A Survey on Solutions for Improving Energy-Efficiency in GPUs

Ehsan Sharifi Esfahani
Department of Computer Science
Faculty of Science
Vrije University of Amsterdam
Amsterdam, Netherlands
e.sharifiesfahani@student.vu.nl

Abstract: **The high energy consumption of Graphics Processing Units (GPUs) is nowadays an important issue in high-performance computing systems. It leads to high energy-related costs as well as environmental consequences. Although energy efficiency has been studied a lot in CPUs, this topic is in its early stage of investigation in GPUs. For these reasons, it is fundamental to work on the development of more energy-efficient parallel programs. In this literature study, we review the motivation and the related challenges of reducing energy consumption in GPUs. Then, we present several metrics used for evaluating energy-efficiency and different methods to measure energy consumption. In addition, we summarize and classify several approaches for enhancing energy consumption in this context.**

Key words: Energy-efficiency, GPU, Power model, Energy consumption, Power consumption.

I. INTRODUCTION

Accelerators, such as Graphic Processing Units (GPUs), are becoming increasingly common in the field of massively parallel computing. Figure 1 demonstrates the number of supercomputers equipped with GPUs and many-core CPUs in the TOP500 list since June 2010 [1]. Obviously, there is an upward trend of using GPUs as accelerators to build more powerful machines. It is because of the fact that GPUs can provide us a high-performance computing environment with better energy-efficiency than CPUs in massively parallel applications [14].

Contrary to the trend in GPU popularity, the popularity of many-core CPUs, such as Intel Xeon Phi, has been gradually decreasing since September 2015. This is compatible with the conclusion of the authors in paper [9] that one of the main reasons to build supercomputers with GPUs instead of many-core CPUs is energy-efficiency since hybrid CPU-GPU systems are more energy-efficient than other solutions.

There is always a high demand for more performance in High-Performance Computing

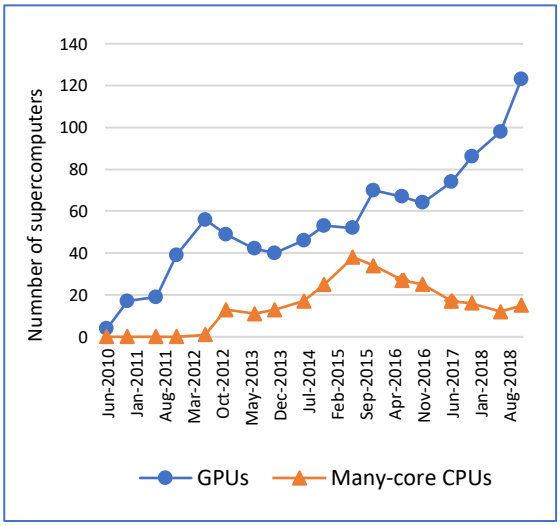

Figure 1: Trend of using GPUs and many-cores as accelerators in TOP500 list since June 2010

(HPC) systems. High energy consumption and running costs are two of the main challenges to design and implement such systems [3]. GPUs are significantly energy-efficient as opposed to CPUs. A x86-based CPU, for instance, consumes roughly 86 times more energy for a floating-point operation than a GPU [28]. Despite of this fact, GPUs are still consuming a large amount of energy in every computer system. The best machine in the TOP500 on November 2018, the Summit supercomputer, consumes about 13MW in its peak power consumption [13] which would cost about 9.36 million Euros per year, according to Dutch electricity bill in 2017 [4]. Moreover, in Summit, much more than half of total energy in every compute node is consumed by 6 NVIDIA Tesla V100 GPUs, each with peak power consumption of 300 watts [10], for a total of about 1800 watts [1]. In systems with such high-power consumption, even a small reduction in power consumption can induce significant environmental and financial savings.

In this study, we present a survey on solutions for improving energy efficiency in GPUs. To do so, we, firstly, make a motivational observation about the importance of energy-efficiency and characterize the associated challenges in the section

II and III. Then, in the section IV, we discuss the prevalent metrics used for evaluating energy-efficiency in GPUs and generalize a common metric from server environments to GPUs platforms.

The rest of our survey is represented as follows: Section V describes the different methods to measure power consumption in GPUs. In section VI, we survey different approaches and studies related to energy-efficiency and classified them based on our proposed classification. Section VII proposes different two-class taxonomies for improving energy-efficiency solutions. The conclusion is presented in the last section.

II. MOTIVATION

Energy-efficiency in GPUs has recently attracted a lot of attention among researchers since this topic is in its early stage of investigation. [5]. Rather than enhancing absolute performance, the main challenge of several GPU applications is improving their energy-efficiency. High power consumption is a critical issue in embedded systems with limited source of energy or applications with large scale. As an example, the estimated power consumed by the high-performance computer center in the Square Kilometer Array project, as a large radio telescope of the future, is in the order of 50MW [45]. The high-power consumption, which leads to high total life cycle cost, along with problems related to providing the infrastructure of power supplier are two main issues in this project [22]. Challenges of this kind require the development of more energy-efficient solutions in HPC platforms.

In addition, high energy consumption causes more heat dissipation and increasing hardware temperature has an adverse effect on hardware lifetime [12]. Anderson el al demonstrated that 15 degrees increment in temperature makes the reliability of the system decline up to 50 percent, [11]. As a result, when the amount of energy consumed by GPUs is decreased, the heat dissipated by the hardware decreases and less energy is required for cooling. For instance, up to 50 percent of electricity expenditure in data centers can be consumed by cooling systems [12]. It should be noted that GPUs are becoming the main part of energy consumption in these systems [24] [1].

Furthermore, reducing energy consumption of current machines will make it possible to build future exascale supercomputers. Keckler et al stated that [28]:
"The reasonable power envelope for future supercomputers has been projected to be 20 MW. The supercomputing community is now aiming to design exascale (10^{18} operations/second) systems. To build such a system within 20 MW requires an energy efficiency of approximately 20 pico joules (pJ) per floating point operation. Future servers and mobile devices will require similar efficiencies."

As another motivation, a higher energy consumption incurs tremendous energy-related costs as well as environmental consequences. For example, CO2 emission from data centers worldwide is estimated to increase from 80 Megatons (MT) in 2007 to 340 MT in 2020, more than double the amount of current CO2 emission in the Netherlands (145 MT) [15]. Lower scalability is another adverse effect of high energy consumption in GPUs since we cannot increase the number of nodes in a supercomputer within a specific power budget. These motivation has led to more efforts on energy-efficiency in GPUs.

