Ernane Rosa Martins
(Organizador)

FUNDAMENTOS DA CIÊNCIA DA COMPUTAÇÃO

Atena
Editora
Ano 2019

Ernane Rosa Martins
(Organizador)

Fundamentos da Ciência da Computação

Atena Editora
2019

2019 *by Atena Editora*
Copyright © da Atena Editora
Editora Chefe: Profª Drª Antonella Carvalho de Oliveira
Diagramação e Edição de Arte: Lorena Prestes e Geraldo Alves
Revisão: Os autores

Dados Internacionais de Catalogação na Publicação (CIP)
(eDOC BRASIL, Belo Horizonte/MG)

F981 Fundamentos da ciência da computação / Organizador Ernane Rosa Martins. – Ponta Grossa (PR): Atena Editora, 2019.

Inclui bibliografia
ISBN 978-85-7247-157-2
DOI 10.22533/at.ed.572190703

1. Computação. I. Martins, Ernane Rosa.

CDD 004

Elaborado por Maurício Amormino Júnior – CRB6/2422

APRESENTAÇÃO

A Ciência da Computação estuda as técnicas, metodologias e instrumentos computacionais, visando automatizar os processos e desenvolver soluções com o uso de processamento de dados. Este livro, possibilita conhecer os elementos básicos desta ciência por meio do contato com alguns dos conceitos fundamentais desta área, apresentados nos resultados relevantes dos trabalhos presentes nesta obra, realizados por autores das mais diversas instituições do Brasil.

Assim, são abordando neste livro assuntos importantes, tais como: desenvolvimento de sistema mobile utilizando as plataformas iOS e Android; desenvolvimento de protótipo que trabalha em cenário real de sala de aula e na comparação de algoritmos usados no reconhecimento facial; criação do jogo que explora a criptografia em um ambiente de computação desplugada; construção de simulador que mostra especificamente o comportamento do escalonador First-in First; apresentação de abordagem para orquestração do conhecimento curricular em Ciência da Computação baseado nas matérias do currículo referência para a Ciência da Computação e em estruturas curriculares de cursos de graduação.

Espero que este livro seja útil tanto para os alunos dos cursos superiores de Ciência da Computação quanto para profissionais que atuam nesta importante área do conhecimento. O principal objetivo deste livro é ajudar na fascinante empreitada de compreender a computação perante os mais diferentes desafios do século XXI. Desejo a todos uma excelente leitura e que está obra contribua fortemente com o seu aprendizado.

Ernane Rosa Martins

SUMÁRIO

CAPÍTULO 9

POWER CONSUMPTION USING INTERNAL SENSORS: AN ANALYSIS FOR DIFFERENT GPU MODELS

André Yokoyama
Laboratório Nacional de Computação Científica
Petrópolis – RJ

Vinicius Prata Klôh
Laboratório Nacional de Computação Científica
Petrópolis – RJ

Gabrieli Dutra Silva
Laboratório Nacional de Computação Científica
Petrópolis – RJ

Mariza Ferro
Laboratório Nacional de Computação Científica
Petrópolis – RJ

Bruno Schulze
Laboratório Nacional de Computação Científica
Petrópolis – RJ

ABSTRACT: GPUs has been widely used in scientific computing, as by offering exceptional performance as by power-efficient hardware. Its position established in high-performance and scientific computing communities has increased the urgency of understanding the power cost of GPU usage in accurate measurements. For this, the use of internal sensors are extremely important. In this work, we employ the GPU sensors to obtain high-resolution power profiles of real and benchmark applications. We wrote our own tools to query the sensors of two NVIDIA GPUs from different generations and compare the accuracy of them. Also, we compare the power profile of GPU with CPU using IPMItool.

KEYWORDS: Power, energy, GPU, HPC.

1 I INTRODUCTION

Scientific computing generally requires huge processing power resources' to perform large scale experiments and simulations in reasonable time. These demands have been addressed by High Performance Computing (HPC), allowing many scientific domains to leverage progress. The design of high performance supercomputers is on the boundary of constant and significant changes. With the arrival of petascale computing in 2008, we see another potential paradigm shift in the construction of parallel computing and the use of hybrid designs employing heterogeneous computational accelerators. In this context are the architectures with reduced energy consumption for improved energy efficiency, such as ARM and GPGPU heterogeneous computing.

However, despite the impressive theoretical peak performance of petascale supercomputers, several areas require more computational power, like the exascale supercomputers expected for the coming decade. For example, in the energy industry simulations, for different energy sources like wind energy, efficient combustion systems for biomass and exploration geophysics in the oil and gas industry. For the latter, seismic applications, as used in this work, are targets of this type of processing (MENEZES et al., 2012). But, it is expected that to achieve exascale will require a nearly 30-fold increase in performance with only a 1.2-fold increase in power consumption (ADHINARAYANAN; SUBRAMANIAM;FENG, 2016).

GPUs have become prevalent in HPC, as by offer exceptional performance, especially for scientific applications highly parallelizable, like seismic applications, as by power-efficient hardware. The position GPUs have established in high-performance and scientific computing communities has increased the urgency of understanding the power cost of GPU usage (BRIDGES; IMAM; MINTZ, 2016) (ADHINARAYANAN; SUBRAMANIAM;FENG, 2016).

