NVIDIA & SIMD Case Study also comparing it with Intel Xeon Phi

CS 6070

Project Documentation

Students Name:

Eric Kane, Hamid Shekarforoush

Brian Sigurdson, Sara Yarshenas

Academic Supervisor:

Dr. Hassan Rajaei

Spring 2017

Department of Computer Science

Bowling Green State University, Ohio

NVIDIA & SIMD Case Study also comparing it with Intel Xeon Phi

Students:
Eric Kane, Hamid Shekarforoush
Brian Sigurdson, Sara Yarshenas
Supervisor: Dr. Rajaei
Dept. of Computer Science
Bowling Green State University, Ohio
Reporting date: 05/01/2017

Abstract

Many Task Computing (MTC) is a concept aimed to run a high number of tasks in a short time. To achieve this goal in previous years, GPUs were the tool of choice to process many tasks in a short time. In 2013, Intel introduced Intel Xeon Phi coprocessors, as a direct competitor for GPUs used for MTC. This paper describes both GPUs with a focus on NVIDIA and Intel Xeon Phi in terms of architecture, design and performance. In terms of performance, there is no single tool that was studied that performs best in all cases. In general, GPUs and Intel Xeon Phi both have areas in which they perform best.

Table of Contents

[This page is left intentionally blank for printing purpose]

1. Introduction

Motivation

Throughput is an important concern in virtually every aspect of computing. Nobody can escape the fact that more throughput generally results in more responsive user interfaces, more data or transactions processed per unit of measure, more efficiency; and frequently higher throughput implies a lower cost of processing information per unit of measure.
Graphical Processing Units (GPU) have existed in one form or another since the 1970's, mainly to benefit the user's visual experience. In roughly 2006, there began a shift toward using GPU's for general purpose and scientific calculations, often referring to them as GPGPU for General Purpose GPU's.

Problem Statement

Although GPU's have many benefits, such as massive parallelism and reasonable cost (even for retail consumers), GPU's are not the only solution to such issues. Intel's Xeon Phi coprocessor has similar benefits. Both GPU's and Xeon Phi can be considered SIMD solutions, yet each attempt to solve the problem of throughput in a different manner. NVIDIA's GPUs are generally considered to be a competitor with Intel's Xeon Phi coprocessor. Applications always can utilize Intel Xeon Phi coprocessors, if they have a satisfactory outcome working with GPUs, but the reverse is not necessarily true because systems with coprocessors have much more applicability and also their tuning is too different to GPU.

Approach

We asked how are these approaches, NVIDIA's GPUs and Xeon Phi, are similar, how they are different, and how do they compare under certain performance metrics? Is there an obvious winner?

Aside from the knowledge gained in our course, CS 6070, and its text, Computer Architecture - A Quantitative Approach, we examined various industry and research articles to more deeply understand how each technology works and where it was being implemented. We reviewed articles discussing their historical development, to understand their transition over time. We evaluated benchmark results in order to compare key performance metrics.

Results

At present, both NVIDIA and Intel are popular SIMD solutions, but we have no evidence to suggest that NVIDIA's or Intel's approach is superior in all cases. Currently, we believe that there is not a clear winner, or superior approach, as the notion of "winner" will depend on the criteria compared and the user's needs. It is likely that each approach has advantages and disadvantages, and whether NVIDIA's GPU or Intel Xeon Phi is best depends on the computing environment.

Research Questions

1. How do GPUs and Intel Xeon Phi compare, which architecture performs better in which situation(s)?
2. In what computation category should a software developer use GPUs and in which others should Xeon Phi be used?
3. What was the historical development of GPUs and Xeon Phi and how did that shape current architectures?
4. What are CUDA Cores and how do they relate to GPUs?

2. Related work

2.1. Summary of Reviewed Work

The brief introduction to the history of GPU development and use relied on the following entry in Wikipedia.com, https://en.wikipedia.org/wiki/Graphics_processing_unit, and information from chapter 1 of Professional CUDA C Programming.

Wikipedia gave a more detailed account of the historical development of GPUs than could be covered in this section of our report. Only small parts of Professional CUDA C Programming were referenced for this write-up. This is because it is difficult to discuss GPUs without eventually including NVIDIA, and it is difficult to discuss the popularity and utility of NVIDIA GPUs with including CUDA.

These articles, Wikipedia and chapter 1 Professional CUDA C Programming, relate to our research question concerning the historical development of GPUs.

Intel Many Integrated Core architecture is a highly parallel and efficient processor architecture that achieves high performance. Intel Xeon Phi coprocessor is one of these accelerators based on the Intel MIC architecture. The evolution of Intel Xeon Phi Coprocessor has been reviewed through this project as a background and different generations of Intel MIC architecture are mentioned with all their hardware characteristics.

As a part of real world performance of our studied platforms, there are several research available that compare both platform in a same hardware environment with different types of software. The importance of using different software and benchmarks is shown in the results. Each of our platform will perform differently depending on the benchmark.

We find that there are a great many examples of uses for Intel Xeon Phi and NVIDIA architectures [21], [22], [23], [24]. In these specific examples, we gather first-hand accounts of how these platforms are being used today and what the strengths and weaknesses are for each architecture. This work is closely related to ours in that we need to use the evidence gleaned from these sources as proof that these architectures work in the real world and that each has its own advantages and disadvantages. These papers in conjunction with our case studies have helped us answer the questions related to which architecture is in fact better and in which situations.

2.2. Analysis and Discussions

Many articles are written on the state of technology at a point in time. We've taken an approach that examines the longitudinal development of NVIDIA GPUs and Intel Xeon Phi coprocessors, while also examining their various strengths and weakness over such duration. We feel that such an examination of these two competing technologies facilitates a better way of comparing them than just at a single point in time.

A problem with the researches was that they were to specific with limited real world usage. A comparison between Intel Xeon Phi and NVIDIA GPUs that covers a massive size of data like Facebook or Amazon would be great.

In the papers used for the examples section of this study, the authors wrote informal accounts about the uses of these architectures. The work that has been done is giving information about how these tools can be used in the real world. They have not completed statistical analysis to state what the performance of these tools are in their applications, but that was not the aim of the writings, as they are meant to give an idea of how these tools can be used. Our work is different from these papers in that we use them as support for our arguments that each architecture has its own strengths and weaknesses that can be seen in these examples. In conjunction with the case studies, these papers give us the needed evidence to make conclusions about our research questions.

3. Background

The following provides some background information concerning the development of GPUs and Intel Xeon Phi coprocessors.

3.1. GPU

Given the apparent similar functionality provided by CPUs and GPUs, readers may be surprised to know that the modern GPU, such as NVIDIA Tesla C2075, shown in Figure 1 below, was developed from the architecture used to power popular arcade games from the 1970's and 1980's, referred to as graphical accelerators, and not from CPUs.[1]

Figure 1: NVIDIA Tesla C2075 [2]

One such game, Space Invaders, shown in Figure 2, was quite popular from its inception in 1978, and on into the 1980's. In the 1970's RAM was expensive, so early game developers had to rely

on specialized hardware to power such games. Early accelerators "used specialized graphics hardware supporting RGB color, multi-colored sprites and tile-map backgrounds."

