Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 340 (2018) 3–22

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

A Holistic Resource Management for Graphics Processing Units in Cloud Computing

Abdulaziz Alnori[1] and Karim Djemame[2]

School of Computing
University of Leeds
Leeds, UK

Abstract

The persistent development of Cloud computing attracts individuals and organisations to change their IT strategies. According to this development and the incremental demand of using Cloud computing, Cloud providers continuously update the Cloud infrastructure to fit the incremental demands. Recently, accelerator units, such as Graphics Processing Units (GPUs) have been introduced in Cloud computing. This updated existence leads to provide an increase in hardware heterogeneity in the Cloud infrastructure. With the increase in hardware heterogeneity, new issues will appear. For instance, managing the heterogeneous Cloud infrastructure while maintaining the Quality of Service (QoS) and minimising the infrastructure operational costs will be a substantial issue. Thus, new management techniques need to be developed to manage the updated Cloud infrastructure efficiently. In this paper, we propose a systematic architecture to manage heterogeneous GPUs in a Cloud environment considering the performance and the energy consumption as key factors. Moreover, we develop a Heterogeneous GPUs analyser as the first step in the implementation of the proposed architecture. It aims to quantitatively compare and analyse the behaviour of two different GPUs architectures, NVIDIA Fermi and Kepler, in terms of performance, power and energy consumption. The experimental results show that adequate blocks and threads per block numbers allocations lead to 13.1% energy saving in Fermi GPU and 11.2% more energy efficient in Kepler GPU.

Keywords: Cloud Computing, Graphics Processing Units, Quality of Service, Heterogeneous GPUs Analyser.

1 Introduction

The existence of programming platforms that deal with Graphics Processing Units (GPUs), such as the Compute Unified Device Architecture (CUDA) [11] and Open Computing Language (OpenCL) [9] have significantly shifted GPU usage from its standard purpose, showing images and video games on computer screens, to a computational usage. The advent of these programming platforms has led to design applications to run on GPUs for general purpose use with high performance capabilities.

[1] Email:scasal@leeds.ac.uk

[2] Email:K.Djemame@leeds.ac.uk

https://doi.org/10.1016/j.entcs.2018.09.002
1571-0661/© 2018 Published by Elsevier B.V.

Cloud computing leverages the virtualisation of computing resources to allow end-users to provide them at an acceptable price. In increasing the computation demands, GPUs have been introduced in Cloud data centres because of their performance abilities and their suitability for some applications [16]. Moreover, GPU clusters will play an important role in the future of Cloud computing data centres since some compute-intensive applications need to involve GPUs with CPUs [36]. Cloud computing providers like Amazon [2], Microsoft Azure [1], IBM Bluemix [5], NIMBIX [6] and recently Google [4] have enabled the users to access GPUs located in their Cloud data centres. Therefore, this recent situation is changing the taxonomy of the Cloud data centres and the methods of managing these resources. The Cloud service provider has to continuously provide the performance for the end-user to avoid Service Level Agreement (SLA) violations. Maintaining the performance of the end-user application requires some corrective actions in the Cloud infrastructure, such as the Virtual Machine (VM) live migration technique; these actions incur a tremendous amount of energy consumption in the Cloud infrastructure and increase the operational costs. The US data centres energy usage report [33] states that in 2014 the total estimated energy consumed by the US data centres was 70 Billion kWh, which was approximately 1.8% of the total energy consumed in the US. The report also shows that the expected increase in energy consumption in the US data centres will be 4% from 2014 to 2020. Thus, the bill for the operational costs will continue to increase. With this massive energy usage, energy efficient solutions in Cloud data centres have become a major research concern. The Cloud physical infrastructure (i.e. CPU, memory and network) has been intensively studied by researchers in terms of performance, energy consumption and cost. However, heterogeneous GPU resources in the physical Cloud infrastructure in terms of performance and energy consumption need more consideration. Therefore, it is important to establish a provisioning framework of resources to support the applications to achieve Quality of Service (QoS) and reduce the operation costs.

Thus, the aim of this paper is to enhance the usability of GPU applications in Cloud computing environments with a steady performance, and at the same time reduce the operational costs by minimising energy consumption and increasing GPU resources sharing.

The contributions in this paper are: 1) a systematic architecture to manage the sharing of GPU resources in Cloud computing environments for general purpose use. The architecture will focus on the application deployment time and ensure the applications QoS is fulfilled in the operation time. Moreover, in this architecture, we will consider the performance and the energy consumption as key factors. 2) We perform a comparative experimental study that reveals the architectural impact on the performance, power and energy consumption on heterogeneous GPUs.

The remainder of this paper is structured as the follows. Section 2 introduces the related work. Section 3 presents the proposed architecture. Section 4 introduces the heterogeneous GPU Benchmarking and Analysis. Section 5 explains the GPU benchmarking experiments and the results. Finally, Section 6 concludes the paper and describes the future work.

2 Related Work

Cloud computing provides the end-user with access to a pool of resources at a suitable price. However, data centres and supercomputers built with CPUs and GPUs that provide GPU applications consume a tremendous amount of electrical power and this will increase the operation cost. For instance, in 2015 the consumed energy of Titan Cray XK7 supercomputer was 8.2 Million Watts [7]. One way to handle the energy consumption is by using virtualisation technologies, for example, a study that was conducted in [19] aims to reduce the number of the physical GPUs in the system by using virtualisation technologies. However, this study does not mention the resource management activities and the impact of these activities on performance and energy consumption.

It is necessary to establish a provisioning framework of resources to support the applications to reduce the costs and achieve QoS. Studies that were performed in [15], [21], [32], [27], [14] and [35] deal with predicting the required resources to execute the applications in Cloud computing to reduce the operation cost and decrease the energy consumption and provide a stable performance to the end-user. In Cloud computing environments, a number of studies that deal with energy efficiency for managing the standard resources, such as CPU, memory and network have been greatly considered. For instance, the studies that were conducted in [22], [20], [18] and [28] propose energy aware mechanisms to manage the resources within Cloud computing environments. However, they do not consider the GPU applications and the supplies of resources to run these applications in Cloud computing.

Although the studies were performed in [24], [30], [25], [31] and [34] deal with GPU scheduling in Cloud computing as first-class scheduling, they merely consider the performance as a key factor for allocating virtual GPUs to physical GPUs, neglecting the energy consumption factor when allocating these VMs. Further, the previous studies do not consider the energy consumption prediction for allocation purposes.

In addition, by using live migration, VMs can be dynamically migrated to a lower number of PMs with the conditions of their requirements [17]. So, it is crucial to develop resource management mechanism in Cloud computing since deployed applications may experience variable workloads which cause a dynamic resource usage. Subsequently, continuous live migration techniques may bring performance degradation when the application demands are not fully fulfilled, and this may cause SLA violations [13]. Therefore, Cloud providers should be aware of the trade-off between energy consumption and performance and operational costs as well. Thus, it is important that Cloud systems include self-adaptive management to automatically fulfil the QoS requirements of the end-user and prevent SLA violations. In [37], an adaptive management framework was conducted to manage GPU resources in Cloud computing, but it merely concerns the performance to meet SLA requirements. In [23], the authors developed an adaptive management framework to guarantee SLA considering the energy efficiency, but they only consider the Cloud gaming perspective and do not deal with GPU applications with general purpose usage in this study.

