applied sciences

Article

Agnostic Energy Consumption Models for Heterogeneous GPUs in Cloud Computing

Abdulaziz Alnori [1,*], Karim Djemame [2] and Yousef Alsenani [1]

[1] Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia; yalsenani@kau.edu.sa
[2] School of Computing, University of Leeds, Leeds LS2 9JT, UK; k.djemame@leeds.ac.uk
* Correspondence: asalnori@kau.edu.sa

Abstract: The adoption of cloud computing has grown significantly among individuals and in organizations. According to this growth, Cloud Service Providers have continuously expanded and updated cloud-computing infrastructures, which have become more heterogeneous. Managing these heterogeneous resources in cloud infrastructures while ensuring Quality of Service (QoS) and minimizing energy consumption is a prominent challenge. Therefore, unifying energy consumption models to deal with heterogeneous cloud environments is essential in order to efficiently manage these resources. This paper deeply analyzes factors affecting power consumption and employs these factors to develop power models. Because of the strong correlation between power consumption and energy consumption, the influencing factors on power consumption, with the addition of other factors, are considered when developing energy consumption models to enhance the treatment in heterogeneous infrastructures in cloud computing. These models have been developed for two Virtual Machines (VMs) containing heterogeneous Graphics Processing Units (GPUs) architectures with different features and capabilities. Experiments evaluate the models through a cloud testbed between the actual and predicted values produced by the models. Deep Neural Network (DNN) power models are validated with shallow neural networks using performance counters as inputs. Then, the results are significantly enhanced by 8% when using hybrid inputs (performance counters, GPU and memory utilization). Moreover, a DNN energy-agnostic model to abstract the complexity of heterogeneous GPU architectures is presented for the two VMs. A comparison between the standard and agnostic energy models containing common inputs is conducted in each VM. Agnostic energy models with common inputs for both VMs show a slight enhancement in accuracy with input reduction.

Keywords: cloud computing; GPU; power modeling; energy modeling; machine learning

check for updates

Citation: Alnori, A.; Djemame, K.; Alsenani, Y. Agnostic Energy Consumption Models for Heterogeneous GPUs in Cloud Computing. *Appl. Sci.* **2024**, *14*, 2385. https://doi.org/10.3390/app14062385

Academic Editor: Jose Machado

Received: 6 February 2024
Revised: 29 February 2024
Accepted: 4 March 2024
Published: 12 March 2024

1. Introduction

Cloud-computing usage has grown significantly, and its popularity has rapidly increased in recent years. In 2021, around 50% of the total workload was executed outside the organization's servers [1]. The size of the global market for public cloud computing has increased significantly in recent years; the market size was approximately 58.6 billion US dollars in 2009, and it is expected to grow to 362.3 billion US dollars in 2022 [2]. Due to the rapid and continuous growth of cloud computing, cloud service providers (CSPs) need to expand and update their cloud infrastructures to cover the cloud users' demands. However, several challenges arise due to the expansion of the cloud infrastructures to fit their users' requests. To expand the cloud infrastructure to cover the rapid increase of cloud user requests, CSPs need to address the issue of hardware resource heterogeneity. Different types of processors, storage, and other resources are used in cloud-computing infrastructures. Moreover, cloud-computing infrastructures have widely adopted accelerator units, such as Graphics Processing Units (GPUs), to boost their computing power capabilities in order to cover the rapid demand for intensive computing and data applications. Several prominent cloud-computing providers including Amazon [3] have enabled the usage of different types of GPUs for their customers. Because of the programmability of GPUs, GPU usage has shifted from its standard purpose, showing images and video games on computers' screens, to computational usage, and this shows a substantial leap in the execution of high-performance intensive parallelism. Thus, this new trend in GPU usage has been utilized in a wide range of fields and applications, producing a significant performance increase. Moreover, the nature of the GPU architecture facilitates the spread of the usage of Deep Learning (DL) applications, such as image classification [4]. Training DL applications is a time-consuming task with CPUs since these applications contain massive amounts of data that need to be trained. Therefore, GPUs dramatically reduce the DL training phase.

Maintaining the services' performance for end-users' demands can lead to performing some operations in the cloud infrastructure, such as live migration. These operations produce unexpected operational costs. Energy consumption is considered one of the main costs incurred by CSPs [5]. For instance, 42% of the total budget for Amazon EC2 was spent on energy consumption, divided between 19% for direct power consumption and 23% for cooling operations [6]. According to [7], in 2030, data centers will consume between 3% and 13% of global electricity compared to 1% in 2010. The estimated amount of energy consumed by these data centers will be approximately 8000 TWh in the worst-case scenario. Moreover, this increase in energy consumption harms the environment. ICT sectors produce 2% of the total CO_2 emissions globally, while data centers produce 0.3% [8]. Therefore, finding solutions to reduce the energy consumed by cloud-computing infrastructures is essential. The accurate and direct measuring of power and energy consumption is not always achievable. Therefore, energy models are an alternative way to predict energy with a decent level of accuracy by using several modeling techniques, such as Machine Learning (ML) techniques. Power and energy modeling can be beneficial in several ways, such as optimizing the power consumption of applications without harmfully affecting performance. Energy modeling can also save time and financial expenses [9]. Since there is a strong relationship between power consumption and energy consumption, factors influencing power consumption will be reused for energy models with other factors.

Different modeling techniques are adopted to model GPU power consumption, most of which are performed in non-cloud environments. Studies presented in [10,11] have presented analytical models using low-level micro-architectural components. This approach can produce acceptable estimation results. Yet, such kinds of models need a deep knowledge of the GPU micro-architecture [12], and it is hard to acquire information on heterogeneous GPU micro-architecture components when running several applications, since the power consumption of each micro-architectural component needs to be measured and the access rate when running every application needs to be calculated [13]. Other studies, such as [14,15], have used low-level micro-architectural GPU power modeling and are aimed to be used as GPU power simulators. Simulation tools are useful for cost reduction. However, these simulations produce significant overhead and require a high level of knowledge for the configuration setup. Additionally, simulation tools are unsuitable for several generations of GPU architectures such as the Kepler architecture [12]. Therefore, we consider using the Machine Learning (ML) approach to deal with these difficulties, and it can produce remarkably accurate results as well. Even though there are already several research studies considering ML for the development of GPU power consumption models like [16,17], they use hardware performance counters as input factors in non-cloud environments.

Moreover, modeling energy consumption is not an effortless task. Serval studies can be performed to model energy consumption since different factors should be handled. Some recent studies have developed energy models. Hardware heterogeneity increases the difficulty of developing an energy consumption model. Although some studies deal with heterogeneous GPUs in terms of energy modeling, they handle these GPUs individually or handle homogeneous GPUs performed in non-cloud environments, such as [9,18]. A few studies have been performed in cloud-computing environments.

Therefore, this work aims to investigate other factors affecting GPU power consumption and consider them in the development of GPU power models for heterogenous GPU architectures combined with performance counters in cloud computing. Moreover, this work aims to develop a unified GPU energy model to handle different GPU architectures in cloud computing. The contributions of this work are as follows:

- Novel GPU power consumption models considering hybrid inputs for heterogeneous GPU architectures in cloud computing. This model aims to predict the power consumed by GPU applications running on a heterogeneous cloud-computing infrastructure.
- A novel agnostic GPU energy consumption model. This model aims to automatically estimate the energy consumed by GPU applications executing on a heterogeneous cloud-computing infrastructure and abstract the heterogeneity of GPU architectures.

The remaining sections in this work are structured as follows: Section 2 presents the related work in GPU power-modeling and GPU energy-modeling techniques in cloud-computing and non-cloud-computing environments. Section 3 discusses the power and energy models in detail. This section begins with the details of developing the GPU power model. The GPU energy consumption model development details are then introduced. Section 4 shows the results of this work. Section 5 concludes this work and discusses future work directions.

2. Related Work

In this section, we discuss studies in GPU power modeling and energy modeling in non-cloud- and cloud-computing environments. Several studies investigated GPU energy modeling in cloud and non-cloud environments using machine-learning algorithms and performance counters including [9,10,19–21]. However, these studies did not consider the heterogeneity of GPU architectures.

Several studies have been performed for GPU power modeling in conventional (non-cloud) systems with different techniques. For example, an agnostic power model for heterogeneous GPU architectures was proposed by Abe et al. [17] using multiple linear regression. The authors used hardware performance counters to develop the model by dividing them into two categories. These categories were GPU core frequency counters and GPU memory counters. The model was applied to GTX 285, GTX 460, GTX 480, and GTX GPUs. However, the authors have merely investigated the linear models to build the power model, which is not appropriate for modern GPU generations. Moreover, an online GPU power estimation model was proposed by Adhinarayanan et al. [12]. The model aimed to instantaneously estimate the power consumption of GPU applications on the runtime. The model only relied upon performance counters that were carefully selected using correlation analysis. The authors used several types of statistical regression techniques. These statistical regression models were Simple Linear Regression (SLR), Multiple Linear Regression (MLR), MLR with Interaction, Basic Quadratic Regression, and Quadratic Regression with Interaction. The authors grouped the power model into application-independent and application-dependent models. The model was evaluated on two heterogeneous GPU generations from NVIDIA (Fermi C2075 and Kepler K20c NVIDIA GPUs, Taiwan Semiconductor Manufacturing Company (TSMC), Hsinchu, Taiwan). However, the power models have produced overheads in some applications at runtime and dealt just with performance counters to develop GPU power models. Braun et al. [22] proposed an accurate, simple, and portable GPU power model by using the Random Forest (RF) algorithm. Independent source code instructions, such as memory instructions and kernel configurations like block numbers, were used as the model's inputs. Several CUDA application types of GPU architectures were used to evaluate the model. However, the model has not performed well in applications with a short execution time, and they merely focused on the software side to build the model.