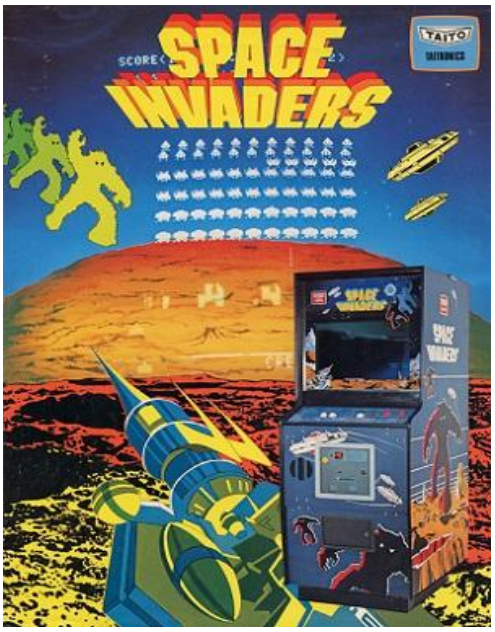

Figure 2: 1978 Advertisement [3]

The 1980's saw improvement in accelerator technology, with companies such as NEC, Commodore, Texas Instruments, and IBM all competing to build better accelerators. The decade began with NEC's popular PD7220 "graphics display controller as a single Large Scale Integration (LSI) integrated circuit chip…" In 1987 IBM released "one of the first video cards for the IBM PC," but before the end of 1988 Fujitsu has released a graphics card "with support for a full 16,777,216 color palate." Additionally, accelerators, often referred to simply as graphics cards, were becoming accessible to the home pc market.

In the 1990's, "2D GUI acceleration continued to evolve," and "real-time 3D graphics were becoming increasingly common in arcade, computer and console games, which led to an increasing public demand for hardware-accelerated 3D graphics." Initially, 3D accelerators required their own hardware, exclusive of 2D support, but eventually both 2D and 3D processing was integrated onto a single chip.

During this time, the open graphics library, OpenGL, emerged and struggled due to a lack of performance. After quickly correcting such issues, OpenGL became quite popular, helping to spur demand for comparable hardware support. At the same time, Microsoft's DirectX was popular with Windows users, and Microsoft began to coordinate its release to "coincide with those of the supporting graphics hardware."

By the end of the 1990's, features of professional graphics hardware, such as accelerated transform and lighting, had begun to make their way into consumer products, such as NVIDIA's GeForce 256.

In the early part of the next decade, graphical processors made a jump in sophistication and speed. As stated in the graphical processing unit article in Wikipedia:

> By October 2002, with the introduction of the ATI Radeon 9700 (also known as R300), the world's first Direct3D 9.0 accelerator, pixel and vertex shaders could implement looping and lengthy floating point math, and were quickly becoming as flexible as CPUs, yet orders of magnitude faster for image-array operations.

In 2007, NVIDIA's GeForce 8 series introduced an even more general computing device, ushering in the use of graphical processing units for more general purpose computing, often referred to as GPGPUs or general purpose GPU. Although, GPGPU use increased in popularity, initial applications required an "abuse of hardware", due to the fact that GPU devices only understood the language of graphics processing, therefore requiring applications to express themselves in terms of graphics processing.

NVIDIA introduced CUDA in 2007, "the earliest widely adopted programming model for GPU computing." CUDA was developed by NVIDIA to facilitate general purpose-parallel processing on NVIDIA GPUs, and is an extension of the ANSI C programming language. CUDA, through its libraries, provides bindings to languages such as C, C++, Fortran, and Python [4].

GPUs have always been adept at computationally intensive procedures such as calculating and rendering geometric manipulations for display devices. Many of these calculations involve significant use of vectors and matrices, making GPU processing attractive to scientists and engineers. GPUs benefit applications that display a lot of data parallelism, which can exploit the GPUs SIMD architecture. GPU computing is now widely used in diverse areas such as machine learning, statistics, graphics processing, oil exploration, financial analysis, cell phones, and virtual reality.

3.2. Xeon Phi

Xeon Phi is the first product of Intel's Many Integrated Core(MIC) architecture which has many names such as Intel Xeon-Phi Coprocessor, Knights Corner, etc. It benefits from X86 architecture which is a CPU with multiple simpler cores with better throughput. In this processor, a lot of power hungry operations have been removed and it has wider hardware thread count and vector unit. Xeon Phi uses X86 programming models, runs Linux and it is cache coherent. The Intel Xeon Phi coprocessors are designed for the applications that can benefit from memory bandwidth and processor vector capabilities completely, so they provide vector support, local memory bandwidth and power efficient scaling for these applications. Figure 3 shows the

necessity of Intel Xeon Phi Coprocessor according to the performance while the number of threads increases because Intel Xeon Processor has a limitation in practical situation [13]. Intel Xeon Phi Coprocessors use the same programming languages and parallelism model which are used by other Intel products and this counts as an advantage.

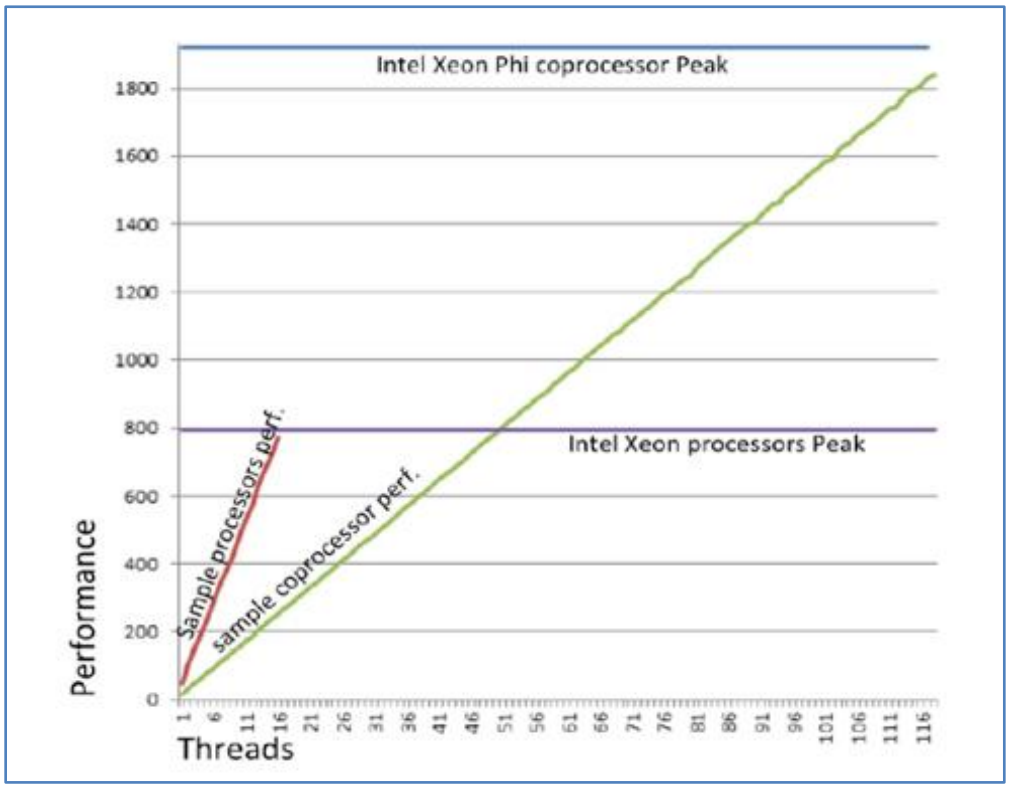

Figure 3: For understanding Intel Xeon Phi Coprocessors

Processors are cache coherent and shared access to main memory with other processors. Coprocessors are cache coherent SMP-on-a-chip devices that connect to other devices via the PCI express bus, and are not hardware cache coherent with other processors or coprocessors in the system, thus a platform cannot consist of only coprocessors. The number of cores in an Intel Xeon Phi coprocessor depends on the situation. Generally speaking, Intel Xeon Phi has up to 61 Intel Architecture cores, 1.1 GHz, 244 threads, 512-bit SIMD instructions, up to 8 GB memory with up to 352 GB/s bandwidth, Linux operating system, IP addressable and Standard programming languages and tools. The best way to prepare for Intel Xeon Phi coprocessors is to fully exploit the performance that an application can get on Intel Xeon processors first. Intel Xeon Phi coprocessors offer a corollary to this: higher performance comes from pairing highly parallel software with highly parallel hardware. The best place to start is to make sure your application is maximizing the capabilities of an Intel Xeon processor. In getting an application ready for utilizing an Intel Xeon Phi coprocessor, nothing is more important than scaling. An application must scale well past one hundred threads to qualify as highly parallel. Efficient use of vectors and/or memory bandwidth is also essential. Intel Xeon Phi coprocessors can reach

heights in performance beyond that of an Intel Xeon processor, but it requires more parallelism to do so.

