Student's name and surname: Grzegorz Koszczał
ID: 175405
Cycle of studies: postgraduate
Mode of study: Full-time studies
Field of study: Informatics
Specialization: Distributed Applications and Internet Services

MASTER'S THESIS

Title of thesis: Research on comparison of various methods for energy measurement for computational systems

Title of thesis (in Polish): Badanie porównawcze różnych metod pomiaru zużycia energii systemów obliczeniowych

Supervisor: dr hab. inż. Paweł Czarnul

GDAŃSK UNIVERSITY OF TECHNOLOGY

FACULTY OF ELECTRONICS, TELECOMMUNICATIONS AND INFORMATICS

DECLARATION regarding the diploma thesis titled:
Research on comparison of various methods for energy measurement for computational systems

First name and surname of student: Grzegorz Koszczał
Date and place of birth: 11.02.1999, Olsztyn
ID: 175405

Faculty: Faculty of Electronics,Telecommunications and Informatics
Field of study: informatics

Cycle of studies: postgraduate
Mode of study: Full-time studies

Type of the diploma thesis: master's thesis

Aware of criminal liability for violations of the Act of 4th February 1994 on Copyright and Related Rights (Journal of Laws 2021, item 1062 with later amendments) and disciplinary actions set out in the Act of 20th July 2018 on the Law on Higher Education and Science (Journal of Laws 2022 item 574 with later amendments),[1] as well as civil liability, I hereby declare that the submitted diploma thesis is my own work.

This diploma thesis has never before been the basis of an official procedure associated with the awarding of a professional title.

All the information contained in the above diploma thesis which is derived from written and electronic sources is documented in a list of relevant literature in accordance with art. 34 of the Copyright and Related Rights Act.

15.08.2023, Grzegorz Koszczał

Date and signature of the student or authentication on the university portal Moja PG

**) The document was drawn up in the IT system, on the basis of paragraph 15 clause 3b of the Decree of the Ministry of Science and Higher Education of 12 May 2020, amending the decree concerning university studies (Journal of Laws of 2020, item 853). No signature or stamp required.*

[1] The Act of 20th July 2018 on the Law on Higher Education and Science:

Art. 312, section 3. Should a student be suspected of committing an act referred to in Art. 287 section 2 items 1–5, the rector shall forthwith order an enquiry.

Art. 312, section 4. If the evidence collected during an enquiry confirms that the act referred to in section 5 has been committed, the rector shall suspend the procedure for the awarding of a professional title pending a judgement of the disciplinary committee and submit formal notification on suspicion of committing a criminal offence.

ABSTRACT

The goal of the project is to conduct experiments and compare various methods of power draw measurement of parallel applications. The used measurement tools are both software (Intel RAPL, NVIDIA NVML) and hardware (Yokogawa WT310E). Moreover, the conditions of implementing mentioned solutions are evaluated, as well as their limitations for newest CPUs and GPUs.

Keywords: Informatics, parallel programming, High-Performance Computing Systems, green computing, NAS Parallel Benchmark, Linux Perf, Intel RAPL, NVML, PAPI, Internal Power Sensors, External Power Meters

Field of science and technology in accordance with OECD requirements: [PLACEHOLDER] ask Dr. Czarnul

CONTENTS

LIST OF IMPORTANT SYMBOLS AND ABBREVIATIONS

GPU – Graphics Processing Unit

INTRODUCTION

The Information and Communication Technology sector is responsible for a significant share of global electricity use. In 2020, the data centers, communication networks and user devices accounted for an estimated $4-6\%$ of global electricity use [33]. This value has grown exponentially over the last years, mainly due to technological advances such as cloud computing and the rapid growth of the use of Internet services [4]. Moreover, it is predicted that ICT energy use is likely to increase until 2030 and may reach approximately 13% of global electricity use. According to the increased energy demand, vast efforts has been put in order to improve the energy efficiency across the ICT sector with a success. As a result the overall energy use remains mostly flat according to some estimates.

The goal of energy efficiency increase is reached using various methods. Some of them involve the kernels optimizations and using the energy methods of computing [11]. Such methods focus mainly on load imbalance, mixed precision in floating-point operations. Other methods of increasing computational efficiency are related to power-capping [14]. Such an approach assumes setting a certain power limit level on CPU or GPU in order to achieve a power/performance trade-off. Appropriate limitations of power draw results in slightly longer execution times and significant savings, which render this method a viable option.

In order to make such implementation meaningful, we must make sure that the tools used to validate them are also as precise as possible. For the measurement of the power draw of a CPU, Intel provides its own interface, called Running Average Power Limit (RAPL) [22]. On the GPU side, NVIDIA provides the NVIDIA Management Library (NVML) [26]. Unfortunately, concerns rise on the precision of such softwares, mainly because their providers don't share any information about estimated error of power draw readings, leaving researchers questioning their practical use. In order to verify the precision of software measurement tools, the external measurement tool is used – the Yokogawa WT310E [37] [38]. Its high precision, backed up by certificates, makes it an excellent tool to benchmark the precision of Intel RAPL and NVML.

1. RESEARCH GOAL

1.1. PURPOSE AND RESEARCH QUESTION

The goal of the project is to verify the accuracy and reliability of the CPU and GPU power draw measurement tools during the computational-straining benchmark applications.

1.2. SCOPE AND LIMITATION

The scope of the project is to verify whether the software power measurement tools are precise, based upon the results of the certified external measurement tool and to specify their error range in case of inaccuracy.