To attempt this urgency, the use of internal power sensors for more accuracy power and energy measuring is a current area of research. But, this is not trivial, since the documentation on these sensors is scarce. So, understand exactly how to obtain power, and other measures of performance, for the GPU board is unclear. In addition, as GPUs continue to evolve, understanding their power and performance profiles on real applications is increasingly difficult (BRIDGES; IMAM; MINTZ, 2016).

In this work, extended from (FERRO et al., 2017), we performed experiments on three computational environments: two of them based on X86-64 CPU architecture with NVIDIA Tesla GPUs from different generations and the third one an ARM based CPU with NVIDIA Pascal GPU Cores, and collect data from its internal sensors. We wrote our own tools to query the sensors that enable a high-resolution power profiles of GPU kernel functions and low overhead (SILVA et al., 2018). We compare the accuracy of profiles via the NVIDIA Management Library (NVML) (NVIDIA, 2012) or IPMItool. Also, we compare the power profile of GPU with CPU using IPMItool for a deeper understanding. The experiments were conducted through either a benchmark from Rodinia (CHE et al., 2009) and a real application of seismic area, used by the Brazilian Oil Company. Understand the behavior of this application and its relation with the energy consumption are among the objectives of this work.

2 I BACKGROUND

In this section, we present some background about approaches for monitoring systems and power measurement in GPUs. The discussion of these concepts could be very broad, however, we will only discuss some concepts that are closely related to this research, as well, related works only focused on GPUs.

Power consumption of computer components can be obtained either for directly or indirectly methods. Directly measurements could be made via internal or external hardware sensors which periodically collect samples to estimate the power used during a time interval (BRIDGES; IMAM; MINTZ, 2016). An estimate of total energy is calculated as the integral of the power over the execution time (BURTSCHER; ZECENA; ZONG,2014). External power meters are connected between the power supply unit and a component under investigation (BRIDGES; IMAM; MINTZ, 2016). Internal power meters are obtained directly from built-in sensors, allowing users convenient access to power data via profiling software by sampling.

According to (BRIDGES; IMAM; MINTZ, 2016), directly measuring power via internal or external hardware sensors is considered the most accurate source of power consumption information. However, external power meters are generally not suitable for comprehensive power profiling, giving not accurate readings, especially in HPC settings. Internal sensors, when available, can be used for more accurate measurement. Besides accuracy, another advantages are that any expensive extra hardware is no need and allow component level profiling. However, not all hardware has these internal sensors and when there is, there is no standard feature nor consistency in accuracy across different hardwares. So, in some cases there is a need, especially in the HPC industry, for indirect methods (BRIDGES; IMAM; MINTZ, 2016).

Indirect measurements cover modeling and simulation techniques for power and performance estimation. The modeling approach estimates the power consumption using a model that correlates power with hardware performance counters. A great number of works in the area of energy efficiency are focused on using models to estimate power consumption (BURTSCHER; ZECENA; ZONG, 2014). Some limitations to counter-based models are that the number and type of counters available being not uniform across hardwares. So, there is a great variety of ways for accessing and visualizing hardware counter data but, the models generally are hardware dependent. Example of hardware monitoring software is the suite of profiling tools offered by NVIDIA that enable capabilities for accessing, visualizing, optimizing, monitoring and profiling applications and GPU hardware (NVIDIA, 2012).

In Section 3 we present some management and monitoring tools that use the method of direct measurements via internal hardware and how we use them to develop our way for power measurements.

3 I POWER MEASURING APPROACH

For each GPU model the sensor is different and enables a kind of measurement, accuracy and requires an approach to collect the data from sensors. In this work, as mentioned, we employ the GPU power sensors to obtain the energy consumption of the tested applications. We wrote our own tools to query the sensors, via the IPMItool for Tesla M2050 and NVML interface for Tesla K40, which implement a direct method by sampling monitoring approach. For the Jetson TX2 we used a shell script routine using architecture specific system calls to query the internal sensors directly. This kind of method periodically collect samples of the parameters. The power is obtained directly from sensors, and the total energy is calculated as the integral of the power over the execution time.

The Intelligent Platform Management Interface (IPMI) is a standard interface for hardware management used by system administrators to control the devices and monitor the sensors. For these, it is necessary the IPMI Controller called Baseboard Management Controller (BMC) and a manager software (for example, IPMItool). It provides an interface to manage IPMI functions in a local (in-band) or remote (out-of-band) system.

We are using the IPMI, in-band, to monitor the temperature and power consumption sensors by sampling methodology (both for CPU and GPU Tesla M2050). The IPMI Controller is used to identify how the application influences power consumption and increases the temperature. The driver included in our architecture is the OpenIPMI Linux Kernel Driver (*/dev/ipmi0*) and the manager software adopted was IPMItool. To collect these parameters, we are using with a sudo user following command: *$ ipmitool dcmi power reading.*

To use the IPMItool with the above command and to select the desired parameters (power and temperature), we developed a Shell-based tool to collect CPU data. This tool also collects the CPU and Memory utilization rates through the Linux's Top utility. Since the tool runs in parallel with the monitors application, competing for the same CPU resources, there is an overhead in this usage. This overhead was measured to be about 2.5% of CPU utilization rate. To collect Tesla M2050 GPU data we developed only a Shell monitoring script that queries the sensor readings, using IPMItool commands.

The NVIDIA profiling tools and APIs can be used to monitor and manage states of the NVIDIA GPU devices. The NVIDIA Management Library (NVML) (NVIDIA,2012) is an API, based on C language, for monitoring and managing states of NVIDIA GPU devices, such as GPU utilization rate, running process, clock and performance state, temperature and fun speed, power consumption and power management. The *nvmlDeviceGetPowerUsage* function retrieves the power usage reading for the device, in milliwatts, with an error of around the 5 watts. This is the power draw for the entire board, including GPU, memory, among others (KASICHAYANULA et al., 2012).

