Noname manuscript No.
(will be inserted by the editor)

A First Look at Integrated GPUs for Green High-Performance Computing

T. R. W. Scogland, H. Lin and W. Feng
Dept. of Computer Science
Virginia Tech
{**tom.scogland, hlin2, wfeng**}**@vt.edu**

Received: date / Accepted: date

Abstract

The graphics processing unit (GPU) has evolved from a single-purpose graphics accelerator to a tool that can greatly accelerate the performance of high-performance computing (HPC) applications. Previous work evaluated the energy efficiency of discrete GPUs for compute-intensive scientific computing and found them to be energy efficient but very high power. In fact, a compute-capable discrete GPU can draw more than 200 watts by itself, which can be as much as an entire compute node (without a GPU). This massive power draw presents a serious roadblock to the adoption of GPUs in low-power environments, such as embedded systems. Even when being considered for data centers, the power draw of a GPU presents a problem as it increases the demand placed on support infrastructure such as cooling and available supplies of power, driving up cost. With the advent of compute-capable *integrated* GPUs with power consumption in the tens of watts, we believe it is time to re-evaluate the notion of GPUs being power-hungry.

In this paper, we present the first evaluation of the energy efficiency of integrated GPUs for green HPC. We make use of four specific workloads, each representative of a different *computational dwarf*, and evaluate them across three different platforms: a multicore system, a high-performance discrete GPU, and a low-power integrated GPU. We find that the integrated GPU delivers superior energy savings and a comparable energy-delay product (EDP) when compared to its discrete counterpart, and it can still outperform the CPUs of a multicore system at a fraction of the power.

Keywords: GPU, performance evaluation, power-aware computing, energy-efficient computing, green, embedded high-performance computing

This work supported in part by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program and NSF I/UCRC Grant IIP-0804155 with equipment support from the Air Force Research Laboratory.

T. R. W. Scogland
2202 Kraft Drive
Knowledge Works II Building

H. Lin
2202 Kraft Drive
Knowledge Works II Building
Room 2219

W. Feng
2202 Kraft Drive
Knowledge Works II Building
Room 2209

1 Introduction

Through the mid-2000s, performance improvements for the general-purpose processor, also known as the CPU, were driven by increasing clock frequencies (f) that had to be supported by correspondingly higher supply voltages to the processor (V). However, because the thermal design power (TDP) of such processors is directly proportional to the clock frequency f and the *square* of the supply voltage V, these higher frequencies and voltages led to increasingly higher TDPs, as exemplified by the Itanium™ processor at 130 watts. This excessive power density led many to "brag" about the fact that it approached the power density of a nuclear reactor, as noted in (Feng et al, 2002; Hsu and Feng, 2005).

Due in large part to the aforementioned power density, clock frequencies finally stalled around 3 GHz in the mid-to-late 2000s, and instead, the number of cores started doubling every 24 months. Thus, performance improvements would no longer come from increased clock frequencies but rather would come from harnessing the parallelism of multi-core and many-core processors. Though the stalling of clock frequencies and advances in process technology (65 nm → 45 nm → 32 nm) have kept the TDP at bay, despite the doubling of cores every 24 months, the TDP has resumed its upward march with the dual-core Intel® Xeon™ 7130 CPU at 150 watts and emerging graphics processing units (GPUs) for general-purpose computation such as the 1600-core AMD/ATI® Radeon™ 5870 at 188 watts, the 240-core

NVIDIA® GTX 280 at 236 watts, and the 512-core NVIDIA® Fermi at 295 watts.

Despite the arguably voracious power appetite of the GPU, its popularity continues to grow quite rapidly because it is a commodity solution that can deliver personal supercomputing to the desktop for everything from a common desktop operating system (Munshi, 2008) to scientific applications (Eastman and Pande, 2010; Sandes and de Melo, 2010; Zhang et al, 2010; Tan et al, 2009; Anandakrishnan et al, 2010; Hardy et al, 2009; Stone et al, 2007). However, the adoption of the GPU in *high-performance embedded environments* has been limited due to the aforementioned TDPs. In addition, future exascale computing environments face daunting power consumption issues, as detailed in (Kogge et al, 2008) — power issues that may be exacerbated by GPUs.

While the most common and most studied GPUs for general-purpose computation are large, dedicated, and discrete many-core processors with 240 to 3200 cores, their TDPs are similarly large. As a consequence, such discrete many-core GPUs *cannot* be used in embedded systems.

Recently, however, low-power GPUs have materialized for general-purpose computation as well. Such GPUs have been designed as part of the chipset of the system motherboard, i.e., an integrated GPU and system chipset, or hereafter referred to simply as an *integrated GPU*. This integrated GPU provides full support for general-purpose computational acceleration via CUDA™ (NVIDIA, 2010) but at a TDP that is an *order of magnitude better* (i.e., less) than a traditionally large, dedicated, and discrete many-core processor such as the NVIDIA GTX 280.

Historically, integrated GPUs have been designed to provide 2D and 3D rendering within a low power and energy budget, especially laptops and miniature PCs. Now that these integrated GPUs also have compute capabilities, can they deliver the energy efficiency of their larger brethren while maintaining the low-power characteristics that made them attractive in portable and embedded systems from the start? Our results show that integrated GPUs with low TDPs possess the capacity for higher performance than their power draw suggests and provide a compelling performance compromise between traditional CPUs and full-sized GPUs with a power draw lower than either.