Intel Many Integrated Core architecture is a highly parallel and efficient processor architecture that achieves high performance through utilization of a large number of cores, wide vector registers and the high bandwidth on package memory [16]. Intel Xeon Phi coprocessor is the brand name of computing accelerators based on the intel MIC architecture (Many Integrated Core) and so those two names will be used interchangeably. The govern generation of Intel Xeon Phi coprocessor is based on the chip code name, Knights Corner (KNC) and we will also occasionally use this name to refer to Intel Xeon Phi coprocessors. The next generation of Intel MIC architecture will be based on Knights Landing chip (KNL). It will be available not only as a coprocessor, but also as a standalone processor. Xeon Phi coprocessors are PCI express end-point devices and they are primarily used for offload of performance critical parts of an application and for heterogeneous clustering. For highly parallel applications, Xeon Phi coprocessors had theoretical peak performance up to 1.2 TGLOP/s in double precision and 2.4 TFLOP/s in single precision. The theoretical peak bandwidth of the onboard memory is up to 384 GB/s, however in practice we observed about 160-180 GB/s. Connectivity with the host system is provided by x16 PCI express Gen 2 interconnect. Compared to the multicore Xeon processor, Xeon Phi coprocessor delivers better performance per watt because its cores are more simple than Xeon cores. Xeon Phi also has better performance per unit of rack space because you can put multiple coprocessor in each compute unit.

The Xeon Phi coprocessors came in performance shelves, the 3100 series, 5100 series and 7100 series. The 3100 series has 57 cores and 6GB of RAM, this is the price optimal shelf. the 5100 series has 60 cores with 8GB of RAM at 225 Watts TDP, this could response to optimal performance per watt. The 7100 series has 61 cores and 16GB of RAM with theoretical maximum performance of 1.2 TFLOPs/s double precision, this is the highest performance shelf. In the enumeration convention for Xeon Phi models also pay attention to the second digit (3**1**00, 5**1**00 and 7**1**00), this is the coprocessor generation. Currently the only available generation is 1 which corresponds to the Knights Corner chip [17]. Finally, the letter after four numbers indicates the cooling solutions, most people will need either passive cooling solution indicated by a letter B or active cooling solution with a fan on board, a letter A. What is important here is that the passive solution is only for servers while the active solution may be used in workstations. Figure 4 shows this information briefly [15].

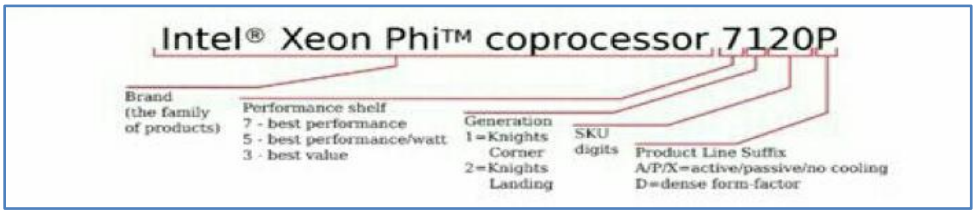

Figure 4: Intel Xeon Phi SKU Lineup

3.3. Intel Xeon Phi Architecture

Figure 5 is the architecture of the entire Intel Xeon Phi board including its chip and this board plugs into PCI express slot [15]. In Intel Xeon Phi architecture there is a flash memory which is a small piece of memory that contains boot up code. There are 16 channels around main coprocessor which connect to the memory and the memory used in this coprocessor is GDDR5 memory which is a higher bandwidth memory than the memory you find in a regular computer, because the coprocessors running 244 threads through 61 cores which needs higher performance of faster memory. Thus, they decided to use this GDDR5 memory which was originally used in graphics processors and graphics processors typically have multiple cores. SMC stands for system management controller which is a component with several sensors to measure physical characteristics such as temperature and voltage and it can decide turn fan control on or off, but if chips do not have this component they will be put into racks with own cooling system or liquid cooled. Hard drive is missing from this diagram since you have your own hard drive on your computer and when you boot up your computer you then start up separate services that you need to install. Many Core Platform Software Stack Service (ESS) is a service that takes files and pushes them across to this processor and GDDR5 memory, then processor executes that code which is Linux kernel. There are also some configuration text files that you can modify and configure Linux.

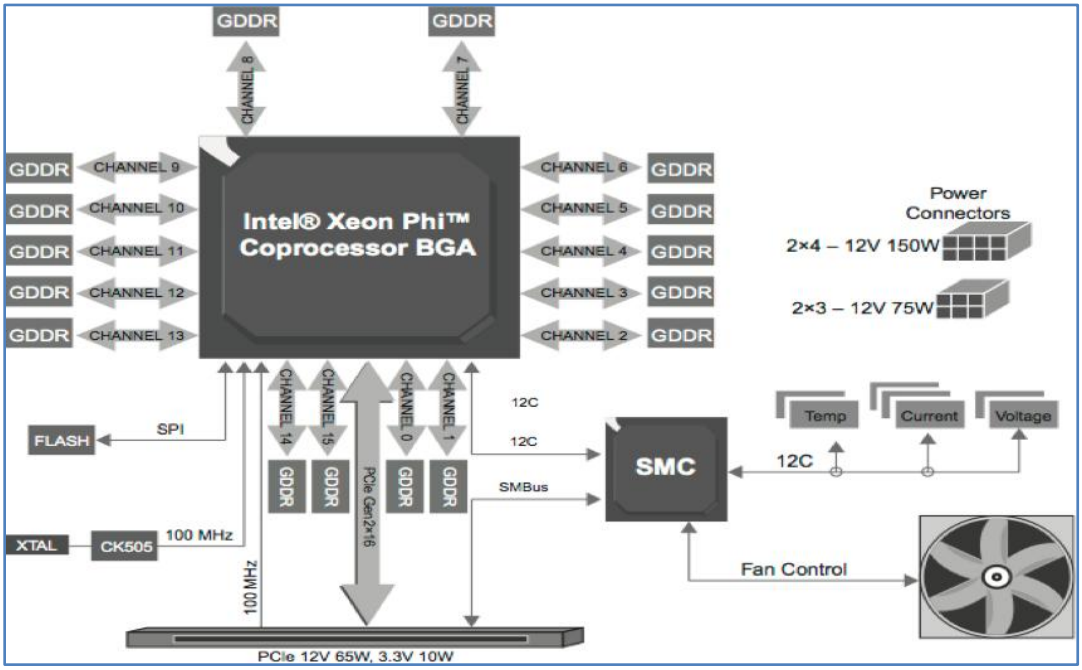

Figure 5: Architecture of Intel Xeon Phi board

Figure 6 is the die organization diagram for the Knights Corner chip on which the Xeon Phi coprocessor is built [17]. The coprocessor contains 62 cores of which 57, 60 or 61 are active. Those cores are connected to each other with the data bus called the core ring interconnect