In terms of GPU power consumption analysis, the authors in [29] presented a study that analyses the correlation between power consumption and workload characteristics. The study was performed in a simulated environment but did not consider heterogeneous GPUs architectures.

Having reviewed the current work on managing heterogeneous GPUs in Cloud computing for general purpose usage, we conclude there is a lack of the research regarding how to manage the life cycle of GPU applications in the deployment and operation time to run these applications in a heterogeneous Cloud infrastructure taking into account two factors: performance and energy consumption. Therefore, we propose a systematic architecture to manage heterogeneous GPUs in a Cloud environment considering performance and the energy consumption as key factors in the deployment and the operation times.

3 The Proposed Architecture

To achieve the research objectives, an adaptive systematic architecture is proposed, as shown in Figure 1, to manage the GPU applications that run within VMs focusing on two parameters: energy consumption and performance in Cloud computing environments in two phases: deployment and operation times. The proposed architecture considers the life cycle of GPU applications in Cloud computing starting from the deployment time to the operation time with the possibility of a self-adaptation framework to maintain the application QoS at the runtime.

Prior to service deployment, the energy prediction modeller will estimate the energy consumed by the GPU applications. This will allow the VM scheduler to allocate the service to the most energy efficient VM. In the operation time, the self-adaptation manager will continuously monitor the application's performance and will take proactive and corrective actions when the performance degrades. The proposed architecture consists of interacting components to achieve the goals of this research, and each component has a certain role as shown next.

The Heterogeneous GPUs Analyser aims to analyse and compare heterogeneous GPU architectures, e.g. Fermi and Kepler, in terms of performance, power and energy consumption, see Section 4. **The Energy Prediction Modeller** is responsible for predicting the energy consumption of running GPU applications on Virtual Machines (VMs) taking into account the power consumption in the deployment and operation phases. **The VM Scheduler** allocates (VMs) to the Physical Machines (PMs) based on the output from the prediction model and during the operation phase. **The Infrastructure Monitor** is responsible for observing the performance and power consumption of the physical infrastructure and sending the monitored data to the self-adaptation manager and the VM Scheduler. **The Self-Adaptation Manager** is a component that ensures that QoS is fulfilled during GPU applications operating within the VMs and implements the MAPE-K [26] (Monitor, Analyse, Plan, Execute and Knowledge) technique. The self-adaptation manager will need to invoke the VM scheduler and the prediction modeller to maintain the application's QoS. The purpose of invoking the prediction modeller is to predict the future behaviour of the application. When the future behaviour of the application shows a performance degradation, the self-adaptation manager will invoke the VM scheduler to reschedule the VM to stabilise the application's performance.

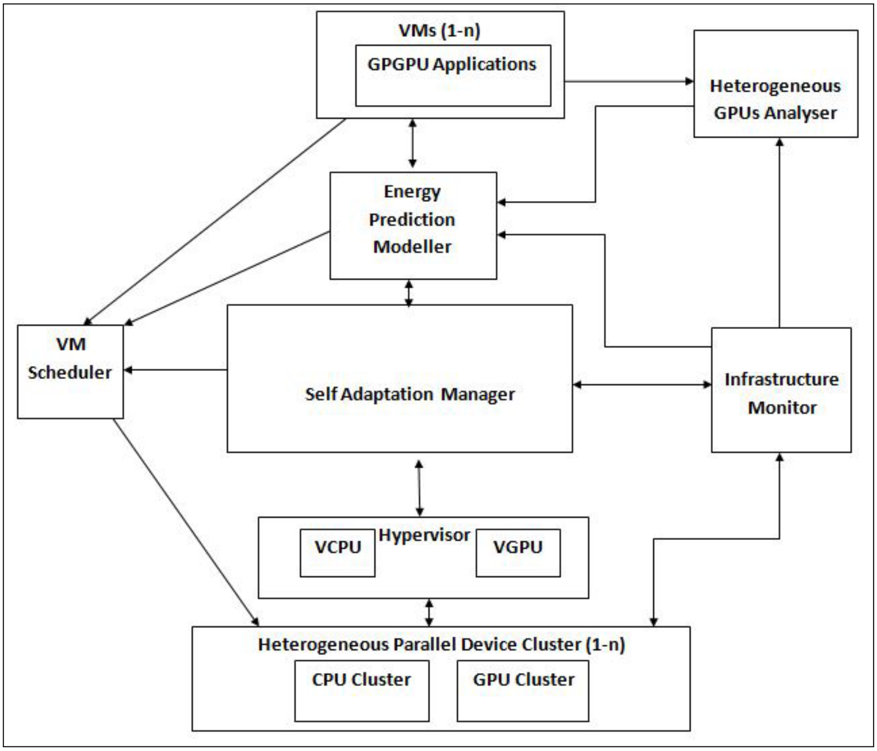

Figure 1: High Level of the Proposed Architecture

4 Heterogeneous GPU Benchmarking and Analysis

As a first step to implement the proposed architecture, it is important to analyse and compare the heterogeneity of the GPU architectures in the Cloud infrastructure in accordance with an adequate resource management development. There are two generations of NVIDIAs GPUs architectures dealt with in this study: Fermi and Kepler. Kepler architecture is newer and more energy efficient than Fermi. C2075 and K40c are examples of Fermi and Kepler architectures respectively.

We analyse the architectural behaviour of GPUs in terms of three criteria: performance, power and energy consumption. We study the impact of the software side on the architecture side of the GPUs in the aforementioned criteria. The software side is defined as the number of blocks and the number of threads per block assigned by the developer to run the kernel which is the function that is executed by the GPU. This is performed by using a specific programming language that deals with GPUs. The selected programming language is CUDA which is supported by NVIDIA. Moreover, we study the factors that have an impact on the performance and power consumption. These factors are the hardware block scheduling, the GPU

Occupancy and the memory hierarchy, such as the device memory. In this study, we do not consider the impact of the CPU and the main memory in terms of performance, power and energy consumption.

The hardware block scheduling can be defined as the number of blocks which can be allocated in a Stream Multiprocessor (SM). We use the equations in the CUDA Occupancy calculator [3] to find the number of blocks allocated in each SM and the GPU Occupancy. To find the number of blocks per SM, we first calculate the number of warps per block in a given kernel, by using the following formula:

$$\#\text{ warps per block} = \frac{\#allocated\ threads\ per\ block}{\#warp\ size} \tag{1}$$

Where warp size = 32 threads

Then, we find the number of blocks per SM by using the following formula:

$$\#\text{ blocks per SM} = min(\#max\ blocks\ per\ SM, \frac{\#max\ warps\ per\ SM}{\#warps\ per\ block}) \tag{2}$$

Occupancy is an important metric to analyse the performance when dealing with GPUs for general purpose use. GPU Occupancy is defined as the ratio of the active number of threads to the maximum number of the threads in the SM. The value of the GPU Occupancy is between 0 and 1. To calculate the GPU Occupancy, we use the following formula:

$$\text{GPU Occupancy} = \frac{\#blocks\ per\ SM \times \#warps\ per\ block}{\#max\ warps\ per\ SM} \tag{3}$$

The percentage of active threads per block is the allocated thread over the maximum number of threads per block. It is calculated by the following formula:

$$\text{Active Threads per Block} = \frac{\#allocated\ threads\ per\ block}{\#max\ number\ of\ threads\ per\ block} \times 100 \tag{4}$$

We assume that the percentage of the active threads per block represents the GPU workload since it is an adequate representative for the GPU utilisation for both GPU architectures.