The main contribution of this paper is a performance evaluation and demonstration of low-power integrated GPUs in enhancing performance and energy efficiency for green high-performance computing (HPC). Specifically, we compare the performance, power consumption, and energy efficiency for a set of scientific applications across three accelerated architectures: an 8-core CPU system, a 240-core discrete GPU, and a low-power integrated GPU. Our results show that the integrated GPU offers a comparable energy-delay product (EDP) and superior energy savings over a discrete GPU, while still delivering better performance than multicore CPUs at a fraction of their power consumption. These properties make compute-capable, integrated GPUs very promising for green HPC applications.

The remainder of the paper is organized as follows, we present a background on the NVIDIA GPU architecture and CUDA™as well as other related work in Section 2 and Section 3. Section 4 discusses the methodology we used for this study, followed by an evaluation of the energy consumption and performance trade offs of a modern multicore architecture and two popular versions of the GPU architectures in Section 5. Finally, Section 6 presents our conclusions and Section 7 our plans for future work.

2 Background

This section provides a brief overview of the CUDA architecture, programming model, and compute capability.

2.1 CUDA Architecture

The CUDA architecture consists of a set of multiprocessors, each of which contains 8 scalar processors. Examples of such GPU devices include the 240-core NVIDIA GTX 280 or the 16-core NVIDIA ION™GPU. Meaning that the NVIDIA GTX 280 has 30 multiprocessors and the NVIDIA ION GPU only 2.

While each core could execute a sequential thread, the cores generally execute in what NVIDIA calls single instruction, multiple-thread (SIMT) fashion. That is, all cores in the same group execute the same instruction at the same time, much like SIMD. Where SIMT differs is in its ability to transparently handle divergent instructions. In addition, rather than scheduling at the granularity of a thread (as a standard CPU would) or at the granularity of 8 threads (as one might expect from SIMD-like execution), CUDA schedules threads in groups of 32, known as warps, which execute as a single unit across *four* time steps on a multiprocessor.

The memory model is similarly different from standard system memory, but its structure is only tangentially relevant to the results in this paper. For more information on NVIDIA GPUs and their architecture in general, see the CUDA programming guide (NVIDIA, 2007).

2.2 CUDA Programming Model

Certain extensions are necessary to program the architecture described above within current programming languages. In the case of CUDA C was chosen and a set of language extensions have been released along with a compiler and various examples by NVIDIA.

The primary concept in CUDA is the *kernel*, which represents a set of computations to be run in parallel on the GPU, and is specified much like a function in C. A kernel is a grid of blocks arranged in a 1 or 2 dimensional array, which are each one- or two-dimensional arrays of threads. Threads in this case, while they are implemented differently, behave for consistency purposes the same as threads in a multicore or multiprocessor scenario. The block abstraction is useful in that all threads in a block are guaranteed to run on the same multiprocessor at the same time, allowing them to synchronize with one another and share data. Between blocks, however, there is no on-chip synchronization that is natively available.

2.3 Compute Capability

NVIDIA uses "compute capability" to represent versions of their hardware and to allow for comparisons between different chips which have the same underlying properties. The compute capability is effectively a version number for the CUDA features and behavior of compute-capable graphics cards. This is especially relevant to this evaluation because our integrated and discrete GPUs have different compute capabilities, 1.1 and 1.2, respectively. In a nutshell, this means that the discrete card has more registers per multiprocessor, allows more warps and threads to be active at a time per multiprocessor, and has greater support for atomic operations. In terms of the architecture and general functioning of the chips, they are still the same.

3 Related Work

Energy efficiency for general-purpose computing on GPUs remains a largely unexplored space. The most directly relevant previous work (Huang et al, 2009) evaluates the energy efficiency of a computationally intensive scientific application across a GPU and a multicore system with and without threading. They demonstrate that the GPU has high potential for energy efficiency for compute-intensive workloads due to its high performance, but leave the high instantaneous power of a GPU as an issue yet to be addressed.

In (Rofouei et al, 2008), the authors evaluate the energy efficiency of GPUs with an emphasis on energy cost and cost/performance. This paper focuses more on their LEAP server, which is the tool used to get "real-time" energy monitoring than on the GPU results themselves. That said, they do present a good evaluation of the cost/performance of GPUs and energy cost of instructions.

The earliest work on GPU power consumption is a tool called QSilver, a simulator designed for the analysis of performance characteristics of graphics hardware (Sheaffer et al, 2005). QSilver focuses on analyzing the graphics pipeline by intercepting OpenGL calls. One of the hardware improvements investigated is dynamic voltage and frequency scaling (DVFS) across multiple clock domains, which achieved an estimated power reduction of 10%.

Though little work has been done in direct relation to general-purpose computing on low-power GPUs, a great deal of effort has been expended on creating low-power embedded GPUs for graphics (Nam et al, 2007). They explore the use of logarithmic arithmetic for power efficiency and propose a significant rearrangement of the arithmetic units in the GPU to produce significant power and space savings.

