Comprehensive Analysis of CPUs, GPUs and Intel Xeon Phi: Performance, Energy, and Efficiency

Jiawen Liu, Yonghong Yan
Department of Computer Science and Engineering
Oakland University
{jliu, yan}@oakland.edu

***Abstract*—In the area of high performance computing (HPC), parallel benchmarks are constantly used to explore and evaluate the performance of heterogeneous computing systems on the horizon. Most previous and related work concentrates on measuring the performance of emerging heterogeneous architectures that consist of multi-core CPU, GPU and Intel Xeon Phi architectures, while other factors such as power consumption, efficiency and cost are considered as less important metrics. In this paper, we report our systematical evaluation of five representative benchmarks of Rodinia on modern CPU, GPU, and Intel Xeon Phi in terms of performance, power consumption, temperature, productivity and monetary cost. We propose a unified metric that combines those results as a means for overall comparison of the three architectures. The metric helps users to comprehensively evaluate heterogeneous architecture from both software and hardware perspectives.**

***Keywords*—Performance analysis, heterogeneous architecture, benchmark suite, power consumption, temperature, productivity, Xeon Phi coprocessor, Nvidia GPU**

I. INTRODUCTION

High performance computing using accelerators or coprocessors has been popular. In addition to CPU, Nvidia GPU and Intel Xeon Phi Many Integrated Cores (MIC) have gradually become two main architectures in the market of high performance and scientific applications, which requires high throughput and intensive computing power. MIC employs a classic x86 based (many-core) architecture and uses classical programming languages such as C/C++/FORTRAN with OpenMP/MPI [7, 10] while GPU uses a totally different architecture and programming language such as CUDA [18].

The architectures and programming models of MIC are similar to CPUs, but are highly different from GPU. This heterogeneity results in a difficult problem how we can compare heterogeneous architectures comprehensively and then choose the best accelerator or coprocessors for specific applications to get the best performance and overall efficiency. Understanding the performance characteristics of heterogeneous architectures can help us in designing more efficient applications, choosing the appropriate accelerators for an application, and developing more effective task scheduling and mapping strategies. In this paper, we investigate and analyze the performance of representative benchmarks of Rodinia [5] within different domains on CPU, GPU and MIC architectures.

The development of heterogeneous architectures promotes researchers to develop a lot of benchmark suites to evaluate the emerging class of architectures and to find out the strength and weakness between heterogeneous architectures. The most popular open source benchmark suites targeting heterogeneous architectures include Rodinia, Parboil [21] and the Scalable Heterogeneous Computing (SHOC) [8]. These benchmark suites provide applications implemented in different programming models for heterogeneous architectures and fulfill roles similar to PARSEC [4] and SPEC [12] that are other popular benchmark suites managed by an industry consortium. These benchmarks also help architects study programming models and applications concurrently.

The Rodinia applications are designed for heterogeneous computing infrastructures using OpenMP, CUDA, and OpenCL [20] targeting CPUs and GPUs to evaluate these two architectures separately. It provides representative real world applications from multiple domains such as Data Mining, Linear Algebra, Fluid Dynamics, Graph Algorithms, Physics Simulation ,Bioinformatics, etc. We carry out a set of comprehensive performance analysis by using the OpenMP and CUDA programming models for these applications. Although the performance analysis is our main motivating purpose, our metric and approaches for comprehensive performance analysis can be used for other heterogeneous benchmark such as Parboil and SHOC.

This paper makes the following contributions:

- We present a comprehensive performance analysis including performance, power consumption, temperature, productivity and monetary cost for five representative benchmarks of Rodinia on CPU, Nvidia GPU and Intel MIC architectures.
- We evaluate the performance efficiency with regard to practical and theoretical performance, energy efficiency with regard to performance and power consumption, and monetary cost efficiency with regard to performance and monetary cost of the three architectures.
- We propose a unified efficiency metric to compare the overall benefits of the three architectures.

The rest of the paper is organized as follows: Section II reviews the related work. Section III describes the overview of architectures and programming models. Section IV provides short overview of benchmarks of Rodinia. Section V presents comprehensive performance analysis of the benchmarks for CPU, GPU, and MIC. Section VI concludes the paper.

II. RELATED WORK

It is well known that as the computing technology advances, the size of compute intensive problems also increases so that GPU and MIC, as the most powerful accelerators, have become the latest demanding market for HPC computing compared with

CPU. Thus comprehensive performance analysis of heterogeneous architectures has become an important role. In the report made by You et al. [25], they gave out a new method to evaluation, which is performance efficiency. The formula of it is obtained performance/theoretical performance. The performance efficiency gap between two devices is small in the report. The performance of operations in an important class of applications in the fields of microscopy image analysis on GPU, MIC and CPU architectures are investigated and characterized by George Teodoro et al. [23]. They systematically implement and evaluated the performance of operations on modern CPU, GPU, and MIC systems using approaches that exhibit different data access patterns (regular and irregular), computation intensity, and types of parallelism, from a class of applications.

For the cluster systems, Massimo Bernaschi et al. [3] presented and compared the performances of GPU and MIC both in a single system and in cluster configuration for the simulation of two physical systems showing that the performances of an MIC change dramatically depending on the need to pad arrays to avoid dramatic performance drops due to L2 TLB trashing effects. In this paper [16], authors perform a rigorous performance analysis and find that after applying optimizations appropriate for both CPUs and GPUs, the performance gap between an Nvidia GTX280 processor and the Intel Core i7 960 processor narrows to only 2.5x on average. They also discuss optimization techniques for both CPU and GPU, analyze what architecture features contributed to performance differences between the two architectures, and recommend a set of architectural features which provide significant improvement in architectural efficiency for throughput kernels. Performance and evaluation comparison of Multi Text Keyword Search algorithms into MIC and GPU has also been proposed and evaluated by performance analysis [1]. Authors used Nvidia K20c and Nvidia K40 for their GPUs and Intel Xeon Phi 5100 for MIC, and found out that K20c and K40 outperformed MIC for this particular algorithm.