In case of benchmark applications, a hypothesis worth considering is whether the change of computational data impacts the power draw measurements between hardware and software tools[NEED QUOTATION]. There are experiments[NEED QUOTATION] that proved that increasing the data used in the benchmarks influences the increasingly different results from the tools. Another hypothesis, however, makes the claim that the application of power capping does not impact the results. In order to make a conclusion, all those configurations should be investigated independently. Another aspect worth investigating is the use of various power supply units, both the server and consumer grade. Those tests would give us insight, whether they differ in total power draw or is it more likely associated with the certificates they have.

The project is limited to testing CPUs and software released by Intel Corporation and to testing the GPUs and software released by NVIDIA Corporation.

1.3. PROJECT REQUIREMENTS

The project requires a reliable testbed in order to run the computations, verified benchmark applications and certified external measurement tool. Moreover, the computational station must be exclusively reserved for the time of research in order to prevent other user's applications from interfering in the tests results, as well as the tests itself should be repeated several times in order to maintain credibility.

The workstation on which the experiments will be conducted is a university computational node that consists of two Intel Xeon Silver 4210 CPUs and eight NVIDIA Quadro RTX 6000 GPUs, as well as a custom power strip that supports the use of external measurement tool. The tests

Benchmark applications are considered suitable for the purpose of the project, when they are able to strain the workstation's hardware, using their entire computational resources. The chosen applications for this task are NAS Parallel Benchmarks [5] [24] – a set of benchmarks designed to evaluate the performance of parallel computing systems. The NPB suite contains a variety of benchmarks, including linear algebra, FFT, stencil computations and others. The benchmarks are intended to be representative of some important real-world application problems and can be used to assess the performance of various systems under different conditions.

The external measurement tool is Yokogawa WT310E. It's a precise and reliable equipment that will serve as the ground truth in verification of reading from the software tools.

2. THEORETICAL BACKGROUND

2.1. MEASUREMENT SOFTWARE

2.1.1. INTEL RAPL

Intel RAPL (Running Average Power Limit) [17] is an interface [8], which allows software to set a power limit that hardware ensures and any power control system takes it as an input and tunes behavior to ensure that this operating limit is respected. That way, it is possible to set and monitor power limits both on processor and DRAM, and by controlling the maximum average power, it matches the expected power and cooling budget. RAPL exposes its energy counters through model-specific registers (MSRs) It updates these counters once in every 1 ms. The energy is calculated as a multiple of model specific energy units. For Sandy Bridge, the energy unit is 15.3 µJ, whereas it is 61 µJ for Haswell and Skylake.

Moreover, the Intel RAPL divides the platform into four domains, which consists of:

- PP0 (Core Devices) – Includes the energy consumption by all the CPU cores in the socket(s).
- PP1 (Uncore Devices) – Includes the power consumption of integrated graphics processing unit (unavailable on the server platforms)
- DRAM – The energy consumption of the main memory.
- Package – The energy consumption of the entire socket including core and uncore.

2.1.2. NVIDIA NVML

NVIDIA Management Library (NVML) [26] – A C-based API for monitoring and managing various states of the NVIDIA GPU devices. It provides direct access to the queries and commands exposed via nvidia-smi [30]. The runtime version of NVML ships with the NVIDIA display driver, and the SDK provides the appropriate header, stub libraries and sample applications. Each new version of NVML is backwards compatible and is intended to be a platform for building third party applications.

There are various techniques of computing the energy consumption using the NVIDIA Management Library, which query the onboard sensors and read the power usage of the device. Such techniques are either from the native NVML API, like Sampling Monitoring Approach (SMA) or Multi-Threaded Synchronized Monitoring (MTSM) or from CUDA component and it's called Performance Application Programming Interface (PAPI) [28].

Sampling Monitoring Approach (SMA) – The C-based API provided by NVML that can query the power usage of the device and provide an instantaneous power measurement. Therefore, it can be programmed to keep reading the hardware sensor with a certain frequency. The *nvmlDeviceGetPowerUsage()* function is used to retrieve the power usage reading for the device, in milliwatts. This function is called and executed by the CPU. The highest frequency possible is 66.7 Hz, which means the measurements are done every 15 ms.

Performance Application Programming Interface (PAPI) provides an API to access the hardware performance counters found on modern processors. The various performance metrics can be read through simple programming interfaces from C or Fortran. It could be used as a middleware in different profiling and tracing tools. PAPI can work as a high-level wrapper for different components. Previously it used only Intel RAPL's interface to report the power usage and energy

consumption for Intel CPUs, but recent updates added the NVML component, which supports both measuring and capping power usage on modern NVIDIA GPU architectures. The major advantage of using PAPI is that the measurements are by default synchronized with the kernel execution. The NVML component implemented in PAPI uses the function, *getPowerUsage()* which query *nvmlDeviceGetPowerUsage()* function. According to the documentation, this function is called only once when the command "papi end" is called. Thus, the power returned using this method is an instantaneous power when the kernel finishes execution.

Multi-Threaded Synchronized Monitoring (MTSM) – This method differs from SMA approach in the measurement period, a specific, exact window of the kernel execution is identified which results in recording of only the power reading of the kernel solely. Monitoring part is performed by the host CPU in that way the master thread calls and then monitors the kernel and other threads records the power, therefore it requires the use of parallel programming execution models, such as Pthreads [23] or OpenMP [31]. This approach at first initializes the volatile variable (at master thread) that is used later in recording of power readings. Then, the remaining threads execute the monitoring function in parallel and start measuring the time and power draw as the benchmark kernel starts doing the computation. After its work is done, the timing is ended and the stored measurements are synchronized, giving the precise logs of power consumption during the test period.