(CRI). There are 8 memory controllers (GBOX) on the core ring interconnect providing access to the onboard GDDR5 memory. The ring also interconnects components of the distributed tag directory which helps to maintain cache coherency in parallel. Each core has slice of a level two with hardware pre fetcher. Together, the caches of all cores form a full coherent aggregate cache. Inside the core, the level 2 cache is interned by a level 1 or L1 cache. This memory hierarchy is extremely important from programmers' point of view because data locality and better memory access can dramatically improve the performance of parallel applications [16]. When an application running on the core accesses a data element at some memory address, there are four cases to consider. The first case is when data is not in the cache, either because it has never been accessed or because it has been evicted. The core will first check the local level 1 cache and because the data is not there the core will refer to the tag directory to check if the address is cached somewhere in the level 2 cache. After that, the core will request the cache line which is 64 bytes long from memory controller which then accesses the GDDR5 memory bank to get the cache line. This line is then copied over the level 2 cache of the local core and then to level 1 cache before it is finally used. This is extremely costly, taking hundreds of cycles to complete. The second case, you saved the cache line is in cache of neighboring core. In this case, the core will refer to the tag directory to find out the location of cache line and then request the cache line directly from the remote L2 cache. This instruction is far or less costly and only takes tens of cycles. The third and fourth cases are the best-case scenarios which are one the data is in the local level 2 and level 1 caches. In these cases, the cache line can be obtained in about 10 cycles or one cycle from the level 2 and level 1 caches respectively. Therefore, data locality is the key toward using the latency overhead of accessing memory and improving performance especially for the problems with intensive memory traffic [15].

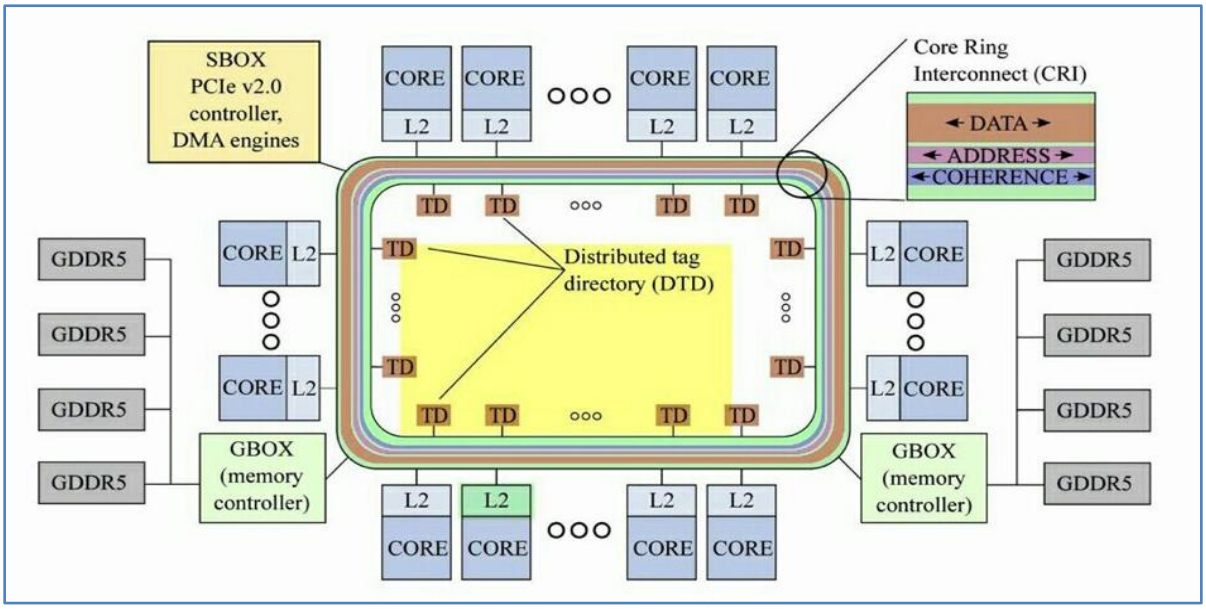

Figure 6: Die organization diagram for Knights Corner Chip

3.4. Maximizing Parallel Performance

When choosing whether to run an application on Intel Xeon processors or Intel Xeon Phi coprocessors, we can start with two basic points to obtain high performance:

I) Scaling
An application must be able of using the highly parallel capabilities of an Intel Xeon Phi coprocessor and this should be evaluated at first.

II) Vectorization and Memory usage
The application should either utilize vector units or benefit from more local memory bandwidth in comparison to that of Intel Xeon processors? In the case that both of these basics are correct, running application on Intel Xeon Phi coprocessor are beneficial to do [13].

3.5. Programming and Software Capability Comparison

Regarding GPU, the application which works well by using GPU should utilize Intel Xeon Phi coprocessor, but the flexibility of Intel Xeon Phi coprocessor cannot run on GPU because it has a wider application. The Intel Xeon Phi coprocessors are appropriate for applications using parallelism, while most of them utilize some tuning such as vectorization in order to achieve the maximum rate of performance. There is one challenge in using of Intel Xeon Phi coprocessor which is the temptation to stop tuning before the best performance which can be a good thing or bad thing. For the purpose of achieving ultimate performance, some unnecessary user level optimizations exist for situations that highest performance is needed [13]. Some of these optimizations are: Vectorization, Memory access and loop transformation, large page considerations and so on.

Parallel computation can be achieved by scaling and vector processing and we can gain minimal data movement by with higher bandwidth between cores and memory. Thus, Intel Xeon Phi coprocessor and Intel Xeon processor benefit from these models and languages such as Fortran, C, C++ and so on. The application will benefit from Intel Xeon processors when scaling, vector usage and memory usage get tuned on Intel Xeon Phi coprocessors. Intel Xeon Phi coprocessor provides four hardware threads and essential floating point abilities per core in order to prevent any limitation and programs should specify the numbers of threads and cores in order to run on processors. There are two approaches for making an application involved to the Intel Xeon Phi coprocessor that an MPI program can work with. The first one is offload model processor, the program runs on processors and offloading select work to coprocessors which is mostly used for finer parallelism. The second one is the native model and a program runs natively on processors and coprocessors and they will communicate with a method [13].

GPUs do not have the programmability of an Intel Xeon Phi coprocessor, but by combination of vectorization, bandwidth and scaling this can be achieved by GPU as well. In other words, applications always can utilize Intel Xeon Phi coprocessors if they had a positive experience with, however the opposite is not true. Generally speaking, a system with Intel Xeon Phi coprocessors has more applicability than that of with GPUs because the flexibility of an Intel Xeon Phi coprocessor includes support for applications that cannot run on GPUs. Moreover, in order to have a dual transforming tuning in Intel Xeon Phi coprocessors, tuning procedure is very different from GPU tuning.

4. CUDA, CUDA Cores, and NVIDIA's GPUs

Discussing the current state of GPUs is difficult, due to the speed at which the technology is constantly changing. It is more prudent to take a brief look at the evolution of the NVIDIA line of GPUs, from their first full fledged GPUs emphasizing throughput, Tesla and Fermi, to their more recent GPUs emphasizing energy efficiency, Kepler and Maxwell. Their relative performance per watt are illustrated in figure 7 below.