5 Performing Heterogeneous GPUs Benchmarking and Results

This section will explain the steps of the Heterogeneous GPUs Analysis.

5.1 Experimental Setup and Design

The experiments are performed in the School of Computing Cloud testbed at the University of Leeds. The experiments are performed on two different Virtual Machines (VMs) supported by two heterogeneous GPUs. These heterogeneous GPUs are NVIDIA Fermi C2075 and NVIDIA Kepler K40c. OpenNebula [10] is used as a Virtual Infrastructure Manager (VIM). The KVM hypervisor is used. Additionally, the Operating System (OS) used is Linux CentOS. Table 1 shows the resources of each VM, and Table 2 shows the details of Fermi C2075 and Kepler K40c GPUs.

CPU	Intel Xeon E5-2630 v3 2.4GHz	Intel Xeon E5-2630 v3 2.4GHz
VCPU	8	8
RAM Size	32 GB	64 GB
GPU	NVIDIA Fermi C2075	NVIDIA Kepler K40c
Hypervisor	KVM	
CUDA Compiler Version	7.5	
OS	Linux CentOS	
VIM	OpenNebula	

Table 1
VMs Details

Details	**F**ermi C2075	**K**epler K40c
CUDA Cores	448	2880
SMs	14	15
Cores/SM	32	192
Core frequency(MHz)	1150	745
Memory Size (GB)	6	12
Max Power Consumption (W)	225	235
Max Threads/ Block	1024	1024
Max Warp/SM	48	64
Max Thread Blocks/SM	8	16

Table 2
Fermi C2075 and Kepler K40c GPUs Characteristics

We use a CUDA matrix multiplication application with $o\ (n^3)$ complexity in these experiments. CUDA Compiler Version 7.5 is used to compile the matrix multiplication CUDA codes. We use several tools supported by NVIDIA, as shown in Figure 2. We choose the NVIDIA CUDA Complier (NVCC) to compile the different matrix multiplication application sizes. We use the NVIDIA System Management Interface (nvidia-smi) [8] monitoring tool to profile the GPU power consumption and the temperature at the runtime. Additionally, the NVIDIA Profiler (nvprof) [12] is utilised to measure the hardware performance counters in the runtime.

The objectives of experiments are to:

- Investigate the relationship between GPU workload and power consumption and influential factors as well
- Explore the blocks and the threads per block allocation's impact on energy consumption
- Explore the temperature impact on power consumption

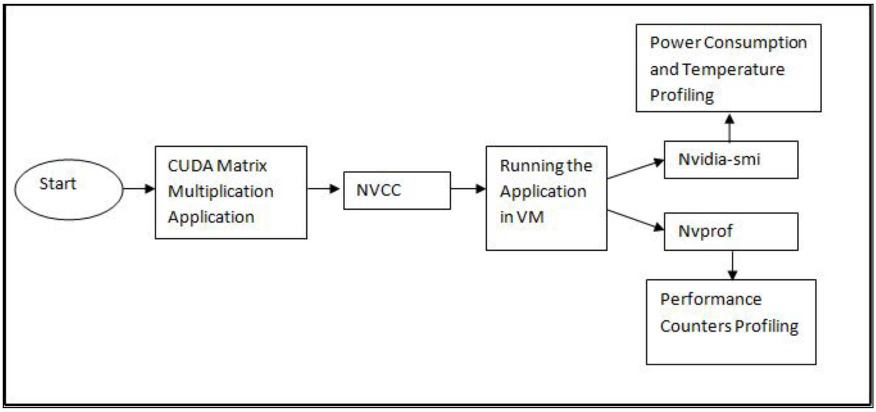
Start
CUDA Matrix Multiplication Application
NVCC
Running the Application in VM
Nvidia-smi
Power Consumption and Temperature Profiling
Nvprof
Performance Counters Profiling

Figure 2: The Analysis Workflow and the Utilised Tools

5.2 *Relationship between GPU Workload and Power Consumption and Influential Factors*

The aim of the design of this experiment is to find the relationship between the GPU workload and the GPU power consumption in both Fermi C2075 and Kepler K40c GPUs and the influential factors on performance and power consumption.

We gradually increased the number of the threads per block up to the maximum number (1024 threads per block), and froze the number of blocks. The number of blocks was 80 x 80 to ensure that SMs were working simultaneously. By increasing the number of threads per block, we increased the size of the memory as well. Then, we ran each matrix multiplication size five times and calculated the average of the power consumption and the execution time. We profiled the GPU power consumption every 50 milliseconds.

5.2.1 *Fermi C2075 Results*

Table 3 shows the results of this experiment in the Fermi C2075 GPU. We applied the regression analysis (linear and nonlinear) to find the relationship between power consumption and the active threads per block. After applying this analysis, we found that the relationship tends to be more nonlinear by applying the quadratic regression since the R-square value in the quadratic regression is greater than the R-square value (0.9528) in the linear regression (0.4225). Figure 3 shows this relationship between the active threads per block and the power consumption in C2075

Fermi GPU.

Matrix size	Threads Number	Active Threads per Block	Average Execution time (s)	Average Power Consumption (W)
480x480	36	4%	0.00828	87.79
800x800	100	10%	0.02717	94.48
1120x1120	196	19%	0.06493	126.27
1440x1440	324	32%	0.12215	137.53
1760x1760	484	47%	0.2238	149.11
2080x2080	676	66%	0.39725	160.05
2400x2400	900	88%	0.56202	136.03
2560x2560	1024	100%	0.65631	133.51