Researchers generally compare heterogeneous architectures by performance of applications instead of other critical factors such as performance efficiency, energy efficiency, monetary cost, temperature, productivity, etc. Some performance analysis of heterogeneous architectures are only based on the applications they developed instead of well known benchmarks that can be a standard comparison to fairly evaluate the heterogeneous architectures. In addition, researchers compare GPU with CPU while MIC has played an important role in high performance computing, or compare GPU with MIC while CPU has strength for some applications. Thus the importance of comprehensive comparison between GPU, MIC and CPU is obvious. Analytical models of CPUs [24] and GPUs [13] have also been proposed. They provide a structural understanding of throughput computing performance, which only focus on a homogeneous architecture for CPU or GPU instead of heterogeneous architectures. Base on the reasons above, we propose comprehensive performance analysis and a unified efficiency metric to evaluate heterogeneous architectures.

III. OVERVIEW OF ARCHITECTURES AND PROGRAMMING MODELS

Before diving into the description of our contribution, we introduce the state-of-the-art heterogeneous architectures and corresponding programming models for comprehensive performance analysis.

A. Multi-core CPU Architecture

Instead of increasing clock frequencies, the growth of parallelism has become the main purpose of traditional CPUs, which relies strongly on the increase of processor clock speed to increase the ability of computing. A multicore CPU is possible where each core is an independent processor containing multiple parallel pipelines, each pipeline being superscalar. Some processors also include vector capability. In addition, vector processing units have evolved to new instruction sets and longer vector registers. At the same time, modern CPU employs Non-uniform memory access (NUMA), which is a computer memory design used in multiprocessing, where the memory access time depends on the memory location relative to the processor. Under NUMA, CPU can access its own local memory faster than non-local memory (memory local to another processor or memory shared between processors). In addition, the CPU architecture is resource-rich running for task and data parallelism in the applications by using AVX extensions (Advanced Vector Extensions) to quickly operate on floating-point vectors.

B. Graphics Processing Units (GPU) Architecture

With the requirement and potential of accelerators for high performance computing, Nvidia develops Tesla series that are designed for massively parallel processing and large scale computing, which goes far beyond basic graphics controller functions, and is a programmable and powerful computational accelerator for the high performance computing mark. For the architecture, CPUs are much different from GPUs. CPUs are composed of few cores with lots of cache memory. It handles a few software threads at a time. In contrast, GPUs are composed of hundreds of cores that can handle thousands of threads simultaneously.

C. Many Integrated Core (MIC) Architecture

With the growing success of GPUs as a massively parallel processor well suited to more general purpose applications, Intel has introduced a line of products based on a Many Integrated Core (MIC) architecture, marketed under the name Xeon Phi. The Xeon Phi co-processors are symmetric multi-processors; they physically look similar to a GPU in that they plug into a host system via PCI Express (PCIe). They run a Linux Operating System. However, a Xeon Phi coprocessor cannot be used as a stand-alone processor, and requires a host system to operate.

From an architectural perspective, they also have many similarities to GPUs. Xeon Phi is based on an x86 Pentium core architecture from the early 1990s. This is a much simplified and hence smaller core compared to modern x86 CPUs. To make these simplified cores more computationally powerful, 512-bit-wide vector units have been added to the core. Because of the simplified nature of the core, each co-processor features 60+ cores clocked at 1 GHz or more, supporting 64-bit x86 instructions. The exact number of cores depends on the model and the generation of the product. These in-order cores support four-way hyper-threading, resulting in more than 240 logical cores. The cores of an Xeon Phi coprocessor are interconnected by a high speed bidirectional ring, which unites the L2 caches of the cores into a large coherent aggregate cache over 30 MB in size. The coprocessor is equipped with 6 to 16 GB of on-board GDDR5 memory. The speed and energy efficiency of Xeon Phi coprocessors come from their vector units. Each core contains a vector-processing unit (VPU) with 512-bit SIMD (Single Instruction Multiple Data) vectors supporting a new instruction set called Intel Initial Many-Core Instructions (Intel IMCI). The Intel IMCI includes, among other instructions the Fused Multiply-Add (FMA), reciprocal, square root, power and exponential function operations, commonly used in physical modeling and statistical analysis[11].

D. Programming Models

The programming tools and languages employed for code development for MICs are the same as those used for CPUs. This is a significant advantage as compared to GPUs that alleviates code migration overhead for the co-processor. For the applications, the implement uses OpenMP for MIC and CPU, and uses CUDA for GPU. MIC supports two execution modes: native and offload. In the native mode the application runs entirely within the co-processor. The offload mode allows for CPU to execute regions of the application code with the co-processor. These regions are defined using pragma tags and include directives for transferring data. The offload mode also supports conditional offload directives, which may be used to decide at runtime whether a region should be offloaded to the co-processor or should be executed on CPU. This feature is used in the applications. Since we execute the codes on MIC using the offload mode, a computing core is reserved to run the offload daemon, and a maximum of 240 computing cores are launched.

IV. RODINIA BENCHMARK SUITE

Ever growing field of HPC requires parallel benchmarks to facilitate research in architecture, programming languages, and operating systems. Rodinia currently presents OpenMP and CUDA implementations of a set of applications to use as benchmarks in the architecture, compiler, and programming-language research. It provides codes, which are reference implementations for research on parallelizing compilers as well as design patterns. For users to develop their own parallel codes, the benchmark is basically intended to evaluate the performance of heterogeneous computing. It incorporates applications and distinguishes kernels targeting the multicore architectures. To reflect major computational kernels in real-world scientific and engineering applications, this work focuses on five representative benchmarks, i.e., LUD, CFD, NW, BFS and HotSpot. Table 1 provides a summary of Berkeley dwarves [2], domains and problem size for the benchmarks.