The NVIDIA System Management Interface (NVSMI) (NVIDIA,2017) is a command line program that allows the system administrators to query, with appropriate privileges, states of the GPU devices. Its functions are provided by the NVML library. By using the NVSMI it is possible to collect some of the same parameters as those obtained with our tool and with the sampling approach. However, when using NVSMI a much wider range of parameters is collected, which could result in overhead. In addition, since NVSMI is a high level utility, the rate of sampling power usage is very low (1 Hz). Such sampling rate might not be enough to notice the change in power, unless the kernel is running for a very long time.

For these reasons, we developed a CUDA C-based tool, using the NVML functions, to collect only the following parameters: Device name, Memory (Free, Used, Total), Utilization Rate (Memory, GPU), Power Consumption (W) and Temperature (C). In addition, this tool uses two threads: one thread responsible for launching, on GPU devices, the application to be monitored and another thread to collect by sampling, those parameters. So, an important feature of this tool is related to overhead, since the second thread only runs on CPUs, which does not impact the execution of the application on GPUs, the overhead is almost zero.

This NVML based monitoring application was developed with the intent of maximizing the sampling rate in order to increase the accuracy of the measurements, particularly for applications with small kernels. For this purpose the monitoring application only queries for the necessary parameters and the sampling rate is not fixed at a preset value. During the tests we observed a sampling rate of about 500 Hz, which provided some good measurements for the application used during the tests. This measurements, as well as the details of the methodology used to collect them, will be presented in the next section.

4 I EXPERIMENTAL EVALUATION

The experimental evaluation was developed using three computational environments with GPUs. The Santos Dumont Supercomputer (http://sdumont.lncc.br/machine.php?pg=machine - SDumont) uses the Tesla K40 GPUs and 2 CPUs Intel Xeon E5-2695v2 Ivy Bridge, with 12 real cores 2,4GHZ, 64GB of memory. The ComCiDis cluster uses the Tesla M2050 and 2 CPUs Intel(R) Xeon X5650 with 6 real cores 2.67 GHz each, 24 GB of memory. Details of Tesla M2050 and Tesla K40 are in Table 1. The third environment is the NVIDIA Jetson TX2, with a dual-core Denver 2 and quad-core A57 ARM CPUs and 256 NVIDIA Pascal CUDA Cores (GPU), it has 8 GB 128 bit LPDDR4 memory with 59.7 GB/s of bandwidth.

	Tesla M2050	**Tesla K40**
Processor	GF100	GK110B
Memory (ECC off) (GB)	3	12
Memory bandwidth (ECC off) (GB/s)	148	288
Clock (GHz)	1,546	3,0
Core	448	2880
GPU clock (MHz)	1150	745 (875MHz boosted)
TDP (Watts)	225	235
GFLOPS single precision	1030	4291-5040
GFLOPS double precision	515	1430-1680

Table 1. Details of the two GPU models used in the experiments.

The main application used in the experiments is an acoustic test implemented and used by Oil and Gas Industry. This is a *3D Wave Propagation* which simulates the propagation of a single wavelet over time by solving the acoustic wave propagation equation. This equation is solved by using finite differences that have a high degree of parallelization, given the interdependence between the data (MENEZES et al., 2012). The program was written in standard C and CUDA and spreads computation over a single CPU and a single GPU. For the experiments with 3D Wave Propagation were used different input sizes, described in Table 2. The Size ID corresponds to different sizes of the problem, which represents the discretization of the domain on a 3D grid. The X, Y and Z correspond to the size of the problem on the three axis and the MB line is the size of memory used for each group of the input size problem.

	Size ID											
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**
X	128	256	256	256	256	256	512	512	768	768	1024	512
Y	128	256	256	512	256	256	512	512	768	768	1024	512
Z	128	256	512	512	1024	2048	512	1024	768	1024	704	2832
MB	32	256	512	1024	1024	2048	2048	4096	6912	9216	11266	11328

Table 2. Input sizes used in experiments with 3D Wave Propagation.

In some experiments, particularly to compare CPU and GPU measurements with IPMItool, we used the LUD benchmark from Rodinia (CHE et al., 2009). We choose this kernel to perform a set of experiments with more controlled results and also that could be executed in both GPU (CUDA version) and CPU (OpenMP). LUD is an algorithm to calculate the solutions of a set of linear equations that decomposes a matrix as the product of a lower triangular matrix and an upper triangular matrix to achieve a triangular form that can be used to solve a system of linear equations easily. In this work, we presented results for matrix size of 16384.

5 I METHODOLOGY

Since the IPMItool requires *sudo* permission it was used only on our own cluster (ComCiDis). It was the only way to get power and temperature reading for the Tesla M2050, since it doesn't have the built-in sensors. The measurement data are collected using a shell script that repeatedly queries the sensor readings, using the IPMItools commands. The output of the queries are written in an output file. To collect the data, the monitoring script is launched 30 seconds before the application to be monitored. This padding allows us to establish the baseline of power drain when only the monitoring script is running, and by querying the sensors manually. We can get the idle power drain (the amount of power been consumed by the system) when no application is running. This information allows us to estimate the real energy consumed by the application by removing the overhead (in power drain) caused by the monitoring script. The experiments performed on ComCiDis cluster with CPU, followed the same methodology to collect the data.