Table 3
The Results in Fermi C2075 GPU

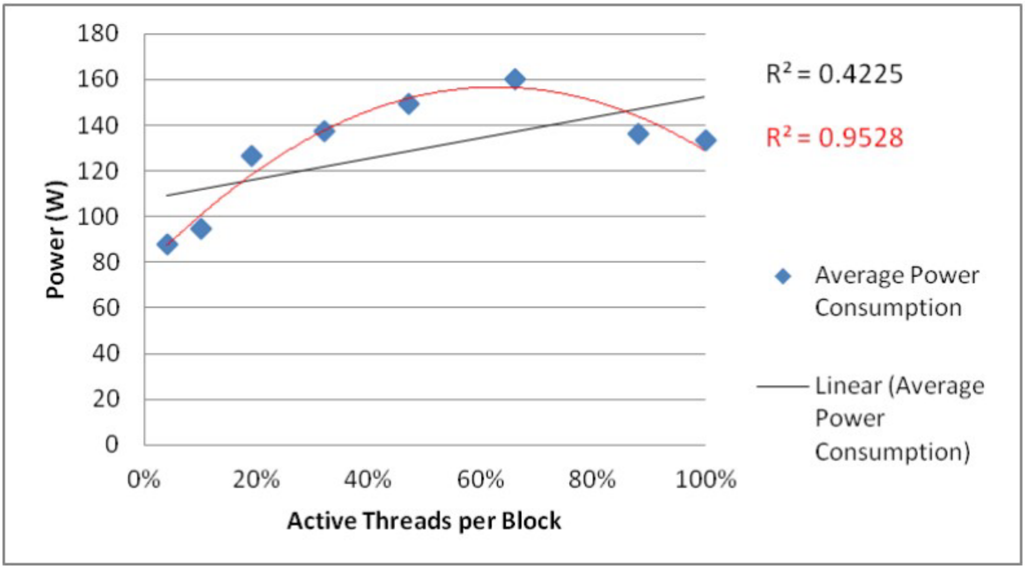

Figure 3: The Regression Analysis Power Consumption and the Active Threads per Block in Fermi C2075 GPU

5.2.2 *Kepler K40c Results*

Table 4 shows the results of this experiment. We applied the regression analysis (linear and nonlinear) to find the relationship between power consumption and the workload in Figure 4. After applying this analysis, we found that the relationship tends to be nonlinear, applying the quadratic regression, since the R-square value in the quadratic regression is greater than the R-square value in the linear regression. However, the difference between them is not so high, being .9875 and .8976 respectively.

Matrix Size	Threads Number	Active Threads per block	Average Execution time (s)	Average Power Consumption (W)
480x480	36	4%	0.00835	52.5
800x800	100	10%	0.02032	57.34
1120x1120	196	19%	0.0468	76.09
1440x1440	324	32%	0.08846	80.3
1760x1760	484	47%	0.15599	98.4
2080x2080	676	66%	0.24055	106.43
2400x2400	900	88%	0.38215	111
2560x2560	1024	100%	0.2745	111.57

Table 4
The Results in Kepler K40c GPU

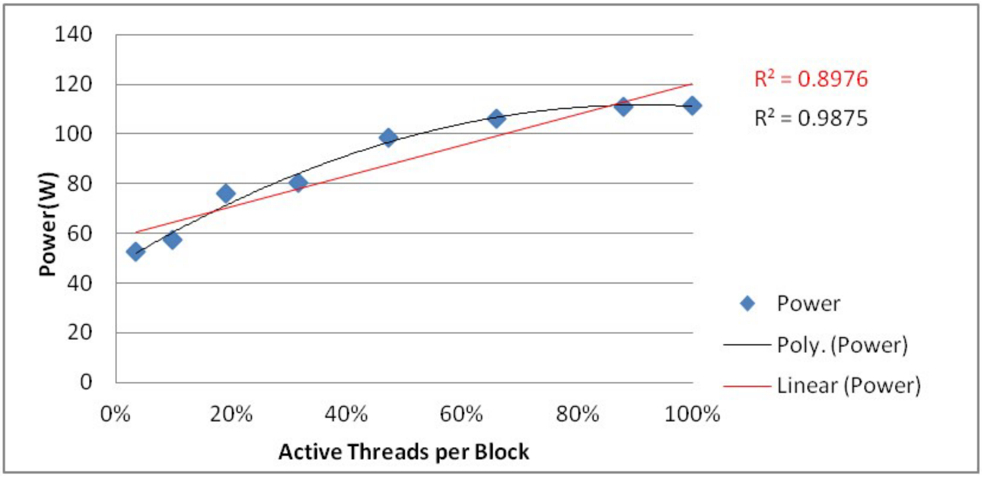

Figure 4: The Regression Analysis Power Consumption and the Active Threads per Block in Kepler K40c GPU

5.2.3 Results Analysis

Considering Fermi C2075 GPU, we found that there is a gradual increase in power consumption up to a certain level of the number of threads per block percentage, when the active threads per block percentage is 66%. After that, power consumption significantly decreases to 136 Watts. To explain the trend of the power consumption during an increase in the GPU workload, we need to compose a performance and architectural analysis for the applications running on the Fermi C2075 GPU at the runtime.

After analysing the GPU microarchitecture disposal at the runtime by applying the hardware performance counters, we found that the behaviours of some these counters have unexpected values, specifically, the memories behaviour, such as the device memory, L2 and the L1 cache memories. Table 5 shows the values of the performance counters related to some memory types for the 2080 x 2080, 2400 x 2400 and 2560 x 2560 matrices because the drop of power consumption began when the size of the matrix was 2080 x 2080.

We found the performance counters values the 2080 x 2080 matrix size was greater than the performance counters values in 2400 x 2400 and similarly in 2560 x 2560, even their memory size (2400 x 2400 and 2560 x 2560) was larger than the previous one. Yet, the counter value of `gst_transactions` in the 2400 x 2400 matrix was greater than the counter value of `gst_transactions` in the 2080 x 2080 matrix.

Counter Name	Counter Description	Counter Value (2080x2080)	Counter Value (2400x2400)	Counter Value (2560x2560)
gst_transactions	Global Store Transactions	411362	520331	204960
dram_read_throughput	Device Memory Read Throughput	19.179GB/s	3.3041GB/s	2.8727GB/s
dram_write_throughput	Device Memory Write Throughput	94.811MB/s	61.674MB/s	47.643MB/s
l2_l1_read_hit_rate	L2 Hit Rate (L1 Reads)	91.85%	76.42%	67.85%
l2_read_transactions	L2 Read Transactions	1960588232	174374480	131120172

Table 5
Performance Counters values of the Memory types in Fermi C2075 GPU

Then, when increasing the number of threads per block, the way to scheduling these blocks in each SM, shown in Figure 5, was not fixed. The number of blocks that were allocated to the SM decreased when the active threads per block percentage were increased.

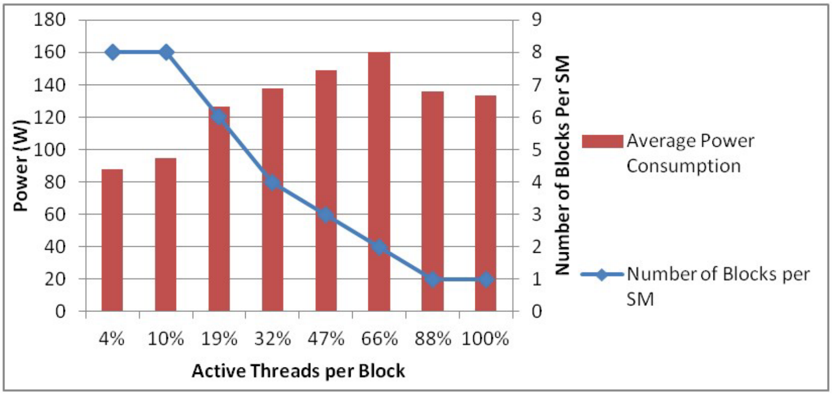

Figure 5: The Number of Blocks per SM in Fermi C2075 GPU

After calculating the GPU Occupancy for each workload (in Figure 6), we found that power consumption was affected by the GPU Occupancy. Therefore, even by increasing the size of memory and the number of threads per block in the Fermi C2075 GPU, power consumption went approximately towards the GPU Occupancy value. When the percentage of active threads per block was 66%, the GPU Occupancy was greater than the GPU Occupancy when the percentage of active threads per block was 88%. Thus, the 2080 x 2080 matrix that has 66% of active threads per block consumes more power.