More generally, high-performance computing (HPC) as a field has sought ways to lower its energy consumption for quite some time, especially with methods such as DVFS, e.g., the power-aware run-time system proposed in (Hsu and Feng, 2005). They propose an algorithm that monitors the percentage of the cycles spent on memory operations and decides when it is appropriate to lower the frequency of the processor such that the performance degradation, if any, is tightly bound. While we do not directly analyze DVFS in this paper, our results suggest that a similar methodology may be in use on the GPUs we study.

4 Methodology

In order to evaluate the effectiveness of integrated GPUs for energy-efficient and low-power HPC, we selected a set of candidate metrics and evaluated them across a diverse set of applications.

4.1 Energy Efficiency Metrics

Currently, the most commonly used energy-efficiency metric, as adopted by the Green500 List (Feng and Cameron, 2008), is *flops/watt*, where flops refers to floating-point operations per second. However, the number of floating-point operations is not applicable to all applications. Also, getting an accurate measure of the floating-point operations per run on each platform for a large number of applications is difficult and prohibitively time consuming. Another option considered was to use available implementations of the LINPACK (Dongarra, 1990) benchmark, but the available GPU implementations of LINPACK do not support cards with memory lower than 4GB, such as those we are evaluating.

An alternative metric for measuring energy efficiency is the *energy-delay product (EDP)*. The EDP is defined as the amount of energy consumed during the execution of a program multiplied by the execution time of the program. It has been a metric used in previous studies of GPU energy efficiency (Huang et al, 2009; Hamano et al, 2009).

This EDP metric, and more generally, ED^n, where n is an integer, is commonly used in circuit design. However,

as noted in (Hsu et al, 2005), the ED^n product emphasizes performance over energy, particularly as n increases.

A metric of increasing interest is the amount of computational work completed per joule. The question here becomes "What constitutes work?" In (Kapasi et al, 2003), work completed per joule is defined as operations per joule. In contrast, SPECPower (Lange, 2009) defines work per joule as "server-side Java operations per joule." However, our set of applications do not use that unit of work, nor do they share a common unit of work other than instructions, and those are debatable since different instructions may be chosen by the compilers for each platform. As such, we decided to treat a complete run of a given application as a single unit of work and report the total energy consumed per application run.

4.2 Power Measurement

With the purpose of this study centered around the power consumption and energy efficiency of GPUs, we seek to investigate the static power vs. the dynamic power of GPUs and the EDP of the computing platforms under different application workloads, respectively. However, directly measuring the power consumption or energy efficiency of the GPU alone is extraordinarily difficult. For example, the ION GPU is not physically separable from the rest of the system, not to mention that it includes more than just the processing element within itself. Another issue is that for the GPU, the static power of just the GPU is not necessarily useful since other components are necessary for the GPUs to function. As such, we measure the *idle and loaded power* of the entire system rather than the static and dynamic power of the GPU, respectively. The idle power is the power of the whole system when running but not loaded. The loaded power is the same as the idle power except with a computational load running on the system in question.

4.3 Applications

This study aims to evaluate the performance and energy efficiency of integrated GPUs for high-performance scientific applications. To this end, we run four benchmark applications on multicore and CUDA-accelerated GPU systems. We chose these four applications, each representing a Berkeley computational dwarf (Asanovic et al, 2006), in order to cover a range of common scientific application signatures — compute intensive, synchronization intensive, communication intensive, and fine-grained parallelism.

For our compute-intensive benchmark, we use GEM (Fenley et al, 2008; Gordon et al, 2008), a molecular dynamics application that calculates the electrostatic potential along the surface of a macromolecule. The other three applications are from the open-source Rodinia Benchmark Suite (Che et al, 2009). We describe these test applications in detail below.

4.3.1 GEM

GEM is a molecular dynamics application that visualizes the electrostatic potential along the surface of a macromolecule. We developed accelerated versions of the application in conjunction with our byiophysics colleagues and have CUDA and pthreads versions along with the original serial version. GEM represents the *n-body dwarf* (Asanovic et al, 2006), but while most n-body applications are all-pair computations over a single set of particles, GEM performs all-pair computations between two different sets. The input of GEM consists of a list of all atoms in the structure, and a list of surface points for which the potential is of interest. As such, the computational complexity is $O(n * m)$ where n is the number of atoms and m is the number of surface points. Like most n-body applications, approximations can be applied to reduce the computational complexity. However, we eschew such approximations in favor of accuracy for the implementation used in our experiments. Since different pairwise computations are independent, GEM requires no synchronization between different threads and thus can scale almost linearly across a large number of compute units given enough input. In addition, it is capable of coalescing almost every memory access. As a result, GEM can efficiently utilize GPU resources to deliver high-performance speedups over CPU.

4.3.2 Speckle Reducing Anisotropic Diffusion (SRAD)

SRAD (Wilson and Gallant, 1998) is a smoothing algorithm designed to clean-up speckles on an image without materially altering the important features. It is primarily used in sonic and radar imagery applications where image noise is a common issue. Its application signature represents the *structured grid* computational dwarf. Inputs and parameters are the same reference images as used for the runs in the original Rodinia paper (Che et al, 2009). SRAD executes a number of iterations, each of which involves two kernel launches, and between which memory is copied to and from the GPU. When the program is set to iterate enough times to take over a second, the minimum for an accurate power measure on our equipment, the performance depends greatly on kernel launch time.