For the experiments on SDumont, only NVML based monitoring application was used, since we don't have *sudo* permission to use IPMItool. K40 has built-in sensors, that no need *sudo* for temperature and power readings. These sensors have a higher precision than those read by the IPMItool and we can get a much more accurate amount of energy. Nevertheless, as we can't use the IPMItool, we are unable to get the total energy consumed by the entire computing node.

As mentioned in Section 3, the NVML based monitoring application uses 2 threads, the first one queries the sensors of all GPUs installed in the computing node and write the readings in an output file, the second thread is responsible for launching the application that's been evaluated, it is also responsible for signaling the monitoring thread to stop collecting data. The monitoring thread starts collecting data as soon as it is launched and only stops when signaled by the second thread. This thread starts the application 30 seconds after the monitoring thread starts, and signal the stop 30 seconds after the end application. This 30 seconds padding allows us to establish a baseline consumption while all the GPUs in the node are in the idle state.

5.1 Results On GPUs

In this section we present and discuss results of the experiments highlighting some aspects of power profiles obtained from internal sensors. The results were obtained using the NVML based monitoring application for SDumont and IPMItool, while running the 3D Wave Propagation application.

The graphs of Figures 1 to 3 present the results for problem size G (the largest size that could be performed in all three environments) in SDumont, ComCiDis and Jetson TX2. In Figure 4 is the L size only in SDumont. It was too large to execute in the other two, since this application requires the entire problem to be loaded to the GPU memory and problem size L uses about 11.2 GB, which is too large for the 2.65 GB of the ComCiDis' Tesla M2050 and 7.8 GB of the Jetson TX2.

The parameters collected and presented in Figures 1 and 4 are the memory and GPUs usage, power and temperature, for each device (device 0 and 1). In Figure 2 due the IPMItool limitation, there are only the temperatures of each CPU (CPU1 Temp, CPU2 Temp) and GPU (GPU1 Temp, GPU2 Temp) and Power. In Figure 3 the parameters are, the power drain and temperature of the whole device, of the GPU, of the CPU and of the Memory, due to the low consumption of the device the power readings are collected in MW while in the other two it is collected in W.

The power levels collected by the monitoring application are the individual power levels for each GPU with a sampling rate around 500 Hz, while the IPMItool collects the total power used by the whole computing node with a rate between 1 and 3 Hz. For the Jetson TX2 the monitoring script reads the sensors with a frequency of 1 Hz to prevent or at least minimise the overhead caused by the monitoring script.

Figures 1, 2 and 3 display the results for the problem size G on SDumont, ComCiDis and Jetson TX2. The average time for this size in SDumont was 4 minutes and 12 seconds, consuming 36802 J (30,520 J of the computing GPU and 6,281 J of the idle GPU). For the ComCiDis it was 6 minutes and 32 seconds, more than 1.5 times longer and consuming 101703 J (2.76 times more than SDumont's K40). For the Jetson TX2 the time was 14 minutes 26 seconds almost 3.5 times compared to the SDumont but with a consumption of 6791.799 J (about 0.18 times the consumption of the SDumont's K40). The peak of the power drain observed in the ComCiDis by GPUs was of 290 W, but it stays at 250 W most of the time. For SDumont the peak was about 160 W for the two GPUs together (134 W for de GPU when it was processing and 26 W for the GPU idle). For the Jetson TX2 the peak was 9.04 W.

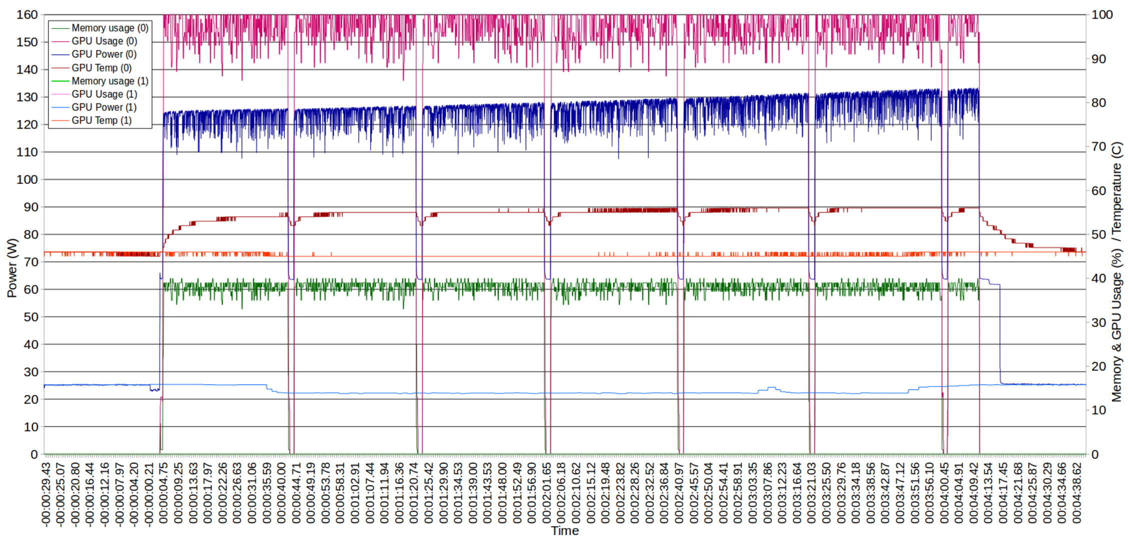

Figure 1: Results for problem size G on SDumont collected by the NVML based monitoring application.