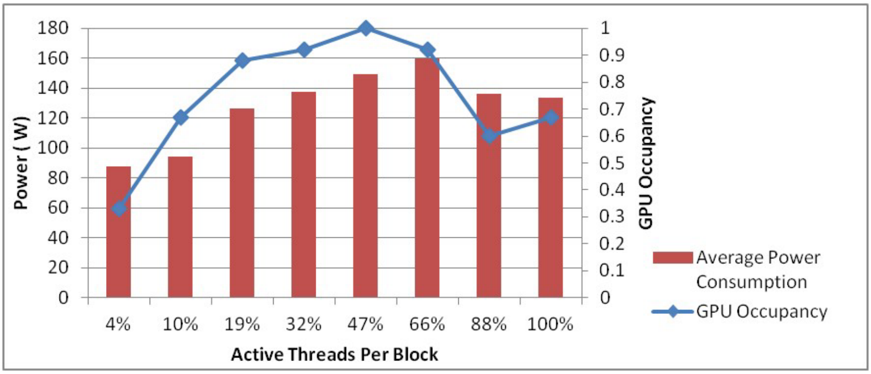

Figure 6: the Power Consumption and GPU Occupancy Values in Fermi C2075 GPU

For Kepler K40c GPU, we found that there was a gradual increase in power consumption up to a certain level of the active threads per block percentage, and the active threads percentage per block was 88%. Then, power consumption values were equal when the percentage of the active threads per block was increased. However, GPU Occupancy has a remarkable impact on the performance. Even the memory size of the 2560 x 2560 matrix is greater than the memory size of 2400 x 2400 matrix, the execution time of 2560 x 2560 is lower than the execution time of 2400 x 2400, as shown in Table 4.

For Kepler K40c, after profiling the same performance counters that were used in Fermi C2075 GPU especially with memory behaviour counters, we found that some of these counters values were correlated with the power consumption trend and a decrease when increasing the workload as well, shown in Table 6. These counters are: `gst_transactions`, `l2_l1_read_hit_rate` and `l2_read_transactions`. The aforementioned performance counters have values contrary to memory size and the active threads per block percentage. Even the size of the 2560 x 2560 matrix is larger than the size of 2400 x 2400 matrix, the values of the aforementioned counters in the 2560 x 2560 matrix is smaller than counters values in the 2400 x 2400 matrix.

We then analysed the effectiveness of scheduling the blocks into SMs on the power consumption in Kepler K40c GPU, shown in Figure 7. When increasing the number of threads per block, the way of scheduling these blocks was not fixed. The number of blocks allocated to the SM decreased when the number of the threads per block was increased. This number was constant when the percentage of active threads per block was 66%. It was observed that the power consumption values of the last three matrices were close to each other. However, they had different values in the GPU Occupancy and the hardware performance counter.

Counter Name	**Counter Description**	**Counter Value (2080x2080)**	**Counter Value (2400x2400)**	**Counter Value (2560x2560)**
gst_transactions	Global Store Transactions	411200	522000	204800
dram_read_throughput	Device Memory Read Throughput	9.8835GB/s	7.8928GB/s	9.4082GB/s
dram_write_throughput	Device Memory Write Throughput	130.00MB/s	95.954MB/s	113.87MB/s
l2_l1_read_hit_rate	L2 Hit Rate (L1 Reads)	97.29%	97.78%	97.41%
l2_read_transactions	L2 Read Transactions	1747268239	2830189274	2621517916

Table 6
Performance Counters values of the Memory types in Kepler K40c GPU

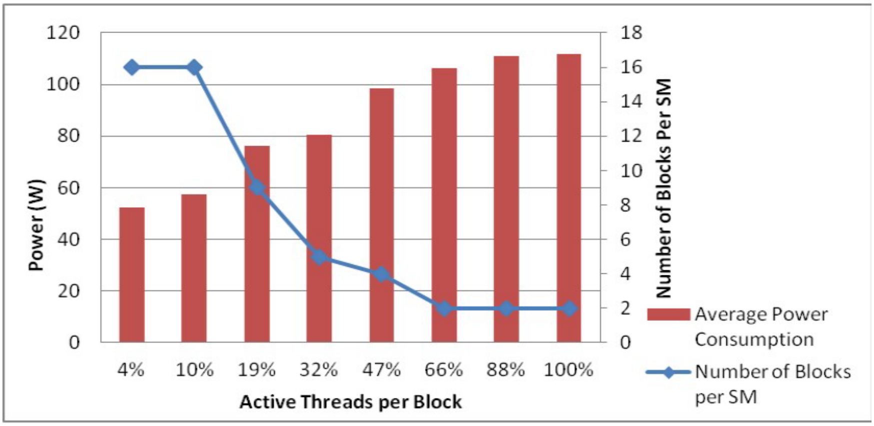

Figure 7: The Number of Blocks per SM in Kepler K40c GPU

For Kepler K40c, we found that the GPU Occupancy values in every workload were greater than or equal to 0.5. Therefore, GPU Occupancy was not a sufficient enough indicator to explain the power consumption trend since the GPU Occupancy here is not correlated with the power consumption values, in Figure 8.

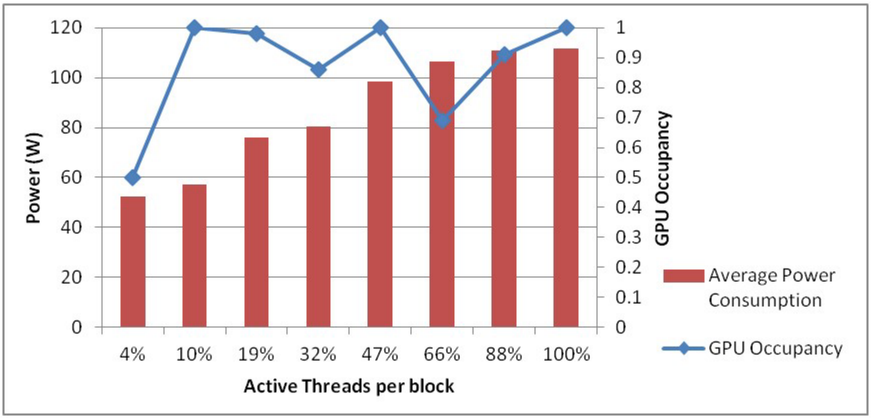

Figure 8: the Power Consumption and GPU Occupancy Values in Kepler K40c GPU

5.3 *The Blocks and the Threads per Block allocations Impact on the Energy Consumption*

The aim of designing this experiment is to explore the impact of the blocks and the threads per block allocations on energy consumption in heterogeneous GPUs architectures.