LU Decomposition (LUD) is an algorithm to calculate the solutions of a set of linear equations. The LUD kernel decomposes a matrix as the product of a lower triangular matrix and an upper triangular matrix. This application has many row-wise and column-wise interdependencies and requires significant optimization to achieve good parallel performance.

Computational Fluid Dynamics (CFD) is an unstructured-grid, finite-volume solver for the three- dimensional Euler equations for compressible flow [6]. Effective GPU memory bandwidth is improved by reducing total global memory accesses and overlapping redundant computation, as well as by using an appropriate numbering scheme and data layout. The CFD solver is released with two versions: one with pre-computed fluxes, and the other with redundant flux computations.

Table 1: RODINIA BENCHMARK SUITE

Applications	Dwarves	Domains	Problem Size
LUD	Dense Linear Algebra	Linear Algebra	12000
CFD	Unstructured Grid	Fluid Dynamics	193000
NW	Dynamic Programming	Bioinformatics	32000
BFS	Graph Traversal	Graph Algorithms	64Million
HotSpot	Structured Grid	Physics Simulation	1024

Needleman-Wunsch (NW) is a dynamic programming algorithm for sequence alignments, which builds up the best alignment by using optimal alignments of smaller subsequences. It consists of three steps: initialization of the score matrix, calculation of scores, and deducing the alignment from the score matrix. The second step is parallelized.

Breath-First Search (BFS) is the Breadth-First Search that is an algorithm for traversing or searching tree or graph data structures and explores the neighbor nodes first before moving to the next level neighbored.

HotSpot is a widely used tool [14] to estimate processor temperature based on an architectural floor plan and simulated power measurements. The thermal simulation iteratively solves a series of differential equations for block. Each output cell in the computational grid represents the average temperature value of corresponding area of the chip.

V. COMPREHENSIVE PERFORMANCE ANALYSIS

This section evaluates and analyzes the performance of benchmarks for CPU, GPU, and MIC. We present the performance efficiency, energy efficiency, monetary efficiency, temperature and productivity using Nvidia K80 GPU, Intel Xeon Phi SC7120P MIC and dual Intel Xeon E5-2699 v3 CPU to evaluate performance. While CPU and MIC executables are generated from the same C source code annotated with OpenMP, GPU programs are implemented using CUDA.

A. Experimental Setup

Performance of CPU is measured with 256 GB of 2133 MHz DDR4 ECC memory, based on two-socket Intel Xeon E5-2699v3 CPUs for the server. The thermal design power (TDP) of each CPU socket is 145 W. The sockets are interconnected by a Quick Path Interconnect (QPI) link and forming a shared-memory NUMA system. Each socket has 18 physical cores (36 cores in the system) clocked at 2.3 GHz (turbo frequency of 3.6 GHz) with two-way hyper-threading. The vector units of the system support the AVX instruction set with 256-bit vector registers. The host operating system is CentOS 6.7 Linux with kernel version 2.6.32-573.12.1.el6.x86_64. The code was compiled with the Intel icc compiler version 13.1.3.

Table 2: Complete System Specification

	GPU	MIC	CPU x 2
Model	Tesla GK210	Xeon Phi SC7120P	Xeon E5-2699 v3
Core	2496	61	36
Core Clock (GHz)	0.56	1.24	2.3
SP TFLOPS	4.37	3.11	1.3
DP TFLOPS	1.45	1.55	0.65
Memory Size (GB)	24	16	-
Mem. Bandwidth(GB/s)	240	352	136
Monetary Cost (USD)	2499	3749	9800
Max Power Usage (TDP)	150	300	290

GPU is measured in the same system as CPU. The GPU model used for the results presented in this paper is the Nvidia Tesla K80 (active-cooled model for workstations), which has two Tesla GK210 GPUs. Characteristics for both GPUs are 2496 cores clocked at 560 MHz, 24 GB of on-board memory with a clock of 2.5 GHz, and a 384-bit memory interface. The code was written in CUDA and C compiled with the nvcc compiler, version 7.5.

With regard to Intel MIC, it is measured in the same system and Intel icc compiler as CPU. The system contains one MIC coprocessor, Xeon Phi QS-7120P (passive-cooled model for servers) with 61 cores at 1.24 GHz, C0 coprocessor stepping, and 16 GB of GDDR5 RAM at 2.75 GHz. The driver stack is MPSS, version 2.1.6720-13.

B. Performance and Performance Efficiency

Performance and efficiency are two most important factors to evaluate architectures of CPU, GPU and MIC. Execution time can be utilized as performance. However, since CPU, GPU and MIC are implemented in different architectures and they are designed in diverse hardware cores that have different definition and standard, it's hard to compare performance efficiency between different architectures. One way to compare their performance is the peak floating-point operations per second (FLOPS). In order to get fair results, we choose problem size of benchmarks that are large enough to eliminate the influence of instant change of performance, power consumption, temperature and data transfer in and out of GPU/MIC. Problem sizes of all applications are fixed at 12000 for LUD, 193000 for CFD, 32000 for NW, 64M for BFS and 1024 with 20000 loops for Hotspot. Threads available are assigned in maximum for CPU, GPU and MIC. The execution time shown is an average over ten runs.

The performance are summarized as speedup in Fig. 1. It presents the average normalized execution time observed for LUD, CFD, NW, BFS and HotSpot. The result shows the speedup of each application running on a K80 GPU and a SC7120P MIC relative to a dual Intel Xeon E5-2699v3 CPU. The speedups of GPU range from 2.3-103X as CPU implementations and the speedups of MIC range from 1.2-24X as CPU implementations excluding BFS.

With respect to GPU and MIC, data is copied to and from the accelerator/coprocessor under the explicit control from the programmer and runtime. Programmers mark code blocks with C/C++ pragmas for MIC and leverage runtime library for GPU. The compiler runtime transfers data from the host to the device, performs computation on the device and returns results back to the host. From Fig. 1, GPU outperforms MIC by 2 to 33 times. GPU is better than MIC at handling huge data because of its streaming multi-processor (SM) that can handle millions of threads.