III. CHALLENGES

Understanding power management challenges in GPUs is extremely significant for scientists to be able to move toward more energy-efficient computing. GPUs have completely different hardware architecture and execution model than a multi-core CPU. As a result, we cannot directly apply the energy consumption reduction methods in CPUs to GPUs. Moreover, there are different GPU technologies and architectures, and they progress quickly, thereby obtaining different results with applying the same methodology on different generation of GPUs.

For instance, one of the most striking and prevalent techniques to mitigate high energy consumption in processing devices is Dynamic Voltage and Frequency Scaling (DVFS). It refers to scaling voltage and working frequency to improve energy consumption when execution time is not critical. The DVFS technology can be used to increase performance when decreasing energy consumption is not an achievable goal. Interestingly, applying the same DVFS settings on CPUs to GPUs is not always useful. The authors of [8] illustrated that decreasing working frequency in CPUs saves energy; however, Ge et al empirically showed that this conclusion cannot be generalized to GPU environments [20].

Another challenge associated with energy-efficient GPUs is the lack of accurate estimation and simulation tools for performance/energy, especially in DVFS-based approaches [9] [21]. This is can be a big issue when comparing the energy-efficiency of different methods and applications. Moreover, defining a concrete and accurate power model is strikingly complicated, in CPU and GPU alike. This happens power consumption in GPUs depends on a wide range of parameters such as working frequency

of processing units and memory, temperature, voltage, and GPU architecture.

In GPU platform, there are cases in which there is a trade-off between energy-efficiency and other parameters, such as performance. Therefore, designing energy-efficient algorithms in multi-objective environments adds more complexity in the proposed solutions because we should balance between these conflicting goals and cope with a multi-optimization problem instead of a single-optimization one. For example, meeting our requirements in terms of performance remains a critical issue in real-time systems, even when improving the energy-efficiency. Lack of information in GPU hardware and power management is another significant issue in GPUs [21]. These challenges and their related consequences has been listed in Table 1.

IV. ENERGY EFFICIENCY METRICS

In literature, there are several power related metrics to evaluate energy-efficiency in GPUs. Performance/watt is the number of operations per each watt. This metric can be used to understand how efficiently a machine or an algorithm use the energy. Performance/watt is also known as "power-efficiency" and "energy-efficiency and these two terms are used interchangeably. Green500 list uses this metric to rank the top 500 green supercomputers [1]. Power, usually measured in watt, is a typical metric for the rate of heat dissipation or consuming energy. Energy, also usually measured in watt, is summation of power consumed during a period [18].

When there is a trade-off between energy consumption and performance in a computer system, Energy Delay product (EDP) and Energy Delay squared product (E2DP) are used to take into account both of these metrics together. Energy-delay product is a useful metric for quantifying efficiency of a specific system in terms of parallel scalability and representing the possible trade-offs between power and performance in the DVFS technique [18]

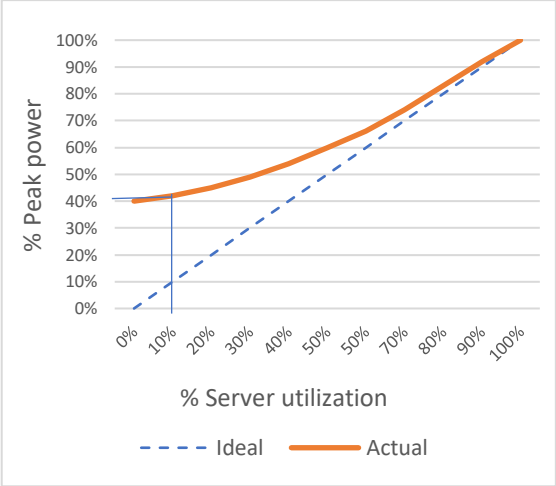

Figure 2: Energy proportionality curve of a server

[37]. Here, delay is a performance metric such as execution time or Instruction Per Second, IPS. There is more emphasis on performance in E2DP as opposed to EDP. Energy per word or byte is also a common metric used for energy consumed by communication links, such as the link between the CPU and GPU.

In 2007, two researchers at Google presented the "*energy proportionality curve*", as depicted in figure 2, which shows the power usage over server utilization [17]. The ideal energy proportionality curve of a server is represented by the dashed line and the actual energy proportionality curve is illustrated by the solid line. According to this curve, Wong at el defined energy proportionality as [19]:

$$EP = 1 - \frac{Area_{actual} - Area_{ideal}}{Area_{ideal}} \qquad (1)$$

The low energy proportionality of a server and inefficient trend in server energy usage have been a major obstacle for achieving overall server energy efficiency. In general, it is said that a component, such as CPU, RAM, cooling system, in a computer system is "*energy proportional*" if the its power

Table 1: Challenges and their related consequences when improving energy-efficiency in GPUs

Challenges	Consequences
Different hardware architecture and execution model than the CPUs	We cannot apply directly the same energy reduction method form CPU to GPU
There are different and progressing quickly of GPU architectures	We get different results when applying the same energy reduction method to different generations.
Lack of estimation and simulation tools for energy and performance for GPU platforms	We mostly use real platform to compare the energy-efficiency of different methods or applications.
Difficulty of defining an accurate power model	It is difficult to estimate the GPU power/energy consumption.
Trade-off between energy-efficiency and performance	It leads to a multi-objective environment with more complexity.
Lack of information from GPU hardware	It makes it more difficult to define a solution to reduce energy consumption

consumption is proportional to its utilizations, the rate of doing useful works. As of writing this paper, many scientists have investigated the problem of "energy proportionality" in different components except for the GPUs. We also believe that this problem can be generalize to the GPUs because of two reasons. Firstly, the main source of energy usage has been trending to GPUs in the nowadays servers and supercomputers [1] [24].

Secondly, GPUs are consuming a significant amount of energy even when there is no useful computation to do. For instance, we measured the idle energy consumption of a GeForce GTX TITAN X, when the GPU is in idle mode for a short time, at about 85W and it can increase to 250W in a period of overloading. It shows that this GPU has a low energy proportionality since it consumes about 85 watts when its utilization is zero. Poor programming, leading to low utilization of the GPU, is one of the main reasons of low energy efficiency. For instance, on supercomputers, many legacy programs are not fully ported to GPUs yet. So, a large fraction of the time, the utilizations of GPUs are zero. An improvement in energy proportionality of GPUs will increase energy-efficiency in the overall computing systems.