2.2. MEASUREMENT HARDWARE

2.2.1. YOKOGAWA WT310E

In order to perform comparison and to check the reliability of software power measurement methods, such as mentioned above Intel RAPL or NVIDIA NVML, it is mandatory to use a certified tool that would serve as the ground truth in such tests. Such a tool is Yokogawa WT310E – an external power meter that will serve this purpose in tests in this paper. It is a digital power analyzer that provides extremely low current measurement capability down to 50 micro-Amps, and a maximum of up to 26 Amps RMS. This device follows standards and certificates such as Energy Star® [32], SPECpower [34] and IEC62301 [20] / EN50564 [35] testing. This model comes from a WT300E's family of appliances that offer a wide range of functions and enhanced specifications, allowing handling of all the measurement applications from low frequency to high frequency inverters using a single power meter. The WT300E series with the fast display update rate of 100ms, offer's engineers a short tact time in their testing procedures. The basic accuracy for all input ranges is 0.1% rdg + 0.05% rng (50/60Hz) and DC 0.1% rdg + 0.2% rng.

To use the Yokogawa WT310E power meter, a special software has been written for easy use – the Yokotool [18]. Yokotool is a command-line tool for controlling Yokogawa power meters in Linux. The tool is written in Python and works with both Python 2.7 and Python 3. The tool comes with the 'yokolibs.PowerMeter' module which can be used from Python scripts.

(Work-In-Progress – Here I can cover a bit more of the use of Yokotool in this project)

(Work-In-Progress: add honorable mentions, such as WattsUp and Kill-A-Watt [12] and their use in previous works)

2.3. BENCHMARK APPLICATIONS

(Work-In-Progress)

[TO DO:

1. Cite 2 works from NPB-CPP and NPB-CUDA github repos
2. Write more about my own multi-gpu benchmark, maybe cite myself]

For the purpose of the experiments, benchmark applications should fully utilize the resources of the tested CPUs and GPUs, as well as be able to run on various configurations parameters. Such parameters are: being able to run in parallel on various numbers of logical processors, being able to run on one or more GPUs and being able to execute on various input parameters class sizes.

There are four benchmark sets, that satisfies mentioned goals:

- NAS Parallel Benchmarks (C++ with OMP)
- NAS Parallel Benchmarks (Fortran with MPI)
- NAS Parallel Benchmarks (CUDA)
- Custom deep learning model, based on XCeptionnet with MPI communication

In a general sense, the NAS Parallel Benchmarks are a set of programs designed and created in order to help evaluate the performance of parallel supercomputers. They are based on computational fluid dynamics applications and originally consisted of five kernels and three pseudo-applications. Later on, the benchmark suite has been extended with more kernels, such as adaptive meshes, parallel I/O, multi-zone applications, and computational grids. Every application comes with predefined and indicated problem size, labeled as class size. Moreover, the benchmark kernels are available in commonly-used programming models like MPI and OpenMP, which allows for easy configuration of use with various number of CPU threads (NPB-OMP and NPB-MPI) [15], as well as execution on two or more computational nodes (NPB-MPI only). For the computations on GPUs, different set has been created, which excels in tests on a single devices (NPB-CUDA) [2] [3].

The original set consists of eight benchmarks, which are tested later in the experiments conducted for the purpose of this work.

Five kernels:

- IS – Integer Sort (random memory access)
- EP – Embarassingly Parallel
- CG – Conjugate Gradient (irregular memory access and communication)
- MG – Multi-Grid on a sequence of meshes
- FT – Discrete 3D fast Fourier Transform (all-to-all communication)

Three pseudo applications:

- BT – Block Tri-diagonal solver
- SP – Scalar Penta-diagonal solver
- LU – Lower-Upper Gauss-Seidel solver

The three pseudo applications mentioned earlier also comes with the multi-zone versions, designed to exploit multiple levels of parallelism. Moreover, NPB suite also offers benchmarks for unstructured computation, parallel I/O and data movement. For the purpose of this work only the original single-zone kernels and applications were used, therefore these benchmarks suites are mentioned only.

In addition to solving different computational problems, each kernel or application operates on various sizes of input data, determined during compilation, known as classes. Those classes

helps choosing the right benchmark in term of execution time, which helps in measurements. Too short benchmarks may cause measurement error, due to low sampling rate of measurement instruments and too long benchmarks are unnecessary, because they prolong the overall experiments.

Benchmark classes:

- Class S – Very small, used for quick test purpose. Nowadays obsolete.
- Class W – The so-called '90's workstation' size, nowadays consisted small.
- Classes A, B, C – Standard test problems (4 times size increase from each of the previous classes)
- Classes D, E, F – Large test problems (16 times size increase from each of the previous classes)

2.3.1. NPB FOR CPU, C++ WITH OMP

(Work-In-Progress)

Explain those benchmarks, how do they works, what problems do they tackle, what are class sizes, how do they differ, why is it useful and so on.

Write that it works on one a two CPUs and on different number of threads – and prove it by screenshots or something.

DON'T EXPLAIN WHICH ONE YOU CHOSE, ADD THAT LATER IN ANOTHER CHAPTER FOR EXAMPLE, 'TEST METHODOLOGY' CHAPTER

2.3.2. NPB FOR CPU, FORTRAN WITH MPI

[PLACEHOLDER]

2.3.3. NPB FOR GPU, CUDA

(Work-In-Progress)

Most of content will be covered in previous subsection, so mention how does it work for GPUs

Mention that it works on SINGLE GPU

2.3.4. CUSTOM DEEP NEURAL NETWORKS MODEL

Explain thoroughly Your own model and the fact it works on GPUs in distributed manner.

3. RELATED WORK

3.1. 'A COMPARATIVE STUDY OF METHODS FOR MEASUREMENT OF ENERGY OF COMPUTING'

Authors of this work [10] investigated the accuracy of measurement of energy consumption during an application execution in order to ensure the application-level energy minimization techniques. They mentioned three most popular methods of measurement: (a) System-level physical measurements using external power meters; (b) Measurements using on-chip power sensors and (c) Energy predictive models. Later, they presented a comparison of the state-of-the-art on-chip power sensors as well as energy predictive models against system-level physical measurements using external power meters, which played the role of a credible measurement appliance.