4.3.3 K-Means

K-Means (Kaufman and Rousseeuw, 2005) is a popular clustering algorithm that identifies correlated observations by grouping them into clusters by their locations. Its application signature conforms to the *dense linear algebra* com-

	Kernel launches	Explicit synchronization between launches	Per launch data transfer
GEM	78	No	None
K-Means	37	Yes	Large
SRAD	4000	Yes	Small
NW	255	No	None

Table 1 Application properties factored into our decision to use these applications.

putational dwarf. The algorithm picks a set of starting centroids for the clusters, finds all observations for which a given centroid is the nearest, and then computes a new centroid based on the location of those observations. K-Means is a communication-intensive algorithm, moving data to and from the GPU between kernel launches and explicitly synchronizing the GPU with the CPU after every kernel. This iterative process continues until convergence is achieved.

4.3.4 Needleman-Wunsch (NW)

Needleman-Wunch (NW) (Needleman and Wunsch, 1970) is one of a set of alignment algorithms commonly used in DNA sequence analysis. It is based on the *dynamic programming* dwarf. All potential pairings of characters in the two input sequences are organized in a 2-D matrix, which is filled with scores representing the quality of the match ending at that location. Once the matrix is filled, a traceback is performed to find not only the highest score of the matches but also the match itself, with insertions and deletions included. Needleman-Wunsch was initially chosen to fill the role which SRAD ended up taking, but the implementation in the Rodinia suite is made coarse-grained by blocking of the matrix into chunks, reducing the number of kernel launches necessary to complete the matrix filling process. Even so, NW still requires a significant number of kernel launches to synchronize between computation of the aforementioned chunks, not to mention a significant amount of intra-block synchronization in the kernel itself.

4.4 Application Characteristics

Table 1 shows three characteristics — kernel launches, explicit synchronizations between launches, and per-launch data transfer — that capture the diversity of the above applications when executing on GPUs. The first characteristic indicates the number of kernel launches during a single run of the application. The second characteristic refers to whether an explicit barrier synchronization is called on the CPU side between launches of consecutive kernels. While this explicit synchronization prevents the overlapping of loading the next kernel to be run with execution of the previous kernel and increases the synchronization time, it is necessary to ensure safe data-transfer to and from the CPU card between kernel launches for applications such as K-Means and SRAD. Finally, the third characteristic gives a coarse-grained characterization of the amount of data transferred between kernel launches. As shown in Section 5, these characteristics will measurably impact the performance, power and energy efficiency of different applications on different platforms.

5 Evaluation

To evaluate the performance and energy efficiency of integrated GPUs for parallel computing, we conduct a comparative study between three systems: (1) an integrated GPU system, (2) a modern multicore system and (3) a discrete-GPU-accelerated desktop supercomputer.

5.1 Experimental Setup

To conduct our experimental tests, we used two commodity desktop computers that are commonly available in today's market. The first desktop computer is built for the purpose of high-performance computing (HPC). It consists of two 2.0-GHz Intel Xeon E5405 quad-core CPUs and 4GB of RAM. Graphics and compute acceleration are provided by an NVIDIA GTX 280 GPU, which has 1GB of graphics memory and 30 multiprocessors (240 stream cores) at compute capability 1.2 and a clock rate of 1.3 GHz. This configuration is similar to a Colfax CXT personal supercomputer (Colfax, 2010). The second computer is a Zotac miniature desktop PC with a 1.6-GHz Intel Atom 230 dual-core CPU, 3GB of RAM, and an NVIDIA MCP79 chipset. The MCP79, better known as a GeForce 9400, contains an integrated ION graphics chip with 256MB of graphics memory, 2 multiprocessors (16 stream cores) at compute capability 1.1 and a clock rate of 1.1 GHz.

The first computer is used for collecting results for both the multicore and the discrete-GPU experiments, and the GPU is removed for the multicore and serial CPU experiments. The integrated-GPU results are collected from the second computer.

In all experiments, we use a WattsUp? PRO ES power meter, in-line with the system under test and connected to a separate measurement-logging PC. We configure the power

meter to collect measurements at the finest granularity supported by the power meter, i.e., one second. Basic DVFS mechanisms, i.e., *cpufreq's ondemand governor* were available, but ineffective for the CPU as this model does not support Intel's Enhanced Speedstep interface. The GPU on the other hand does have some working DVFS in the form of NVIDIA's built-in PowerMizer.

All experiments are repeated five times, and the average numbers are reported for performance, power, and energy results. The experimental results for GPU measure the main computational kernel of each application, including the memory allocations and transfers to and from the GPU. For a fair comparison, the CPU results include only the computational kernel as well. In other words all operations outside the computational kernel, including disk I/O, are excluded in both CPU and GPU versions. For brevity, we use *Xeon serial*,[1] *Xeon SMP*, *ION*, and *GTX 280* to refer the the CPU serial, CPU multicore, integrated GPU, and discrete GPU results, respectively.

5.2 Performance

Discrete GPUs have become recognized for their high performance when running general-purpose applications. Integrated GPUs, on the other hand, are traditionally designed for low-demand graphics processing in business workstations and servers or for low-power environments such as laptops and miniature desktops. Newer models of integrated GPUs have begun to more substantially increase in performance and gained the capabilities necessary to run general-purpose applications. However, their computational capabilities for HPC have not been studied.