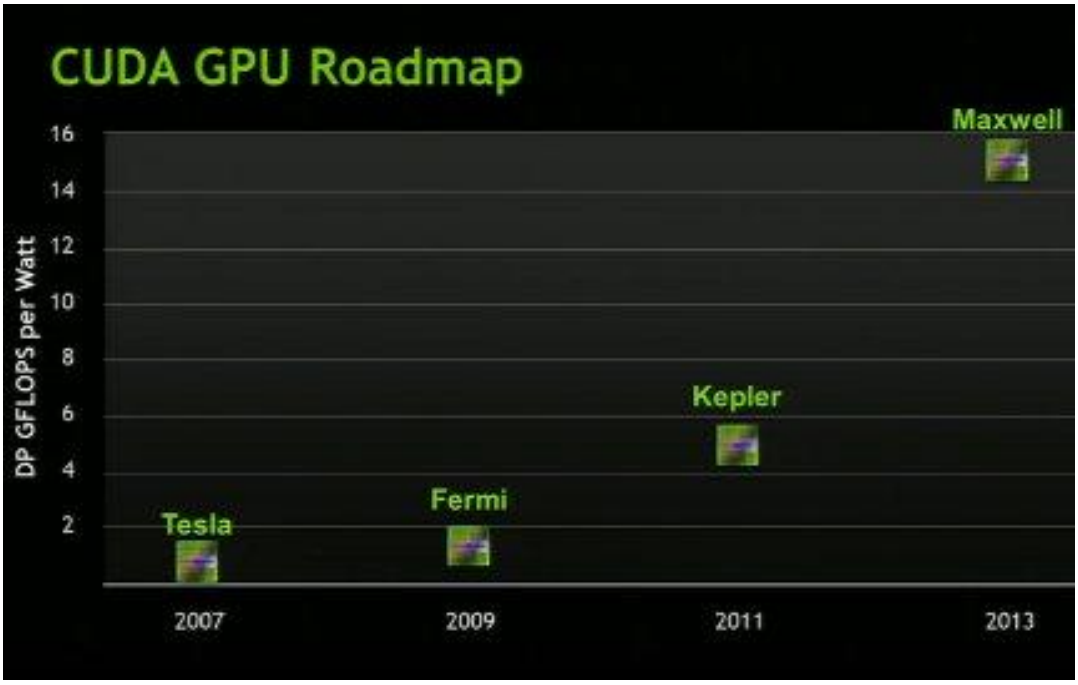

Figure 7: NVIDIA's CUDA GPU Roadmap [5]

From figure 7, based on double precision giga-flops per watt, you can see that between Fermi and Kepler models, NVIDIA shifted its emphasis from that of throughput to that of energy efficiency. In the following sections we'll discuss some of the significant changes to NVIDIA's GPU technology for the four models already mentioned.

4.1. Tesla - CUDA

Tesla, introduced in 2007, represented the first full-fledged general purpose GPU. It still performed well with graphics, but now programmers had a device that more easily allowed for more general purpose programming, such matrix multiplication and scientific computing.

Although Tesla implemented many changes to their previous micro-architectures, "the most obvious change being the move from the separate functional units (pixel shaders, vertex shaders) within previous GPUs to a homogeneous collection of universal floating point processors (called "stream processors") that can perform a more universal set of tasks [6]." These streaming processors (SP) are also known as CUDA Cores, and were previously known as shader units in graphical programming parlance.

The unifying structure of Tesla along with the introduction of the C-language extension CUDA, discussed previously, made big impact on the productivity of non-graphical programming, and ushered in the way for GPUs acceptance in more mainstream applications, such as image processing, machine learning, simulations, and many other disciplines also.

4.2. Fermi - FP64

Tesla was followed by Fermi, introduced in 2010[6], represented the next step in NVIDIA's GPU series. As was mentioned in the discussion on Tesla, in Fermi, these streaming processors (CUDA Cores) are further grouped to form what is known as streaming multiprocessors (SM), or groups of CUDA Cores. A 32 core SM can be seen in figure 8 below.

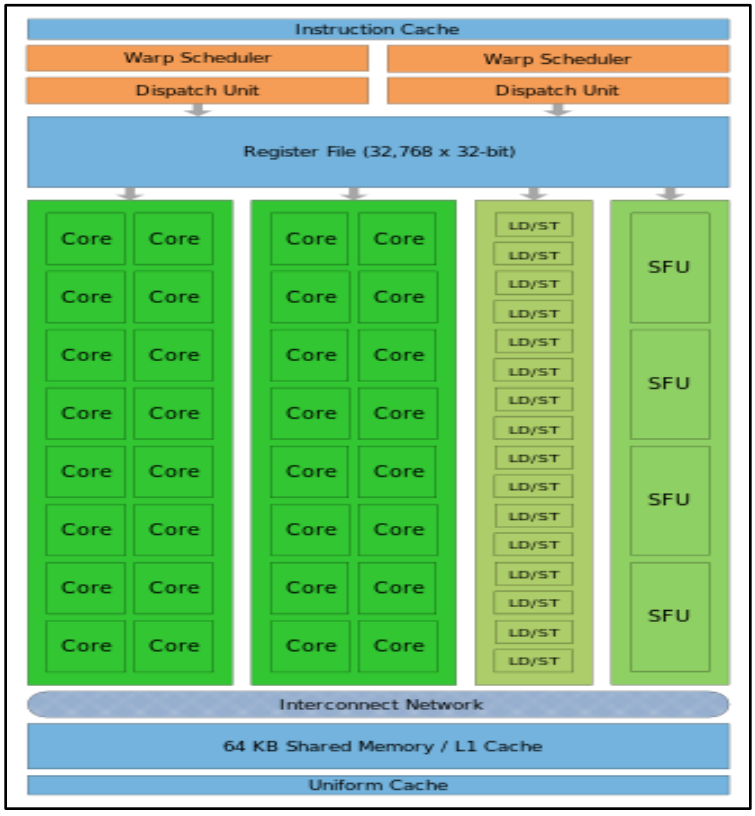

Figure 8: Streaming Multiprocessor (32 core) on Fermi [6]

The Fermi micro-architecture features 32 CUDA cores, implementing 32-bit single precision floating point operations [6], also "the Fermi architecture, each SM has 64 KB of on-chip memory that can be configured as 48 KB of Shared memory with 16 KB of L1 cache or as 16 KB of Shared memory with 48 KB of L1 cache. [7]"

4.3. Kepler

The GPU micro-architecture following Fermi was named Kepler. With the introduction of Kepler, NVIDIA made a conscious effort to shift their focus from throughput to energy efficiency, as illustrated in figure 7 above. Some significant changes to the Kepler micro-architecture are briefly discussed below.

4.3.1. Kepler – SMX

SMX refers to Kepler's new streaming multiprocessor architecture. The efficiency of the GPU was significantly enhanced by moving to a single internal clock, which allowed two Kepler cores to consume less energy than a single Fermi core [8].

4.3.2. Kepler – Dynamic Parallelism

Dynamic parallelism is the ability for the GPU to launch its own kernels. Previously, GPU kernels could only be launched by the CPU, requiring it to pause and consume CPU cycles for a relatively simple task. This "allows less-structured, more complex tasks to run easily and effectively, enabling larger portions of an application to run entirely on the GPU [9]."

4.3.3. Kepler – NVIDIA GPU Direct

NVIDIA GPU Direct works similar to direct memory access (DMA), in that GPUs within the same machine, or through network connections, can directly access each others memory, thus bypassing the use of CPU memory [9]. This reduces the demands on the CPU and memory bandwidth.

4.4. Maxwell

The final architecture we discuss is the Maxwell micro-architecture, the successor to the Kepler micro-architecture, and introduced in early 2014[10]. Some of the improvements in Maxwell are discussed below.