The methodology is as follows: The ground truth tests were performed at first, using WattsUp Pro Meter[9]. The group of components that were running the benchmark kernel were defined as abstract processor – a whole that consists of multicore CPU processor, consisting of a certain number of physical cores and DRAM. In order to perform meaningful measurements, only a certain configuration of application was run, that executed solely on an abstract processor, without need of using any other system resources, such as solid state drives or network interface cards and so on. The result of such an approach is that HCLWattsUp reflects solely the power drawn from CPU and DRAM.

The researchers took other important precautions during the experiment. At first, all the resources they used have been reserved and fully dedicated to the experiment, making sure that no unwanted, third-party applications are being run in the background. During the tests, they actively monitored the disk consumption and network to ensure that there is no interference in the measurements. Another important factor of the testbed that draws power and generates heat are the fans. In default configuration, the fans speed is dependent on the increasing temperature of the CPUs, which rises as the time of the training goes on. That generates a dynamic energy consumption that impacts negatively on the outcomes of the experiment. To rule this phenomenon out, all the fans in the entire testbed are set at full speed, therefore they draw the same amount of power regardless of the actual strain put on CPUs and their temperature. All the procedures mentioned above ensure that the dynamic energy consumption obtained using HCLWattsUp, reflects the contribution solely by the abstract processor executing the given application kernel.

After determining the "ground truth" measurements using the configuration with HCLWattsUp, the researchers conducted a series of tests that determined the dynamic energy consumption by the given application using RAPL. At first the Intel PCM/PAPI was used to obtain the base power of CPUs, both core and uncore as well as DRAM, with no straining workload applied. Then, in the next phase, using HCLWattsUp API the execution time of the given application has been obtained. After that, the Intel PCM/PAPI has been used in order to obtain the total consumption of the CPUs and DRAM, all within the execution time of a given benchmark. Lastly, the researchers responsible for the experiment calculated the dynamic energy consumption of the abstract processor by subtracting the base energy from the total energy drawn during the kernel execution. To determine the dynamic energy consumption using HCLWattsUp, all the steps mentioned before has been repeated, but using the HCLWattsUp software instead of the Intel RAPL.

The execution time of the benchmark kernels were the same for both of the power draw measurement tools, so any difference between the energy readings of the tools comes solely from their power readings. Finally, tests were conducted on three different sets of experiments in order

to receive three different types of patterns.

In the first set of experiments, the FFTW and MKL-FFT energy consumption has been explored, by using a given workload size. For many tests on various problem sizes, the Intel RAPL reports showed less dynamic energy consumption for all application configurations than HCLWattsUp, but it follows the same pattern as HCLWattsUp for most of the data points. Therefore it is possible to calibrate the RAPL readings, which resulted in significant decrease of average error for the dynamic energy profile.

Second set of tests was conducted using OpenBLAS DGEMM. Executions were, again, performed using various configurations of data sizes, but results were less satisfactory than in the first tests. Like the first set of experiments, RAPL profiles lag behind the HCLWattsUp profiles. Unlike the first set of experiments where the error between both the profiles could be reduced significantly by calibration, the reduction of the average error for most of the application configurations was only as half as effective contrary to the first set of tests. This calibration, however, is again not the same for all the application configurations.

In the third and last set of experiments the team studied the dynamic energy behavior of FFTW as a function of problem size N x N. The tests were performed in different problem ranges. Researchers claim that for many data points, RAPL reports an increase in dynamic energy consumption with respect to the previous data point in the profile whereas HCLWattsUp reports a decrease and vice versa. Therefore it is impossible to use calibration to reduce the average error between the profiles because of their interlacing behavior.

As a conclusion, readings from Intel RAPL and HCLWattsUp differ strongly based on executed benchmark and data size. In the first set of experiments, the FFTW and MKL-FFT energy consumption test, the Intel RAPL readings followed the pattern of HCLWattsUp readings, being even more accurate after calibrations. The second test however, showed that RAPL does not follow most of the energy consumption pattern of the power meter. This could be tuned to some extent by calibration, but not as good as in the first test case. In the last experiment, however, the RAPL does not follow the energy consumption pattern of the power meter and can not be calibrated, leaving the readings quite troublesome.

Next experiment conducted in this paper was the comparison of measurements by GPU and Xeon Phi Sensors with HCLWattsUp. The methodology of work is similar to the one explained before – the entire testbed is reserved solely for the purpose of the experiment, the fans are set to the maximum speed and only the abstract processor is measured. To strain the hardware, two applications were used:

The first one was the matrix multiplication (DGEMM), the second one was 2D-FFT. Tests were performed on two NVIDIA GPUs: K40c [13] and P100, and one Intel coprocessor, the Intel Xeon Phi 3120P [16]. To obtain the power values from on-chip sensors on NVIDIA GPUs, the dedicated libraries were used, called NVIDIA NVML [27] and to obtain power values from Intel Xeon Phi, the Intel System Management Controller chip (SMC) [21] was used. Values from the Intel Xeon Phi can be programmatically obtained using Intel manycore platform software stack (MPSS) [17]. The methodology taken to compare the measurements using GPU and Xeon Phi sensors and HCLWattsUp is similar to this for RAPL. Briefly, HCLWattsUp API provides the dynamic energy consumption of an application using both CPU and an accelerator (GPU or Xeon Phi) instead of the components involved in its execution. Execution of an application using GPU/Xeon Phi involves the CPU host-core, DRAM and PCIe to copy the data between CPU host-core and GPU/Intel Xeon Phi. On-chip power sensors (NVML and MPSS) only provide the power consumption of GPU or Xeon Phi only. Therefore, to obtain the dynamic energy profiles of applications, the Intel RAPL was used to determine the energy contribution of CPU and DRAM. Energy contributions

from data transfers using PCIe were considered as not significant.