Boughzala et al. [23] presented a simple and lightweight energy consumption prediction model for CUDA applications using a simulation tool in non-cloud environments. Instead of adopting hardware performance counters and low-level architectural details, the model merely considered the block number per Stream Multiprocessor (SM) as an essential input parameter. To model energy consumption, a performance model was developed using simple linear regression. Then, the energy consumed by a single block was calculated including static and dynamic energy consumption. Finally, the energy consumption of all number of blocks in SMs was calculated by integrating with single linear regression. The model was conducted on a simulation tool called SimGrid [24] to simulate two heterogeneous NVIDIA GPU architectures and compare the simulated energy values with the real measured energy values. The selected GPUs were Kepler K20Xm and Tesla M2075, Taiwan Semiconductor Manufacturing Company (TSMC), Hsinchu, Taiwan. However, the model is not applicable to all CUDA applications, and the energy model is not able to estimate the energy consumption of CUDA applications that have a small number of blocks. Moreover, simulation tools produce an unavoidable amount of overhead compared to real experiments.

Makaratzis et al. developed an analytical energy model based on the binary model to estimate GPU energy consumption [13]. The authors focused on GPU-intensive applications. The used GPUs were Maxwell GeForce GTX 980, Kepler Tesla K2 and Pascal Tesla P100 NVIDIA GPUs, Taiwan Semiconductor Manufacturing Company (TSMC), Hsinchu, Taiwan. The authors claimed that the developed model is suitable for integration with cloud environment simulators. However, the used technique in this work required a deep knowledge of the GPU architecture and was created in several substages.

Therefore, after the related studies have been investigated, this work will investigate the feasibility of merely considering hardware performance counters for developing a power consumption model evaluated on heterogeneous GPU architectures using DNN. A further investigation will be performed to determine other factors that enhance power models. Then, an agnostic and straightforward energy model, without conducting substages to estimate energy consumption, is developed in a cloud-computing environment.

3. Power and Energy Models

This section will delve into the power and energy models and is divided into two main parts. The first part will discuss the methodology of developing the power consumption model in detail, while the second part will cover the steps of developing the agnostic energy model. This work distinguishes between power and energy consumption as follows:

Power consumption is the electrical usage when conducting a workload at a certain level of time; power consumption is measured in Watts (W).

On the other hand, energy consumption is the amount of the average power consumption (P) within a period of time (T) until we finish executing the workload measured by Seconds (S); energy consumption (E) can be measured in Joules (J), and it is represented by Equation (1):

$$E = P \times T \tag{1}$$

According to Equation (1), there is a strong relationship between power and energy. When power is increased, energy consumption is also increased, and vice versa. The objective of this work is to predict energy consumption when GPU applications are executed on VMs in cloud-computing environments through ML techniques, more specifically, shallow neural networks and Deep Neural Networks (DNNs). We separately develop GPU power and energy consumption for several reasons. GPU power and energy consumption can be beneficially used for different purposes. GPU power consumption models are used to develop and optimize GPU architectures for power consumption [15]. GPU energy models are utilized for energy-aware GPU resource management [25]. Unlike power consumption, energy consumption is affected by other factors, such as execution time and temperature in the long execution time. Moreover, we develop GPU power consumption for applications with short execution times like [26], and we develop GPU energy consumption for applications having long execution times for measurable energy consumption. Cloud environments consist of heterogeneous infrastructures. The heterogeneity type in this research deals with heterogeneous GPUs, each of which contains different characteristics and capabilities. In this work, we handle two different GPU generations. The GPUs used for the power and energy models are Nvidia Fermi C2075 and Nvidia Kepler K40c, Taiwan Semiconductor Manufacturing Company (TSMC), Hsinchu, Taiwan.

3.1. Power Models

This section illustrates the methodology for developing the power consumption model in detail. Firstly, we present the execution of GPU applications in the experiment. Secondly, we provide a brief overview of the data collection stage. Thirdly, we delve into the input selection process utilized in the model. Then, we will comprehensively examine the power model design. Furthermore, we showcase the training of the model. Finally, we will discuss the hybrid input model.

(1) Applications

The power model involves the training phase that contains collected data from several GPU applications to develop, train, and test the prediction accuracy of the developed model. Several GPU applications that contain different computational and communication characteristics and implementation techniques are selected to train and test the model, and evaluate the strength of the model as well. Different GPU application types including compute-intensive and memory-intensive applications are considered to develop the power model. N-body is an example of a compute-intensive application, and Spmv is an example of a memory-intensive application, chosen to harden and generalize the model's usage. Selected applications are found in several fields, such as linear algebra, image processing, simulation, and fluid dynamics. The main dataset is divided into training and testing sets. The training phase contains 30 applications, and the testing phase contains 10 applications. The applications in the training and testing phases are selected from well-known benchmarks in the field of GPU computing. These benchmarks are Rodinia [27], Parboil [28], and CUDA SDK applications [29]. We separately run two different profiling tools to collect data (nvprof and nvidia-smi), and nvprof has considerable overhead. Additionally, we consider two different GPUs to collect the required data for the model. Table 1 represents the application name, its source, and the application's size in the training set. Table 2 similarly illustrates the details of the selected applications in the testing set.

Table 1. Training set characteristics in the power model.

Application	Source	Description	Size
RecursiveGaussian	Cuda SDK	Gaussian blur implementation using Deriche's recursive way or recursive Gaussian filter	512 × 512
HSOpticalFlow	Cuda SDK	A clear movement of objects estimation in a picture	640 × 480
Interval	Cuda SDK	Interval Newton method calculation	65,536 equations
SobolQRNG	Cuda SDK	Sobol quasirandom sequence implementation	1,000,000 vectors and 1000 dimensions
Reduction	Cuda SDK	Parallel reduction technique implementation	16,777,216 elements
ScalarProd	Cuda SDK	Scalar product implementation	2048 vectors and 131,072 elements
StereoDisparity	Cuda SDK	Stereo disparity computation implementation	1800 × 1800
ThreadFenceReduction	Cuda SDK	Implementation of array reduction operation using thread fence instruction approach	1,048,576 elements; 128 threads; 64 blocks
Sgemm	Parboil	Single matrix multiplication implementation	A × B; A = 2048 × 1984, B = 1984 × 2112
Spmv	Parboil	Sparse matrix vector multiplication implementation	146,689 × 146,689
Stencil	Parboil	3D seven-point stencil implementation	128 × 128 × 32; 200 iterations

Table 1. *Cont.*

Application	Source	Description	Size
Heartwall	Rodinia	Ultrasound image for heart wall tracking	656 × 744 frame
Kmeans	Rodinia	Clustering algorithm	494,020 objects and 34 features
Gaussian	Rodinia	Gaussian elimination technique for solving equations	1024 × 1024
Leukocyte	Rodinia	Microscopy tracking of white blood cells	640 × 480 frames
Nw	Rodinia	Needleman–Wunsch method for optimized DNA alignment	6400 × 6400
Hotspot	Rodinia	Simulation For estimating a processor temperature	4096 × 4096
Dwt2d	Rodinia	Two-dimensional discrete wavelet transform algorithm	3900 × 4200 pixels
CFD	Rodinia	Computational fluid dynamic solver	97,000 elements
AlignedTypes	Cuda SDK	Memory aligned access implementation type	49,999,872 bytes
Binomial Options	Cuda SDK	The European pricing options between seller and buyer	1024 options
BlackScholes	Cuda SDK	The European pricing options using the Black–Scholes model	8,000,000 options and 512 iterations
Dxtc	Cuda SDK	DirectX texture compression algorithm	512 × 512 pixels
ConvolutionSeparable	Cuda SDK	Convolution technique for image filtrations	3072 × 3072 pixels
Histogram	Cuda SDK	Analysis tool for applications	64 and 256 bins
Transpose	Cuda SDK	Matrix transpose implementation	1024 × 1024
FDTD3d	Cuda SDK	Three-dimensional finite difference time domain model	376 × 376 × 376
MergeSort	Cuda SDK	Merge sort implementation	4,194,304 elements
RadixSortThrust	Cuda SDK	Parallel radix sort implementation	1,048,576 elements
GuasirandomGenerator	Cuda SDK	Niederreiter quasirandom sequence implementation	31,148,576 elements

Table 2. Testing set characteristics in the power model.

Application	Source	Size
ConvolutionTexture	Cuda SDK	3072 × 1536 Pixels
FastWalshTransform	Cuda SDK	8,388,608 data length
MontCarlo	Cuda SDK	8192 options and 262,144 paths
Nbody	Cuda SDK	16,640 bodies
Cutcp	Parboil	96,603 atoms
Histo	Parboil	256 × 8192
Mri-q	Parboil	64 × 64 × 64
Bfs	Rodinia	1,000,000 nodes
Streamcluster	Rodinia	65,536 points
Srad_2	Rodinia	8192 × 8192 data points

(2) Data Collection

A built-in nvidia-smi [30] profiling tool is used to collect power consumption, and a built-in nvprof [31] profiling tool is used to collect the hardware performance counters of the GPU architecture components by querying the counter name during the execution of the application in each GPU for 40 real applications in the training and testing sets. An appropriate application size is selected for suitable power consumption measurements since the Kepler GPU has a better performance and is newer than the Fermi GPU. After measuring the power consumption for each application, the average power consumption is calculated. The nvprof tool separately profiles the performance counters of each kernel, and some applications have more than one kernel. If the application has more than one kernel, the average value is calculated to unify all counters' values.

(3) Input Selection

The first GPU power models' inputs mainly rely upon hardware performance counters to evaluate them with other input criteria. Selecting appropriate model inputs can increase the model's accuracy. Each generation of GPU architecture has a different number of hardware performance counters than others. The nvprof tool can profile 98 and 111 performance metrics in the Fermi C2075 and Kepler K40c GPUs (Taiwan Semiconductor Manufacturing Company (TSMC), Hsinchu, Taiwan) [32]. Using all existing performance counters to feed the model can increase the model's accuracy. However, considering all existing performance counters as model inputs will increase the model's training complexity and the cost of data collection because of the overhead incurred by the usage of the profiling tools. Therefore, we develop the candidacy algorithm to find the performance counters that have a reasonable impact on power consumption in each GPU (Fermi and Kepler), as shown in Algorithm 1. It is difficult to statistically compute the significance between each performance counter and power consumption differing from zero in two different GPUs because of the large number of performance counters in each GPU. Thus, we follow [12,33] to develop the candidacy algorithm by using the correlation analysis to find the collection of hardware performance counters that have a reasonable correlation with power consumption in each GPU and assess this way of selecting the model's inputs.