The stream mechanism offered by CUDA allows the hiding of the communication overhead, in particular of the copies between CPU and GPU, provided that the computation executed concurrently with the communication is large enough. MIC is incapable of overcoming the extra processing capabilities even though MIC has superior memory transfer latency. The overheads of MIC, e.g., data offloading overheads of memory transfer and computation overheads between CPU and MIC, are much more than GPU in the benchmarks. In native mode of MIC, application directly executes on coprocessor without being offloaded to coprocessor. Such applications are required to be compiled and built on the host system using -mmic supporting flag which enables the compiler to generate an object file for Xeon Phi architecture. Executing applications in native mode requires all the dynamic libraries and executable must be present on coprocessor. MIC can take advantage of the native mode and would outperform GPU. But with respect to the benchmarks, the implementation for MIC is offload mode instead of native mode. Running on an MIC in offload mode is possible to emulate the CUDA streaming mechanism that supports independent execution flows. The performance of an MIC remains low in offload mode compared to a GPU with the same number of computing unit for a problem of fixed size.

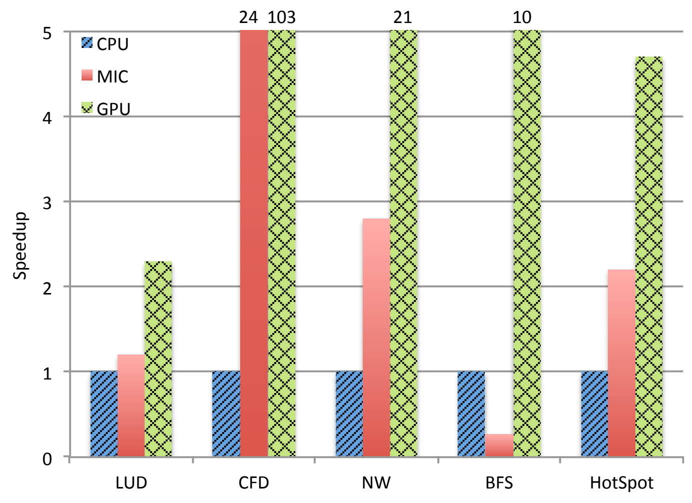

Fig. 1: Performance comparison among CPU, GPU and MIC on speedup is normalized against CPU for LUD, CFD, NW, BFS and HotSpot. Each value is the average of ten runs. GPU outperforms CPU and MIC for execution time performance.

For traditional homogeneous architectures, the performance efficiency is defined as follows. Let T_s represent execution time for serial computing of one core, P is the number of cores and T_p is the execution time for concurrent computing with the number of P cores in a given application. Thus the speedup S_p and performance efficiency E_p are defined as:

$$S_p = \frac{T_s}{T_p} \quad and \quad E_p = \frac{S_p}{P} \tag{1}$$

Since the benchmarks are designed using single precision floating point for computation, the theoretical peak FLOPS of CPU, GPU and MIC in this system are respectively 1.3 TFLOPS (Tera FLOPS), 3.11 TFLOPS and 4.87 TFLOPS. We use CPU as the baseline and normalize the performance efficiency. Let T_b denote base execution time and T_t denote target execution time. S_p represents speedup calculated as the ratio of T_b to T_t. The theoretical peak FLOPS performance of heterogeneous architectures is denoted as F_t. Thus the formula of performance efficiency E_p is as follows:

$$S_p = \frac{T_b}{T_t} \quad and \quad E_p = \frac{S_p}{F_t} \tag{2}$$

Performance efficiency represents the gap between practical performance and theoretical performance in heterogeneous architectures, which is a critical consideration to find out the factors to improve architecture designs. From Fig. 2, we conclude that CPU has a better performance efficiency than GPU in LUD and better than MIC in LUD, BFS and HotSpot. Since the theoretical performance of GPU is 3.4X as CPU, but the practical speedup is only 2.3X as CPU, the performance efficiency of CPU is better than GPU in LUD. With respect to MIC, similarly even though MIC has 1.2X and 2.2X better practical performance than MIC in LUD and HotSpot respectively, MIC has 2.4X better theoretical performance than CPU. Therefore, the performance efficiency of CPU is better than MIC in LUD and HotSpot.

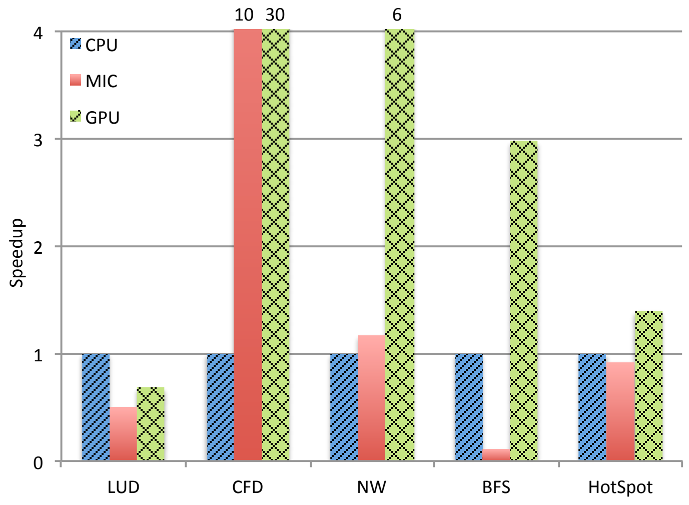

Fig. 2: Performance efficiency is based on practical performance and theoretical performance. The comparison among CPU, GPU and MIC uses CPU execution time as baseline for LUD, CFD, NW, BFS and HotSpot.