It is generally agreed that there is a trade-off between performance and energy-efficiency in the concurrent applications. However, we argue that energy-efficiency and performance in parallel programs may not have any conflict to each other. Energy efficiency and performance can be improved by efficient structured programming, such as using less synchronization or shared data, thereby increasing the GPU utilization. This matter is more observable in hybrid CPU-GPU systems where high utilization of computing resources can improve both the total energy consumption and performance. For instance, Luk et al suggested an algorithm for work distribution among CPU and GPU to increase processing elements utilizations [36]. In comparison with static work distribution by programmers, their algorithm could improve energy consumption and performance by 20% and 25%, respectively. Moreover, in [39], the authors observed that the relationship between the number of blocks and performance follows the same pattern as the relationship between the number of blocks and energy consumption when they run a well-known parallel biological code on GPU. That is, performance and energy-efficiency are not conflicting, and they can both be improved through optimizations at the same time.

Additionally, there are two common metrics to evaluate a power model. The error percentage of the proposed model represents the error of our power prediction to its true value. It is calculated by the formula in (2), where the $Actual_{Value}$ *and* $Predicted_{Value}$ are the actual power consumption measured by a power meter and the power estimated by the power model, respectively.

$$Error = \frac{Actual_{Value} - Predicted_{Value}}{Actual_{Value}} \times 100\% \quad (2)$$

Accuracy of a power model represents how accurate the power model is, and can be calculated using formula in (3):

$$Accuracy = 100\% - Error\ percentage \quad (3)$$

The mentioned power-related metrics and their applications has been listed in Table 2.

V. ENERGY AND POWER MEASUREMENT

A. *Power Model*

There are different power models proposed in literature to estimate the energy consumption of GPUs. Some models are more high-level and simpler, while others are more detailed, but also more complex. That is, there is a trade-off between the simplicity of power model and its accuracy.

Table 2: The mentioned power-related metrics and their applications to evaluate energy efficiency in GPUs

Metrics	**Units**	**Applications**
Power consumption	Watt	To measure the rate of heat dissipation or consuming energy
Energy consumption	Watt	To calculate the summation of power consumed during a period
Energy efficiency (Power efficiency)	Performance per watt	To understand how efficiently a machine or an algorithm use the energy
Energy Delay Product	Performance watt	To take into account performance and energy consumption together when there is trade-off between them
Energy Delay Square Product	Performance^2 watt	To take into account performance and energy consumption together when there is trade-off between them, but with more emphasis on performance
Energy per word (byte)	Watt per word (byte)	To measure the energy consumption by communication links
Accuracy	-	Measuring the accuracy of a proposed power model

Multiple researchers have proposed that only the power consumption of processing units in GPUs can be taken into account as they are the main power consumers**.** The power consumption of a core is derived from the power consumption model in CMOS logic circuits. As it is formulated in (4), the power consumption is the summation of $P_{dynamic}$ and P_{static}, in which $P_{dynamic}$ (dynamic power dissipation) is traditionally thought to be the main source of total power consumption. The static power is mostly assumed to be a small fixed value, depending on the voltage and leakage current.

$$P = P_{dynamic} + P_{static} \quad (4)$$

$P_{dynamic}$ is composed by three main components, namely $P_{switching}$, $P_{short-circuit}$, $P_{leakage}$, which Dayarathna defines as [24]:

"… switched capacitance power (caused by the charging and discharging of the capacitive load on each gate's output), short-circuit power (caused by short-circuit current momentarily flowing within the cell), and leakage power (caused by leakage current irrespective of the gate's state)"

Hence, the total power consumption can be defined by (5):

$$P=(P_{switching} + P_{short-circuit} + P_{leakage}) + P_{static} \quad (5)$$

The main contribution to dynamic power is caused by the switching component, $P_{switching}$, whose power model is:

$$P_{switching} = ACV^2f \quad (6)$$

In equation (6), A is the number of switches per each clock cycle, C is the total capacitance load, V represents the supply voltage, and f is the frequency at which the processor is working. In the DVFS-based solutions, $P_{switching}$ is used to model the total power consumption of the GPUs. Although $P_{switching}$ represents a high abstraction and general power model, such a basic model can provide us enough accurate to model power.

B. INTERNAL AND EXTERNAL POWER SENSORS

Although external power meter provides us an accurate power measurement method for a specific hardware component (e.g. RAM, CPU, GPU, Network Card), they suffer from several disadvantages [2]. This power measurement technique is less portable and scalable since it needs extra costly hardware components as well as physical access to the system [24]. Moreover, it can only provide us coarse-grained power profiling. In most cases, we need to know the power consumption of different parts of a component to be able to analyze the energy consumption of our HPC systems with more details. Low sampling frequency, and lack of available tools in the market for some specific HPC systems, are two other drawbacks of external power measuring. Even though PowerSensor 2 was recently introduced as a cheap external power meter prototype tool with higher sampling frequency, easier to use and more publicly available in comparison with other kinds of external power meter, the existing problems are still an obstacle to use this method ubiquitously [50].

Internal power sensors are becoming more common in state-of-the-art GPUs to monitor real-time power consumption without needing to purchase external hardware. They provide online ability to measure the instantaneous power consumption in the GPU using specific tool such as NVIDIA-smi or NVIDIA Management Library (NVML) library [2].

Power measurement using internal power sensors is the current area of research [31]. Low sampling frequency, lack of ubiquity, error measurement (the instantaneous power draw is in the range of +/- 5 watts) and lack of documentation about the way of measuring power are four disadvantages of this method [23] [31]. However, ease of use is the main advantage, since we need no extra hardware or physical access to measure instantaneous power consumption [2] [31]. The mentioned problems have been mitigated in the next generations of GPUs. For example, Ferro et al showed that Kepler provides much more accurate power measurement with a better sampling rate in comparison with Fermi platform [31].

VI. PROPOSED SOLUTIONS

A. DVFS GPU

Among the different kinds of proposed solutions for energy-efficiency in GPUs in the literature, it seems that the most common and investigated one is using DVFS [9]. In this section, we first describe the features of DVFS in GPUs, and then summarize several available researches.

DVFS features in GPUs. Although research into GPU DVFS started a few years ago, GPUs provide a better environment to apply DVFS technique than CPU. This happens because the peak power consumption of a modern GPU is almost double that of the common modern CPU. Hence, we can save more energy with DVFS in GPUs in comparison with CPUs. Moreover, unlike CPUs, we can scale the working frequency of processing

components and memory in GPUs. As a result, applying DVFS in GPUs is more complex since we should firstly determine which resource is underutilized, memory or processing component, to adjust its working frequency. [16]

There is a trade-off between energy-efficiency and performance in CPUs when applying DVFS. However, this assumption may not be valid in GPU environments [16]. Moreover, the frequencies of GPUs do not only have a larger range than CPUs, but they are also more granular. Unfortunately, one of the drawbacks of scaling the operating frequencies of GPUs is that at the moment it requires administrative access to the system [51]. Lack of proper tools to scale the core voltage in Linux environments is another drawback of this technique.