We selected the matrix multiplication size after dropping the power consumption to analyse the impact on the energy consumption in experiment 1. We implemented the same matrix multiplication size (2400 x 2400) with different workload allocations (a different number of blocks and threads per block). The first implementation had 100 x 100 number of blocks and 24 x 24 number of threads per block. The second implementation had 80 x 80 number of blocks and 30 x 30 number of threads per block. Then, we calculated energy consumption (Joules) by multiplying the execution time (seconds) and the power consumption (Watts). Table 7 shows the execution time and the energy consumption of these matrices in Fermi C2075. We found that there was an energy saving of 5.8% in the matrix that had a larger number of the threads per block and 9.1% in performance loss.

Matrix size	**Number of blocks**	**Number of threads per block**	**Average Power(W)**	**Average Execution time (s)**	**Energy(J)**
2400x2400	100x100	24x24	158.88	0.51102	81.19
2400x2400	80x80	30x30	136.03	0.56202	76.45

Table 7
The Execution time and the Energy Consumption of the Same Matrix size in Fermi C2075 GPU

The second scenario checked the effectiveness of the execution time on the performance and the energy consumption. We increased the block size to five times larger than the block size in the previous experiment for both the matrices to increase the execution time. We repeated the experiment five times and calculated the average of the power consumption and the execution time, as shown in Table 8. Also, even by increasing the execution time in the matrix that had 24 x 24 number of threads per block, there was a 13.1% energy saving with the matrix that had 30 x 30 number of threads per block, which was similar to the previous experiment and had a lower execution time. In this case, by increasing the execution time, the performance loss decreased to 5.6% compared to the previous case.

Matrix size	**Number of blocks**	**Number of threads per block**	**Average Power(W)**	**Average Execution time (s)**	**Energy(J)**
12000x12000	500x500	24x24	179.439	68.355	12265.55
12000x12000	400x400	30x30	147.138	72.427	10656.76

Table 8
The Execution time and the Energy Consumption of the Same Matrix size in Fermi C2075 GPU by increasing the Number of Blocks

Thus, in Fermi C2075 GPU, the energy consumption reduction moved towards the blocks and threads per block allocation which had a lower power consumption.

Therefore, there is an affordable tradeoff between energy consumption and performance in Fermi C2075 GPU in this case. Moreover, the tradeoff between energy consumption and performance reduced when increasing the execution time.

Subsequently, we implemented the exact matrix multiplication size and the same way of calculating energy consumption in the Fermi C2075. Table 9 shows the execution time and the energy consumption of these matrices in Kepler K40c.

In Kepler K40c, we found the opposite situation. We found that there was an energy consumption saving of 9.1% in the matrix that had a faster execution time and a larger number of blocks (100 x 100) since there was no substantial difference in power consumption between the first and the second workload allocation. The power consumption difference was merely 3.08 Watts between them. Then, the second scenario was to increase the block size five times larger than the block size in the previous experiment for both the matrices to increase the execution time similarly in Fermi C2075 GPU, as shown in Table 10. The matrix size that had 24 x 24 number of threads per block was 11.2% more energy efficient. Thus, in Kepler K40 GPU, the energy consumption reduction moved toward the blocks and threads per block allocation which had a fast execution time. Therefore, this experiment can make developers aware of selecting energy aware blocks and threads per block number allocation based on the GPU architecture. In this experiment, Kepler K40c GPU was 46.5% more energy efficient than Fermi C2075 GPU.

Matrix size	**Number of blocks**	**Number of threads per block**	**Average Power(W)**	**Average Execution time (s)**	**Energy(J)**
2400x2400	100x100	24x24	114.08	0.33763	38.51
2400x2400	80x80	30x30	111	0.38215	42.41

Table 9
The Execution Time and the Energy Consumption of the Same Matrix in Kepler K40c GPU

Matrix size	**Number of blocks**	**Number of threads per block**	**Average Power(W)**	**Average Execution time (s)**	**Energy(J)**
12000x12000	500x500	24x24	141.213	40.366	5700.20
12000x12000	400x400	30x30	140.695	45.672	6425.82

Table 10
The Execution time and the Energy Consumption of the Same Matrix size in Kepler K40c GPU by increasing the Number of Blocks

5.4 Temperature Impact on Power Consumption

The aim of the design of this experiment is to explore the temperature impact on the power consumption in Fermi C2075 and Kepler K40c.

We increased the size of the matrix and the size of the blocks to 1000 x 1000. We executed a 2000 x 2000 matrix multiplication application on the both GPUs

(Fermi C2075 and Kepler K40c), see Figure 9 and Figure 10. Power consumption and temperature were profiled every five seconds.

We found that there was a linear increment in power when temperature was increased in both GPUs. However, there was a resistance in power in Kepler K40c GPU at some level but the power consumption continues increased after finishing this resistance. GPU memory utilisation which profiled alignment with power consumption by the nvidia-smi management tool could have influenced the occurrence of this resistance, as shown in Figure 11.