The candidacy algorithm consists of two main phases. Phase 1 aims to find the main performance counters in each GPU that have an impact on power consumption. In the CUDA programming language, there are several implementations to handle the GPU memory types, such as global, shared, and texture memories. The basic implementation for dealing with GPU memories is global memory, which is used in Phase 1. In this phase, we aim to reduce the nvprof profiling tool overhead since we found that some of the training set applications have a significant time delay when the profiling tool is executed. The output of Phase 1 is used as an input for Phase 2. Phase 2 aims to find the performance counters that have a notable impact on power consumption on applications located in the training set.

The algorithm's inputs are all existing performance counters in each GPU (Fermi and Kepler) and all the applications considered in the training set. The algorithm's output is performance counter sets that have reasonable impacts on power consumption that will be used to train the model. In Phase 1, the matrix size is gradually increased starting from 480×480 up to 2560×2560 with a regular increase, to analyze the number of threads per block in terms of power. This increase creates eight different matrix sizes. Then, the power consumption and the performance counter values of each matrix size are profiled. Once the profiling of the power consumption and values of certain performance counters in all the matrices is completed, the level of strength between them is measured. The Pearson correlation coefficient is applied to evaluate the strength of the relationship between the power consumption and the performance counter values. The Pearson correlation coefficient aims to measure the linear correlation between two variables sets X and Y between -1 and 1. When the absolute value of the coefficient is 1 or close to 1, this means that there is a linear high correlation. In [33], the authors selected the top 10 performance counters that have a strong correlation with GPU power consumption. However, since we consider two heterogeneous GPUs containing different features, we initially selected the hardware performance counters that are greater or equal to 0.5 as a predefined threshold that represents the middle of the value between 0 and 1, as mentioned in [34]. After applying the Pearson correlation coefficient, a threshold is set to select the Pearson coefficient Pt_c (line 14). The outcome of Phase 1 is the *FC* set which includes the essential performance counter metrics that are greater than the assigned 0.5, and the *FC* set will be used as an input for the second phase.

Similar to Phase 1, the power consumption and the performance counter value of each application in the training set that contains 30 applications are profiled. After profiling the power consumption and performance counters values, another Pearson correlation coefficient $Pt2_c$ is applied between the counter value and the profiled power consumption in all applications in the training set. After applying the correlation analysis in Phase 2, we found that there is low overlapping in the correlation relationship between the selected performance counters and power consumption. The reason behind this low overlapping in Phase 2 is that we applied the Pearson correlation coefficient on 30 applications with different sizes and different characteristics to find the strength of the relationship between each selected performance counter and power consumption. Increasing the number of applications and applying this correlation analysis will show us how strong the correlation is, and this correlation analysis will give us a clear and more confident view of the relationship between each selected performance counter and power consumption, unlike applying the correlation analysis on only one application with a regular increase, such as in Phase 1. Therefore, Ω is calculated to find the average of the correlation analysis values of all selected counters in each GPU and used as an appropriate threshold like [35,36] using Equation (2):

$$\Omega = \sum_{c=1}^{n} \left| \frac{\mathrm{P}}{\mathrm{n}} \right| \tag{2}$$

where P is the value of every performance counter and n is the total number of performance counters in Phase 2 in every used GPU. Therefore, Ω aims to calculate the average of all performance counters in Phase 2 for every GPU after using the absolute value. The value of Ω is 0.232 and 0.087 for Fermi C2075 and Kepler K40c (Taiwan Semiconductor Manufacturing Company (TSMC), Hsinchu, Taiwan), respectively. Then, Ω is assigned to be a threshold value to compare it with each selected counter in Phase 2. If the Pearson correlation coefficient $Pt2_c$ is greater than or equal to the threshold value Ω (line 32), the performance counter metric will be added to the PC set. Otherwise, the metric will be rejected. The candidacy algorithm applies to the performance counters twice on Fermi and Kepler GPUs' performance counters.

According to the candidacy algorithm's outcomes, there are nine and six selected counter metrics for the model inputs for Fermi C2075 and Kepler K40c, respectively. After applying the developed candidacy algorithm, we found that performance counters can not only be considered to develop GPU power models, as shown in Tables 3 and 4.

Table 3. Selected performance counters for Fermi C2075.

Metric Name	Description	$Pt2_c$ Value
Gst_transactions	Transactions of global memory store number	0.281
Ecc_transactions	Error-correcting code memory transactions sent between L2 cache memory and DRAM memory number	0.421
L2_read_transactions	L2 cache memory read transactions number	0.236
L2_write_transactions	L2 cache memory write transactions number	0.281
L2_L1_read_transactions	Memory read transactions requested by L1 cache memory and seen in L2 cache memory number	0.310
L2_L1_write_transactions	Memory write transactions requested by L1 cache memory and seen in L2 cache memory number	0.310
Eligible_warps_per_cycle	The average number of eligible wraps to be issued in every cycle	0.291
DRAM_read_transactions	The GPU device RAM read transactions number	0.293
DRAM_write_transactions	The GPU device RAM write transactions number	0.284

Table 4. Selected performance counters for Kepler K40c.

Metric Name	Description	$Pt2_c$ Value
Gld_transactions_per_request	The average number of transactions of the global memory load conducted for every global memory request	−0.187
Gld_transactions	Transactions of global memory load number	0.102
Inst_per_warp	The average number of instructions run by every warp	0.165
DRAM_read_transactions	The GPU device RAM read transactions number	−0.090
IPC	Instructions performed per cycle number	0.275
Issued_IPC	Issued number of instructions for every cycle	0.214

Algorithm 1: CANDIDACY ALGORITHM

```
Input: C [ ], A [ ]
Output: HC [ ]
1: set HC [ ], P1 [ ], P2 [ ], FC [ ] = null
2: m [ ] = 480 $\times$ 480
3: begin phase 1
4: for each c in C [ ] do
5:     for (i = 1; i $\leq$ 8; i++) do
6:         set PC1 [ ] = null
7:         increase m (the matrix size) gradually
8:         profile average $p_i$ (power consumption) in every matrix size
9:         profile $c_i$ (performance counter) in every matrix size
10:        enqueue $p_i$ into P1 [ ]
11:        enqueue $c_i$ into PC1 [ ]
12:    end for
13:    apply the Pearson correlation test $Pt_c$ between P1 and $PC1_c$
14:    if $|Pt_c| \geq 0.5$ then
15:        enqueue $c_i$ into FC [ ]
16:    else
17:        reject $c_i$
18:    end if
19: end for
20: end phase 1
21: begin phase 2
22: for each c in FC [ ] do
23:    for (i = 1; i $\leq$ 30; i++) do
24:        set PC2 [ ] = null
25:        profile average $p_i$ (power consumption) in every application
26:        profile $c_i$ (performance counter) in every application
27:        enqueue $p_i$ into P2 [ ]
28:        enqueue $c_i$ into PC2 [ ]
29:    end for
30:    apply Pearson correlation test $Pt2_c$ between P2 and $PC2_c$
31:    calculate $\Omega$
32:    if $|Pt2_c| \geq \Omega$ then
33:        enqueue $c_i$ into HC [ ]
34:    else
35:        reject $c_i$
36:    end if
37:    end for
38:    end phase 2
```

(4) Model Design

Some studies such as [26] have confirmed that the linear regression method is appropriate for old GPU architectures. Nevertheless, the study conducted in [16] has argued that linear models can not fit the complexity of modern GPUs. The relationship between the workload and power consumption in both GPUs tends to be non-linear. Therefore, DNN is selected to predict the power consumption for heterogeneous GPUs connected with VMs since DNN is a non-linear model and shows a better performance than other machine-learning algorithms [37].

Since every model's input has a different range of value scale, the collected data in each input attribute have been pre-processed by using the z-score technique since it is a popular technique [38]. The aim of using this technique is to simplify the procedure of model training. The z-score is represented by Equation (3):

$$Z = \frac{X - \mu}{\sigma} \tag{3}$$

where X is the element value—in this case, it will be the counter value for every application; μ is the mean of all application values for a certain selected counter in the training set; and σ is the standard deviation of a single selected counter. We similarly use the process of μ to calculate σ. Then, we calculate Z for the remaining applications on other selected counters. After the inputs are pre-processed, they will be fed to the model's training stage.

The model aims to develop a function f that maps the X inputs set to the Y outputs set f: $(X \rightarrow Y)$ to estimate the power consumption P of a GPU (Fermi and Kepler) connected with a VM. Two power models are developed for the Fermi and Kepler GPUs. For the Fermi power model, the input layer has nine neurons representing every selected input, as shown in Table 3.

After performing some experiments to find the adequate number of hidden layers and neuron numbers in every layer, we found that four hidden layers and eight neurons in each hidden layer are appropriate numbers. Therefore, the model contains four hidden layers to enable the layers to extract features from the data, and each hidden layer contains eight neurons; moreover, the output layer contains one neuron representing the predicted power consumption.

For the Kepler GPU power model, the input layer consists of six neurons indicating the selected inputs for Kepler GPU. Like the Fermi Power model, the Kepler model has four hidden layers, and each hidden layer has eight neurons. Finally, the output layer contains one neuron to represent the predicted power consumption.

Suppose $x \in X = \{x_1,\ x_2,\ x_3,\ \ldots, x_n\}$ is a selected input variable and w_{ij} represents a certain synapse weight S in each layer j which contains i neurons. Every s represents the summation of the product of all w_{ij} with x to every neuron by using Equation (4):

$$s_i = \sum\nolimits_j w_{ij} x_i \tag{4}$$

After obtaining s for every neuron, the value will be passed to the activation function f(s_i) by using the appropriate non-linear function. In this model, the rectified linear unit (ReLU) function is used as an activation function.

Since ReLU cannot be an activation function for the output layer, the activation function in the output layer s_o is the linear activation function.