It should be noted that energy optimization and power optimization are not similar. For instance, if decreasing power by the factor of 2 makes the program execution time increase by the factor of 3, then the total energy consumption can go up by the factor of $\frac{3}{2} = 1.5$. Therefore, scaling up the frequency may increase power consumption but also may decrease total energy consumption. It is dependent on the application characteristics and GPU specifications [20].

Several studies on DVFS GPU. DVFS technology in GPUs behaves differently in comparison with CPUs regarding performance and energy efficiency. These behaviors were experimentally studied by Ge et al using three different compute-bounded applications on a heterogeneous system equipped with a NVIDIA Kepler GPU and an Intel Sandy Bridge CPU [20]. They observed that the achieved worst performance and energy-efficiency (GFLOPS/Watt) on GPU were at least 3× and 5× better than the best corresponding numbers in CPU, respectively.

Additionally, their experiments showed that the problem size can impact the obtainable energy-efficiency in CPU and GPU with different slope, in matrix multiplication problem. That is, the achievable energy-efficiency increases faster in the CPU than the GPU, when the problem size increases. Furthermore, it is observed that higher GPU core frequency provides better performance without any increase in the total system energy consumption, unlike typical behavior of the CPU. That is, performance and energy-efficiency had a typical trade-off behavior in CPUs, however, this conclusion could not be generalized to GPUs. It was also experienced that the proper working frequency which can provide the best energy-efficiency is dependent to the application characteristics, e.g. how much it is compute-bound or memory-bound, in GPU.

Jiao et al conducted another study to observe the core and memory frequency scaling on three diverse application benchmarks: matrix multiplication (compute-bounded), matrix transpose (memory-bounded), and fast Fourier transformation (hybrid) [29]. These applications showed three different types of behavior toward the same core and frequency scaling. Performance and energy-efficiency in matrix multiplication application could be improved by decreasing memory frequency and increasing core frequency. However, to achieve the same goal for matrix transpose, we need to scale up the memory frequency and scale down the core frequency. The best performance and energy-efficiency could be achieved in the hybrid benchmark using increasing both the memory and core frequency.

Accordingly, analysis about application characteristics, e.g. compute-bound or memory-bound, can aim to analyze the memory and processor utilization. This also leads to the opportunity of improving energy-efficiency by decreasing the working frequency of the underutilized component. To do so, Guerreir et al proposed a novel methodology to classify the applications based on the DVFS effects on their power consumption, energy consumption and execution time [44]. This method predicts the behavior of GPUs under different memory and core working frequencies to understand the effect of DVFS on a specific application. The advantage of this method is that it can easily provide near-optimal outcomes in comparison with using complicated and accurate power and performance models. The classifier trained based on the underlying hardware and a set of synthetic benchmarks. They validated their hardware-independent approach on two new generations of NVIDIA GPU architectures, Maxwell and Pascal, with 35 benchmark applications. The proposed technique could predict the DVFS working frequencies of GPUs in some classes with up to 22% energy saving and only 2% performance loss. The average achievable energy savings on Maxwell and Pascal were 16% and 20%, respectively.

The energy consumption in GPUs can be affected not only by working frequency but also by voltage. Frequency/voltage of memory or cores may scale continuously within a specific range. According to a study conducted by Mei et al, core frequency and voltage on Maxwell and Fermi platforms had a sublinear behaviour, while it is generally believed core frequency increases linearly to the core voltage [9]. Based on their experimental results, they also concluded that the relationship between core frequency and performance does not follow a linear model.

Interestingly, different generations of GPUs behave differently with respect to DVFS. Mei at al experimentally observed that they always could save energy with the maximum memory frequency in a Maxwell GPU. However, decreasing core voltage was the key factor to conserve energy in a Fermi architecture.

Discussion. In [37], the authors claimed that in general, there is a trade-off between performance and energy-efficiency in concurrent applications using DVFS technique for GPUs. However, this trade-off depends on the applications characteristics and also GPU features. We observed several cases of using DVFS where performance and energy could also support each other.

Theoretically, it seems that selecting the best working frequency for cores and memory in GPUs depends on the application type [29] [9]. It would be better to increase core frequency and decrease memory frequency in compute-bound applications. However, to improve energy efficiency, it was suggested to increase the memory frequency and decrease core frequency in memory-bound programs. Practically, selecting the best working frequency for memory and processing cores, to obtain energy-efficiency, depends not only on the application specifications, but also on the underling GPU, energy measuring technique, input data and problem size.

B. Other proposed solutions and research

GPU architectures consists of different parts which Zhao et al categorized into three sections: (1) processing units and caches, (2) off-chip memory and (3) memory controllers [6]. According to their experimental analysis on NVIDIA Quardo 6000 and AMD Radeon HD 7970, the most significant energy usage in GPUs is caused by processing units and caches, and this accounts for about 50% of the total energy consumption. In addition to this, graphic memory contributes a large portion of the power consumption, up to about 30 percent. These percentages can somewhat vary for different GPUs architectures and with different memory bandwidth utilizations (the fraction of time which data-bus is busy). They illustrated that the more memory bandwidth is utilized, the more the percentage of total energy consumption is consumed by memory.

Adjusting fan speed to GPU workload and temperature is another method to obtain energy-efficiency on GPUs [32]. This method can reduce the energy consumption of the cooling system and improve functionality of GPU in terms of energy consumption and performance.

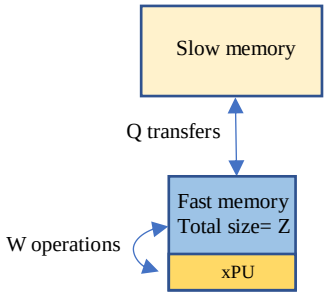

Figure 3: The memory architecture used in the proposed energy roofline model

In another study, Price et al experimentally study how temperature, core frequency and voltage can influence the energy efficiency. They used the xGPU code, a common compute-bound code in the radio astronomy context, as a benchmark on a platform equipped with NVIDIA K20 GPU. In their experimental results, energy efficiency increased up to 48% in comparison with GPU default settings through increasing core frequency, decreasing supply voltage and keeping the die temperature of the GPU constant to 30 degrees by throttling fan speed.