4.4.1. Maxwell – SMM

With Maxwell, NVIDIA again improved on the streaming multiprocessor. In addition to continued efficiency improvements, illustrated in figure 9 below, the Maxwell micro-architecture added just over 500 CUDA cores, for a total or 2048 cores for the Maxwell GPU [11].

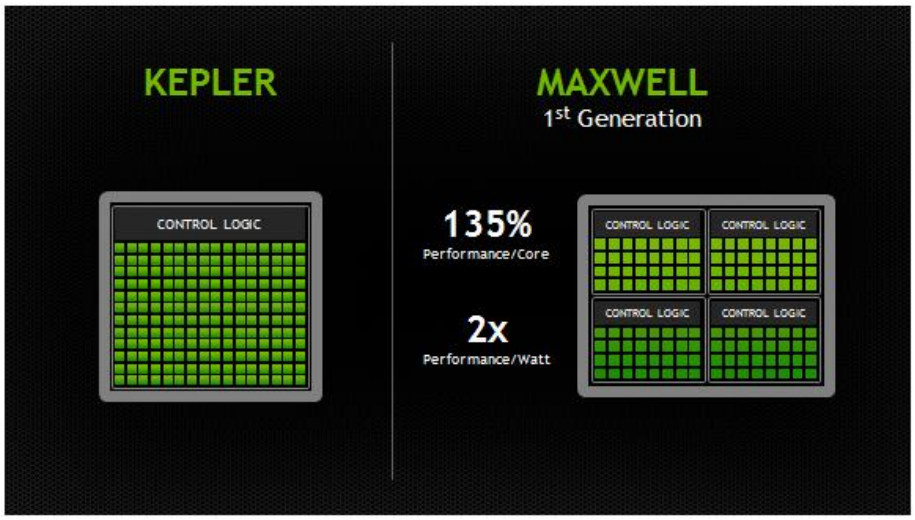

Figure 9: Maxwell [12]

Maxwell's new streaming multiprocessors have streamlined scheduling and "control logic partitioning, workload balancing, clock-gating granularity, instruction scheduling, number of instructions issued per clock cycle, and many other enhancements allow the Maxwell SM (also called "SMM") to far exceed Kepler SMX efficiency [12]."

4.5. GPU vs Xeon Phi architecture, design and specification

In terms of architecture comparison the first thing to compare is the number of cores. Xeon Phi Coprocessors has an average of 60 core with 1 GHz of clock speed. On the other hand, we have the Tesla GPUs with an insane number of smaller cores usually around 2500 cores with lower clock speed of 0.7 GHz. and both of them has the same amount of TDP (Thermal Design Power) of around 230. The other main difference is in their bandwidth and cache memory. In terms of memory, different generations of Xeon Phi Coprocessors and Tesla GPUs usually have 6 or 8 gigabytes of ram. So we can assume that the memory capacity is not real game changer in here. For L1 cache each of Xeon Phi Coprocessor cores have a 32 KB of cache and the Tesla GPUs have the same amount of cache but per CUDA core. We talk about CUDA cores later. In L2 cache again each Xeon Phi Coprocessor core have its own 512 KB of cache and Tesla GPUs have 1.5 MB of cache per CUDA core. From the marketing perspective both of latest product from each of them goes for around 5000$ and the old generation's price are relatively the same. Additionally, both of them uses PCI express 3 connections to work with the system.

In conclusion we have two different architectures, one with fewer cores, more clock speed, and more cache size, while the other architecture has 45X more cores, but smaller clock speed and cache size that both have the same power usage and price. There are tons of research papers that comparing both architecture in terms of benchmark in different categories of processing. We try to summarize some of the result in later section. Figure 10 shows specifications of different platform.

	Xeon E5-2670	Xeon Phi 5110P	Tesla M2050	Tesla K20c	Tesla K20X	GeForce GTX Titan
Performance (SP)	333 GFlops	2022 GFlops	1030 GFlops	3520 GFlops	3.95 GFlops	4500 GFlops
Memory Bandwidth	51.2 GB/s	320 GB/s (ECC off)	148 GB/s (ECC off)	208 GB/s (ECC off)	250 GB/s (ECC off)	288 GB/s (nonECC)
Memory Size	---	8 GB	3 GB	5 GB	6 GB	6 GB
number of cores	8	60/61	448	2496	2688	2688
Clock Speed	2.6 GHz	1.053 GHz	1.15 GHz	0.706 GHz	0.732 GHz	0.837 GHz

Figure 10: specification of different platforms [20]

4.6. Programming and Software Capability Comparison

GPUs do not have the programmability of an Intel Xeon Phi coprocessor, but by combination of vectorization, bandwidth and scaling this can be achieved by GPU as well. In other words, applications always can utilize Intel Xeon Phi coprocessors if they had a positive experience with, however the opposite is not true. Generally speaking, a system with Intel Xeon Phi coprocessors has more applicability than that of with GPUs because the flexibility of an Intel Xeon Phi coprocessor includes support for applications that cannot run on GPUs. Moreover, in order to have a dual transforming tuning in Intel Xeon Phi coprocessors, tuning procedure is very different from GPU tuning.

5. Case Studies, Examples, Discussions, and Analysis

This section analyzes some benchmark case studies with different software categories. These benchmarks show the performance of Intel Xeon Phi and NVIDIA GPUs compare to each other. We try to include different application types with different data access, data type and parallelism level.

5.1. Case Study 1: Image Processing

Teodoro and Kurc [18] perform an investigation to compare the performance of GPUs, Intel Xeon Phi and normal CPUs in microscopy image analysis applications. These applications usually use low dimensional spatial dataset, mostly 2D and 3D. These features can apply to some other image processing domains like satellite image processing, thus the authors were confident

that the result can be similar on other applications domain with the same type of dataset. Their research consists of different operation categories like data cleaning, Feature computation, classification and others that chained together sequentially. There are data dependencies in the data and different levels of parallelism can be found in different stages of the application. So, the result is a good combination of different scenario and can be trusted. To optimize the application for running on different platforms and provide a level playing field, they use OpenMP for coding in Intel Xeon Phi and CPU and CUDA for implementing the code for GPU.

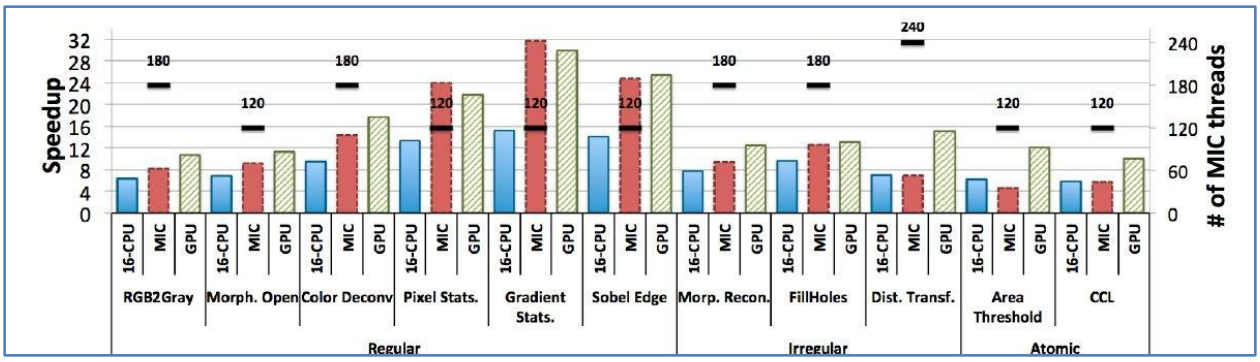

Figure 11: "Speedups achieved by operations on the CPU, MIC, and GPU, using the single core CPU version as a reference. The number above each dash refers to the number of threads that lead to the best performance on the MIC."[18]

The overall performance of research is presented in Figure [7]. The MIC means Many Integrated Core, which refers to Intel Xeon Phi architecture. As you can see there is high variability in different systems on different operations. To better analyzing of the results the authors divide the operations into three different sections. First the regular, who considers the operations with regular data access, second the operation with irregular data access, and lastly operations that heavily rely on the use of atomic functions, which include the Area Threshold and Connected Component Labeling. We are going discuss in the conclusion section, different data access types have a huge impact between different platforms.