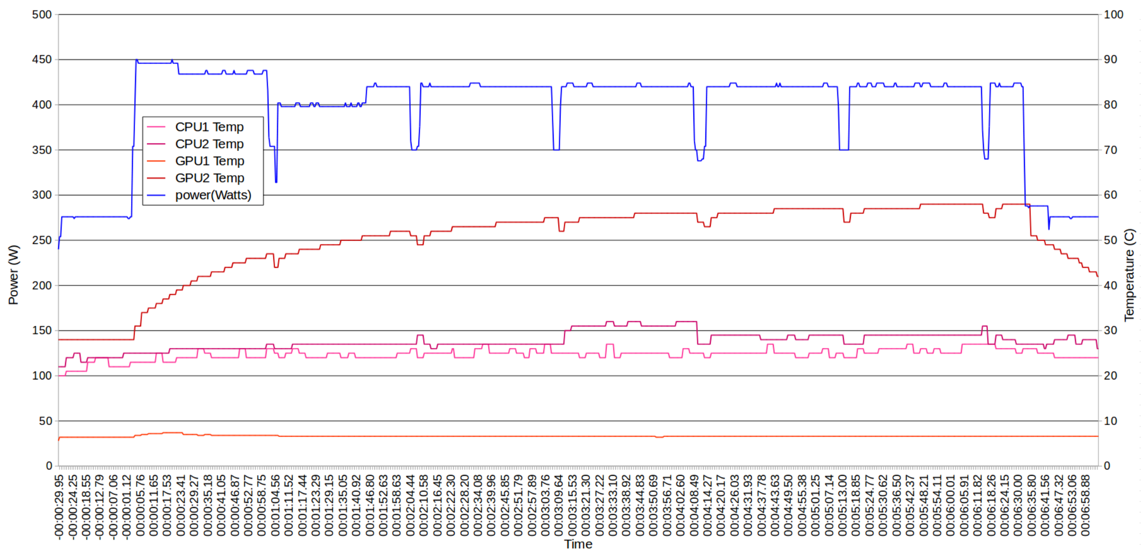

Figure 2: Results for problem size G on the ComCiDis collected by the IPMITool application.

In Figure 1 we can observe a high level of fluctuation on the power readings (GPU Power 0). In this graph it is possible to see, particularly looking at the GPU and memory usage for the running device (Memory usage 0 and GPU Usage 0), that the pattern of increase and decrease in usage matches that of the power readings, with only the power reading showing a steady increase over time. This could be related to the GPU's temperature and this behavior has been observed also in (BURTSCHER; ZECENA; ZONG,2014).

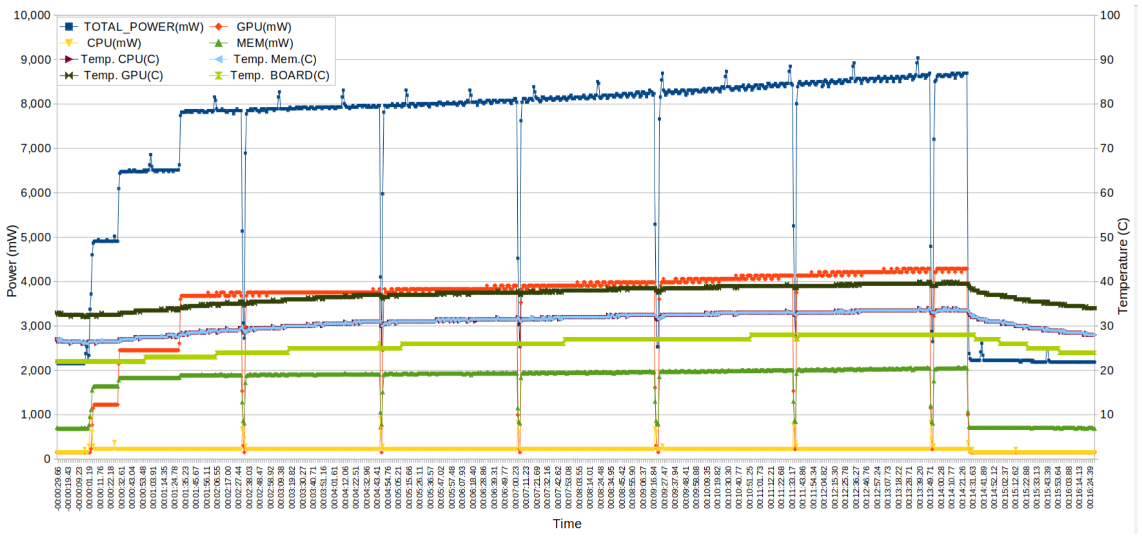

Figure 3: Results for problem size G on the Jetson TX2 collected by internal sensors.

Looking at the behavior of the application for the size G (Figures 1, 2 and 3), there is a significant drop in the power readings in regular intervals. This happens because the application has pauses in the processing, in regular intervals, to take "snapshots" (copying the data from the GPU memory to the CPU memory). During this time the GPU goes into a idle state, but ready to run a new kernel. In this state, the power drain is higher than a completely idle GPU. What we can see in the graphs is that during this pauses in the processing, the temperature drops as expected. But, when the application resumes the processing, the temperature rises faster than in the previous sequence o kernels. Also, the temperature gets higher, as can be seen in Figure 1, than the maximum temperature observed in the last sequence, being about 2 degrees higher than in the first sequence. In addition to that, it also gets to that temperature faster. This increase in the temperature could be contributing to the increase in power drain, but it could also be caused by it. We need to perform more tests in order to properly verify this behavior, including the execution of tests after pre-heating the GPU, or super-cooling it.