Choi et al proposed an energy roofline model [35] inspired by William's roofline model. They modelled an energy-efficiency roofline for an algorithm running on a specific machine, using its arithmetic intensity. Arithmetic intensity is defined as the number of arithmetic, floating point, operations per byte of memory that is accessed [25]. The authors also elaborated the conditions which there is a trade-off between performance and energy. They used an abstract model of the van Neumann architecture with a two-level memory hierarchy of an infinite slow memory, a fast finite memory, and a processing unit (xPU), as shown in Figure 3. Here, it is assumed that a specific algorithm does W arithmetic operations and Q memory operations between the slow and fast memory. With the assumption of the perfect overlap between memory access and computations, the execution time of algorithm can be modelled by (7) and (8), where τ_{flop} and τ_{mem} are the time per flop and time per memory operations, respectively.

$$T = \max\left(W \times \tau_{flop}, Q \times \tau_{mem}\right) \qquad (7)$$

$$= W \times \tau_{flop} \times \max\left(1, \frac{B_\tau}{I}\right) \qquad (8)$$

$B_\tau \equiv \tau_{flop}/\tau_{mem}$ and it is called time-balanced point of the machine with flops per byte unit. I is the intensity of the algorithm and obtained by I≡W/Q. They also used an energy model as represented in (9), where ϵ_{flop} ,ϵ_{mem} and π_0 are energy per flop, energy per memory operation and power leakage respectively.

$$E = W \times \epsilon_{flop} + Q \times \epsilon_{mem} + \pi_0 \times T \qquad (9)$$

This energy roofline model was validated on two different hardware platforms, an Intel multi-core CPU and a NVIDIA GPU.

Choi et al extend their primary roofline energy model later by adding the contribution of the cache memory and a power cap [34]. Given an arithmetic intensity, their improved model can show which underlying hardware for a specific application is more or less energy-efficient or performance-efficient. Additionally, the proposed model represents how instructions, using arithmetic intensity, can be throttled to satisfy the power cap. The main drawback of their work is using its only-benchmarking evaluation and a very simple assumption regarding the underlying hardware. For instance, they have not taken into account the other existing components in the GPUs, such as warp scheduler and other type of memory, which also can impact the total energy consumption.

Demmel et al did an interesting research on energy-efficiency of GPUs by generalizing the strong scaling concept [33]. They showed that total energy consumption remains constant for a fixed problem size in matrix multiplication and n-body problem, when the number of processing unit increases. In [29], the authors showed that energy consumption and performance in GPUs was influenced by two factors: the rate of issuing instructions and the ratio of global memory transactions to computation instructions. The former describes that how much the application is compute-bound and the latter characterizes how much the application is memory-bound.

Memory access can impact energy efficiency and performance. Collange et al showed that energy required for a memory access done in Tesla GPUs with CUDA framework was up 7 to 15 times more than any computations [38]. Hence, high memory access rate can decrease not only energy efficiency, but also performance.

In the CUDA environment, programmers should take a decision about the optimal number of blocks and threads per blocks. These two factors influence not only performance, but also total energy consumption. Huang et al and Dreßler et al experimentally showed that energy-efficiency can be improved when the number of blocks and thread per blocks, thread topology, in their applications increased, thereby increasing warp occupancy [39] [40].

Another technique used for improving energy consumption in GPUs is scheduling. Unlike single thread scheduling in CPU, in the CUDA environment, a group of threads called warp, is scheduled on each SM. ElTantawy et al suggested a mechanism to find fine-grained synchronizations, mostly implemented in busy-wait method, in code, and then reduce the priority execution of the warp running the synchronization code with a novel hardware warp scheduler, since fine-grained synchronizations are expensive [41]. That is, their proposed scheduler does not allow to the spinning warp, the warp contained the fine-grained synchronization, competing with scheduler to acquire GPU resources. The novel scheduler allows other critical warps to make more progress. Their approach improved energy-efficiency and performance with the average ratio of 1.7 and 1.4, respectively. It should be noted that warp scheduling is a hardware-based technique, since it is one of the constant GPU characteristics determined by producer.

GPU resources can be underutilized since multi-threaded parallelism is dependent on the different parameters such as block size, grid size and problem size. To mitigate this issue, Fermi, and the next generations of NVIDIA GPUs, provided concurrent kernel execution. This increases resource utilizations using concurrent execution of multiple thread blocks of different CUDA kernels on SMs. The order of issuing and scheduling block threads from different ready kernels is according to the kernel launch order. Thread blocks from different kernels cannot run simultaneously in a *wave*[1] when there is conflict among the resources. Li et al proposed a novel scheduling algorithm, called symbiotic, to improve total energy consumption and performance through augmenting resource utilization in concurrent kernel executions environments [43]. Their symbiotic method balances the execution of memory-bound and compute-bound kernels.

Studies conducted by Wang et al illustrated that kernel fusing can be employed as a software power optimization method in GPUs [42]. Kernel fussing consists in integrating two or more kernels into a single and large one. It can provide two features: reducing the redundant data communication between CPU and GPU, and reusing the data in shared memory or registers. As a result, kernel fusing provides demand balancing of hardware resources, which can increase the unitizations of resources, thereby improving the total energy-efficiency. To improve the use of kernel fusion, they also proposed an effective methodology to reduce

[1] According to NVIDIA definition, a wave is a set of thread blocks that run concurrently on GPU.

the usage of shared memory since kernel fusion cause more demand of shared memory, and this can degrade energy-efficiency and performance.

A code analyzer can analyze which part of code is compute-bound, and which part is memory-bound. Then, the working frequency of memory and processing unit can be adjusted on-the-fly. However, Calore et al concluded that this technique is a feasible method on state-of-the-art Xeon processors, but hardly on Nvidia GPUs [7], which is because the overhead of clock management libraries in GPUs outweighs the achievable energy-efficiency.

Every SM has its own components such as warp scheduler, instructions fetch and decode components. In many cases, SMs have the same behaviour. Therefore, if we can group the SMs in a lock-step fashion, and they can work together with a shared component, it is possible to reduce total energy consumption. Zhang et al, proposed a new architectural master-slave model for GPU according to this idea [46]. A SM can choose as a master SM and it shares its component with the slave SMs, which they put their unused component in idle mode. Their simulation shows that this novel architecture model can reduce 7.5% of total energy consumption.