New challenges need to be solved in order to analyze monetary cost corresponding to the performance of GPU, MIC and CPU especially in computing centers where makes an efficient or suitable decision balancing the performance and cost. Thus the monetary cost efficiency is important to consider. Let S_p and C represent speedup of performance and monetary cost, respectively. We utilize CPU as the baseline and normalize the performance efficiency so the formula of monetary cost efficiency E_c is as follows:

$$E_c = \frac{S_p}{C} \tag{3}$$

C. Power Consumption and Energy Efficiency

There are growing numbers of commercial high performance computing solutions using various accelerators. Therefore, power consumption has increasingly become a concern. In addition, a better understanding of the power tradeoffs of heterogeneous architectures is needed to guide usage in server clusters and data centers.

NVML (Nvidia Management Library) [19] is a runtime library to measure power consumption when executing the kernel. However, since nvidia-smi is a high-level interface allowing administrators to query GPU information in command line, the rate of sampling power consumption is very low so that we could not record the change in power until the kernel is running for a very long time. The nvmlDeviceGetPowerUsage function in the NVML library obtains the power consumption for GPU in milliwatts. The power consumption is obtained for the entire board. The power consumption is accurate within +/- 5 watts error with precision in milliwatt and is updated at roughly 60Hz. To measure the runtime power of a kernel with the NVML library for lowest overhead, we run NVML on a thread and the kernel on other threads. We choose Pthread because it reduces overhead, and the only communication we would like to have with the main thread is a flag variable and a variable to store power readings which is set to be volatile. The thread running NVML stops when the flag is reset, which is when the GPU kernel stops executing. The temperature is provided by integrated sensors inside the device.

Power consumption of MIC is calculated by the approximation of power timeslices provided by MIC System Management Control (micsmc) [22] which has direct connection to the coprocessor I2C interface [15], an on-board I2C sensor bus and a third interface through the SMBus pins of the PCI Express connector to the system management solution which is used for obtaining data including frequency, power, temperature, memory usage, and CPU utilization per core. However, as additional types of data are requested, the delay between calls increase. Therefore, to capture the energy readings in the smallest timeslice available, only temperature and power data are recorded (both are printed when the same input parameter is given). The micsmc tool measures and reports every 22–28ms, which is sufficient for a detailed evaluation.

Recent Intel SandyBridge chips include the "Running Average Power Limit" (RAPL) interface [9] which is designed to provide an infrastructure for keeping processors inside of a given user-specified power envelope. RAPL supports power capping PKG, PP0 and DRAM power planes by writing into the relevant model specific register (MSR). The internal circuitry can estimate current energy usage based on a model driven by hardware counters and temperature. The results of this model are available to the user via a MSR which can be accessed to get power consumption for each power plane with an update frequency on the order of milliseconds.

Idle power is the power consumed by the devices when the device is in a long idle state, i.e., when the devices are doing nothing for a long period of time. As Fig. 3 illustrates, at the idle state the power consumption of CPU is much lower than GPU and MIC and is evaluated 28% and 19% for GPU and MIC respectively. With respect to the benchmarks, MIC consumes more power than GPU except HotSpot and CPU except NW. The power consumption of MIC is 1.01-2.37X as GPU and 1.20-1.68X as CPU. GPU consumes more power than MIC and CPU in HotSpot while CPU consumes more power than MIC and GPU in NW. One reason is that certain factors of power consumption of MIC such as the execution of the graphics pipeline, bus communication and memory access consume more power than the corresponding parts of GPU. For certain benchmarks, GPU and MIC are more effective than CPU for the computations that use a small size of input data in communication between the host and devices and perform well in compute-intensive operations which reduce the workload of communication and computation to decrease power consumption. For some workload, memory-bound causes a decrease in power consumption, as the duration of execution is reducing, consuming less cumulative power. In other words, intensive workload of MIC and GPU should generally use the high core clock so that it completes operation as soon as possible to minimize energy.

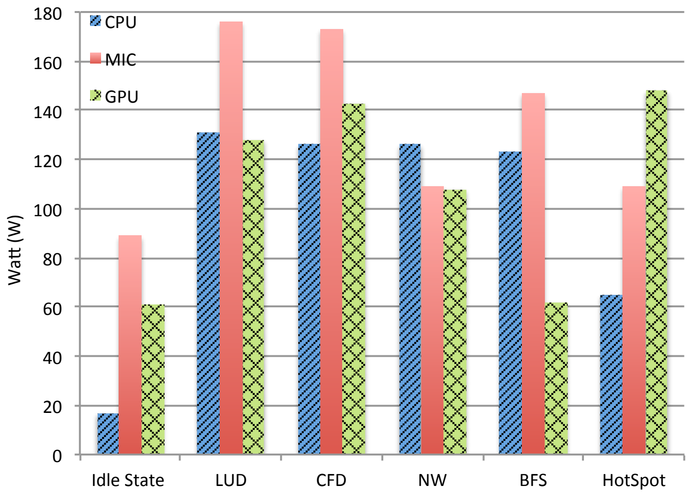

Fig. 3: Power consumption comparison among CPU, GPU and MIC for idle state, LUD, CFD, NW, BFS and HotSpot.

With regard to scientific computing programs, FLOPS per watt has gained popularity in evaluating energy efficiency of high performance computing applications. Practical FLOPS and power consumption are respectively denoted as F_p and P. We utilize CPU as the baseline and normalize the energy efficiency so the formula of energy efficiency E_e is as follows:

$$E_e = \frac{F_p}{P} \tag{4}$$

According to our measurement in Fig. 4, GPU outperforms MIC and CPU in all benchmarks while MIC outperforms CPU in CFD, NW and HotSpot. Because FLOPS of GPU in the benchmarks is higher than MIC and CPU, even though the power consumption of GPU is more than MIC and CPU in HotSpot, GPU still has better energy efficiency. With respect to LUD between MIC and CPU, because the power consumption of MIC in LUD is 1.4X as CPU and MIC performs 1.2X better FLOPS performance than CPU, CPU outperforms MIC.