Figure 1 shows the performance of running four benchmark applications on the test platforms. In order to make the results more visually comparable, the figure shows the speedup over the Xeon serial version, which is almost universally the slowest. Table 2 shows the same results in raw seconds for comparisons within an application. One important observation from Figure 1 is that the ION chip can clearly accelerate the performance of the test applications — except for K-means, where ION performs slightly worse than Xeon serial. In particular, ION delivers 30-fold and 13-fold speedups over Xeon serial for the GEM and NW applications, respectively. For these two applications, ION also outperforms Xeon SMP, a modern multicore system. The slightly worse performance of K-means on ION is due to the large amounts of memory transfers between CPU and GPU at each kernel launch. This is also indicated by the relatively lower speedup of K-means on the GTX 280 as compared to the other applications. It is worth noting that the host-device memory bandwidth is much higher on GTX 280 than on ION despite the host and device sharing physical memory in the case of ION.

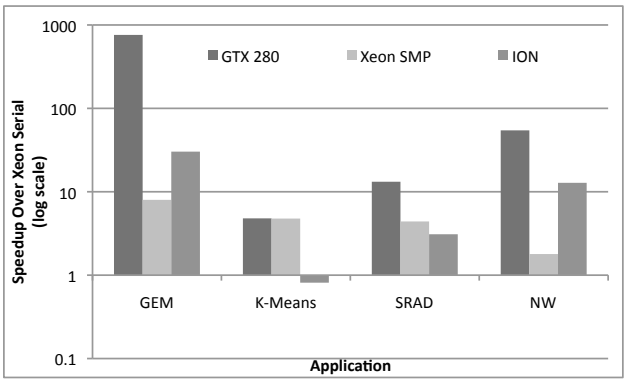


Fig. 1 Speedup of each version over Xeon serial

Time (seconds)	GEM (capsid)	K-Means	SRAD	NW
Xeon serial	63,029.5	7.9	788.5	377.0
Xeon SMP	7,878.7	1.7	179.0	210.5
GTX 280	82.9	1.7	59.8	6.9
ION	1,998.5	9.7	254.9	29.5

Table 2 Run time in seconds for all platforms and applications

Nonetheless, GTX 280 significantly outperforms ION for all the applications, suggesting the obvious performance advantage of discrete GPUs over the current generation of integrated GPUs. Specifically, GTX 280 is 4 to 25 times faster than ION for the test applications.

The Needleman-Wunsch (NW) results are aberrant in that the 8-core run of NW shows little improvement over the serial version. Our chosen implementation of the CPU version, from the Rodinia suite, appears to scale poorly. In fact, it only keeps up to four of the eight cores utilized during a run, resulting in only a slight performance speedup, as shown in Figure 1. In the future, we intend to investigate more optimized CPU versions of the NW application.

[1] In the *Xeon serial* case, the tests are performed on the multicore system with all but one core, i.e., the active core, being idle. Though ideally we would like to report the power numbers consumed by a *single CPU core* for the *Xeon serial* results, such power numbers are difficult to measure given that four CPU cores are tightly integrated on a single CPU processor. Note: Running our tests with the second CPU physically *removed* would reduce the power consumption consistently by 17 watts (W), i.e., 118.6 W vs. 135.7 W at idle with similar differences when at load. We chose *not* to use this latter configuration as typical end users would not normally remove a CPU from their systems.

5.3 Power

As noted in Section 4.2, we evaluate the idle and loaded power of our test systems rather than the static and dynamic of the GPU, respectively.

5.3.1 Idle Power

We first measure the idle power for the three test systems, as shown in Table 3. Most noticeably, the ION system consumes significantly less power at idle than the other two systems. The idle power of ION is a mere 20 watts, less than 1/6 of the power of the Xeon SMP system and 1/9 of a GTX 280 accelerated system. The difference between the idle power of Xeon SMP and GTX 280 implies that the static power of a GTX 280 GPU card alone is 51 watts. Recall that the GTX 280 system idle power also includes the Xeon multicore CPU because we cannot remove it from the tested system.

	Idle Power (Watts)
Xeon SMP	135.7
GTX 280	186.4
ION	20.1

Table 3 Static power for the three test platforms.

5.3.2 Loaded Power

While traditional discrete GPUs can deliver much higher performance than CPUs, it comes at the cost of significantly higher power consumption. For example, a top-of-the-line discrete NVIDIA GPU e.g., GTX 280, is rated with a thermal design power (TDP) of 236 watts, not to mention the other components necessary to drive it. The next generation of NVIDIA® discrete GPUs, known as Fermi, is reported to have a TDP of 295 watts. To put this into perspective, the TDP for the Intel quad-core Xeon E5405 is only rated at 80 watts while the ION GPU has a rated TDP of only 12 watts. Of course, these are only *reported* dissipation measures and represent only the compute device itself; thus, in this section, we present real-world power results for each device.