Neighboring concurrent thread arrays usually use a large amount of shared data. However, the GPU scheduler distributed these threads in a round-robin fashion among the SMs to achieve better load balancing, thereby increasing data replication in L1 cache. To synchronize shared data among the different L1 caches of SMs, data movement increases which causes decrease cache-efficiency and power-efficiency. To mitigate this issue, Tabbakh et al proposed a novel scheduler and a cache line allocation/replacement strategy to improve total energy consumption and performance in GPUs [47]. They suggested a sharing-aware scheduler to attempt to spread the threads among the same SMs, so that the data replication in L1 caches decreases. Furthermore, they improve their proposed solution with an allocation/replacement policy that gives higher priority to shared data to stay longer in the L1 cache. The simulation results, based on GPGPU-SIM v3.3.2, and GPUWattch simulators, illustrate that these methods can provide us 10% and 7% energy-efficiency of DRAM and performance improvement, respectively.

In most state-of-the-art mobile devices thermal-aware methods are used to control the working frequency of CPU and GPU. Prakash et al suggested a thermal-aware technique to adjust the working frequency of CPU and GPU in mobile phones to improve performance and meet the temperature constraints [48]. They proposed a power model based on the temperature of the processing units to select the best working frequencies with DVFS. This technique, as the best of our knowledge, has not been studied in other environment such as HPC platforms.

Hardware-based compression of data between the communication links in data-intensive applications can decrease data traffic in both CPU and GPU, thereby improving performance and energy-efficiency. Data compression remarkably increases the number of communication channel switches from 0 to 1 or 1 to 0, called bit-toggle because of decreasing in redundancy and data similarity. As a result, more bit-toggle increases dynamic energy consumption of the communication links. To alleviate this problem, Pekhimenko et al proposed a novel toggle-aware compression algorithm to not only decrease the energy consumption of communications links using less bit-toggle, but also decreasing the data traffic to improve performance [49].

VII. TAXONOMIES AND PROPOSED SOLUTIONS

In the literature, there are different taxonomies of proposed solutions to reduce energy consumption for various environments such as servers, CPU and memory, but there has been little focus on the field of GPUs. Hence, we propose different taxonomies for these proposed approaches and describe them by a two-branch taxonomy as follows:

Hardware-based and software-based: GPU memory architecture, topology network between cores, and the cooling system can influence energy consumption. Any improvement in terms of power consumption in these components requires hardware revising. We call these solutions, which should be applied at design time in hardware by vendors, hardware-based. However, after producing a GPU, we can only make use of energy-efficient software to reduce energy consumption, which is termed as software-based approaches.

Thermal-aware and energy-aware: thermal-aware techniques focus on implementing certain methods that are not only restricted to reduce heat dissipated by GPUs, but also for improving the quality of the cooling system used in GPUs. These approaches take temperature as a core component when building a power model [2]. The temperature depends on different parameters such as the power consumption, dimension, and relative location of the GPU. The goal of thermal-aware approaches is to minimize the GPU temperature, thereby reducing the costs of cooling and the related power consumption in GPU. On the other hand, energy-

aware technologies attempt to reduce energy usage by efficiently making use of current resources. That is, energy-aware methods use power consumption as the core of their power model. Clearly, temperature and power consumption can form a more complicated power model together. Therefore, the proposed solution would be thermal-aware and energy-aware at the same time.

Single and composite: The cost function, also known as objective function and energy function, is a mathematical function that contains one or more different parameters (respectively, single and composite cost function). The proposed solutions can be grouped with respect to the number of parameters considered for optimization [2]. For instance, a cost function which aims at minimizing energy consumption is considered as a single cost function; but if it deals with minimizing both energy consumption and execution time at the same time, the cost model would be composite.

Online and offline: we classified a proposed solution as an offline method if it can be applied before running the program. For example, increasing core utilization using a proper block size and number of threads per block is an offline method since it should be done by programmer before running the program. The auto-tuning technique, which is used for tuning these kind of parameters, can be categorized as an offline method since it does not put any overhead during execution of program. Nevertheless, if the proposed approach should be applied during execution time, it is called online. For instance, selecting optimal working frequency in DVFS-based technique may be done during program execution, based on system state. It should be noted that every online proposed approach introduces some overhead, thereby increasing energy consumption. The energy saving gained by a proposed solution must outweigh the appended energy consumption caused by it.

We classified the mentioned proposed solutions for improving energy-efficiency in GPUs in Table 3. It should be noted that we only take into account the studies that proposed novel solutions to improve energy-efficiency. We extract serval interpretation of this table as follow:

- The majority of the proposed solutions are categorized in the energy-aware group. We could not find any thermal-aware solution in the context of HPC in literature. This can be because the thermal-aware techniques are a proper way to address the related temperature problems in the thermal-sensitive environments.
- In literature, there was almost a balance between single and composite solutions. However, the cost function in the composite proposed solutions mostly consists of performance and power consumption. We can build more complicated cost function with more parameters.
- It seems that there is more interest among researchers to proposed hardware-based solutions. This leads to every generations of Nvidia GPU are becoming more energy-efficient that the previous one.

Table 3: Classification of mentioned proposed approaches for energy-efficiency in GPUs based on our taxonomies.

Proposed Solutions / Classifications	Luk et al [36]	NVIDIA Co [32]	ElTantawy et al [41]	Wang et al [42]	Li et al [43]	Guerreiro et al [44]	Zhang el al [46]	Tabbakh et al [47]	Prakash et al [48]	Pekhimenko et al [49]
Thermal-aware	×	✓	×	×	×	×	×	×	✓	×
Energy-aware	✓	×	✓	✓	✓	✓	✓	✓	×	✓
Single	×	✓	×	✓	×	✓	✓	×	✓	×
Composite	✓	×	✓	×	✓	×	×	✓	×	✓
Online	✓	✓	✓	×	✓	×	✓	✓	✓	✓
Offline	×	×	×	✓	×	✓	×	×	×	×
Hardware-based	×	×	✓	×	✓	✓	✓	✓	✓	✓
Software-based	✓	✓	×	✓	×	×	×	×	×	×
Area of focus	Increasing GPU utilization through better work distribution between CPU and GPU	Adjusting fan speed to the GPU workload	Decreasing the non-useful works through a more efficient thread scheduling	Kernel Fusing to increase GPU utilizations	Increasing GPU utilizations through a better scheduling strategy	Adjusting the memory and core frequency based on the program analysis results.	Changing the GPU architecture to decrease doing the non-useful works	Revising the SM scheduler to decrease doing the non-useful works	A thermal-aware method to select the best working frequencies	Revising the compression protocol in the GPU communications link to reduce the bit-toggles
Year of publish	2009	2009	2018	2010	2018	2018	2015	2017	2016	2016

- In general, there are two common ways to improve energy-efficiency in GPUs: selecting the best working core/memory frequencies, and improving GPU utilization. The latter can be done either by decreasing the redundant works done by GPU, or decreasing the period that each component, such as memory or processing units, is idle.