In DNN, to find the output of every layer l and map them together, Equation (5) can be used to estimate power consumption P_d:

$$P_d = \mathrm{f}(W, X, b) \tag{5}$$

where W is the weight matrix in every layer, X is the input features matrix in every layer, and b is the bias in every layer. In DNN, it is beneficial to randomly initialize weights by small values to avoid the similarity of the values among layers and to rapidly train the model [39]. More specifically, the weights of the synapses in the input layer are randomly initialized under a uniform distribution. The synapse weights in the hidden and output layers are randomly initialized under normal distribution, and bias values are initialized to be zero. Hyper-parameters are the parameters that their values establish before starting to train the model. Some DNN models can have a range of hyper-parameters between ten to fifty based on the model's developer who sets certain hyper-parameters and keeps others as defaults. Hyper-parameters should be tuned carefully by experiments since it is a challenging task [40]. Some hyper-parameters are adjusted in this model, such as the learning rater, number of epochs, and batch size. The learning rate is considered the most substantial type of hyper-parameter [41]. The learning rate is the step size of every iteration to reduce the loss function during the training phase [42], and the suggested value is 0.01 number of epochs which is defined as the iteration number for the training algorithm. The assigned number of epochs is 1000. The batch size is the sample number in the training set used in every step for a faster training process, and it is assigned to be 30.

(5) Model Training

After designing the model, the model training step is established. The loss function L is used to measure the difference between P_d and P_{ac} in each training step using Equation (6):

$$L\left(P_d^i, P_{ac}^i\right) = \frac{1}{2}\left(P_d^i - P_{ac}^i\right)^2 \tag{6}$$

where i is a certain training step. The DNN training process is a time-consuming and hard task [43]. Therefore, the Adam optimization algorithm [44] is the selected algorithm for the model's training stage. Adam is an extension of the classic stochastic gradient descent to update the edge weights. Adam has a better ability to train the model than other training algorithms, such as stochastic gradient descent and AdaGrad [45].

Next, to validate DNN models, a shallow neural network that contains only one hidden layer is developed for the Fermi and Kepler GPUs connected with VMs. We aim to compare the performance of shallow and deep neural networks. The hyper-parameters for this model are identical to the DNN model except for the hidden layer which contains only one layer.

(6) Hybrid Input Model

Since the relationship between the selected performance counters and power consumption in both GPUs is not very correlated, we need another investigation to find other input factors that contain a better correlation. Therefore, another DNN model that contains hybrid inputs is developed. These inputs are a mix of input parameters. These inputs are the selected performance counters combined with the utility of GPU and memory, measured in percentage, to give a comprehensive overview of the GPU behavior during the running of applications for each GPU. The nvidia-smi tool is used to collect GPU and memory utilization. When the Pearson correlation coefficient has been applied on GPU and memory utilization, the $Pt2_c$ values in the Fermi GPU are 0.368 and 0.479, respectively. The $Pt2_c$ values for GPU and memory utilizations in the Kepler GPU are 0.653 and 0.872, respectively. After the candidacy algorithm has been applied on GPU and memory utilization, it can be observed that GPU and memory utilization have an acceptable correlation with power consumption in both GPUs, better than performance counters.

In terms of the model design stage, the number of hidden layers and the output layer is similar to the previous DNN model that only uses performance counters, unless the number of neurons in the input layer is changed in both GPUs since the input parameters are increased. For the Fermi GPU, the number of neurons in the input layer is equivalent to the number of the model's inputs. Therefore, the number of neurons in the input layer is 11 neurons. For the Kepler GPU, the number of neurons in the input layer is 8. The hyper-parameters for this model are similar to the previous DNN, which were created by the only-performance-counters-inputs model, except the number of epochs is 300, which decreased.

3.2. Energy Models

This section illustrates the methodology of developing the energy consumption model in detail. Initially, we outline the execution of GPU applications in the experiment. Subsequently, we will provide a succinct overview of both the data collection and input selection stages. Finally, we will thoroughly discuss the intricacies of the model design.

(1) Applications

The applications that are selected for the energy consumption models are derived from the previous GPU power model. The selected applications are capable of increasing their sizes and being executed for longer periods of time to analyze the increase of the application's execution time on energy consumption. Each application has different patterns and characteristics. The applications' sizes are regularly and gradually increased until the VM cannot execute the application's size based on the GPU resources' computing capabilities that are connected to the VM or when the application's execution time reaches 1 h and 40 min. The range of the execution time is between 5 s up to 1 h and 40 min. Tables 5 and 6 depict the applications in the training set and the testing set in the energy model, respectively.

Table 5. Training set in the energy model.

Application	Size
Gaussian	4000 × 4000
Gaussian	6000 × 6000
Gaussian	8000 × 8000
Gaussian	10,000 × 10,000
Gaussian	12,000 × 12,000
Gaussian	14,000 × 14,000
Gaussian	18,000 × 18,000
Gaussian	20,000 × 20,000
Hotspot	1024 × 1024
Hotspot	2048 × 2048
Hotspot	4096 × 4096
Hotspot	16,384 × 16,384
Nbody	665,600
Nbody	998,400
Nbody	1,331,200
Nbody	1,664,000
Nbody	1,996,800
Nbody	2,329,600
Nbody	2,995,200
Nbody	3,328,000
Srad	3200 × 3200
Srad	4800 × 4800
Srad	6400 × 6400
Srad	8000 × 8000
Srad	9600 × 9600
Srad	11,200 × 11,200
Srad	14,400 × 14,400
Streamcluster	65,536
Streamcluster	131,072
Streamcluster	262,144
Streamcluster	524,288
Streamcluster	2,097,152
Streamcluster	4,194,304

Table 6. Testing set in the energy model.

Application	Size	Abbreviation
Gaussian	16,000 × 16,000	G16000
Hotspot	8192 × 8192	H8192
Nbody	2,662,400	N2662400
Srad	12,800 × 12,800	S12800
Streamcluster	1,048,576	SC1048576

(2) Data Collection and Input Selection

To obtain the energy consumption, the application's execution time should be considered. Moreover, according to the experiments between power and temperature, it has been found that device temperature has a great impact on power consumption when the execution time is increased [46]. Thus, the device temperature also should be considered in order to enhance the energy consumption estimation.

Nvidia-smi is used to profile the temperature, power consumption, and GPU and memory utilizations with 1 s time intervals. The nvidia-smi tool collects power consumption through sensors located in the GPU with an assumed +/− 5% error margin. Then, the average temperature, power consumption, and GPU and memory utilization are calculated after profiling them for every application. Nvprof is used to profile the selected performance counters in the power model. However, nvprof is unable to profile some counters when the application's size and execution time are increased because an overflow happened when profiling these counters. Therefore, the remaining performance counters that nvprof can profile are used as the model's inputs. The remaining counters represent the behavior of GPU memory types.

An agnostic energy model is also developed by selecting the inputs that share both GPUs (Fermi and Kepler). This agnostic energy model aims to standardize the difference and complexity of heterogeneous GPU architectures by selecting the common inputs between the used GPUs. Table 7 illustrates the model's inputs for every VM connected with a different GPU and the agnostic one.

Table 7. The energy model inputs for each VM.

Model Input	VM with C2075	VM with K40c	Common Inputs (Agnostic Model)
Gst_transactions	X		
Ecc_transactions	X		
L2_read_transactions	X		
L2_write_transactions	X		
L2_l1_read_transactions	X		
L2_l1_write_transactions	X		
Dram_read_transactions	X	X	X
Dram_write_transactions	X		
Gld_transactions_per_request		X	
Gld_transactions		X	
Execution time	X	X	X
Temperature	X	X	X
Power consumption	X	X	X
GPU utilization	X	X	X
Memory utilization	X	X	X

(3) Model Design

This model aims to calculate GPU applications' energy consumption and eliminates human intervention directly and automatically. Two different energy models are designed for two VMs that are connected with heterogeneous GPUs (Fermi C2075 and Kepler K40c). The model will calculate the application's energy consumption by Equation (1). However, as has been discussed earlier, other factors affect energy consumption, such as device temperature, GPU utilization, and memory utilization. Therefore, these factors will be considered in order to predict the energy consumption. For every VM connected, there are two groups of models. The first group is for developing the energy model with individual inputs representing every GPU architecture. The second model is for developing the model with common input parameters (the agnostic model) to abstract the heterogeneity of the hardware.

When the relationship between the size of the applications and their execution time was analyzed, we found that most of the relationships are not linear even though the size has been regularly increased. Similarly, the relationship between the size of the applications and power consumption is also not linear. This analysis is performed in both VMs that contain heterogeneous GPUs.

Figure 1 shows the relationship trend between the execution time and the size of Gaussian application running in the Fermi GPU.

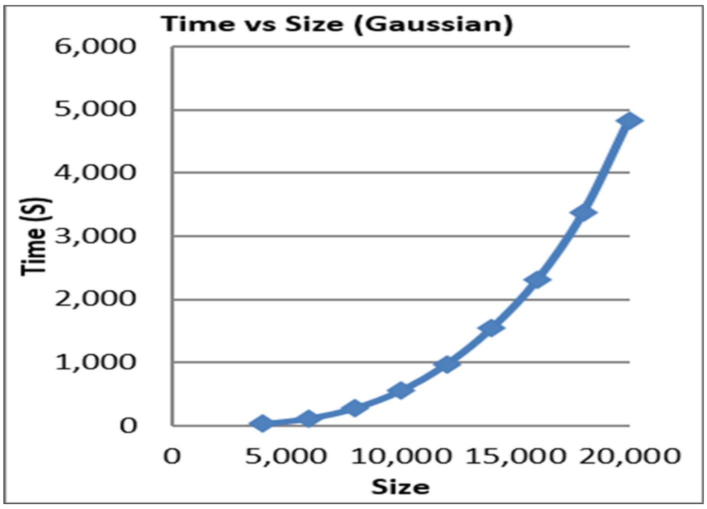

Figure 1. Relationship between execution time and size of the Gaussian application on the VM connected with Fermi GPU.

The calculated energy consumption when the application is increased is not linear. Thus, predicting the application's energy consumption will be complex. Therefore, DNN is an appropriate candidate for automatically calculating energy consumption.