Figure 2 plots the average wattage across a complete run of the computational kernel of each benchmark on the tested platforms. *Overall, the power consumption of the ION version is significantly lower than the others.* Specifically, the loaded power of the ION system is 8.8 to 10.0 times lower than the Xeon SMP version for all tested applications. The power gap between the ION system and the GTX 280 system is even larger, with a 9.9 to 12.7-fold difference, depending on the application. The larger variance in power

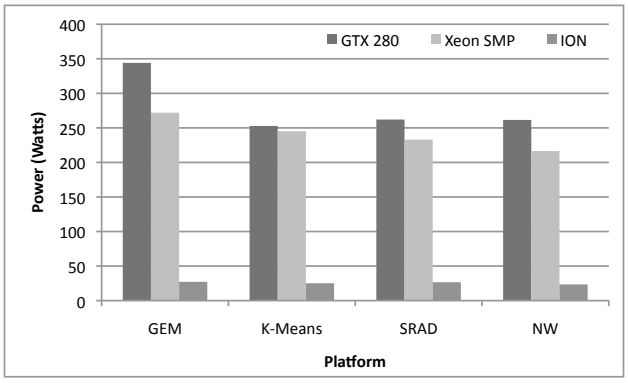


Fig. 2 Power results for the four benchmarks for all platforms.

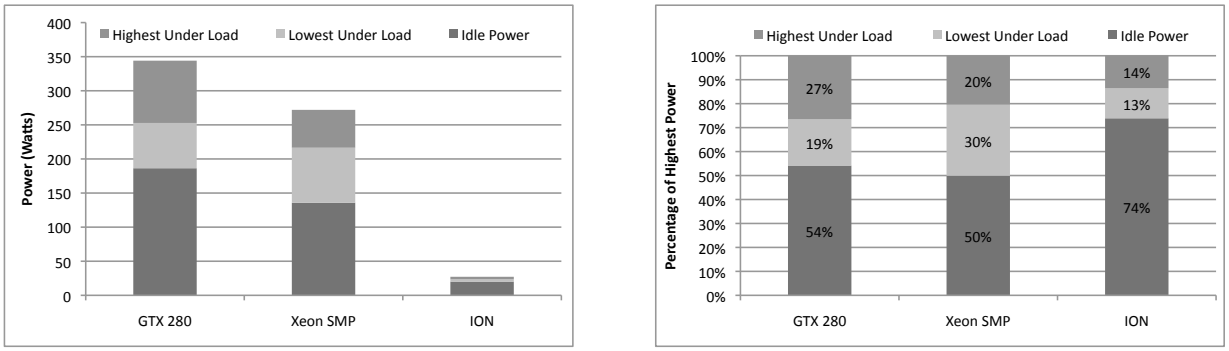
consumption for the GTX 280 system is likely caused by the GPU DVFS policy (i.e., NVIDIA's PowerMizer in this case) reacting differently to various application signatures.

To further characterize the dynamic power behavior, Figure 3(a) shows the difference between the lowest and the highest loaded power relative to the idle power for each platform. The lowest power refers the power drawn by the application which draws the least power out of those tested, and the highest power is similarly defined. The loaded power range of the ION system is an order of magnitude smaller than the other two systems. That is, the increase from the idle power to the highest loaded power on ION is 7.1 watts, while this increase is 157.7 watts and 136.3 watts on the GTX 280 and Xeon SMP, respectively. Figure 3(b) plots the dynamic power increase in percentage of the peak power drawn from each platform. The idle power is 74% of the peak power on the ION system, whereas this percentage is about 50% for the others.

5.4 Energy Efficiency

As previous work has shown (Huang et al, 2009), despite their high power requirements, GPUs can be energy efficient for highly parallel codes due to the massive speedups they can achieve. Even so, we have found that low power consumption is also desirable for many of the environments in which energy efficiency is desirable. As a result, we evaluate whether the high energy efficiency that we observed on discrete GPUs is also applicable to the compute-capable integrated GPUs.

We first show the energy consumption of each of the four tested applications across different platforms in Figure 4 with raw joules listed in Table 4. As the energy consumption is largely different for each experiment, for a cross-experiment comparison, we report the relative energy consumption ratios with regard to the energy consumed by running the serial version of the application on the Xeon machine. Specifically, suppose the power consumption of the



(a) Power range for each platform

(b) Percentage dynamic power

Fig. 3 Power results for all platforms, both absolute and percentage representations

serial version on the Xeon machine is E_s, and the energy consumption of the parallelized version of the application on one of the platforms is E_p, Figure 4 plots E_s/E_p on a log scale. In other words, energy "speedup" numbers are reported. Higher "speedup" means lower energy use.

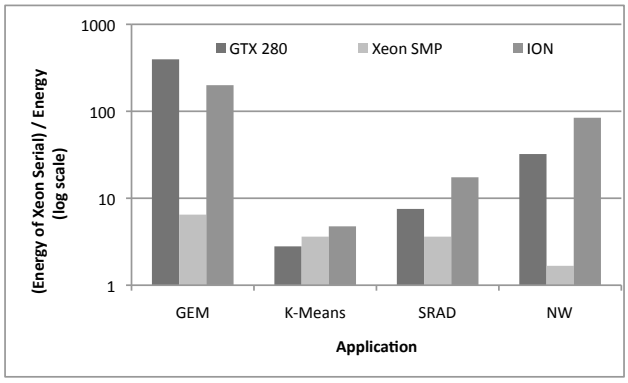


Fig. 4 Energy results for all applications

Energy (joules)	GEM (capsid)	K-Means	SRAD	NW
Xeon serial	11,301,185.4	1,167.4	118,124.2	58,328.1
Xeon SMP	1,743,552.9	322.7	32,648.9	34,904.2
GTX 280	28,544.7	417.4	15,671.5	1,809.6
ION	18,973.9	2,394.52	6,779.9	692.2

Table 4 Energy in joules for all platforms and applications

The ION platform consumes the least energy for all applications except GEM on the GTX 280. On the other hand, the ION platform consumes 30 times less energy than the multicore platform for GEM.