VIII. CONCLUSION AND FUTURE WORK

Energy-efficiency in GPUs got tremendous attentions in academia since it has a crucial impact on the running costs and development of future HPC machines. However, improving energy-efficiency is not easy to achieve in GPUs because of different hardware architectures, fast-progressing technologies, lack of accurate estimation and simulation tools for performance/energy in this context, difficulty of power measurement and the possible trade-off between performance and energy-efficiency.

There are different proposed power models in literature to estimate the energy consumption of GPUs, and there is a trade-off between their simplicity and accuracy. Moreover, there are several solutions to achieve energy-efficiency in GPUs, of which DVFS is the most studied. We classified them from different two-class perspectives: software-based and hardware-based, thermal-aware and energy-aware, single and composite, online and offline.

Although there is a lot of ongoing research about energy-efficiency in GPUs, several topics in this scope have not been investigated so far. One of the most interesting one is energy-efficiency in multi-GPU machines. Multi-GPU machines are getting more common in HPC systems, as they obtain more performance either in loosely coupled architectures or tightly coupled platforms. Another future work can be about defining thermal-aware power model for GPUs in HPC platforms.

REFERENCES

[1] Top500.org. (2018). TOP500 Supercomputer Site. [online] Available at: https://www.top500.org/ [Accessed 2 Aug. 2018].

[2] Bridges, R., Imam, N. and Mintz, T. (2016). Understanding GPU Power: A Survey of Profiling, Modelling, and Simulation Methods. ACM Computing Surveys, 49(3), pp.1-27.

[3] Price, D., Clark, M., Barsdell, B., Babich, R. and Greenhill, L. (2015). Optimizing performance-per-watt on GPUs in high performance computing. Computer Science - Research and Development, 31(4), pp.185-193.

[4] Statista website. (2018). Industrial prices for electricity in the Netherlands 1995-2017 | Statistic. [online] Available at: https://www.statista.com/statistics/596254/electricity-industry-price-netherlands/ [Accessed 2 Aug. 2018].

[5] Mittal, S. and Vetter, J. (2014). A Survey of Methods for Analyzing and Improving GPU Energy Efficiency. ACM Computing Surveys, 47(2), pp.1-23.

[6] Zhao, J., Sun, G., Loh, G. and Xie, Y. (2013). Optimizing GPU energy efficiency with 3D die-stacking graphics memory and reconfigurable memory interface. ACM Transactions on Architecture and Code Optimization, 10(4), pp.1-25.

[7] Calore, E., Gabbana, A., Schifano, S. and Tripiccione, R. (2017). Evaluation of DVFS techniques on modern HPC processors and accelerators for energy-aware applications. Concurrency and Computation: Practice and Experience, 29(12), p.e4143.

[8] D.H. Kim, C. Imes, H. Hoffmann. (2015). Racing and pacing to idle: theoretical and empirical analysis of energy optimization heuristics, in: Proceedings of the IEEE 3rd International Conference on Cyber-Physical Systems, Networks, Applications, pp. 78–85.

[9] Mei, X., Wang, Q. and Chu, X. (2017). A survey and measurement study of GPU DVFS on energy conservation. Digital Communications and Networks, 3(2), pp.89-100.

[10] NVIDIA. (2018). NVIDIA Tesla V100 Data Center GPU. [online] Available at: https://www.NVIDIA.com/en-us/data-center/tesla-v100/ [Accessed 6 Aug. 2018].

[11] D. Anderson, J. Dykes, and E. Riedel. (2003). More than an interface-SCSI vs. ATA. In Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST'03). pp. 245–257.

[12] El-Sayed, N., Stefanovici, I. A., Amvrosiadis, G., Hwang, A. A., & Schroeder, B. (2012). Temperature management in data centers: why some (might) like it hot. ACM SIGMETRICS Performance Evaluation Review, 40(1), 163-174.

[13] Oak Ridge Leadership Computing Facility. (2018). Summit. [online] Available at: https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/ [Accessed 7 Aug. 2018].

[14] Dong, T., Dobrev, V., Kolev, T., Rieben, R., Tomov, S., & Dongarra, J. (2014). A step towards energy efficient computing: Redesigning a hydrodynamic application on CPU-GPU. In Parallel and Distributed Processing Symposium, 2014 IEEE 28th International (pp. 972-981). IEEE.

[15] A. Hameed, A. Khoshkbarforoushha, R, Ranjan, P. Jayaraman, J. Kolodziej, P Balaji and et al, "A Survey and Taxonomy on Energy Efficient Resource Allocation Techniques for Cloud Computing Systems", Journal of Computing, Vol 98, Issue 7, Jul 2016, pp.751-774.

[16] Boyer, M. (2013). Improving Resource Utilization in Heterogeneous CPU-GPU Systems. Ph.D. Thesis. University of Virginia.

[17] Barroso, L. A., & Hölzle, U. (2007). The case for energy-proportional computing.

[18] Feng, X., Ge, R. and Cameron, K. (2005). Power and Energy Profiling of Scientific Applications on Distributed Systems. In: 19th IEEE International Parallel and Distributed Processing Symposium. IEEE. pp. 10-pp. IEEE.

[19] Wong, D., & Annavaram, M. (2012). Knightshift: Scaling the energy proportionality wall through server-