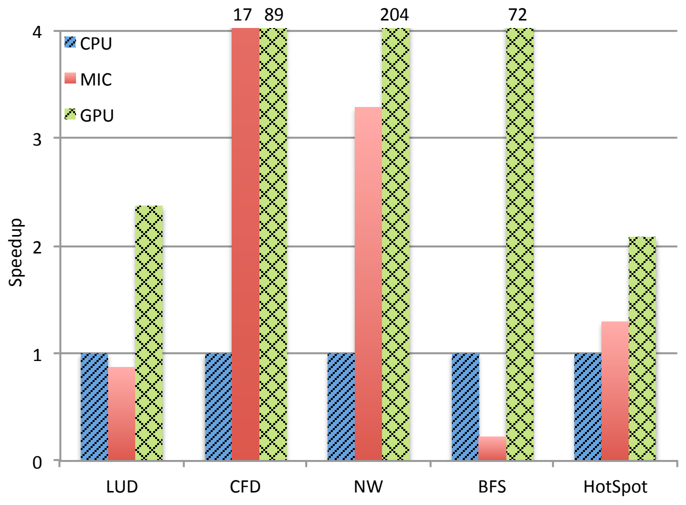

Fig. 4: Energy efficiency is the efficiency of the practical FLOPS to power consumption on devices of heterogeneous architectures for each benchmark using CPU as baseline. GPU performs better than CPU and MIC for five benchmarks on energy efficiency.

D. Temperature

Temperature of running kernels on heterogeneous architecture devices is also a critical consideration for high performance computing in server clusters and data centers because we need to pay for cooling down systems and to utilize them efficiently.

By comparing results in Fig. 5, MIC stays in higher temperature than GPU and CPU, and GPU stays in higher temperature than CPU. In addition, we observe that for MIC and GPU even though the temperature of GPU is only 80% of MIC in HotSpot, power consumption of GPU is 1.4X as MIC, which means that for some benchmarks even though GPU consumes more power for calculation than MIC, MIC stays in higher temperature than GPU. With regard to other benchmarks, the power consumption of MIC is 1.01-2.37X as GPU which the temperature is 1.04-1.25X as GPU. From Fig. 5, CPU performs lower than MIC and GPU especially in idle state. The temperature of CPU is 63-83% of MIC for benchmarks and 59% in idle state while CPU is 77-100% of GPU excluding HotSpot and 63% in idle state. Because the design for CPU is for general purpose and task scheduling in the whole system, the temperature of CPU must stay low in idle for some applications using less intensive computing. But the purpose of MIC and GPU is for high throughput and intensive computing, which in architecture design is one of the reasons why temperature of MIC and GPU is higher than CPU.

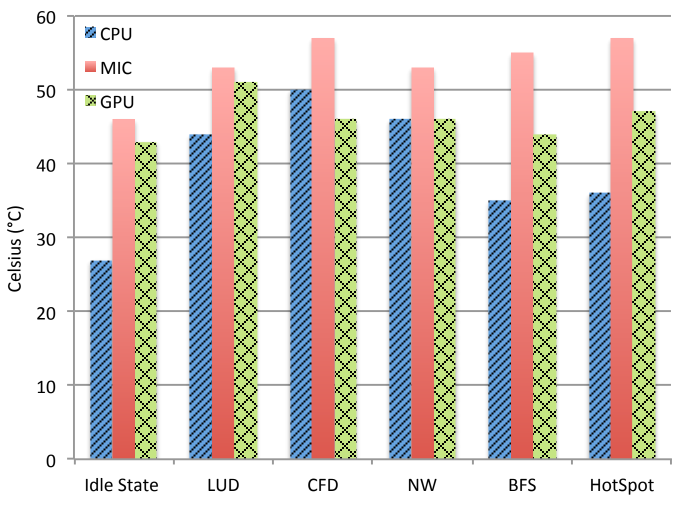

Fig. 5: Temperature comparison among CPU, GPU and MIC for idle state, LUD, CFD, NW, BFS and HotSpot.

E. Monetary Cost Efficiency

Monetary cost efficiency represents the efficiency of the performance speedup to monetary cost paid for heterogeneous architectures on specific mainstream CPU, GPU and MIC, which also refers to the performance speedup of heterogeneous architectures per dollar. The monetary cost is presented in Table 2. From Fig. 6, the monetary cost efficiency of GPU performs well than CPU and MIC for the benchmarks because the monetary cost of CPU is 3.9X as GPU and the monetary cost of MIC is 1.5X as GPU. With regard to CPU and MIC, the monetary cost efficiency of MIC is better than CPU for the benchmarks excluding BFS because the speedup performance of CPU is 3.3X as MIC but the monetary cost of CPU is only 2.6X as MIC. Since the speedup performance of MIC in BFS is only 30% of CPU, the monetary cost efficiency of CPU is still better than MIC in BFS.

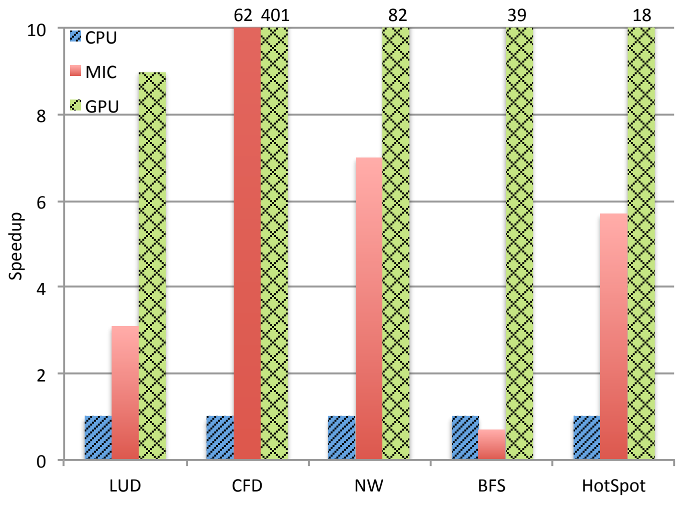

Fig. 6: Monetary cost efficiency is the efficiency of the performance speedup using CPU as baseline to monetary cost paid for devices of heterogeneous architectures. GPU performs better than CPU and MIC for five benchmarks on monetary cost efficiency.