As illustrated in the power model section, the procedures for designing the energy consumption model are similar to those for the power consumption model. Firstly, the model's inputs in the training set have been pre-processed using z-score normalization to unify the scale range. Then, the model has been designed using a DNN to estimate energy consumption E_d by the following equation:

$$E_d = \mathrm{f}(W, X, b) \tag{7}$$

where W is the weight matrix in every layer, X is the input features matrix in every layer, and b is the bias value in every layer. For the VM connected with the Fermi GPU, for the DNN model that has the standard input, the input layer contains 13 neurons equivalent to the number of input attributes. Similar to the power model, the number of hidden layers is four layers, and each layer contains eight neurons. The output layer has one neuron to represent the predicted energy consumption. Another energy model for the VM connected with the Fermi GPU is the common inputs (the agnostic model). The features of this model are similar to the standard model except the number of neurons in the input layer is six neurons and the number of neurons in every hidden layer is six neurons.

For the VM connected with the Kepler GPU, for the structure of the DNN model that has the standard inputs, the input layer contains six neurons equivalent to the number of input attributes, the number of hidden layers is four layers, and each layer contains six neurons. The output layer has one neuron to represent the predicted energy consumption.

Another energy model for the VM connected with the Kepler GPU is with the common inputs (the agnostic model). The input layer has six neurons, and the other layers are identical to the DNN model with the standard inputs for the VM connected with Kepler. Then, we compare the agnostic energy model's accuracy with the standard inputs model. The number of hidden layers in all models is fixed. The selected activation function for all DNN layers except the output layer is the ReLU activation function. The activation function of the output layer is the linear function.

Similar to the power models, the weights of the synapses in the input layer are randomly initialized under the uniform distribution, the weights of the synapses in the hidden and output layers are randomly initialized under normal distribution, and the bias values are initialized to be zero.

The tuned hyper-parameters are the learning rate, epochs, and batch size. The learning rate, epoch, and batch size are 0.01, 1000, and 33, respectively. Then, the energy models are trained using the Adam algorithm to adjust the weights of the edges in the DNN layers, similar to the power models. The difference between the actual energy and the predicted energy is evaluated by the loss function in every training step.

4. Result

This section will first illustrate the details of implementing the experiments and the experiments' setup. Then, the results of the power and hybrid power models in both GPUs will be shown. After that, the energy and agnostic models' results will be given. Finally, a discussion of the results will be illustrated in this section.

4.1. Implementation

To evaluate the developed model, several experiments have been performed in a cloud testbed located in the School of Computing at Leeds University to generate historical data for the models' training and testing. Several applications are selected for their well-known benchmarks and CUDA SDKs containing different features and characteristics to evaluate the models. The DNN models for power and energy predictions are implemented using the Keras platform [47] which is based on the Python programming language.

4.2. Experimental Set-Up

The experiments are performed on two different virtual machines (VMs) supported by two heterogeneous GPUs, mainly used for GPU computing. These heterogeneous GPUs are NVIDIA Fermi C2075 (released in 2010) and NVIDIA Kepler K40c (released in 2012). OpenNebula [48] is used as a Virtual Infrastructure Manager (VIM), and the KVM is the used hypervisor, as shown in Table 8. Additionally, the Operating System (OS) used is Linux CentOS in the VM. Moreover, the used CUDA compiler version is 7.5. The used GPUs have distinctive features and resources, as shown Table 9.

Table 8. Two VMs' details in the cloud testbed.

VM Characteristics	VM1	VM2
CPU	Intel Xeon E5-2630 v3 2.4 GHz	Intel Xeon E5-2630 v3 2.4 GHz
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 9. Fermi C2075 and Kepler K40c GPUs' characteristics.

GPU Details	Fermi C2075	Kepler 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
ECC Mode	Enabled	Enabled

Moreover, the VMs connected with the GPUs were set with different RAM sizes. For both VMs, the physical RAM was allocated to the virtual RAM. In all models, we selected application sizes to be compatible with both GPUs' resources, since the Kepler GPU and the connected VM have better resources and performance, and to enable fairness between the GPUs as well. Additionally, the nvprof profiling tool measured the time of data transfer between the RAM and the GPU, which is included in the execution time. This work does not consider the impact of other factors on energy consumption when GPU applications are executed in a cloud-computing infrastructure, including I/O traffic and networking, since the majority of power consumption is consumed by processing units.

To measure the model accuracy with the actual value for each application in the testing set, the absolute error percentage is utilized, and it is calculated by the following formula:

$$\text{Error }(\%) = \frac{Predicted - Actual}{Actual} \times 100 \quad (8)$$

4.3. Power Models' Results

This section shows the results of power models when the selected performance counters are utilized for developing the GPU power models.

Figure 2 summarizes the difference between the actual and estimated application power consumption of the shallow and deep neural networks when they run on a VM connected with the Fermi C2075 GPU.

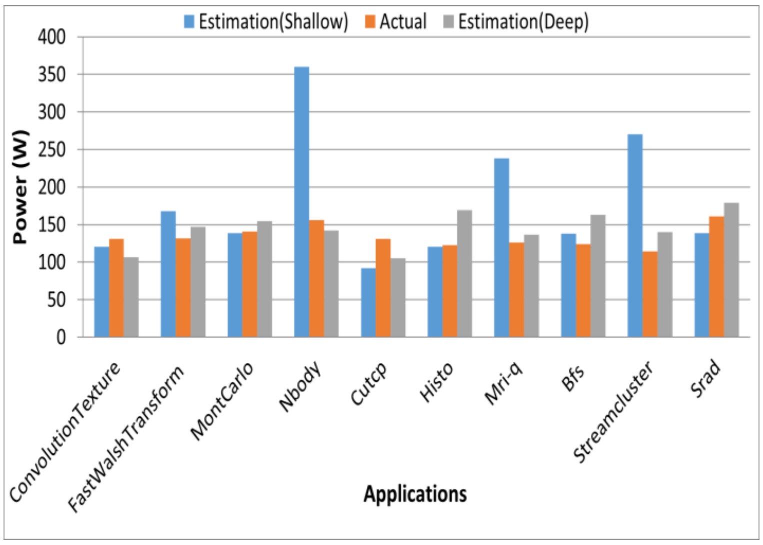

Figure 2. The difference between actual and estimated power consumption in VM connected with Fermi C2075 GPU.

Figure 3 shows the absolute percentage error difference between the shallow and deep neural network models for predicting applications' power consumption when they are executed on a VM connected with the Fermi C2075 GPU.

Figure 4 summarizes the difference between the actual and estimated applications' power consumption of the shallow and deep neural networks running on the VM connected with the Kepler GPU.

Figure 5 depicts the absolute percentage error difference between the shallow and deep neural network models for predicting applications' power consumption when they are executed on a VM connected with the Kepler K40c GPU.

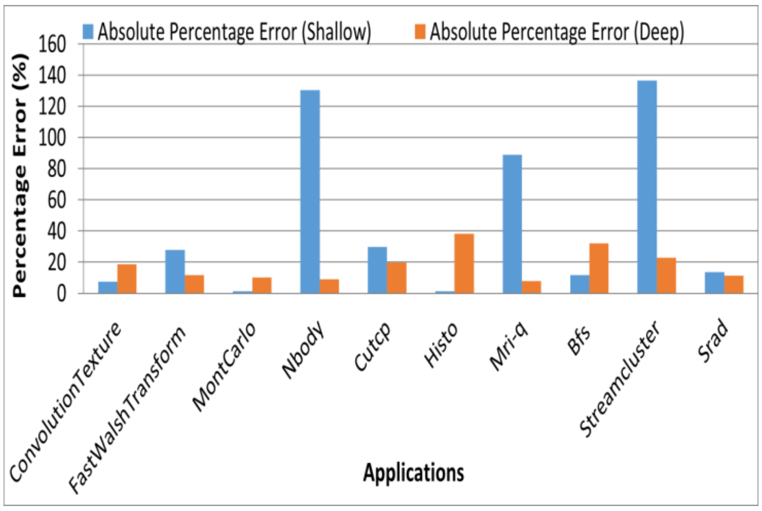

Figure 3. The absolute percentage error between shallow and deep neural network power models in VM connected with Fermi C2075 GPU.

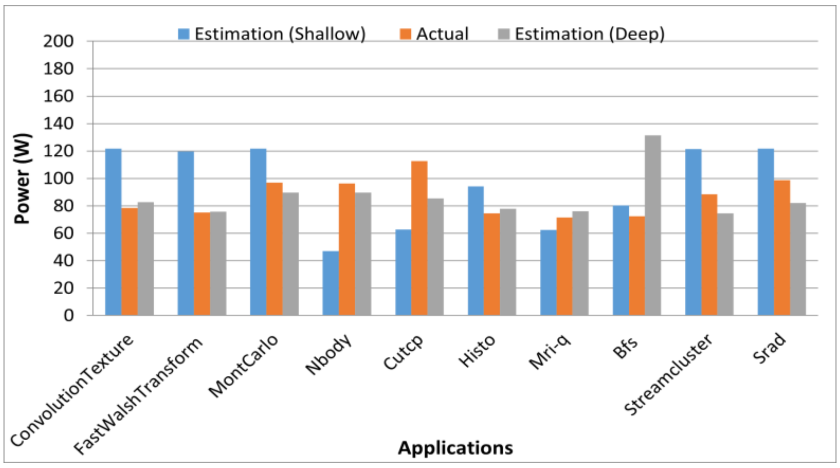

Figure 4. The difference between actual and estimated power consumption in VM connected with Kepler K40c GPU.