At first, the DGEMM was used as the test benchmark with various workload sizes. The energy readings from the GPU NVIDIA K40c sensors exhibit a linear profile whereas HCLWattsUp does not. Moreover, the sensor does not follow the application behavior exhibited by HCLWattsUp for approximately two-thirds of the data points. In the case of the Intel Xeon Phi coprocessor, the results seemed to be better – sensors follow the trend exhibited by HCLWattsUp for third-fourth of the data points. However, sensors report higher dynamic energy than HCLWattsUp, but that can be reduced significantly using calibration.

In the case of the second benchmark, the 2D-FFT, the measurements by NVML follow the same trend for the majority of the data points, compared to the results from NCLWattsUp. The sensor of the Intel Xeon Phi followed the trend of HCLWattsUp for over 90% of all data points, which is a good result. Overall, Intel RAPL and NVML both exhibit the same trend for FFT. Therefore, the difference with HCLWattsUp comes from both sensors collectively.

The results of this test allows to draw several conclusions. First, the average error between measurements using sensors and HCLWattsUp can be reduced using calibration, which is, nevertheless, specific for an application configuration. Another important finding is that CPU host-core and DRAM consume equal or more dynamic energy than the accelerator for FFT applications (FFTW 2D and MKL FFT 2D),which means that data transfers (between CPU host-core and an accelerator) consume same amount of energy as that for computations on the accelerator for older generations of NVIDIA Tesla GPUs such as K40c and Intel Xeon Phi such as 3120P. However, for newer generations of Nvidia Tesla GPUs such as P100, the data transfers consume more dynamic energy than computations. It suggests that optimizing the data transfers for dynamic energy consumption is important.

3.2. 'VERIFIED INSTRUCTION-LEVEL ENERGY CONSUMPTION MEASUREMENT FOR NVIDIA GPUS'

Authors of this research paper [1] investigated the actual cost of the power/energy overhead of the internal microarchitecture of various NVIDIA GPUs from four different generations. In order to do so, they compared over 40 different PTX instructions and showed the effect of the CUDA compiler optimizations on the energy consumption of each instruction. To measure the power consumption, they used three different software techniques to read the GPU on-chip power sensors and determined their precision by comparing them to custom-designed hardware power measurement. The motivation of their work comes from the fact that in order to increase the performance of the GPUs, their power consumption must be correctly and reliably measured, because it serves as a primary metric of performance evaluation. This issue is proven even more challenging, since the GPU vendors never publish the data on the actual energy cost of their GPUs' microarchitecture, therefore the independent research should be conducted in order to verify the power measurement software they provide.

The authors of the research paper prepared a set of special micro-benchmarks to stress the GPU, in order to capture the power usage of each PTX instruction [29], so the instructions were written in PTX as well. PTX is a virtual-assembly language used in NVIDIA's CUDA programming environment whose purpose is to control the exact sequence of instructions executing without any overhead. The researchers prepared two kernels for the purpose of this work – first one is tasked with adding integers and second one responsible for dividing variables with unsigned values. Since it is impossible to capture power draw of an execution of a single instruction, a different approach was proposed: the same instruction has been repeated millions of times and the power

drawn during the entire test case has been measured. Then the amount of power reported by the measuring system was divided by the total number of instructions, giving the power consumed by a single PTX instruction. It is worth noting that GPUs drain power as static power and dynamic power. The static power is a constant power that the GPU consumes to maintain its operation. To eliminate the static power and any overhead dynamic power other than the instruction power consumption, the kernel's was computed twice and the energy consumption was measured both times. First, the kernel was run in a configuration to measure the total energy drawn for the operation. In the second run the back-to-back instructions were omitted and the energy measured was defined as overhead energy. Energy used on instruction was defined as subtraction of total energy and overhead energy, divided by the total number of instructions.

In this experiment, the ground truth of energy drawn by the GPUs was set by the external power meter. It was pointed out that the GPUs have two power sources: one is direct DC power, provided by a PSU, another one is the PCI-E power source, provided through the motherboard. In order to capture the total power, the measurement of current and voltage has been done for each power source simultaneously. A clamp meter and a shunt series resistor were used for the current measurement. For voltage measurement, a direct probe on the voltage line using an oscilloscope has been used. In case of measurements of current and voltage on the direct DC power supply source, everything was measured using an oscilloscope, therefore the power draw calculations were performed using a certain formula. The measurement of the PCI-E power source was more difficult. Since there wasn't any possibility to directly receive current or voltage, the authors of this paper decided to set up a special PCI-E riser board that measures the power supplied through the motherboard. Two in-series shunt resistors are used as a power sensing technique. Using the series property, the current that flows through the riser is the same current that goes to the graphics card, same with the voltages.

The experiment has been conducted for four NVIDIA GPUs from four different generations/architectures: GTX TITAN X from Maxwell architecture, GTX 1080 Ti from Pascal architecture, TITAN V from Volta architecture and TITAN RTX from Turing architecture. To compile and run the previously prepared benchmarks, the CUDA NVCC compiler [25] has been used. The results of the tests show that NVIDIA TITAN V has the lowest energy consumption per instruction among all the tested GPUs. Additionally the tests were performed on both CUDA optimized and non-optimized versions of code, and overall the optimized versions of instruction proved to be less energy hungry than the non-optimized ones. In terms of differences between various software power measuring techniques, namely PAPI versus MTSM, The dominant tendency of the results is that PAPI readings are always more than the MTSM. Although the difference is not significant for small kernels, it can be up to 1 µJ for bigger kernels like Floating Single and Double Precision div instructions. Different software techniques (MTSM and PAPI) have been compared against the hardware setup on Volta TITAN V GPU. Compared to the ground truth hardware measurements, for all the instructions, the average Mean Absolute Percentage Error (MAPE) of MTSM Energy is 6.39 and the mean Root Mean Square Error (RMSE) is 3.97. In contrast, PAPI average MAPE is 10.24 and the average RMSE is 5.04. The results prove that MTSM is more accurate than PAPI as it is closer to what has been measured using the hardware.