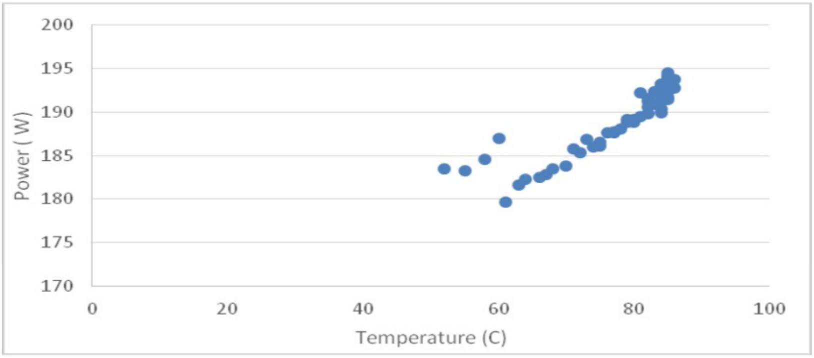

Figure 9: Power Consumption and Temperature in Fermi C2075 GPU

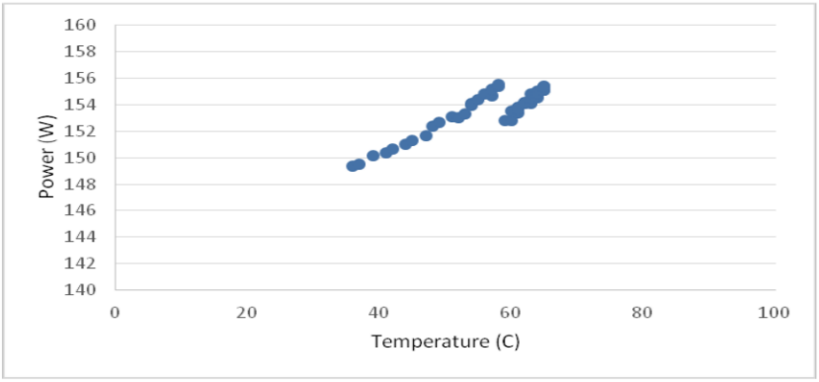

Figure 10: Power consumption and Temperature in Kepler K40c GPU

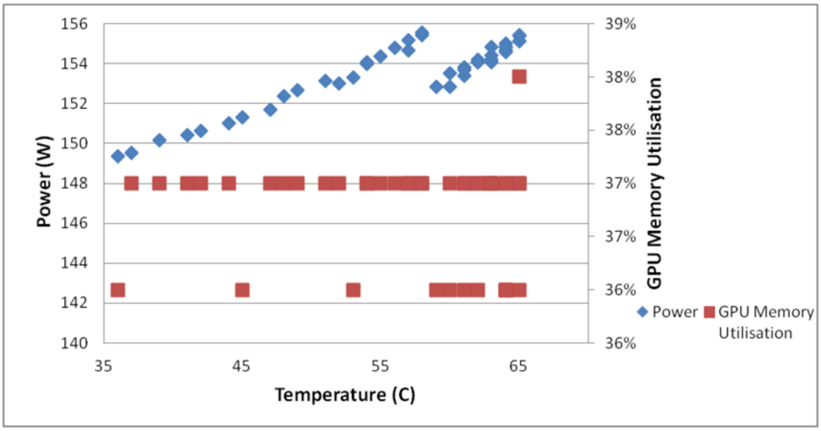

Figure 11: Power Consumption and GPU Memory Utilisation in Kepler K40c GPU

5.5 Overall Discussion of Results

The linearity level between the workload and power consumption in Kepler K40c is greater than Fermi C2075.

In Fermi C2075, raising the block numbers lead to an increase in the hardware resource usage. Thus, the resident blocks number in the SM is decreased from 8 to 1. This reduction produces inefficient parallelism behaviour to cover the instruction pipeline and the memory latency. Therefore, it leads to a performance decrease. This performance decrease affects power consumption. Similarly, GPU Occupancy deceases alongside with the resident blocks in the SM leading to a performance and power decrease. However, in Kepler K40c, although the resident blocks number in the SM is decreased from 16 to 2, the performance of the matrix with the size 2560 x 2560, which is the largest matrix size, not affected by this reduction, and its execution time is faster than the previous matrix, 2400 x 2400. The reason lies in the effectiveness of the GPU Occupancy on performance since the GPU Occupancy of 2560 x 2560 is greater than 2400 x 2400.

GPU memory types have an impact on the power consumption in the Fermi GPU as some types affect the power consumption in the Kepler GPU.

GPU Occupancy, GPU memory types and hardware block scheduling factors have a strong correlation with power consumption in Fermi C2075 GPU. However, the effectiveness of GPU Occupancy on power consumption in Kepler K40c GPU is not reliable. It has a clear effect on the performance. The remainding factors, some GPU memory types and block scheduling, can be considered in terms of effectiveness on the power consumption.

Moreover, blocks and the threads per block allocations affect energy consumption. The impact depends on the type of GPU architecture. In Fermi C2075 GPU, there is a trade-off between the performance and energy consumption. Increasing the number of blocks will increase the performance and also increase energy consumption. However, in Kepler K40c GPU, increasing the number of blocks will increase the performance and become more energy efficient.

Finally, temperature has a strong impact on power consumption for both Fermi and Kepler GPU architectures when the execution time is increased.

6 Conclusion and Future Work

In this paper, we have proposed an adaptive architecture in Cloud computing environments. The aim of this architecture is to manage heterogeneous GPUs resources for general purpose in Cloud computing environments. The architecture considers the deployment and the runtime by focusing on performance and energy consumption factors. The Heterogeneous GPUs Analyser has been introduced as the initial step to develop the aforementioned architecture. The Heterogeneous GPUs Analyser aims to analyse the architectural behaviour of heterogeneous GPUs in terms of performance, power and energy consumption. Additionally, Kepler architecture is 46.5% more energy efficient than the Fermi architecture.

After analysing the heterogeneous GPU architectures in terms of performance, power and energy consumption, a novel energy consumption prediction model will be developed to estimate the energy consumed by the GPU application. The energy consumption prediction model will be developed by selecting the highest influential factors on energy consumption for both GPU architectures. These influential factors will be set as the model inputs. Then, energy efficient scheduling policy will be developed to allocate the GPU applications to the most energy efficient VM. The decision made by the energy efficient scheduling policy will rely upon the energy consumption prediction model. Additionally, this scheduling policy will consider the execution time and the energy consumption as key factors. Finally, we will develop an adaptive management framework to automatically maintain the QoS of the allocated application during the operation time. To maintain the QoS of the GPU applications during the operation time, there should be a trade-off in terms of energy efficiency, performance and cost. Therefore, another research aim is to find the aforementioned trade-off.

References

[1] *Applications that scale using GPU Compute*, https://channel9.msdn.com/Events/Microsoft-Azure/AzureCon-2015/ACON303, 2016-02-22.

[2] *AWS — High Performance Computing - HPC Cloud Computing*, http://aws.amazon.com/hpc/, 2016-02-21.

[3] *CUDA Occupancy Calculator - Nvidia*, 2017-10-23. developer.download.nvidia.com/compute/cuda/CUDA{_}Occupancy{_}calculator.xls,

[4] *Graphics Processing Units (GPU) — Google Cloud Platform*, https://cloud.google.com/gpu/, 2017-08-20.

[5] *IBM Bluemix - GPUs Cloud Computing - More processing power*, https://www.ibm.com/cloud-computing/bluemix/gpu-computing, 2017-08-20.

[6] *Nimbix: High Performance Computing & Supercomputing Platform*, https://www.nimbix.net/, 2017-08-20.

[7] *November 2015 — TOP500 Supercomputer Sites*, http://www.top500.org/lists/2015/11/, 2016-02-22.

[8] *nvidia-smi*, http://developer.download.nvidia.com/compute/DCGM/docs/nvidia-smi-367.38.pdf, 2017-10-20.

[9] *OpenCL - The open standard for parallel programming of heterogeneous systems*, https://www.khronos.org/opencl/, 2016-02-21.

[10] *OpenNebula*, https://opennebula.org/, 2017-10-26.

[11] *Parallel Programming and Computing Platform — CUDA — NVIDIA—NVIDIA*, http://www.nvidia.com/object/cuda{_}home{_}new.html, 2016-02-21.

[12] *Profiler User's Guide*, http://docs.nvidia.com/cuda/profiler-users-guide/index.html{#}gpu-trace-and-api-trace-modes, 2017-10-20.

[13] Beloglazov, A. and R. Buyya, *Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers*, Concurrency Computation Practice and Experience **24** (2012), pp. 1397–1420.