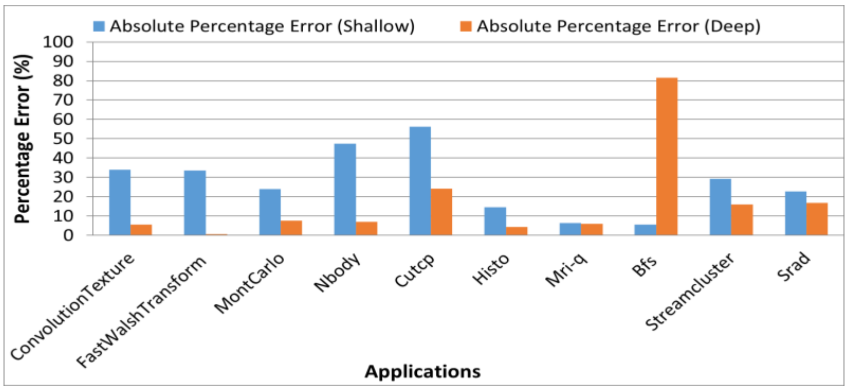

Figure 5. The absolute percentage error between shallow and deep neural network power models in VM connected with Kepler K40c GPU.

4.4. Hybrid Inputs Power Models

Figures 6 and 7 summarize the absolute percentage error difference between the shallow and deep neural network models developed by hybrid inputs for predicting applications' power consumption when they are executed on the VM connected with the Fermi GPU and the VM with the Kepler GPU, respectively.

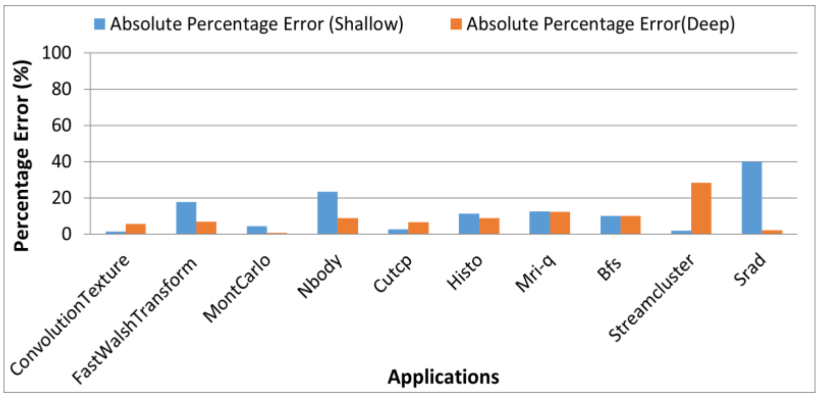

Figure 6. The absolute percentage error between shallow and deep neural network power models of hybrid inputs in VM connected with Fermi C2075 GPU.

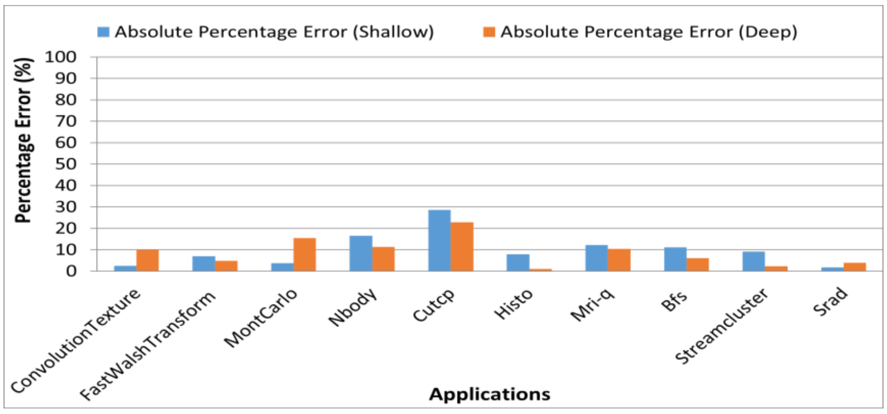

Figure 7. The absolute percentage error between shallow and deep neural network power models of hybrid inputs in a VM connected with Kepler K40c GPU.

4.5. Energy Models

Figure 8 depicts the energy consumption differences among the estimated values by the model with the standard inputs, the actual values, and the estimated values by the model developed by the common inputs (the agnostic model) of the applications in the testing set when these applications are executed on the Fermi GPU VM.

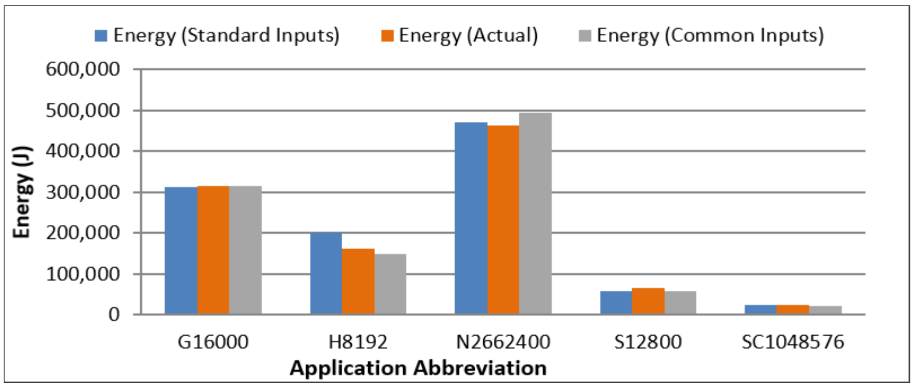

Figure 8. The energy difference between actual and estimated values of standard and agnostic models on a VM connected with Fermi C2075 GPU.

Figure 9 shows the absolute percentage error value difference between the energy model with standard inputs and the agnostic energy model on the applications in the testing set when these applications are executed on the Fermi GPU VM.

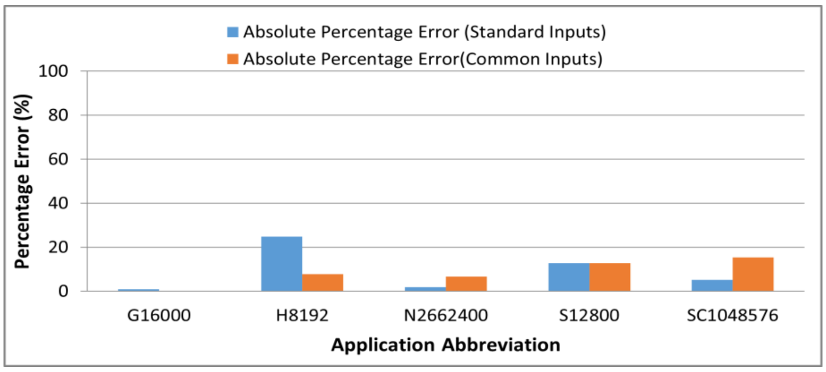

Figure 9. Absolute percentage error of energy models (standard and agnostic) on a VM connected with Fermi C2075 GPU.

Figure 10 depicts the energy consumption differences between the actual and the estimated values of the developed models (standard and agnostic) for the applications in the testing set when these applications are executed on the Kepler GPU VM.

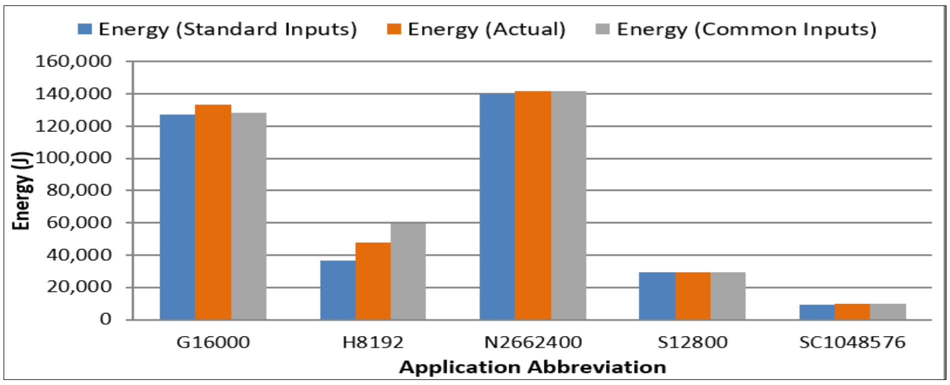

Figure 10. The energy difference between actual and estimated values (standard and agnostic) on a VM connected with Kepler K40c GPU.

Figure 11 shows the absolute percentage error values of the energy models (standard and agnostic) on the applications in the testing set when these applications are executed on the Kepler VM.

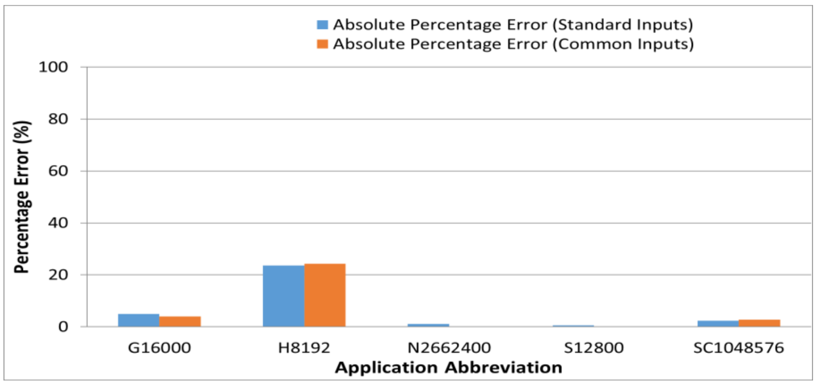

Figure 11. The absolute percentage error of energy models (standard and agnostic) on a VM connected with Kepler K40c GPU.

5. Discussion

Several models have been developed to predict power and energy consumption on heterogeneous GPUs connected with VMs in a cloud environment; each GPU has different characteristics. The absolute percentage error has been calculated to measure the overall accuracy in the testing set.

(1) Power Consumption Models

A comparison between shallow neural networks and DNNs to validate DNN models has been performed in power consumption models. Acceptable accuracy differences between actual and predicted power values have been conducted using neural networks (shallow and deep) by merely using performance counters as models' inputs. According to the power models' results on heterogeneous VMs, DNN models systematically produce a better accuracy compared with shallow networks in both VMs connected with heterogeneous GPUs.