3.3. 'MEASURING GPU POWER WITH THE K20 BUILT-IN SENSOR'

Authors of this research paper [7] investigated accurate profiling of the power consumption of GPU code when using the on-board power sensor of NVIDIA K20 GPUs. Moreover, two major anomalies that happened during the tests were more thoroughly analyzed – the first one being related to the doubling a benchmark kernel's runtime resulted with more than double energy usage, the second indicated that running two kernels in close temporal proximity inflates the energy consumption of the later kernel. Based on previous work in a similar field and set of preliminary tests, a new, reliable methodology [19] has been proposed as the conclusion of this experiment.

GPUs used in this project are NVIDIA Tesla K20, equipped with on-board sensors for querying the power consumption at runtime. As noted by the authors of the work, measurement of the power draw of the GPU using its built-in sensor is more complex than it would seem at first glance. The straightforward approach of sampling the power, computing the average, and multiplying by the runtime of the GPU code is likely to yield large errors and nonsensical results, hence the anomalies related to more energy used than expected due to increase of kernel's runtime or kernel's energy consumption increase after consecutive runs. Therefore another approach must be adopted. Methodology of the experiment is as follows: a number of unexpected behaviors when measuring a GPU's power consumption have been noted for further investigation, various observations has been noted during the tests runs conducted on the NVIDIA K20 GPUs and based on those observations and other related work, a correct way of measuring the power and energy consumption using sensor has been created. Later on it was validated for reliability by performing it multiple ways on many GPUs based on Kepler architecture, equipped with power sensor, such as the NVIDIA K20c, K20m, and K20x. The custom tool, created by the authors of the work, has been published for future use by other scientists, as an open source code.

Benchmark applications used in this paper solved two different n-body problem implementations. The algorithm models the simulation of gravity-induced motion of stars in a star cluster. The first kernel, called NB (N-Body), performs precise pairwise force calculations, which means that the same operations are performed for all n bodies, leading to a very regular implementation that maps well to GPUs. Moreover, the force calculations are independent, resulting in large amounts of parallelism. The second code, called BH, uses the Barnes-Hut algorithm to approximately compute the forces [9] [36]. It hierarchically partitions the volume around the n bodies into successively smaller cubes, called cells, until there is just one body per innermost cell. The resulting spatial hierarchy is recorded in an unbalanced octree. Each cell summarizes information about the bodies it contains. The NB code is relatively straightforward, has a high computational density, and only accesses main memory infrequently due to excellent caching in shared memory. In contrast, the BH code is quite complex, has a low computational density, performs mostly irregular [6] pointer-chasing memory accesses, and consists of multiple different kernels. Nevertheless, because of its lower time complexity, it is about 33 times faster than the NB code when simulating one million stars.

In order to conduct the energy measurement from the GPU power sensor, the authors of the work wrote their own tool to query the sensor via the NVIDIA Management Library (NVML) interface, which returns the power readings in milliwatts. The sampling intervals of the measurement are lowest possible – 15 ms between measurements. At first, during the tests, there was a noticeable power lag and measurement distortion – power profiling tends to lag behind the kernel activity and shape of the profile does not match the kernel activity in both shape (minor difference) and time (major difference). The key insight in creating a model of correct measurement is the fact

that the power sensor gradually approaches the true power level rather than doing so instantly. Since the 'curved' power readings between time when kernel start running and time when the power curve stabilizes reminded the authors of this work of capacitors charging and discharging, they tested whether the power profiles can be described by the same formulas. This turned out to work very well in the end. It is assumed that this is the case because the power sensor hardware uses a capacitor of some sort. After this revelation, the authors determined the 'capacitance' of the power sensor by using a single capacitor function to approximate the curve between the kernel start time and kernel stop time. After that, they determined the value of the capacitance that minimizes the sum of the differences between the measured values and the function values. As the capacitance is constant, it only needs to be established once for a given GPU, which is C = 833.333 on all tested K20 GPUs. Computing the true power draw value then become a single function of the slope of power profile derived in time domain and is shown in a function below:

Ptrue(ti)=Pmeasured(ti) + C (Pmeasured(ti+1) - Pmeasured(ti-1))(ti+1 - tt-1) [W]

Moving back onto the recommended steps for this experiment, following assumption should be considered: highest possible sample rate for NVML (which is 66.7 Hz / 15 ms between intervals) as well as including the time stamps, removal of consecutive samples of the same value that are no more than 4 ms apart of each other, computation of true power with the equation mentioned above and finally, computation of the true energy consumption by integrating the true power, using the time stamps, over all intervals where the power level is above the 'active idle' threshold of 52.5 W.