ION also exhibits a significant advantage in energy consumption for the NW application when compared to running on Xeon SMP. As mentioned earlier, the OpenMP version of NW on the CPU does not appear to scale well, which may be the primary cause.

In summary, the above observations suggest that the integrated GPU demonstrates great promise for green HPC, especially when power consumption is a major concern. The results also show that while the discrete GPU consumes more power, it is more energy efficient than its integrated counterpart for highly compute-intensive applications.

Figure 5 presents the energy efficiency of each platform by plotting the "speedups" calculated with respect to EDP. The higher the value, the more energy efficient. The GTX 280 GPU is the best in all cases but K-Means, where the 8-core Xeon SMP performs slightly better than the GTX 280. This is due to the fact that the EDP metric favors high performance over lower energy consumption. Even so, the ION platform stays competitive by outperforming the Xeon SMP platform for all but K-Means. The reason why the EDP value is not as good on GPUs, as discussed in Section 4.3, is that the K-Means application requires a large amount of data transfer to and from the GPU device repeatedly during a run. Such data transfers prevent the GPU processing power from being fully utilized.

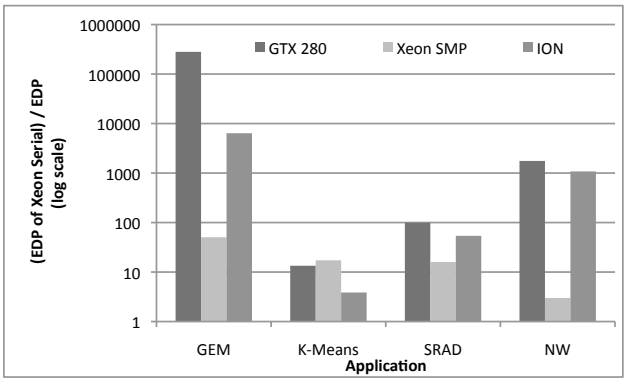


Fig. 5 Improvement in EDP over Xeon serial

6 Concluding Remarks

The new generation of compute-capable integrated GPUs shows great promise for green high-performance computing (HPC). We have demonstrated that when both energy efficiency and low power consumption are important, an integrated GPU can be an attractive option as much as a discrete GPU can be for energy-efficient computing without power constraints. In fact, for the sample of applications we tested, the integrated GPU used the least energy for all tested applications. We believe this is a new opportunity for compute acceleration in clusters and other deployments, where energy efficiency is desirable and discrete GPUs are impractical or impossible to use because of power constraints. We also reinforce the results presented in (Huang et al, 2009) by concluding once again that discrete GPUs can be exceptionally energy efficient, especially with regard to the *performance-per-watt* metric.

7 Future Work

In the future, we intend to evaluate a larger number of platforms in this space as well as provide comparisons with other types of accelerators for embedded HPC. For example, we seek to investigate the effectiveness of embeddable GPUs like ION for jobs that usually employ FPGAs. Beyond that, we will also examine new offerings from AMD such as the ATI Radeon 5450 to get a better sampling of the low-power GPU space.

As part of the above process, we will also explore using OpenCL for fair comparisons between CPUs and GPUs. In addition to further testing, we also hope to investigate the clustering potential using these low-power accelerators in support of greener large-scale HPC solutions.

Acknowledgements We would like to thank the NSF Center for High Performance Reconfigurable Computing (CHREC) for their support, as well as the Air Force Research Laboratory (AFRL) for their support and for access to equipment used in this work.

Thanks also to Timo Minartz and the anonymous reviewers for their constructive feedback to improve our manuscript.

References

Anandakrishnan R, Scogland TR, Fenley AT, Gordon JC, chun Feng W, Onufriev AV (2010) Accelerating electrostatic surface potential calculation with multi-scale approximation on graphics processing units. Journal of Molecular Graphics and Modelling 28(8):904 – 910, DOI DOI:10.1016/j.jmgm.2010.04.001, URL http://www.sciencedirect.com/science/article/B6TGP-4YVY7HW-1/2/ba476f86f0044232c55666d9e9275b2a

Asanovic K, Bodik R, Catanzaro B, Gebis J, Husbands P, Keutzer K, Patterson D, Plishker W, Shalf J, Williams S, et al (2006) The landscape of parallel computing research: A view from berkeley. EECS Department, University of California, Berkeley, Tech Rep UCB/EECS-2006-183 pp 2006–183

Che S, Boyer M, Meng J, Tarjan D, Sheaffer J, Lee S, Skadron K (2009) Rodinia: A benchmark suite for heterogeneous computing

Colfax (2010) Colfax CTX Personal Supercomputer. http://www.colfax-intl.com/ms_Tesla.asp?M=100

Dongarra J (1990) The LINPACK benchmark: An explanation. In: Supercomputing, Springer, pp 456–474

Eastman P, Pande V (2010) OpenMM: A Hardware Abstraction Layer for Molecular Simulations. Computing in Science and Engineering

Feng W, Cameron K (2008) The Green500 List: Encouraging Sustainable Supercomputing. IEEE Computer 40(12):50–55