F. Productivity of Programming

As we know, the productivity of programming is difficult to measure under a standard approach because the ability of each programmer and the characteristics of each programming language is totally different. One way to compare the productivity of programming for different language is the source lines of code (SLOC). SLOC is one of the most widely used sizing metric in industry and literature. It is the key input for most of major cost estimation models such as COCOMO, SLIM, and SEER-SEM. Its long-standing tradition is due to the fact that SLOC is the direct result of programming work as the most perceivable indicator of software cost. The SLOC counting standard [17] is followed for all SLOC measurements for applications of heterogeneous architectures.

The advantage of the programming models for MICs and CPUs is reducing source code size. In principle, one of the main advantages of the Intel MIC technology, with respect to other coprocessors and accelerators, is the simplicity of the porting. Programmers compile their existing source codes specifying MIC as the target architecture. Classical programming languages used in high performance computing – Fortran/C/C++ – as well as well known parallel paradigms – OpenMP or MPI – may be directly employed regarding MIC as a “classic” x86 based (many-core) architecture. An Intel MIC may be accessed directly as a stand-alone system running executables in the native mode. However, offload execution mode is available and used more widely. Adopting the offload execution model, a code runs mostly on the host but selected regions of the code are marked through directives or APIs to be executed on MIC coprocessor. On the contrary, the porting from the classical programming languages such as Fortran/C/C++ to CUDA needs more effort because CUDA is a significantly different programming model.

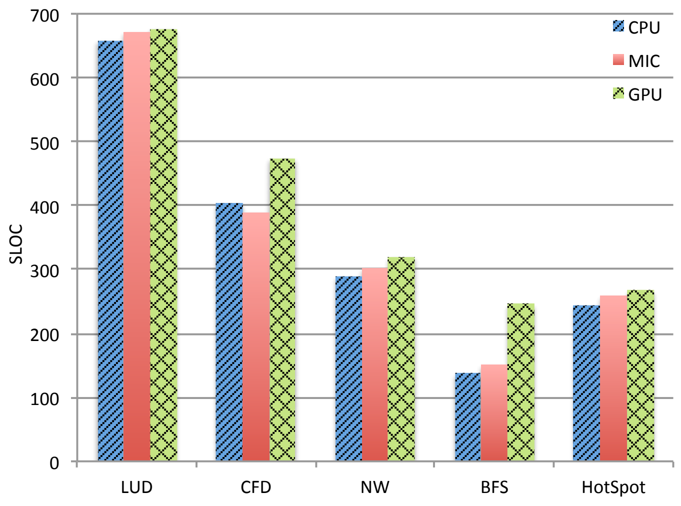

Fig. 7: SLOC comparison among CPU, GPU and MIC for LUD, CFD, NW, BFS and HotSpot.

G. Unified Efficiency Metric

To gain further insights into the performance of each benchmark, this work considers a unified efficiency metrics obtained from performance efficiency, energy efficiency and monetary efficiency for overall comparison of the three architectures. The main idea of the unified efficiency metric is that different efficiency represents the strengths and weaknesses of different architectures.

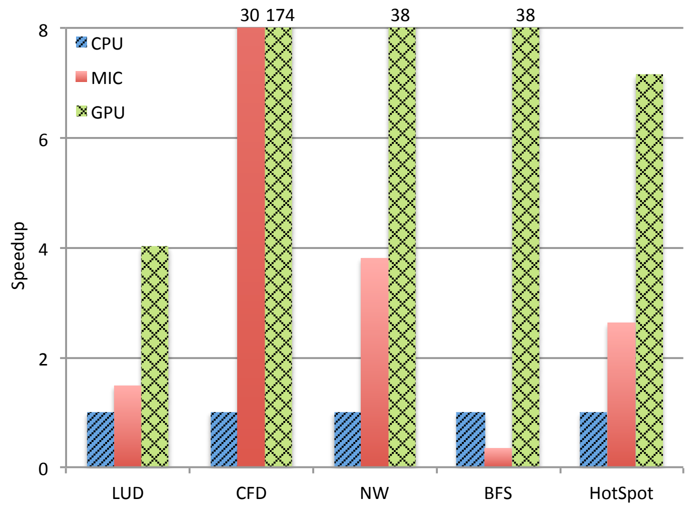

Fig. 8: Unified efficiency metric leverages performance efficiency, energy efficiency and monetary efficiency comparing strengths and weaknesses of CPU, GPU and MIC.

Let E_p represent performance efficiency that is the ratio of speedup (based on CPU) to the theoretical peak performance of the target, E_e be the energy efficiency that is the ratio of practical FLOPS to average power consumption of the specific application for different target devices, and E_c represent monetary cost efficiency that is the ratio of speedup to monetary cost. Let S_p represent speedup; F_t represents theoretical peak FLOPS; F_p represents practical FLOPS; P represents power consumption; and C represents monetary cost. The unified efficiency metric E_u formula is as follows:

$$\begin{aligned} E_u &= E_p + E_e + E_c \\ &= \frac{S_p}{F_t} + \frac{F_p}{P} + \frac{S_p}{C} \end{aligned} \tag{5}$$

Let T_b denote base execution time, T_t denote target execution time and F_a denote the number of floating-point operations. Since $S_p = T_b \,/\, T_t$ and $F_p = F_n \,/\, T_t$, the metric E_u is as follows:

$$\begin{aligned} E_u &= \frac{S_p}{F_t} + \frac{F_n}{P * T_t} + \frac{S_p}{C} \\ &= S_p\left(\frac{1}{F_t} + \frac{F_n}{P * T_b} + \frac{1}{C}\right) \end{aligned} \tag{6}$$

From Fig. 8, GPU is still better than MIC and much better than CPU for the benchmarks. This illustrates that when combining performance efficiency, energy efficiency and monetary cost efficiency, GPU performs better compared to CPU and MIC for the benchmarks. Since MIC outperforms CPU for the benchmarks excluding BFS, for some intensive computing applications designed for HPC in heterogeneous computing infrastructures such as BFS, CPU still outperforms MIC when we consider overall benefits.