level heterogeneity. In Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture (pp. 119-130). IEEE Computer Society.
[20] R. Ge, R. Vogt, J. Majumder, A. Alam, M. Burtscher, and Z. Zong.(2013). Effects of dynamic voltage and frequency scaling on a K20 GPU. In: Proceedings of the IEEE 42nd International Conference on Parallel Processing (ICPP), pages 826–833. IEEE.
[21] Mei, X. (2016). Energy conservation techniques for GPU computing. Ph.D. Thesis. Hong Kong Baptist University.
[22] Barbosa, D., Márquez, G. L., Ruiz, V., Silva, M., Verdes-Montenegro, L., Santander-Vela, J., ... & Kramer, M. (2012). Power Challenges of Large Scale Research Infrastructures: the Square Kilometer Array and Solar Energy Integration; Towards a zero-carbon footprint next generation telescope. arXiv preprint arXiv:1210.4972
[23] NVIDIA Developer. (2018). NVIDIA System Management Interface. [online] Available at: https://developer.nvidia.com/nvidia-system-management-interface [Accessed 24 Dec. 2018].
[24] Dayarathna, M., Wen, Y. and Fan, R. (2016). Data Center Energy Consumption Modeling: A Survey. IEEE Communications Surveys & Tutorials, 18(1), pp.732-794.
[25] Williams, S., Waterman, A., & Patterson, D. (2009). Roofline: An insightful visual performance model for floating-point programs and multicore architectures (No. LBNL-2141E). Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States).
[26] K. Kasichayanula et al., (2012). Power aware computing on GPUS. Master thesis, SAAHPC, pp. 64–73.
[27] Jeffers, J., & Reinders, J. (2015). High performance parallelism pearls volume two: multicore and many-core programming approaches. Morgan Kaufmann.
[28] Keckler, S., Dally, W., Khailany, B., Garland, M. and Glasco, D. (2011). GPUs and the Future of Parallel Computing. IEEE Micro, 31(5), pp.7-17.
[29] Y. Jiao, H. Lin, P. Balaji, and W. Feng. (2010). Power and performance characterization of computational kernels on the GPU. In Int'l Conference on Green Computing and Communications (GreenCom) & Int'l Conference on Cyber, Physical and Social Computing (CPSCom'10). 221–228.
[30] M.Rofouei, T. Stathopoulos, S.Ryffel, W.Kaiser, M.Sarrafzadeh. (2008). Energy-aware high-performance computing with graphic processing units. In Proceedings of the 2008 Conference on Power Aware Computing and Systems (HotPower'08). USENIX Association.
[31] M. Ferro, A. Yokoyama, V. Kloh, G.Silva1, R. Gandra, R. Braganc, A. Bulcao, B. Schulze. (2017). Analysis of gpu power consumption using internal sensors. In Anais do XVI Workshop em Desempenho de Sistemas Computacionais e de Comunicac¸ao˜ , Sao Paulo - SP. Sociedade Brasileira de Computac¸ ˜ao (SBC).
[32] NVIDIA Corp (2009). Regulating power using a fuzzy logic control system. US8700925B2.
[33] J. Demmel, A. Gearhart, B. Lipshitz, andO. Schwartz. (2013). Perfect strong scaling using no additional energy. In Proceedings of the 2013 IEEE 27th International Symposium on Parallel Distributed Processing (IPDPS'13). pp. 649-660. IEEE.
[34] Choi, J., Dukhan, M., Liu, X. and Vuduc, R. (2014). Algorithmic time, energy, and power on candidate HPC compute building blocks. In: 28th International Parallel and Distributed Processing Symposium. IEEE. pp. 447-457.
[35] Choi, J., Bedard, D., Fowler, R. and Vuduc, R. (2013). A Roofline Model of Energy. In: 27th International Symposium on Parallel and Distributed Processing. IEEE. pp. 661-672.
[36] Luk, C., Hong, S. and Kim, H. (2009). Qilin: Exploiting parallelism on heterogeneous multiprocessors with adaptive mapping. In: 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE. pp. 45-55.
[37] Jin, C., de Supinski, B., Abramson, D., Poxon, H., DeRose, L., Dinh, M., Endrei, M. and Jessup, E. (2016). A survey on software methods to improve the energy efficiency of parallel computing. The International Journal of High Performance Computing Applications, 31(6), pp.517-549.
[38] Collange, S., Defour, D. and Tisserand, A. (2009). Power Consumption of GPUs from a Software Perspective. In: International Conference on Computational Science. Berlin, Heidelberg: Springer, pp.914-923.
[39] Huang, S., Xiao, S. and Feng, W. (2009). On the Energy Efficiency of Graphics Processing Units for Scientific Computing. In: International Symposium on Parallel & Distributed Processing. IEEE. pp. 1-8.
[40] Dreßler, S. and Steinke, T. (2012). Energy consumption of CUDA kernels with varying thread topology. Computer Science - Research and Development, 29(2), pp.113-121.
[41] ElTantawy, A. and Aamodt, T. (2018). Warp Scheduling for Fine-Grained Synchronization. In: International Symposium on High Performance Computer Architecture (HPCA). IEEE. p. 375-388.
[42] Wang, G., Lin, Y. and Yi, W. (2010). Kernel Fusion: an Effective Method for Better Power Efficiency on Multithreaded GPU. In: IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing. IEEE. p. 344-350.
[43] Li, T. (2018). Efficient Virtualization and Scheduling for Productive GPU-based High-Performance Computing Systems. Ph.D. Beihang University.
[44] Guerreiro, J., Ilic, A., Roma, N. and Tomás, P. (2018). DVFS-aware application classification to improve GPGPUs energy efficiency. Parallel Computing.
[45] Hall, P. J. (2011, August). Power considerations for the Square Kilometre Array (SKA) radio telescope. In General Assembly and Scientific Symposium, 2011 XXXth URSI (pp. 1-4). IEEE.
[46] Zhang, T., Jing, N., Jiang, K., Shu, W., Wu, M. and Liang, X. (2015). Buddy SM: Sharing Pipeline Front-End for Improved Energy Efficiency in GPGPUs. ACM Transactions on Architecture and Code Optimization, 12(2), pp.1-23.
[47] Tabbakh, A., Annavaram, M. and Qian, X. (2017). Power Efficient Sharing-Aware GPU Data Management. In: International Parallel and Distributed Processing Symposium (IPDPS). IEEE. pp. 698-707.
[48] Prakash, A., Amrouch, H., Shafique, M., Mitra, T. and Henkel, J. (2016). Improving Mobile Gaming Performance through Cooperative CPU-GPU Thermal Management. In: Proceedings of the 53rd Annual Design Automation Conference. ACM, New York, p. 47.
[49] Pekhimenko, G., Bolotin, E., Vijaykumar, N., Mutlu, O., Mowry, T. and Keckler, S. (2016). A Case for Toggle-Aware Compression for GPU Systems. In: International Symposium on High Performance Computer Architecture (HPCA). IEEE. pp. 188-200.

[50] Romein, J. W., & Veenboer, B. (2018, April). PowerSensor 2: A Fast Power Measurement Tool. In Performance Analysis of Systems and Software (ISPASS), 2018 IEEE International Symposium on, IEEE, pp. 111-113.

[51] Rojek, K. (2018). Machine learning method for energy reduction by utilizing dynamic mixed precision on GPU-based supercomputers. Concurrency and Computation: Practice and Experience, p.e4644.