In power consumption models' results on a VM with the Fermi C2075 GPU, the outliers of power consumption values in Nbody, Mri-q, and Streamcluster applications produced by shallow networks have been dramatically decreased in the DNN model. Additionally, the overall accuracy in the DNN GPU power model has exhibited better accuracy than shallow networks with a mean error of 18.1% compared with 45% in shallow networks. Furthermore, in the power consumption model results on a VM with the Kepler K40c GPU, many power consumption outliers have been produced by the shallow network in ConvolutionTexture, FastWalshTransform, Nbody, and Cutcp applications, while these outliers have been mitigated in the DNN power model, as shown in Figure 6.

The overall accuracy of the DNN model is also greater than the shallow network with 17% in comparison with the 27.4% mean error in the shallow network.

A significant accuracy improvement has occurred when the utility percentage of GPU and memory have been added with the selected performance counters—hybrid inputs—for the power consumption prediction in both heterogeneous GPUs connected with two different VMs.

In the power consumption model's results on a VM with the Fermi C2075 GPU, the accuracy of shallow networks has been improved with a mean error of 12.5% compared with 45% in the performance-counters-only model. Similarly, the DNN power model's accuracy has been improved from 18.1% in the performance counters model to 9% in the hybrid inputs model. In terms of the accuracy of each application individually, the greatest absolute percentage error has been dropped by adopting hybrid inputs from 38.1% (Histo) to 28.3 (Streamcluster) in the DNN model. However, in the shallow networks model, the greatest absolute percentage error has been reduced from 137% (Streamclucter) to 40% (Srad) on the VM connected with Fermi C2075. However, the accuracy of the DNN was better than shallow networks with a 9% mean error and 12.5% for shallow networks.

In the power consumption results on a VM with the Kepler K40c GPU, an improvement of the overall accuracy has occurred in the shallow networks model from a 27.4% mean error with the only-performance-counters-inputs model to a 10% mean error in the hybrid inputs model. Moreover, the overall accuracy in DNN has been enhanced from a 17% mean error in only-performance-counters-inputs to a 9% mean error in hybrid inputs. Although the accuracy of the shallow networks has been improved in hybrid inputs, the DNN model has performed better than the shallow networks model with 9% compared with a 10% mean error in the shallow network model. For individual evaluation, the greatest absolute percentage error has been decreased from 82% (Bfs) to 23% (Cutcp) in the DNN model with hybrid inputs. Similarly, in the hybrid inputs model, the greatest absolute percentage error value in shallow networks has dropped from 56.3% (Cutcp) to 29% (Cutcp). Table 10 summarizes the mean errors and the greatest error percentages of the GPU power consumption models on both VMs.

Table 10. The mean errors and greatest errors values summary of power consumption models on Both VMs.

VMs / Power Models	VM with Fermi C2075 GPU		VM with Kepler K40c	
	Mean Error	**Greatest Error**	**Mean Error**	**Greatest Error**
Performance Counters Inputs (Shallow)	45	137	27.4	56.3
Performance Counters Inputs (Deep)	18.1	38.1	17	82
Hybrid Inputs (Shallow)	12.5	40	10	29
Hybrid Inputs (Deep)	**9**	**28.3**	**9**	**23**

According to Table 10, the minimum mean error percentage of all power models on both VMs was 9%, for DNN models with hybrid inputs in both VMs, and the minimum greatest error percentage of all power models was 23%, for the DNN model with hybrid inputs in the VM connected with Kepler K40c. This indicates that adding both the utility percentage of GPU usage and the memory usage improves the model's accuracy.

(2) Energy Consumption Models

The outliers in the energy DNN model results do not exist when compared with the actual energy values in both VMs, as shown in the energy models' section.

In comparing the energy models having standard inputs and the agnostic model that has the common inputs in the VM connected with the Fermi GPU, the overall accuracy in the agnostic energy model performed slightly better than the energy model with standard inputs with a mean error of 8.6% compared with a 9.1% mean error in the standard energy model. In terms of individual accuracy, the maximum percentage error in the DNN energy model with standard inputs was 25% (H8192), while the greatest error in the agnostic DNN energy model with the common inputs was 15.4% (SC 1048576).

In a VM connected with the Kepler GPU, the agnostic energy model performance is slightly better than the energy model with standard inputs with a mean error of 6.3% compared with 6.5% in the energy model with standard inputs. Moreover, the maximum individual percentage error in the DNN energy model with standard inputs is 23.6% (H8192), while the highest individual percentage error in the DNN agnostic energy model is 24.3% (H8192). Therefore, the agnostic DNN energy model has exhibited a better performance in terms of the mean error and the maximum individual error percentage. Table 11 summarizes the mean error and the greatest error percentages of energy models in both VMs.

Table 11. The mean errors and greatest errors values summary of energy models for both VMs.

VMs / Energy Models	VM with Fermi C2075 GPU		VM with Kepler K40c	
	Mean Error	**Greatest Error**	**Mean Error**	**Greatest Error**
DNN with Standard Inputs	9.1	25	6.5	24.3
DNN with Common Inputs (agnostic model)	8.6	**15.4**	**6.3**	23.6

According to Table 11, for a comparison of both VMs, the best mean percentage error for the energy prediction model was found on the VM connected with the Kepler K40c GPU (6.3%), while the VM that connected with the Fermi C2075 GPU had the minimum greatest individual error (15.4%). Thus, the energy agnostic model has a better performance and allows us to reduce the input parameters required, which contributes to a reduction in the cost of data collection.

Another interesting finding is that the agnostic model provides an unintended level of bias in some cases. For example, the overall performance (measured by the mean) of the agnostic model in the Kepler GPU is better than the Fermi GPU. However, the Fermi GPU is better in terms of the greatest error. Therefore, there is a trade-off between accuracy and the cost of data collection in agnostic models.

6. Limitations

The evaluations conducted in this study through direct experiments for the developed models have shown promising results for enabling energy awareness in order to manage the heterogeneous resources of cloud computing. However, this study has a few limitations. These limitations are as follows:

- Each VM in cloud computing consists of different virtual resources, such as CPUs, memory, and network traffic. These resources affect the total VM energy consumption. Although the workload has been executed by the GPU, the developed energy consumption models in this study have only considered the energy consumed by the GPU and neglected other resources that affect the total VM energy consumption.
- Cloud computing is known for the diversity and heterogeneity of its resources, and the cloud-computing infrastructure hosts a great number of VMs and PMs. However, this work has only considered two VMs and two types of NVIDIA GPU architectures in cloud computing. This is because of the high cost of performing experiments on a real cloud testbed that contains a large number of heterogeneous GPUs. Moreover, simulation tools that support heterogeneous GPU architectures in cloud-computing environments are not available.

7. Conclusions and Future Work

The cloud infrastructure has grown more varied in terms of hardware due to cloud service providers' considerations regarding using accelerators, making resource management more difficult. It can be challenging to maintain QoS when operating a heterogeneous cloud architecture. This paper presents and discusses power and energy consumption models. These models were created for two virtual machines, each of which has heterogeneous GPUs from two generations (Fermi C2075 and K40c). Each GPU has a variety of unique characteristics and features. The models have been assessed by studies comparing predicted and actual values generated by models on a cloud testbed. Performance counters were used as inputs in shallow neural networks to test DNN power models. Secondly, employing hybrid inputs considerably improved the outcomes (performance counters, and GPU and memory utilization). Additionally, DNN energy models for two VMs have been presented. In each VM, comparisons between conventional and agnostic energy models with similar inputs have been made. Agnostic energy models that abstract the infrastructure heterogeneity for both VMs have demonstrated a small accuracy improvement with reduced inputs.

The suggested future directions for extending this study are structured as follows: First, we will employ Deep Convolutional Neural Network (CNN) workloads to enhance the generalization of GPU energy models to apply to different types of GPU architectures. Second, we will consider other cloud infrastructure resources including CPU, main memory, and network traffic to estimate the total energy consumption of the VM. Another potential extension of this work is to increase the heterogeneity of GPU architectures, for example, through the consideration of GPU generations—from older to new to the newest, the Maxwell, Pascal, and Volta NVIDIA architectures running together in cloud-computing environments. Finally, based on the enhanced energy models, scheduling policies to reduce energy consumption, and then balancing the trade-off between energy and performance will be developed in a cloud-computing environment.