VI. CONCLUSION

In this paper, we present a comprehensive performance analysis including performance, power consumption, temperature, productivity of programming and monetary cost for benchmarks on CPU, Nvidia GPU and Intel MIC architectures. Regarding efficiency, we evaluate the performance efficiency with respect to practical and theoretical performance, energy efficiency with respect to performance and power consumption, and monetary cost efficiency with respect to performance and monetary cost of the three architectures. To compare the overall benefits, we propose a unified efficiency metric for three architectures. In the future, our effort will focus on comparing three architectures on large-scale parallel and multi-node computer systems, so as to guide architecture and system design.

References

[1] A. Abdullah, K. K. Yong, E. K. Karuppiah, and P. K. Chong, "Multi keyword range search in gpu and mic: A comparison study," in *Open Systems (ICOS), 2014 IEEE Conference on*, Oct 2014, pp. 117–122.

[2] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams *et al.*, "The landscape of parallel computing research: A view from berkeley," Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Tech. Rep., 2006.

[3] M. Bernaschi, M. Bisson, and F. Salvadore, "Multi-kepler {GPU} vs. multi-intel {MIC} for spin systems simulations," *Computer Physics Communications*, vol. 185, no. 10, pp. 2495 – 2503, 2014.

[4] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '08. New York, NY, USA: ACM, 2008, pp. 72–81.

[5] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S. H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*, Oct 2009, pp. 44–54.

[6] A. Corrigan, F. F. Camelli, R. Löhner, and J. Wallin, "Running unstructured grid-based cfd solvers on modern graphics hardware," *International Journal for Numerical Methods in Fluids*, vol. 66, no. 2, pp. 221–229, 2011.

[7] L. Dagum and R. Menon, "Openmp: An industry-standard api for shared-memory programming," *IEEE Comput. Sci. Eng.*, vol. 5, no. 1, pp. 46–55, Jan. 1998.

[8] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, V. Tipparaju, and J. S. Vetter, "The scalable heterogeneous computing (shoc) benchmark suite," in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, ser. GPGPU-3. New York, NY, USA: ACM, 2010, pp. 63–74.

[9] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, "Rapl: Memory power estimation and capping," in *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design*, ser. ISLPED '10. New York, NY, USA: ACM, 2010, pp. 189–194.

[10] M. P. Forum, "Mpi: A message-passing interface standard," Knoxville, TN, USA, Tech. Rep., 1994.

[11] V. Halyo, P. LeGresley, P. Lujan, V. Karpusenko, and A. Vladimirov, "First Evaluation of the CPU, GPGPU and MIC Architectures for Real Time Particle Tracking based on Hough Transform at the LHC," *JINST*, vol. 9, p. P04005, 2014.

[12] J. L. Henning, "Spec cpu2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, Sep. 2006.

[13] S. Hong and H. Kim, "An analytical model for a gpu architecture with memory-level and thread-level parallelism awareness," *SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 152–163, Jun. 2009.

[14] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "Hotspot: a compact thermal modeling methodology for early-stage vlsi design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 5, pp. 501–513, May 2006.

[15] Intel. (2013) Ipmi intelligent platform management interface. [Online]. Available: http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/ipmi-second-gen-interface-spec-v2-rev1-1.pdf

[16] V. W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A. D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, R. Singhal, and P. Dubey, “Debunking the 100x gpu vs. cpu myth: An evaluation of throughput computing on cpu and gpu,” *SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 451–460, Jun. 2010.

[17] V. Nguyen, S. Deeds-Rubin, T. Tan, and B. Boehm, “A sloc counting standard,” in *COCOMO II Forum*, vol. 2007, 2007.

[18] NVIDIA. (2007) Cuda. [Online]. Available: http://www.nvidia.com/object/cuda_home_new.html

[19] ——. (2015) Nvml api reference guide. [Online]. Available: http://docs.nvidia.com/deploy/nvml-api/index.html

[20] J. E. Stone, D. Gohara, and G. Shi, “Opencl: A parallel programming standard for heterogeneous computing systems,” *IEEE Des. Test*, vol. 12, no. 3, pp. 66–73, May 2010.

[21] J. A. Stratton, C. Rodrigrues, I.-J. Sung, N. Obeid, L. Chang, G. Liu, and W.-M. W. Hwu, “Parboil: A revised benchmark suite for scientific and commercial throughput computing,” University of Illinois at Urbana-Champaign, Urbana, Tech. Rep. IMPACT-12-01, Mar. 2012.

[22] Taylor.K. (2014) Intel xeon phi coprocessor power management configuration. [Online]. Available: https://software.intel.com/en-us/blogs/2014/01/31/intel-xeon-phi-coprocessor-power-management-configuration

[23] G. Teodoro, T. Kurc, J. Kong, L. Cooper, and J. Saltz, “Comparative performance analysis of intel (r) xeon phi (tm), gpu, and cpu: A case study from microscopy image analysis,” in *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, May 2014, pp. 1063–1072.

[24] S. Williams, A. Waterman, and D. Patterson, “Roofline: An insightful visual performance model for multicore architectures,” *Commun. ACM*, vol. 52, no. 4, pp. 65–76, Apr. 2009.

[25] Y. You, H. Fu, X. Huang, G. Song, L. Gan, W. Yu, and G. Yang, “Accelerating the 3d elastic wave forward modeling on gpu and mic,” in *Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, ser. IPDPSW ’13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 1088–1096.