Quality Assessment of GPU Power Profiling Mechanisms

Satyabrata Sen
Computational Sciences & Engineering Div.
Oak Ridge National Laboratory, TN 37831
sens@ornl.gov

Neena Imam
Computational Research & Dev. Programs
Oak Ridge National Laboratory, TN 37831
imamn@ornl.gov

Chung-Hsing Hsu
Computer Science & Mathematics Div.
Oak Ridge National Laboratory, TN 37831
hsuc@ornl.gov

***Abstract*—Accurate component-level power measurements are nowadays essential for the design and optimization of high-performance computing (HPC) systems and applications. Particularly, as more and more heterogeneous HPC systems are developed, the characterizations of GPU power profiles have become extremely crucial because, although GPUs provide exceptional performance, they do consume substantial amounts of power. Currently, there are various GPU power profiling mechanisms available; however, there is no standard way to assess the quality of such profiling schemes. To address this issue, in this paper, we develop an assessment methodology to rate the quality and performance of the profiling mechanism itself. Specifically, we present the assessments of four different GPU power profiling techniques: (i) Nvidia's NVML via Allinea MAP, (ii) Nvidia's NVML via direct reads, and (iii) Penguin Computing's PowerInsight (PI) via two vendor-provided drivers, and (iv) PowerInsight via Allinea MAP. In addition, we discuss the effects of moving-average filters to explain the slow variations of some of the measured power profiles. Based on our assessment, the GPU power profiling mechanism using PI device outperforms the other schemes by reliably measuring the ground-truth power profile generated by a GPU stress-test benchmark.**

***Keywords*-GPU; power profiling; quality assessment; Tesla K20c; NVML; Allinea MAP; PowerInsight.**

I. Introduction

The evolution of high-performance computing (HPC) systems has always been driven exclusively by their computing performance in terms of reducing time-to-solution and increasing the resolution of models and problems being solved. However, while making efforts to improve the peak performance of parallel supercomputers beyond Exa-FLOPS, it has been realized over the past decade that energy is becoming one of the most expensive resources; in some cases, the total energy cost of an HPC installation over its lifetime (5-7 years) is already comparable to its initial cost of acquisition and deployment [1]. Now, various components in a computing machine, such as CPU, GPU, memory, etc., consume power at different scales, some are power-hungry and some are not. Therefore, understanding the component-level power consumption behavior is critical for figuring out the overall power-budget of future Exascale HPC machines.

In particular, because of their energy efficiency, the graphics processing units (GPUs) are increasingly becoming prevalent in recent HPC systems; for example, the upcoming Summit supercomputer at the Oak Ridge National Laboratory will have approximately $27,600$ Nvidia Tesla V100 GPUs [2]. Therefore, to accurately collect and analyze the GPU power measurements, currently there are multiple component-level power-profiling mechanisms available on the market. However, no standards exist for component-level mechanisms, and it is very challenging to establish the ground truth. Hence, the question arises 'how to assess or rate the quality/performance of a component-level power profiling mechanism?' In this paper, we address this issue by developing an assessment methodology that characterizes the profiling mechanism itself. It is to be noted here that we consider a profiling mechanism including all the associated hardware and software involved in that profiling workflow, because they all contribute to the overall quality and performance of the mechanism. However, we do not characterize the individual hardware/software modules, but provide an assessment of the overall profiling scheme.

Specifically, we present the assessments of four different techniques that measure power variations on an Nvidia Tesla K20c GPU system: (i) Nvidia's NVML via Allinea MAP tool, (ii) Nvidia's NVML via direct reads, and (iii) Penguin Computing's PowerInsight via two vendor-provided drivers, namely, getRawPower() and powerInsight(), and (iv) PowerInsight via Allinea MAP. First, we develop a GPU stress-test benchmark to generate a series of high-low, square-wave like power profile on the GPU, and apply the above-mentioned four profiling mechanisms to record the measured power values. Then, the measured power profiles are compared with the ground-truth profile to check how reliably they match with the expected power profiles. The quality of the profiling scheme is measured in terms of the smallest period (or highest frequency) of the high-low power variations of the ground-truth profile which the measured power profile can reliably exhibit. Hence, smaller the period (or higher the frequency) of the ground-truth power profile, the better the quality of the profiling mechanism.

The rest of the paper is organized as follows. In Section II, we describe a brief overview of the GPU power profiling mechanisms, and in Section III, we introduce the developed GPU stress-test benchmark and the hardware specifications of the compute node. Then, we provide a detailed assessment of the GPU power profiling schemes in Section IV. We also discuss the effects of moving-average filter, and the performances of two variants of the developed benchmark in Section V. Conclusions are presented in Section VI.

II. Overview of GPU Power Profiling

In this section, we provide a brief overview of the GPU power profiling mechanisms using both internal and external hardware sensors; a comprehensive survey of different GPU power profiling methodologies can be found in [3] and references therein.