Feng Wc, Warren M, Weigle E (2002) The Bladed Beowulf: A Cost-Effective Alternative to Traditional Beowulfs. In: IEEE International Conference on Cluster Computing (IEEE Cluster 2002), Chicago, Illinois

Fenley AT, Gordon JC, Onufriev A (2008) An analytical approach to computing biomolecular electrostatic potential. I. Derivation and analysis. The Journal of Chemical Physics 129(7):075,101, URL http://scitation.aip.org/getabs/servlet/GetabsServlet?prog=normal\&id=JCPSA6000129000007075101000001\&idtype=cvips\&gifs=yes

Gordon JC, Fenley AT, Onufriev A (2008) An analytical approach to computing biomolecular electrostatic potential. II. Validation and applications. The Journal of Chemical Physics 129(7):075,102, URL http://scitation.aip.org/getabs/servlet/GetabsServlet?prog=normal\&id=JCPSA6000129000007075102000001\&idtype=cvips\&gifs=yes

Hamano T, Endo T, Matsuoka S (2009) Power-aware dynamic task scheduling for heterogeneous accelerated clusters

Hardy DJ, Stone JE, Schulten K (2009) Multilevel summation of electrostatic potentials using graphics processing units. Parallel Computing 35(3):164 – 177

Hsu Ch, Feng Wc (2005) A power-aware run-time system for high-performance computing. In: ACM/IEEE SC2005: The International Conference on High-Performance Computing, Networking, and Storage, Seattle, Washington

Hsu Ch, Feng Wc, Archuleta JS (2005) Towards Efficient Supercomputing: A Quest for the Right Metric. In: 1st IEEE Workshop on High-Performance, Power-Aware Computing (in conjunction with the 19th International Parallel & Distributed Processing Symposium), Denver, Colorado

Huang S, Xiao S, Feng W (2009) On the energy efficiency of graphics processing units for scientific computing. In: IPDPS '09: Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing, IEEE Computer Society, Washington, DC, USA, pp 1–8, DOI http://dx.doi.org/10.1109/IPDPS.2009.5160980

Kapasi UJ, Rixner S, Dally WJ, Khailany B, Ahn JH, Mattson P, Owens JD (2003) Programmable stream processors. Computer 36:54–62, DOI http://doi.ieeecomputersociety.org/10.1109/MC.2003.1220582

Kaufman L, Rousseeuw P (2005) Finding Groups in Data: An Introduction to Cluster Analysis. Wiley's Series in Probability and Statistics. John Wiley and Sons, New York

Kogge P, Bergman K, Borkar S, Campbell D, Carlson W, Dally W, Denneau M, Franzon P, Harrod W, Hill K, et al (2008) Exascale computing study: Technology challenges in achieving exascale systems. Washington, DC: DARPA Information Processing Techniques Office 28

Lange KD (2009) Identifying Shades of Green: The SPECpower Benchmarks. Computer 42:95–97, DOI http://doi.ieeecomputersociety.org/10.1109/MC.2009.84

Munshi A (2008) OpenCL. SIGGRAPH

Nam BG, Lee J, Kim K, Lee SJ, Yoo HJ (2007) A low-power handheld GPU using logarithmic arithmetic and triple DVFS power domains. In: GH '07: Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp 73–80

Needleman S, Wunsch C (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of molecular biology 48(3):443–453

NVIDIA (2007) Compute Unified Device Architecture Programming Guide. Nvidia, June

NVIDIA (2010) NVIDIA CUDA. http://developer.nvidia.com/object/cuda.html

Rofouei M, Stathopoulos T, Ryffel S, Kaiser W, Sarrafzadeh M (2008) Energy-Aware High Performance Computing with Graphic Processing Units. In: Workshop on Power Aware Computing and System

Sandes EFO, de Melo ACM (2010) CUDAlign: using GPU to accelerate the comparison of megabase genomic sequences. In: PPoPP '10: Proceedings of the 15th ACM SIGPLAN symposium on Principles and practice of parallel programming, ACM, New York, NY, USA, pp 137–146, DOI http://doi.acm.org/10.1145/1693453.1693473

Sheaffer J, Skadron K, Luebke D (2005) Fine-grained graphics architectural simulation with Qsilver. In: ACM SIGGRAPH 2005 Posters, ACM, p 118

Stone JE, Phillips JC, Freddolino PL, Hardy DJ, Trabuco LG, Schulten K (2007) Accelerating molecular modeling applications with graphics processors. Journal of Computational Chemistry 28(16):2618–2640

Tan G, Guo Z, Chen M, Meng D (2009) Single-particle 3d reconstruction from cryo-electron microscopy images on GPU. In: ICS '09: Proceedings of the 23rd international conference on Supercomputing, ACM, New York, NY, USA, pp 380–389, DOI http://doi.acm.org/10.1145/1542275.1542329

Wilson J, Gallant J (1998) SRAD: a program for estimating radiation and temperature in complex terrain. Trans GIS _

Zhang Y, Cohen J, Owens JD (2010) Fast tridiagonal solvers on the GPU. In: PPoPP '10: Proceedings of the 15th ACM SIGPLAN symposium on Principles and practice of parallel programming, ACM, New York, NY, USA, pp 127–136, DOI http://doi.acm.org/10.1145/1693453.1693472