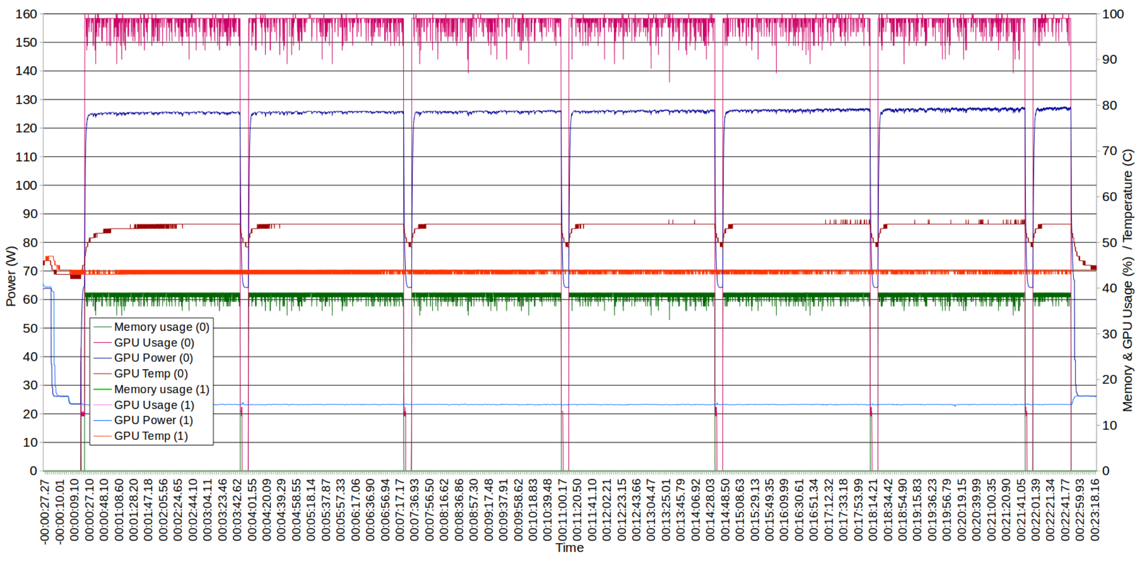

Figure 4: Results for the problem size L, executed on SDumont and collected by the NVML based monitoring application.

Analyzing the results for the largest execution size L in SDumont (Figure 4), it is possible to see some behavior seen also in the graphs for the G size. For example, the steady increase in the power readings over time, as well for the temperature, and the "stop-and-go" behavior, similar for the G size. Although there is also a high level of fluctuation on the usage of the GPU cores and memory, the range of the fluctuation is shorter than that seen in the problem size G. In this case, the readings have a higher concentration, at 99% usage for the GPU and 39% for the memory. We also can see the fluctuation on the power readings, but the range is very narrow, almost negligible. The L size took 22 minutes and 50 seconds to complete, consuming 200,740 J (166,496 J of the computing GPU and 34,244 J of the idle GPU).

Table 3 summarizes the results for all experiments. For the problem size G the SDumont had the best time and energy to solution of the two X86-64 environments, the ARM based environment had the worst time to solution of the three but the best energy to solution consuming slightly more than one sixth of the energy consumed by the SDumont GPUs.

Size	Cluster	Time to solution	Energy to solution
G	SDumont	4 min 12 sec	36,802 J
G	ComCiDis	6 min 32 sec	101,703 J
G	Jetson TX2	14 min 26 sec	6,792 J
L	SDumont	22 min 50 sec	200,740 J

Table 3. Time and Energy to Solution on GPUs.

5.2 Results - CPU x GPU

Here, we will present the results obtained in ComCiDis cluster using both the IPMItool based monitoring application for CPU and the script for GPU (the same used in the experiments presented in Section 4.1. In this experiments we used the application LUD for problem size of 16384, with OpenMP and CUDA parallel models.

Figure 5 displays the results using the OpenMP parallel model in CPU. The average execution time was about 2 minutes and 54 seconds with the sampling rate between 1 and 3 Hz. There are two intervals of 30 seconds, one before T1 and another after T3. In these intervals (`padding'), only the monitoring tool was running to establish the baseline of power drain. The average of the power drain was of 170 W with an energy consumption of 29,240 J and the average of the power consumption measured with the monitoring tool running was of 182 W. So, the monitoring tool increased the power consumption by 12 W, with an energy consumption of 2,064 J. The energy consumption was calculated only when the application was running.

The LUD application started in T1 and finished in T3. While the application was running, it is possible to note two steps: *i)* between T1 and T2 the application was loading the matrix (serial region of code) and the average of power consumption was of 218 W - power consumption increase of 36 W - with an energy consumption of 2,916 J, and *ii)* between T2 and T3, the application was solving the problem (OpenMP region of code), the average of power consumption was of 298 W - increase of 116 W - with an energy consumption of 10,556 J. The rise in temperature in the CPU(0), when the application was loading the matrix, indicates that only that CPU was executing the process. Between the interval T2 and T3 the parallel region of the code is started and both CPUs were executing the process. So, it is possible to observe the rise of the temperature in both CPUs and the peak of the power consumption and CPU utilization rate.