Author Contributions: Methodology, A.A.; Software, A.A.; Validation, Y.A.; Resources, Y.A.; Writing—original draft, A.A.; Writing—review & editing, K.D. and Y.A.; Supervision, K.D.; Funding acquisition, A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministry of Education of Saudi Arabia and funded by King Abdulaziz University under Grant G: 690-611-1443.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets presented in this article are not readily available because they are part of an ongoing study. Requests to access the datasets should be directed to the corresponding authors.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. CISCO. *Cisco Annual Internet Report 2018–2023 White Paper*; CISCO: San Jose, CA, USA, 2020.
2. Public Cloud Computing Market Size 2022 | Statista. Available online: https://www.statista.com/statistics/273818/global-revenue-generated-with-cloud-computing-since-2009/ (accessed on 25 July 2023).
3. Amazon EC2 P3—Ideal for Machine Learning and HPC—AWS. Available online: https://aws.amazon.com/ec2/instance-types/p3/ (accessed on 1 August 2023).
4. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A.A. Inception-v4, inception-ResNet and the impact of residual connections on learning. *arXiv preprint* **2016**, arXiv:1602.07261. [CrossRef]
5. Mukherjee, T.; Dasgupta, K.; Gujar, S.; Jung, G.; Lee, H. An economic model for green cloud. In Proceedings of the 10th International Workshop on Middleware for Grids, Clouds and e-Science, MGC 2012, Montreal, QC, Canada, 3–7 December 2012. [CrossRef]
6. Hamilton, J. Cooperative expendable micro-slice servers (CEMS): Low cost, low power servers for internet-scale services. In Proceedings of the Conference on Innovative Data Systems Research (CIDR'09), Asilomar, CA, USA, 4–7 January 2009; pp. 1–8.
7. Andrae, A.; Edler, T. On Global Electricity Usage of Communication Technology: Trends to 2030. *Challenges* **2015**, *6*, 117–157. [CrossRef]
8. Jones, N. The information factories. *Nature* **2018**, *561*, 163–166. [CrossRef] [PubMed]
9. Ghosh, S.; Chandrasekaran, S.; Chapman, B.M. Statistical Modeling of Power/Energy of Scientific Kernels on a Multi-GPU system. In Proceedings of the 2013 International Green Computing Conference Proceedings, Arlington, VA, USA, 27–29 June 2013; pp. 1–6. [CrossRef]
10. Hong, S.; Kim, H. An integrated GPU power and performance model. In Proceedings of the 37th Annual International Symposium on Computer Architecture, Saint-Malo, France, 19–23 June 2010; pp. 280–289. [CrossRef]
11. Kasichayanula, K.; Terpstra, D.; Luszczek, P.; Tomov, S.; Moore, S.; Peterson, G.D. Power aware computing on GPUs. In Proceedings of the Symposium on Application Accelerators in High-Performance Computing, Argonne, IL, USA, 10–11 July 2012; pp. 64–73. [CrossRef]
12. Adhinarayanan, V.; Subramaniam, B.; Feng, W. Online Power Estimation of Graphics Processing Units. In Proceedings of the IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Cartagena, Colombia, 16–19 May 2016. [CrossRef]
13. Makaratzis, A.T.; Khan, M.M.; Giannoutakis, K.M.; Elster, A.C.; Tzovaras, D. GPU Power Modeling of HPC Applications for the Simulation of Heterogeneous Clouds. In *International Conference on Parallel Processing and Applied Mathematics*; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 91–101. [CrossRef]
14. Leng, J.; Hetherington, T.; ElTantawy, A.; Gilani, S.; Kim, N.S.; Aamodt, T.M.; Reddi, V.J. GPUWattch: Enabling Energy Optimizations in GPGPUs. *ACM SIGARCH Comput. Archit. News* **2013**, *41*, 487–498. [CrossRef]
15. Lucas, J.; Lal, S.; Andersch, M.; Alvarez-Mesa, M.; Juurlink, B. How a single chip causes massive power bills GPUSimPow: A GPGPU power simulator. In Proceedings of the ISPASS 2013—IEEE International Symposium on Performance Analysis of Systems and Software, Austin, TX, USA, 21–23 April 2013; pp. 97–106. [CrossRef]
16. Song, S.; Su, C.; Rountree, B.; Cameron, K.W. A simplified and accurate model of power-performance efficiency on emergent GPU architectures. In Proceedings of the IEEE 27th International Parallel and Distributed Processing Symposium, IPDPS 2013, Cambridge, MA, USA, 20–24 May 2013; pp. 673–686. [CrossRef]
17. Abe, Y.; Sasaki, H.; Kato, S.; Inoue, K.; Edahiro, M.; Peres, M. Power and performance characterization and modeling of GPU-accelerated systems. In Proceedings of the International Parallel and Distributed Processing Symposium, IPDPS, Phoenix, AZ, USA, 19–23 May 2014; pp. 113–122. [CrossRef]
18. Xie, Q.; Huang, T.; Zou, Z.; Xia, L.; Zhu, Y.; Jiang, J. An accurate power model for GPU processors. In Proceedings of the 2012 7th International Conference on Computing and Convergence Technology (ICCIT, ICEI and ICACT), ICCCT 2012, Seoul, Republic of Korea, 3–5 December 2012; pp. 1141–1146.
19. Siavashi, A.; Momtazpour, M. GPUCloudSim: An extension of CloudSim for modeling and simulation of GPUs in cloud data centers. *J. Supercomput.* **2019**, *75*, 2535–2561. [CrossRef]
20. Metz, C.A.; Goli, M.; Drechsler, R. ML-based Power Estimation of Convolutional Neural Networks on GPGPUs. In Proceedings of the 2022 25th International Symposium on Design and Diagnostics of Electronic Circuits and Systems, DDECS 2022, Prague, Czech Republic, 6–8 April 2022; pp. 166–171. [CrossRef]
21. Moolchandani, D.; Kumar, A.; Sarangi, S.R. Performance and Power Prediction for Concurrent Execution on GPUs. *ACM Trans. Archit. Code Optim.* **2022**, *19*, 1–27. [CrossRef]
22. Braun, L.; Nikas, S.; Song, C.; Heuveline, V.; Fröning, H. A Simple Model for Portable and Fast Prediction of Execution Time and Power Consumption of GPU Kernels. *ACM Trans. Archit. Code Optim.* **2021**, *18*, 1–25. [CrossRef]

23. Boughzala, D.; Lefèvre, L.; Orgerie, A.C. Predicting the energy consumption of CUDA kernels using SimGrid. In Proceedings of the SBAC-PAD-IEEE International Symposium on Computer Archi- Tecture and High Performance Computing International Symposium on Computer Archi- Tecture and High Performance Computing, Porto, Portugal, 9–11 September 2020; pp. 1–8.
24. Casanova, H.; Giersch, A.; Legrand, A.; Quinson, M.; Suter, F. Versatile, scalable, and accurate simulation of distributed applications and platforms. *J. Parallel Distrib. Comput.* **2014**, *74*, 2899–2917. [CrossRef]
25. Ilager, S.; Muralidhar, R.; Rammohanrao, K.; Buyya, R. A Data-Driven Frequency Scaling Approach for Deadline-aware Energy Efficient Scheduling on Graphics Processing Units (GPUs). *arXiv preprint* **2020**, arXiv:2004.08177.
26. Nagasaka, H.; Maruyama, N.; Nukada, A.; Endo, T.; Matsuoka, S. Statistical power modeling of GPU kernels using performance counters. In Proceedings of the International Conference on Green Computing, Green Comp, Chicago, IL, USA, 15–18 August 2010; pp. 115–122. [CrossRef]
27. Che, S.; Boyer, M.; Meng, J.; Tarjan, D.; Sheaffer, J.W.; Lee, S.H.; Skadron, K. Rodinia: A Benchmark Suite for Heterogeneous Computing. In Proceedings of the IEEE International Symposium on Workload Characterization, Austin, TX, USA, 4–6 October 2009; pp. 44–54.
28. Stratton, J.A.; Rodrigues, C.; Sung, I.J.; Obeid, N.; Chang, L.W.; Anssari, N.; Liu, G.; Hwu, W.M.W. Parboil: A Revised Benchmark Suite for Scientific and Commercial Throughput Computing. *Cent. Reliab. High-Perform. Comput.* **2012**, *127*, 27.
29. CUDA Code Samples | NVIDIA Developer. Available online: https://developer.nvidia.com/cuda-code-samples (accessed on 23 April 2023).
30. nvidia-smi. Available online: http://developer.download.nvidia.com/compute/DCGM/docs/nvidia-smi-367.38.pdf (accessed on 20 February 2023).
31. Profiler User's Guide. Available online: http://docs.nvidia.com/cuda/profiler-users-guide/index.html#gpu-trace-and-api-trace-modes (accessed on 10 April 2023).
32. Profiler User's Guide. Available online: https://cseweb.ucsd.edu/classes/wi15/cse262-a/static/cuda-5.5-doc/pdf/CUDA_Profiler_Users_Guide.pdf (accessed on 5 May 2023).
33. Curtis-Maury, M.; Shah, A.; Blagojevic, F.; Nikolopoulos, D.S.; De Supinski, B.R.; Schulz, M. Prediction models for multi-dimensional power-performance optimization on many cores. In Proceedings of the Parallel Architectures and Compilation Techniques—Conference Proceedings, PACT, Toronto, ON, Canada, 25–29 October 2008; pp. 250–259. [CrossRef]
34. Zaharia, M.; Konwinski, A.; Joseph, A.; Katz, R.; Stoica, I. Improving MapReduce performance in heterogeneous environments. *OSDI* **2008**, *8*, 7. [CrossRef]
35. Fall, D.; Okuda, T.; Kadobayashi, Y.; Yamaguchi, S. Risk adaptive authorization mechanism (RAdAM) for cloud computing. *J. Inf. Process.* **2016**, *24*, 371–380. [CrossRef]
36. Shao, G.; Chen, J. A load balancing strategy based on data correlation in cloud computing. In Proceedings of the 9th IEEE/ACM International Conference on Utility and Cloud Computing, UCC 2016, ACM, New York, NY, USA, 6–9 December 2016; pp. 364–368. [CrossRef]
37. Lek, S.; Delacoste, M.; Baran, P.; Dimopoulos, I.; Lauga, J.; Aulagnier, S. Application of neural networks to modelling nonlinear relationships in ecology. *Ecol. Model.* **1996**, *90*, 39–52. [CrossRef]
38. Mukhopadhyay, S.; Samanta, P. Deep Learning and Neural Networks. In *Advanced Data Analytics Using Python*; Apress: Berkeley, CA, USA, 2015. [CrossRef]
39. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.R.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.N.; et al. Deep neural network for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.* **2012**, *29*, 82–97. [CrossRef]
40. Bergstra, J.; Bardenet, R.; Bengio, Y.; Kégl, B. Algorithms for hyper-parameter optimization. *Adv. Neural Inf. Process. Syst.* **2011**, *24*, 1–9.
41. Greff, K.; Srivastava, R.K.; Koutnik, J.; Steunebrink, B.R.; Schmidhuber, J. LSTM: A Search Space Odyssey. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 2222–2232. [CrossRef] [PubMed]
42. Borovcnik, M.; Bentz, H.-J.; Kapadia, R. *Machine Learning: A Probabilistic Perspective*; MIT Press: Cambridge, MA, USA, 2012. [CrossRef]
43. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 249–256. [CrossRef]
44. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv preprint* **2014**, arXiv:1412.6980. [CrossRef]
45. Duchi, J.C.; Bartlett, P.L.; Wainwright, M.J. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Ofmachine Learn. Res.* **2011**, *12*, 2121–2159. [CrossRef]
46. Alnori, A.; Djemame, K. A Holistic Resource Management for Graphics Processing Units in Cloud Computing. *Electron. Notes Theor. Comput. Sci.* **2018**, *340*, 3–22. [CrossRef]
47. Home—Keras Documentation. Available online: https://keras.io/ (accessed on 27 July 2023).
48. OpenNebula. Available online: https://opennebula.io/ (accessed on 26 June 2023).