical Report. Technical Report TXSTATE-CS-ECL-2016-1.

[8] Anthony Danalis, Gabriel Marin, Collin McCurdy, Jeremy S Meredith, Philip C Roth, Kyle Spafford, Vinod Tipparaju, and Jeffrey S Vetter. 2010. The scalable heterogeneous computing (SHOC) benchmark suite. In *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*. ACM, 63–74.

[9] Jack Dongarra. 1988. The LINPACK benchmark: An explanation. In *Proceedings of the 1st International Conference on Supercomputing*. 456âĂŞ–474.

[10] Jack Dongarra and Michael A Heroux. 2013. Toward a New Metric for Ranking High Performance Computing Systems. *SANDIA Report* (2013).

[11] Jack Dongarra, Piotr Luszczek, and Antoine Petitet. 2003. The LINPACK Benchmark: past, present and future. *Concurrency and Computation: Practice and Experience* 15, 9 (2003), 803–820.

[12] Joao Guerreiro, Aleksandar Ilic, Nuno Roma, and Pedro Tomás. 2016. Performance and Power-Aware Classification for Frequency Scaling of GPGPU Applications. (2016).

[13] Intel. 2012. Intel Power Governor. [Online] https://software.intel.com/en-us/articles/intel-power-governor. (2012).

[14] Canturk Isci, Alper Buyuktosunoglu, Chen-Yong Cher, Pradip Bose, and Margaret Martonosi. 2006. An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. In *Proceedings of the 39th annual IEEE/ACM international symposium on microarchitecture*. IEEE Computer Society, 347–358.

[15] Michael LeBeane, Jee Ho Ryoo, Reena Panda, and Lizy Kurian John. 2015. Watt Watcher: Fine-Grained Power Estimation for Emerging Workloads. In *Computer Architecture and High Performance Computing (SBAC-PAD), 2015 27th International Symposium on*. IEEE, 106–113.

[16] Bo Li, Hung-Ching Chang, Shuaiwen Song, Chun-Yi Su, Timmy Meyer, John Mooring, and Kirk W Cameron. 2014. The power-performance tradeoffs of the Intel Xeon Phi on HPC applications. In *Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*. IEEE, 1448–1456.

[17] K. Li. 2016. Energy-Efficient Task Scheduling on Multiple Heterogeneous Computers: Algorithms, Analysis, and Performance Evaluation. *IEEE Transactions on Sustainable Computing* 1, 1 (Jan 2016), 7–19. DOI: http://dx.doi.org/10.1109/TSUSC.2016.2623775

[18] Oscar G Lorenzo, Tomás F Pena, José Carlos Cabaleiro Domínguez, Juan Carlos Pichel Campos, Francisco Fernández Rivera, and Dimitrios S Nikolopoulos. 2015. Power and energy implications of the number of threads used on the Intel Xeon Phi. *Annals of Multicore and GPU Programming: AMGP* 2, 1 (2015), 55–65.

[19] Xinxin Mei and Xiaowen Chu. 2017. Dissecting GPU Memory Hierarchy Through Microbenchmarking. *IEEE Transactions on Parallel and Distributed Systems* 28, 1 (2017), 72–86.

[20] Xinxin Mei, Xiaowen Chu, Hai Liu, Yiu-Wing Leung, and Zongpeng Li. 2017. Energy Efficient Real-time Task Scheduling on $CPU-GPU$ Hybrid Clusters. In *INFOCOM*. IEEE.

[21] Xinxin Mei, Qiang Wang, and Xiaowen Chu. 2017. A Survey and Measurement Study of GPU DVFS on Energy Conservation. *Digital Communications and Networks* 3, 2 (2017), 89–100.

[22] Xinxin Mei, Ling Sing Yung, Kaiyong Zhao, and Xiaowen Chu. 2013. A measurement study of GPU DVFS on energy conservation. In *HotPower@SOSP*. ACM, 10:1–10:5.

[23] Ashish Mishra and Nilay Khare. 2015. Analysis of DVFS Techniques for Improving the GPU Energy Efficiency. *Open Journal of Energy Efficiency* 4, 04 (2015), 77.

[24] Sparsh Mittal and Jeffrey S Vetter. 2015. A survey of cpu-gpu heterogeneous computing techniques. *ACM Computing Surveys (CSUR)* 47, 4 (2015), 69.

[25] NVIDIA. 2016. GeForce GTX 1080 Whitepaper. (2016).

[26] NVIDIA. 2016. NVIDIA Management Library . [Online] https://developer.nvidia.com/nvidia-management-library-nvml. (2016).

[27] NVIDIA. 2016. NVIDIA System Management Interface (nvidia-smi). [Online] http://developer.download.nvidia.com/compute/DCGM/docs/nvidia-smi-367.38.pdf. (2016).

[28] Depei Qian. 2016. High performance computing: a brief review and prospects. *National Science Review* 3, 1 (2016), 16–16.

[29] Yakun Sophia Shao and David Brooks. 2013. Energy characterization and instruction-level energy model of Intel's Xeon Phi processor. In *Proceedings of the 2013 International Symposium on Low Power Electronics and Design*. IEEE Press, 389–394.

[30] SPEC. 2012. Power and Performance Benchmark Methodology V2.1. (2012).

[31] John A Stratton, Christopher Rodrigues, I-Jui Sung, Nady Obeid, Li-Wen Chang, Nasser Anssari, Geng Daniel Liu, and Wen-mei W Hwu. 2012. Parboil: A revised benchmark suite for scientific and commercial throughput computing. *Center for Reliable and High-Performance Computing* 127 (2012).

[32] Yash Ukidave, Fanny Nina Paravecino, Leiming Yu, Charu Kalra, Amir Momeni, Zhongliang Chen, Nick Materise, Brett Daley, Perhaad Mistry, and David Kaeli. 2015. Nupar: A benchmark suite for modern gpu architectures. In *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*. ACM, 253–264.