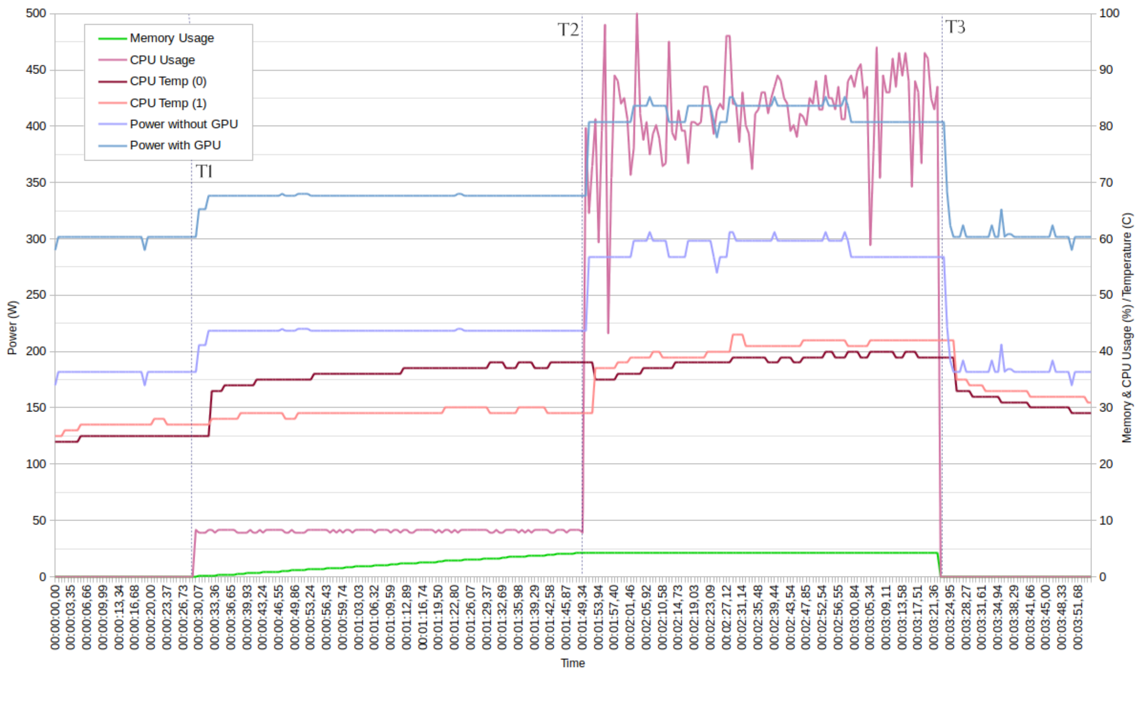

Figure 5: Results for the ComCiDis CPU collected by the IPMItool based monitoring application.

Table 4 compares the results for the energy and power consumption by system, monitor and application, while the application was running on serial and parallel region of the code. In the column System, can be observed that the power consumption by the GPU system was greater than by the CPU system, increased by GPU devices. For the application (App), it's possible to note that power consumption on serial region was much smaller than on parallel region for either CPU and GPU. Even the power consumption on parallel region in the GPU was greater than in the CPU. Although the power consumption was greater, given the execution time of the application, the energy-to-solution in the parallel region in CPU was smaller than for GPU. The power consumptions by the monitors (Monitor) were the same for both CPU and GPU, but the energy consumptions were different because the execution time on GPU was less than on CPU.

		CPU			GPU		
		System	Monitor	App	System	Monitor	App
Energy (J)	Serial Region	13,770	972	2,916	17,989	811	1,758
	Parallel region	15,470	1,092	10,556	5,721	258	4,990
Power (W)	Serial region	170	12	36	266	12	26
	Parallel region	170	12	116	266	12	232

Table 4. Results for the ComCiDis' CPU and GPU.

6 I FINAL CONSIDERATIONS

In this work were analysed three directly measuring methods via internal sensor for two generations of NVDIA GPUs and for the ARM based NVIDIA Jetson TX2. Also, we compare the power profile of GPU with CPU using IPMItool for a deeper understanding of this method. For this we developed two monitoring tools, based on NVML and IPMItool.

Analyzing the results obtained with three monitoring approaches, as well as for different computational environments, it is possible to make some observations. With regard to different generations of sensors, and consequently the available sensors to measure temperature and energy, the precision of K40 in relation to M2050 is very high. This can be clearly observed by visually comparing the graph of Figures 1 and 2, even though the graph for the M2050 had to be compressed to fit within the same width of the graph for the K40 (since it the time to solution of the M2050 was 1.5 times longer), where the profiling power is much coarse-grained, it is not so clear seen in the graph of the Jetson TX2 execution (Figure 3) due to a higher compression, but still visible. Also, the refreshing rate for K40 is very faster and its idle power drain is about 25 W against 60 W for M2050, indicating the improvement in the energy saving of the newer generations.