As shown in irregular section, “the bandwidth attained by the processors is much lower than those with regular data access. The GPU significantly outperforms the other devices. The random writing bandwidth of the MIC processors is notably poor. This is in fact expected because this processor needs to maintain cache consistency among its many cores, which will result in a high data traffic and competition in its ring bus connecting caches. Due the low bandwidth attained by all the processors, all of our irregular operations are necessarily memory bound.”

5.2. Case Study 2: Particle Tracking

This case study tries to compare the performance result of our different platforms in real time particle tracking based on Hough transform at the LHC (Large Hadron Collider). What is

standing out in this case study is the real-time response of the application and the use of server farms, instead of single servers.

The application must deal with type of computations that exponentially grow over execution time. The authors [19] also mentioned that the difficulty with these applications was that they should write and optimize new algorithms as they go on, since the project deals with new idea and undiscovered phenomena in physics. Like the last case study, the authors optimize the application for different platforms, using OpenMP for Intel Xeon Phi and CUDA for NVIDIA GPUs.

"The Hough transform calculation is a highly parallel task. However, it cannot take advantage of the GPU or many-core architecture in an optimal way because of the nature of this calculation."

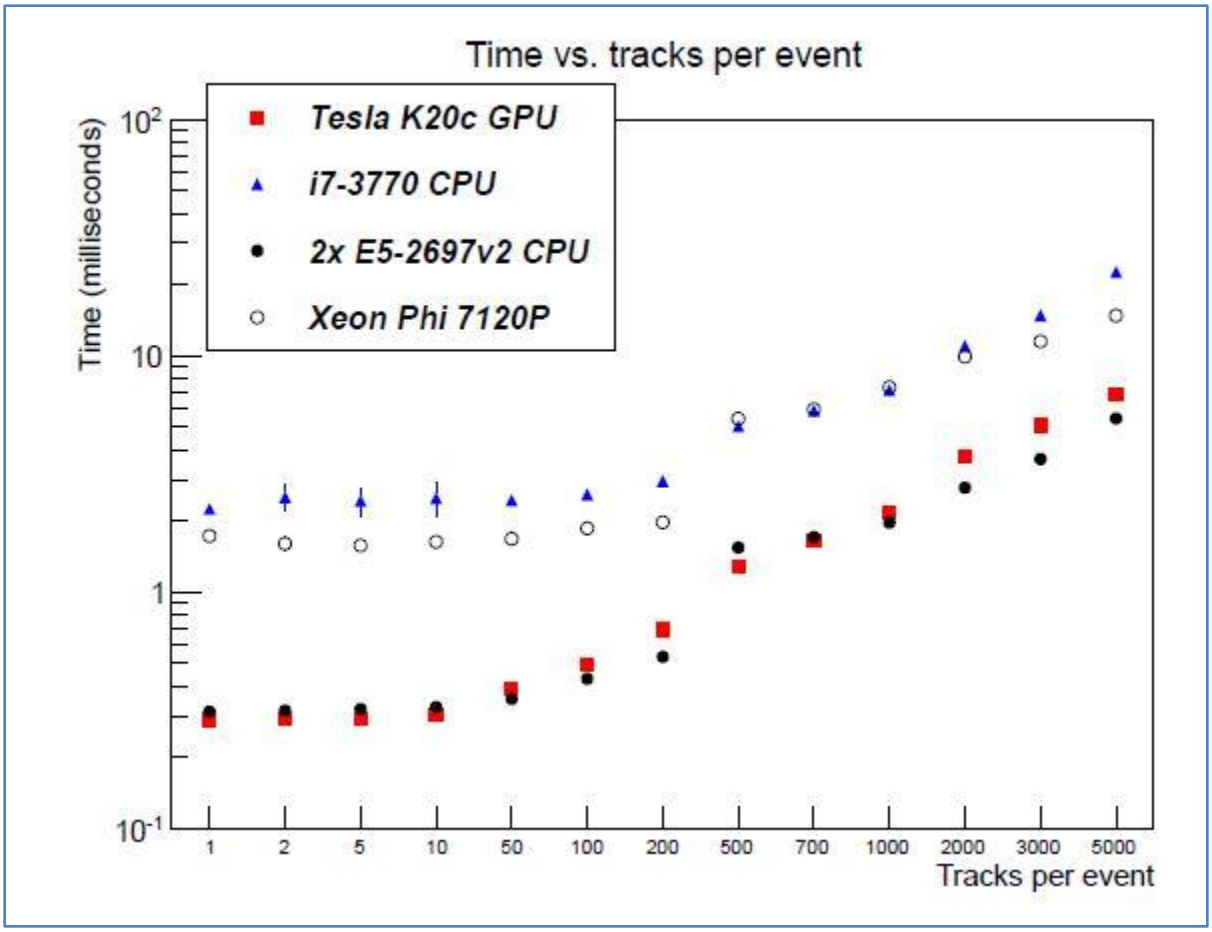

Figure 12: "Performance of the Hough transform algorithm on four platforms: NVIDIA Tesla K20c GPU (red squares), Intel i7-3770 CPU (blue triangles), dual-socket Intel Xeon E5-2697v2 CPU (solid black circles), and Intel Xeon Phi 7120P coprocessor (open black circle

The figure 12 shows the result of application running on 4 different platforms. As you can see from the results the i7 CPU and Intel Xeon Phi are considerably slower than Xeon CPUs and

NVIDIA GPUs. In large number tracking The Intel Xeon Phi is 3 times slower than the NVIDIA GPU. And Xeon CPU is 5 times faster than the i7 CPU. This 5X difference is related to the ratio of number of cores in those platforms.

In a conclusion discussion, the author mentioned that although we get better results with Xeon CPUs, but more CPUs need much more server space and cost. However, with Intel Xeon Phi or GPU approach we can put multiple of them in each server rack without modifying other parts of the server farm.

5.3. Examples

As we have analyzed in the case studies section, there are many uses for both NVIDIA and the Intel Xeon Phi architectures. Below, more examples of their use are cited to add context to the breadth of the usability of these platforms.
For NVIDIA, there are many examples of their use today as we can see with the case studies, and below, we discover some more real world examples of their use. One such example is listed in [21], in which NVIDIA is used for simulating currents and patterns in shallow water. This work has proven to be important because things like tsunami warnings and dam integrity rely upon this type of analysis, so the processing speed given by NVIDIA is a critical piece to this work. In [22], the world of virtual reality and how NVIDIA is used in this space are covered. This paper discusses many uses for NVIDIA GPUs in the VR space, from the carmaker Audi using it for their virtual showroom to Hollywood movies like *The Martian* and even surgical companies using them for training and preparation tools. As we can see, in addition to the case studies that were dissected, there are many everyday applications for NVIDIA GPUs.