[14] Calheiros, R. N., E. Masoumi, R. Ranjan and R. Buyya, *Workload prediction using ARIMA model and its impact on cloud applications' QoS*, IEEE Transactions on Cloud Computing **3** (2015), pp. 449–458.

[15] Caron, E., F. Desprez and A. Muresan, *Forecasting for grid and cloud computing on-demand resources based on pattern matching*, in: *Proceedings - 2nd IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2010*, 2010, pp. 456–463.

[16] Choi, H. J., D. O. Son, S. G. Kang, J. M. Kim, H.-H. Lee and C. H. Kim, *An efficient scheduling scheme using estimated execution time for heterogeneous computing systems*, Journal of Supercomputing **65** (2013), pp. 886–902.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84881370959{&}partnerID=40{&}md5=b32a9579e35bb92b866752c17cd6302b

[17] Clark, C., K. Fraser, S. Hand, J. G. J. Hansen, E. Jul, C. Limpach, I. Pratt and A. Warfield, *Live migration of virtual machines*, in: *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, Vmm, 2005, pp. 273–286.
URL http://dl.acm.org/citation.cfm?id=1251203.1251223{%}5Cnhttp://dl.acm.org/citation.cfm?id=1251223

[18] Dong, J., X. Jin, H. Wang, Y. Li, P. Zhang and S. Cheng, *Energy-Saving virtual machine placement in cloud data centers*, in: *Proceedings - 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2013*, 2013, pp. 618–624.

[19] Duato, J., A. J. Peña, F. Silla, R. Mayo and E. S. Quintana-Ort, *rCUDA: Reducing the number of GPU-based accelerators in high performance clusters*, in: *Proceedings of the 2010 International Conference on High Performance Computing and Simulation, HPCS 2010*, 2010, pp. 224–231.

[20] Feller, E., C. Rohr, D. Margery and C. Morin, *Energy management in IaaS clouds: A holistic approach*, in: *Proceedings - 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012*, 2012, pp. 204–212.

[21] Gong, Z., X. Gu and J. Wilkes, *PRESS: PRedictive Elastic reSource Scaling for cloud systems*, in: *Proceedings of the 2010 International Conference on Network and Service Management, CNSM 2010*, Vm, 2010, pp. 9–16.

[22] Graubner, P., M. Schmidt and B. Freisleben, *Energy-efficient management of virtual machines in Eucalyptus*, in: *Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011*, 2011, pp. 243–250.

[23] Guan, H., J. Yao, Z. Qi and R. Wang, *Energy-Efficient SLA Guarantees for Virtualized GPU in Cloud Gaming*, IEEE Transactions on Parallel and Distributed Systems **9219** (2015), pp. 1–1.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6881719

[24] Gupta, V., A. Gavrilovska, K. Schwan, H. Kharche, N. Tolia, V. Talwar and P. Ranganathan, *GViM: GPU-accelerated Virtual Machines*, in: *the 3rd ACM Workshop on System-level Virtualization for High Performance Computing*, 2009, pp. 1–8.
URL http://dl.acm.org/citation.cfm?id=1519141papers3://publication/uuid/86E8E696-2579-46C8-84E4-2C3EB09C6914

[25] Gupta, V., K. Schwan, N. Tolia, V. Talwar and P. Ranganathan, *Pegasus: Coordinated Scheduling for Virtualized Accelerator-Based Systems*, in: *2011 USENIX Annual Technical Conference (USENIX ATC'11)*, 2011, p. 31.
URL https://www.usenix.org/conference/usenixatc11/pegasus-coordinated-scheduling-virtualized-accelerator-based-systems

[26] IBM, *An architectural blueprint for autonomic computing*, Technical Report June (2006).
URL http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:An+architectural+blueprint+for+autonomic+computing+.{#}0{%}5Cnhttp://users.encs.concordia.ca/{~}ac/ac-resources/AC{_}Blueprint{_}White{_}Paper{_}4th.pdf

[27] Islam, S., J. Keung, K. Lee and A. Liu, *Empirical prediction models for adaptive resource provisioning in the cloud*, Future Generation Computer Systems **28** (2012), pp. 155–162.
URL http://dx.doi.org/10.1016/j.future.2011.05.027

[28] Kim, N., J. Cho and E. Seo, *Energy-credit scheduler: An energy-aware virtual machine scheduler for cloud systems*, Future Generation Computer Systems **32** (2014), pp. 128–137.
URL http://dx.doi.org/10.1016/j.future.2012.05.019

[29] Lal, S., J. Lucas, M. Andersch, M. Alvarez-Mesa, A. Elhossini and B. Juurlink, *GPGPU workload characteristics and performance analysis*, in: *Proceedings - International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, SAMOS 2014*, Samos Xiv, 2014, pp. 115–124.

[30] Qi, Z., J. Yao, C. Zhang, M. Yu, Z. Yang and H. Guan, *VGRIS: Virtualized GPU Resource Isolation and Scheduling in Cloud Gaming*, ACM Transactions on Architecture and Code Optimization **11** (2014), pp. 1–25.
URL http://dl.acm.org/citation.cfm?doid=2639036.2632216

[31] Ravi, V. T., M. Becchi, G. Agrawal and S. Chakradhar, *Supporting GPU sharing in cloud environments with a transparent runtime consolidation framework*, in: *Proceedings of the 20th international symposium on High performance distributed computing - HPDC '11*, 2011, p. 217.
URL http://portal.acm.org/citation.cfm?doid=1996130.1996160

[32] Roy, N., A. Dubey and A. Gokhale, *Efficient autoscaling in the cloud using predictive models for workload forecasting*, in: *Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011*, 2011, pp. 500–507.

[33] Shehabi, A., S. Smith, N. Horner, I. Azevedo, R. Brown, J. Koomey, E. Masanet, D. Sartor, M. Herrlin and W. Lintner, *United States Data Center Energy Usage Report*, Technical Report September, Lawrence Berkeley National Laboratory, Berkeley, California (2016).
URL https://datacenters.lbl.gov/sites/all/files/DCEnergyUseReport{_}2016.pdf

[34] Suzuki, Y., S. Kato, H. Yamada and K. Kono, *GPUvm: GPU Virtualization at the Hypervisor*, IEEE Transactions on Computers **65** (2016), pp. 2752–2766.

[35] Wang, J., C. Huang, K. He, X. Wang, X. Chen and K. Qin, *An Energy-Aware Resource Allocation Heuristics for VM Scheduling in Cloud*, in: *High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on*, 2013, pp. 587–594.

[36] Yang, C.-T., J.-C. Liu, H.-Y. Wang and C.-H. Hsu, *Implementation of GPU Virtualization Using PCI Pass-through Mechanism*, J. Supercomput. **68** (2014), pp. 183–213.
URL http://dx.doi.org/10.1007/s11227-013-1034-4

[37] Zhang, C., J. Yao, Z. Qi, M. Yu and H. Guan, *vGASA: Adaptive Scheduling Algorithm of Virtualized GPU Resource in Cloud Gaming*, Parallel and Distributed Systems, IEEE Transactions on **25** (2014), pp. 3036–3045.