Regard the monitoring applications, the sampling rate for NVML based monitoring application is much higher, reaching 500 Hz versus only 3 Hz for IPMItool. Thus, the accuracy of the power profile obtained with the NVML based application is much higher with incurred overhead of almost zero for the execution time and the power measured for GPU. Another advantage is that this method does not need *sudo* user to query the sensors, in order to enable the monitoring of environments that have high restrictions to access, like SDumont. Using the IPMItool based monitoring it was possible to compare the power consumption and estimate the energy consumption for GPU and CPU. The accuracy is the same for both architectures, since IPMItool query the same kind of sensors. The power profile is not so accurate, as demonstrated in results and there is an overhead about 2.5% of CPU utilization rate. Despite these limitations, in some cases its use is still a nice option, like in generations of GPUs that do not have built-in sensors. Besides that, this monitoring method allows to analyse the power profile, for the same application, in different architectures and parallel models. The major limitation is the need for *sudo* user. For the Jetson we used a shell script using architecture specific system calls to query the sensors directly both because of the availability of the system calls as well as to minimise the overhead caused by a more complex application, since the ARM CPU are much more limited compared to X86-64, for that same reason we choose a sampling rate of 1Hz.

In general, for the X86-64 architecture, both monitoring tools were feasible for our studies on energy consumption of the applications. However, we conclude that the best option could be a monitoring tool that combines the two monitoring tools, since the NVML measures only the individual power levels for each GPU and IPMItool measures only the total power used by the whole computing node. Thus, it would be possible to develop a more accurate monitoring.

As future works we plan to develop this hybrid monitoring tool, to execute tests with a multi-GPU version of the real application and also to perform multiple experiments, similar to those presented using IPMItool, however, using different times of application launches in relation to the launch time of the monitor.

ACKNOWLEDGMENTS

This work has received partial funding from the European Union’s Horizon 2020 Programme (2014-2020) and from Brazilian Ministry of Science, Technology, Innovation and Communication through the RNP under the HPC4E Project, grant agreement n 689772 and from the CNPq and CAPES. The authors would also like to thank CENPES/Petrobras.

REFERÊNCIAS

ADHINARAYANAN, V.; SUBRAMANIAM, B.; FENG, W. CHUN. **Online power estimation of graphics processing units.** In: 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2016, Cartagena, Colômbia, May 16-19, 2016. [S.l.]: IEEE Computer Society,

2016. p. 245–254. ISBN 978-1-5090-2453-7/16.

BRIDGES, R. A.; IMAM, N.; MINTZ, T. M. **Understanding gpu power: A survey of profiling, modeling, and simulation methods.** ACM Comput. Surv., ACM, New York, NY, USA, v. 49, n. 3, p. 41:1–41:27, set. 2016. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/2962131>.

BURTSCHER, M.; ZECENA, I.; ZONG, Z. **Measuring gpu power with the k20 built-in sensor.** In: Proceedings of Workshop on General Purpose Processing Using GPUs. New York, NY, USA: ACM, 2014. (GPGPU-7), p. 28:28–28:36. ISBN 978-1-4503-2766-4. Disponível em: <http://doi.acm.org/10.1145/2576779.2576783>.

CHE, S. et al. **Rodinia: A benchmark suite for heterogeneous computing.** In: IISWC. [S.l.]:IEEE, 2009. p. 44–54. ISBN 978-1-4244-5156-2.

FERRO, M. et al. **Analysis of gpu power consumption using internal sensors.** In: Anais do XVI Workshop em Desempenho de Sistemas Computacionais e de Comunicação . São Paulo - SP: Sociedade Brasileira de Computação (SBC), 2017.

KASICHAYANULA, K. et al. **Power aware computing on gpus.** Application Accelerators in High-Performance Computing, Symposium on, IEEE Computer Society, Los Alamitos, CA, USA, v. 00, p. 64–73, 2012. ISSN 2166-5133.

MENEZES, G. S. et al. **Energy estimation tool fpga-based approach for petroleum industry.** In: Proceedings of the 2012 41st International Conference on Parallel Processing Workshops. Washington, DC, USA: IEEE Computer Society, 2012. (ICPPW '12), p. 600–601. ISBN 978-0-7695-4795-4. Disponível em: <http://dx.doi.org/10.1109/ICPPW.2012.88>.

NVIDIA. **NVML API Reference Manual**. [S.l.], 2012. Disponível em: <http://developer.download.nvidia.com/assets/cuda/files/CUDADownloads/NVML/nvml.pdf>.

NVIDIA. **Nvidia system management interfaces.** 2017. Disponível em: <"https://developer.nvidia.com/nvidia-system-management-interface">.

SILVA, G. D. et al. **Smcis: um sistema para o monitoramento de aplicações científicas em ambientes hpc.** In: Anais do XIX Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD 2018). Sociedade Brasileira de Computação (SBC). São Paulo, 2018. p.277–288. Disponível em: http://www2.sbc.org.br/wscad/current/anais/anais-wscad-2018.pdf.

SOBRE O ORGANIZADOR

Ernane Rosa Martins - Doutorado em andamento em Ciência da Informação com ênfase em Sistemas, Tecnologias e Gestão da Informação, na Universidade Fernando Pessoa, em Porto/Portugal. Mestre em Engenharia de Produção e Sistemas pela PUC-Goiás, possui Pós-Graduação em Tecnologia em Gestão da Informação pela Anhanguera, Graduação em Ciência da Computação pela Anhanguera e Graduação em Sistemas de Informação pela Uni Evangélica. Atualmente é Professor de Informática do Instituto Federal de Educação, Ciência e Tecnologia de Goiás - IFG (Câmpus Luziânia), ministrando disciplinas nas áreas de Engenharia de Software, Desenvolvimento de Sistemas, Linguagens de Programação, Banco de Dados e Gestão em Tecnologia da Informação. Pesquisador do Núcleo de Inovação, Tecnologia e Educação (NITE).

Agência Brasileira do ISBN
ISBN 978-85-7247-157-2

9 788572 471572