After incorporating the steps mentioned above in the tests, the authors of the research paper validated their results. To do so, they checked if the computed power profile follows the GPU kernel activity and also they revisited anomalies that they encountered before in order to check if their new approach eliminates them. In the end, the profiling almost instantly shoots up when the kernel starts, stays at a (more or less) constant level during execution, and almost instantly drops to the aforementioned 52.5 W after the kernel stops. Importantly, the power level during execution coincides with the asymptotic power between kernel start and kernel stop, which verifies the above hypothesis. This observation gives insight that power should be integrated from the time point of kernel start to the point of kernel end. Any energy consumption by the GPU before or after the kernel execution is due to idling (at different power levels) and is a function of time but independent of the kernel. In case of anomalies, the first one was regarding the kernel runtime changes and unintuitive increase of measured power draw. In the early tests that were based solely on readings from NVML, after increasing the kernel's runtime by two times, the power draw readings were not increased proportionally as well, they were higher by approximately 8% than expected. The corrected power profile, however, indicates that, in fact, the energy consumption increase follows the total runtime one-to-one, thus resolving this particular anomaly. The second anomaly was that running the same kernel twice in close temporal proximity inflates the energy consumption of the second invocation. Once again, the power profiling proposed by the author clearly resolves this anomaly as the two corrected profiles are now at the same level (and the power level between the kernel runs is at the active-idle level). In other words, the computed power profile of a kernel is unaffected by prior kernel runs, which is an important advantage of this approach. This means that there is no need to have to delay kernel runs until the GPU reaches its idle power level before one can measure the energy consumption of the next kernel. Those tests were also validated on other GPUs and the results showed that they behave in a similar manner to the first one. The only notable difference is that all of the measurements are a few watts higher on the second GPU. The difference is, however, within the 5 W absolute accuracy of the sensor. The profiled power obtained from tests on other GPUs are all very similar to each other, which

further validates the used methodology.

As a conclusion of this work, many results and insights were obtained, such as: Power profile is distorted with respect to the kernel activity; the measured power lags behind the kernel activity; running multiple kernels one after another inflates the power draw of the later kernels; after a short-running kernel, the power draw can even increase for a while; integrating the power to a discernable time after a kernel stops does not correctly compensate for the power lag; the sampling interval lengths vary greatly; the GPU sensor only performs power measurements once in a while; the true sampling rate may be too low to accurately measure short-running kernels and the PCI-bus activity is not included in the sensor's measurements. This paper proposes and evaluates a power- and energy-measurement methodology that is accurate even in the presence of the above problems. It computes the true instant power consumption from the measured power samples, accounts for variations in the sampling frequency, and correctly identifies when kernels are running and when the GPU is in active-idle mode waiting for the next kernel launch.

4. PROPOSAL OF A SOLUTION TO THE PROBLEM

4.1. [PLACEHOLDER]

5. EXPERIMENTS

5.1. PROPOSED WORKFLOW AND METHODOLOGY

[PLACEHOLDER]

5.2. TESTBED APPLICATIONS AND CONFIGURATIONS

[PLACEHOLDER]

5.3. TESTBED ENVIRONMENT

[PLACEHOLDER]

5.4. TESTS AND RESULTS

[PLACEHOLDER]

5.5. DISCUSSION

[PLACEHOLDER]

6. SUMMARY AND FUTURE WORK

6.1. SUMMARY

[PLACEHOLDER]

6.2. FUTURE WORK

[PLACEHOLDER]

RESEARCH PAPERS

[1] Yehia Arafa et al. “Verified Instruction-Level Energy Consumption Measurement for NVIDIA GPUs”. In: *Proceedings of the 17th ACM International Conference on Computing Frontiers*. CF ’20. Catania, Sicily, Italy: Association for Computing Machinery, 2020, pp. 60–70. ISBN: 9781450379564. DOI: 10.1145/3387902.3392613. URL: https://doi.org/10.1145/3387902.3392613.

[2] Gabriell Araujo et al. “NAS Parallel Benchmarks with CUDA and beyond”. In: *Software: Practice and Experience* 53.1 (2023), pp. 53–80. DOI: https://doi.org/10.1002/spe.3056. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.3056.

[3] Gabriell Alves de Araujo et al. “Efficient NAS Parallel Benchmark Kernels with CUDA”. In: *2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. 2020, pp. 9–16. DOI: 10.1109/PDP50117.2020.00009.

[4] Maria Avgerinou, Paolo Bertoldi, and Luca Castellazzi. “Trends in Data Centre Energy Consumption under the European Code of Conduct for Data Centre Energy Efficiency”. In: *Energies* 10.10 (2017). ISSN: 1996-1073. DOI: 10.3390/en10101470. URL: https://www.mdpi.com/1996-1073/10/10/1470.

[5] David H. Bailey. “NAS Parallel Benchmarks”. In: *Encyclopedia of Parallel Computing*. Ed. by David Padua. Boston, MA: Springer US, 2011, pp. 1254–1259. ISBN: 978-0-387-09766-4. DOI: 10.1007/978-0-387-09766-4_133. URL: https://doi.org/10.1007/978-0-387-09766-4_133.

[6] Martin Burtscher, Rupesh Nasre, and Keshav Pingali. “A quantitative study of irregular programs on GPUs”. In: *2012 IEEE International Symposium on Workload Characterization (IISWC)*. Nov. 2012, pp. 141–151. DOI: 10.1109/IISWC.2012.6402918.

[7] Martin Burtscher, Ivan Zecena, and Ziliang Zong. “Measuring GPU Power with the K20 Built-in Sensor”. In: *Proceedings of Workshop on General Purpose Processing Using GPUs*. GPGPU-7. Salt Lake City, UT, USA: Association for Computing Machinery, 2014, pp. 28–36. ISBN: 9781450327664. DOI: 10.1145/2588768.2576783. URL: https://doi.org/10.1145/2588768.2576783.

[8] Howard David et al. “RAPL: Memory power estimation and capping”. In: *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*. Aug. 2010, pp. 189–194. DOI: 10.1145/1840845.1840883.

[9] Walter Dehnen. “A Hierarchical (N) Force Calculation Algorithm”. In: *Journal of Computational Physics* 179.1 (June 2002), pp. 27–42. DOI: 10.1006/jcph.2002.7026. URL: https://doi.org/10.1006%2Fjcph.2002.7026.