For Intel Xeon Phi, we have a similar analysis, as there are numerous practical applications for the coprocessor architecture. One such example [23] is an application using Intel Xeon Phi Processors for calculating the states of scientific materials. The computing power required for this application is quite large because thousands of atoms are needed to complete the study and the size of the calculation is usually on the order of the number of atoms used in the study to the fourth power, which can become incredibly large. Another such example [24] is a code for calculations of nuclear objects, specifically a representation of a many-body (or n-body) problem. The amount of data worked on for models like this can be trillions of no-zero data points stored in a matrix data structure, so it is easy to see why huge computing power is needed for this problem. As we can see, in addition to the case studies that were dissected, there are many everyday applications for Intel Xeon Phi Processors and Coprocessors.

5.4. Discussions and Analysis

As we have analyzed, the history of the GPU and Intel's Xeon Phi have developed independently into important computing platforms for large data computations. Firstly, we compared NVIDIA and Xeon Phi rigorously in our case studies section, and found that there are numerous differences between the two architectures and also numerous practical benchmarks and applications that can be used to compare the two. We find that although the two architectures are different, they each have their own unique characteristics that make them an effective tool for

different purposes. In general, we find that Intel Xeon Phi processors are used quite often when large datasets are involved in a tabular form and that NVIDIA is used often with image processing and other forms. Because of this and the propensity of the world to move toward more virtual reality (VR) technology, it seems that NVIDIA may be better aligned to the future computing needs of the world. Further analysis and other replications and peer reviews must be done if we are to say unequivocally that this finding is always true. We discussed the evolution of GPUs and traditional processors and found that the two types of architecture evolved separately and in a way such that they can both be viable options for today's problems.

6. Project Steps

Below, this section describes the steps and the details of the study.

6.1. Phases and Efforts

Our research developed over many phases. Initially, we determined our research questions, followed by independent research by each individual. We updated our work at regular intervals with summary write-ups and presentations to the class. We later aggregated our prior and current work into this single document.

6.2. Research Methods used by the Project

The research methods employed for this project were exclusively literature survey.

6.3. Project Strengths and Limitation

We feel that out project benefits from a longitudinal approach, with cross section analysis at various point in time, as opposed to a single comparison at a particular date.

A limitation of our approach is that it tends to limit the depth of analysis, due to the multiple cross section analyses. If were to go into greater depth at each cross section, we would have created an unwieldy document excessive in size.

7. Proposal for Future Work

We feel that a longitudinal approach with cross section analyses could benefit the examination and research of other fast moving technologies in future work. For a slight reduction in depth, the analysis facilitates a broader and more holistic view of such technologies, highlighting and examining their trends, and not only fixating on the here and now.

Even though we believe our approach is useful, it too requires a certain amount to time to analyze article and deduce trends. We feel that our analysis would have benefited from additional time to further our work. Given additional time, we feel that our research may have benefited by the actual use of NVIDIA GPUs and Xeon Phi co-processors to generate some of our own quantitative results.

8. Conclusion

In conclusion, we briefly covered the history for GPU's, from their earliest use in arcade games, when they were know as graphic accelerators, to their use in advanced scientific computing. NVIDIA's focus has transitioned from a throughput oriented GPU to one of energy efficiency.

CUDA is NVIDIA's C-language extension to facilitate general purpose programming on NVIDIA's GPUs. CUDA Cores are the heart of the GPU, acting analogously as a CPU for the GPU. NVIDIA has managed to continually increase the throughput of its GPUs, while at the same time reducing their energy consumption to per core levels less than that of their earliest Tesla models.

Creating an optimal and efficient application that works perfectly with Intel Xeon Phi and GPU is a difficult task. Application developers have to try hard to keep up with every new generation of hardware and make changes to their applications.

However as we saw in our case studies, none of the platforms are a clear winner in all application categories in terms of performance. Coprocessors can do better in applications with regular data access but get dwarfed by GPUs in irregular data access.
From another View since in most cases these hardware works in a server farm form and huge numbers, other considerations like cost, power and physical space can affect the choice between Intel Xeon Phi and GPU.

What we have done in an overarching sense, is undergone an intense literature review to answer our research questions which has led to some possible solutions but also mixed results. We find that Intel Xeon Phi processors are used quite often when large datasets are involved in a tabular form and that NVIDIA is used often with image processing and other forms, and due to real world demands, we must trust that real world applications are using the best tool for their specific application. This leads to a conclusion that with the advent of VR technology and the constant state of image processing today, NVIDIA seems to be better positioned for the future of computing, but as always it will depend upon the scenario. We discussed the evolution of these architectures as well and find how they relate to our conclusions made when comparing the two. Finally, CUDA Cores were explored and how they related to GPUs, and by proxy our two specific architectures.

9. References

1. https://en.wikipedia.org/wiki/Graphics_processing_unit
2. https://en.wikipedia.org/wiki/File:NvidiaTesla2075.JPG
3. https://en.wikipedia.org/wiki/File:Space_Invaders_flyer,_1978.jpg
4. Professional CUDA C Programming 1st Edition, John Cheng, Max Grossman, Ty McKercher, ISBN: 1118739329
5. https://www.acellera.com/maxwell-gtx980-molecular-dynamics-gm204-gpu/
6. https://en.wikipedia.org/wiki/Fermi_(microarchitecture)
7. http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf
8. https://en.wikipedia.org/wiki/Kepler_(microarchitecture)
9. https://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf
10. https://en.wikipedia.org/wiki/Maxwell_(microarchitecture)
11. https://international.download.nvidia.com/geforce-com/international/pdfs/GeForce_GTX_980_Whitepaper_FINAL.PDF
12. https://devblogs.nvidia.com/parallelforall/5-things-you-should-know-about-new-maxwell-gpu-architecture/
13. James Reinders, "An Overview of Programming for Intel Xeon processors and Intel Xeon Phi coprocessors", 2012.
14. Jim Jeffers and Jams Reinders, *The War: Intel Xeon Phi Coprocessor High Performance Programming, 2013*.
15. Andrey Vladimirov, Ryo Asai and Vadim Karpusenko, The *Parallel Programming and Optimization with Intel Xeon Phi Coprocessors*, 2015.
16. "Intel Xeon Phi Coprocessor Developer's Quick Start Guide," http://software.intel.com/en-us/articles/intel-xeon-phi- coprocessor-developers-quick-start-guide.
17. "Intel Xeon Phi Coprocessor (codename Knights Corner)," http://software.intel.com/en-us/articles/intel-xeon-phi- coprocessor-codename-knights-corner.
18. Teodoro, George, Tahsin Kurc, Jun Kong, Lee Cooper, and Joel Saltz. "Comparative performance analysis of Intel Xeon Phi, GPU, and CPU." arXiv preprint arXiv:1311.0378 (2013).
19. Halyo, V., Patrick 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." Journal of Instrumentation 9, no. 04 (2014): P04005.
20. Takayuki Aoki , Global Scientific Information and Computing Center, Tokyo Institute of Technology
21. Cuda Spotlight: Modeling the World in Real Time.

http://www.nvidia.com/content/cuda/spotlights/modeling-world-real-time.html

22. Estes, Greg. How Virtual Reality is Making the Leap Into Everyday Life. https://blogs.nvidia.com/blog/2016/01/05/virtual-reality/

23. Case Study: BerkeleyGW using Intel Xeon Phi Processors. https://software.intel.com/en-us/articles/case-study-berkeleygw-using-intel-xeon-phi-processors

24. Case Study: Many-Fermion Dynamics using Intel Xeon Phi Processors. http://www.nersc.gov/users/computational-systems/cori/application-porting-and-performance/application-case-studies/mfdn-case-study/