[10] Muhammad Fahad et al. “A Comparative Study of Methods for Measurement of Energy of Computing”. In: *Energies* 12.11 (2019). ISSN: 1996-1073. DOI: 10.3390/en12112204. URL: https://www.mdpi.com/1996-1073/12/11/2204.

[11] Chao Jin et al. “A survey on software methods to improve the energy efficiency of parallel computing”. In: *The International Journal of High Performance Computing Applications* 31.6 (2017), pp. 517–549. DOI: 10.1177/1094342016665471.

[12] Kiran Kasichayanula et al. “Power Aware Computing on GPUs”. In: *2012 Symposium on Application Accelerators in High Performance Computing*. July 2012, pp. 64–73. DOI: 10.1109/SAAHPC.2012.26.

[13] Hamidreza Khaleghzadeh et al. “Out-of-Core Implementation for Accelerator Kernels on Heterogeneous Clouds”. In: *J. Supercomput.* 74.2 (Feb. 2018), pp. 551–568. ISSN: 0920-8542. DOI: 10.1007/s11227-017-2141-4. URL: https://doi.org/10.1007/s11227-017-2141-4.

[14] Adam Krzywaniak, Jerzy Proficz, and Pawel Czarnul. “Analyzing energy/performance trade-offs with power capping for parallel applications on modern multi and many core processors”. In: Sept. 2018, pp. 339–346. DOI: 10.15439/2018F177.

[15] Júnior Löff et al. “The NAS Parallel Benchmarks for evaluating C++ parallel programming frameworks on shared-memory architectures”. In: *Future Generation Computer Systems* 125 (2021), pp. 743–757. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2021.07.021. URL: https://www.sciencedirect.com/science/article/pii/S0167739X21002831.

[16] Rezaur. Rahman and SpringerLink (Online service). *Intel® Xeon Phi™ Coprocessor Architecture and Tools*. Berkeley, CA : Apress : 2013. URL: https://doi.org/10.1007/978-1-4302-5927-5.

[17] Efraim Rotem et al. “Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge”. In: *IEEE Micro* 32.2 (Mar. 2012), pp. 20–27. ISSN: 1937-4143. DOI: 10.1109/MM.2012.12.

ADDITIONAL RESOURCES

[18] Artem Bityutskiy, Antti Laakso, and Helia Correia. *Yokotool*. URL: https://github.com/intel/yoko-tool.

[19] Martin Burtscher. *K20Power v1.1*. URL: https://userweb.cs.txstate.edu/~burtscher/research/K20power/.

[20] International Electrotechnical Commission. *IEC 62301:2011 | IEC Webstore | energy efficiency, smart city, standby power*. URL: https://webstore.iec.ch/publication/6789.

[21] Intel. *Intel® Xeon Phi™ Coprocessor DEVELOPER'S QUICK START GUIDE*. URL: https://www.intel.com/content/dam/develop/external/us/en/documents/intel-xeon-phi-coprocessor-quick-start-developers-guide.pdf.

[22] Intel. *Running Average Power Limit Energy Reporting*. URL: https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/advisory-guidance/running-average-power-limit-energy-reporting.html.

[23] Michael Kerrisk. *Pthreads - Linux manual page*. URL: https://man7.org/linux/man-pages/man7/pthreads.7.html.

[24] NASA. *NAS Parallel Benchmarks*. URL: https://www.nas.nasa.gov/software/npb.html.

[25] NVIDIA. *NVIDIA CUDA Compiler Driver NVCC*. URL: https://docs.nvidia.com/cuda/cuda-compiler-driver-nvcc/.

[26] NVIDIA. *NVIDIA Management Library (NVML)*. URL: https://developer.nvidia.com/nvidia-management-library-nvml.

[27] NVIDIA. *NVML API Reference Guide*. URL: https://docs.nvidia.com/deploy/nvml-api/index.html.

[28] NVIDIA. *PAPI CUDA Component*. URL: https://developer.nvidia.com/papi-cuda-component.

[29] NVIDIA. *Parallel Thread Execution ISA Version 8.2*. URL: https://docs.nvidia.com/cuda/parallel-thread-execution/index.html.

[30] NVIDIA. *System Management Interface SMI*. URL: https://developer.nvidia.com/nvidia-system-management-interface.

[31] OpenMP. *OpenMP Reference Guides*. URL: https://www.openmp.org/resources/refguides/.

[32] ICR Polska. *EnergyStar - ICR POLAND - testing and certification*. URL: https://icrpolska.com/en/energystar/.

[33] The Parliamentary Office of Science and Technology. *Energy Consumption of ICT*. URL: https://researchbriefings.files.parliament.uk/documents/POST-PN-0677/POST-PN-0677.pdf.

[34] SPEC. *SPECpower and Performance Committee*. URL: https://www.spec.org/power/.

[35] iTeh Standards. *Electrical and electronic household and office equipment - Measurement of low power consumption*. URL: https://standards.iteh.ai/catalog/standards/clc/371d2d67-a439-4f20-96f0-02675496fd03/en-50564-2011.

[36] ISS Group at the University of Texas. *LonestarGPU*. URL: https://iss.oden.utexas.edu/?p=projects/galois/lonestargpu.

[37] Yokogawa. *WT300E Digital Power Analyzer*. URL: https://tmi.yokogawa.com/solutions/products/power-analyzers/digital-power-meter-wt300e/#Documents-Downloads.

[38] Yokogawa. *WT300E Series Digital Power Meter*. URL: https://cdn.tmi.yokogawa.com/1/2562/files/BUWT300E-01EN.pdf.

LIST OF FIGURES

LIST OF TABLES