Traditionally, the most comprehensive hardware monitoring software has been the Performance API (PAPI) [4], [5], which offers access to internal hardware counters primarily of CPUs. Its extended version Component PAPI (PAPI-C) now provides information from thermal sensors, GPU counters, memory interface chips, network interface cards, and network switches. Nvidia Fermi and Tesla architecture GPUs are now equipped with internal power sensors, whose measured data can be accessed using various profiling tools from Nvidia, including Nvidia Nsight, Nvidia Visual Profiler, Nvidia CUDA Profiling Tools Interface (CUPTI), and Nvidia Management Library (NVML) [6]-[8]. For example, various aspects of measured power profiles using on-board sensors available on K20 GPUs are discussed in [9] using the NVML based mechanisms. Similar to the Nvidia tools, the Mali GPU Shader Development Studio and AMD's PerfApi tools are used to access profiling information from the internal counters of ARM's Mali and AMD's Radeon GPUs respectively [10], [11]. Additionally, third-party profiler, such as Allinea MAP (which is a part of Allinea Forge tool suite) [12], can be used to report component and node level power measurements.

To measure the power usage using external hardware sensors, a common method is to measure the current drawn by the server/system under investigation from the power supply unit (PSU) [13]. Recently, to standardize the component-level power measurement process, some efforts have been made to develop multilevel power monitoring solutions for computing clusters. For example, *PowerPack* [14], [15] is a profiling system, comprised of several hardware sensors and a software package, that simultaneously measures component, node, and system level power data. *PowerMon/PowerMon2* [16] are newer power measurement devices designed by the Renaissance Computing Institute; they use six/eight collection channels with sampling rate up to $3,072$ Hz, and both are limited to in-band collection. *PowerInsight* (PI) [17], [18] is another customized power profiling and measurement device commercially available from Penguin Computing [19]; it offers both in-band and out-of-band power monitoring with a sampling rate on the order of 1 kHz. On the PI device, two vendor-provided drivers getRawPower() and powerInsight() can be used to collect and report accurate measurements.

In order to acquire component-level power consumption data, in general benchmark programs are repeatedly executed to stress the CPU or GPU in some particular manner. In particular, the HPC researchers consider LINPACK benchmark, whereas the consumer performance enthusiasts and overclockers prefer Prime95 benchmark. However, as stress test utilities, these compute intensive routines are unnecessarily hard to use. Alternatively, FIRESTARTER [20], [21] is an open-source benchmark application that is specifically designed to simulate various loads on the CPU for a specific amount of time. For example, it can create a high load for half of a second and a low load for another half, and can then repeat such a high-low load pattern for certain amount of time. An equivalent version of this program as the GPU stress test utility is the FIRESTARTER-CUDA, which implements a matrix-multiplication operation on the GPUs.

Algorithm 1 matrix-CUDA program

```
1: Input: matrix-size $m$, repetition no. $r$, sleep duration $s$
2: Initialize: Matrix $A(m,m)$, $B(m,m)$
3: for ($ii = 1$; $ii <=$ r; $ii++$) do
4:     cudaMalloc():    to allocate device memory space
5:     cudaMemcpy():  to copy from host to device memory
6:     gpu-matrix-multiplication($A$,$B$,$C$,$m$)   for high load
7:     cudaMemcpy():  to copy from device to host memory
8:     cudaFree():    to free up device memory
9:     waitFor(usleep(),s)    for low load
```

We ran the FIRESTARTER-CUDA program with various parameters; for example, 0%, 50%, 100% high-load, from 0.1 sec to 10 sec period, from 1024 to the default 12288 matrix size, and 30 sec of total runtime. The resulting power data were collected via two methods: (i) Allinea MAP, and (ii) getRawPower() code of PI device. We noticed from these measurements that CPU power pattern reliably captured the high-low variations of constructed load; however, the GPU power pattern only showed high-load output throughout the entire execution time of the FIRESTARTER-CUDA program. Therefore, we need to develop a simple GPU stress-test benchmark that can reliably generate the high-low power pattern on the GPUs.

III. Benchmark and Hardware Specifications

A. Benchmark for Generating Ground-Truth Profiles

We developed a GPU benchmark, matrix-CUDA, in order to generate the ground-truth power profiles with high-low load variations on the GPU; a pseudo-code is shown in Algorithm 1. Similar to the FIRESTARTER-CUDA benchmark, the matrix-CUDA code also implements a matrix-multiplication operation to simulate high-load on the GPUs. The low-load is generated using the built-in *usleep()* function. In the current version, matrix-CUDA accepts three arguments: (i) size of the matrix (m), (ii) number of repetitions of high-low pattern (r), and (iii) sleep-time in micro-seconds (s). For example, a usage like *./matrix-cuda* 4800 15 1000000 implies that the matrix-CUDA program would first perform multiplication of two matrices of size 4800 on the GPU, then sleep for 1 sec, and repeat these two steps one after another for 15 times.

B. Compute Node Specifications

We conducted the power profiling tests on a single compute node from a hybrid HPC 31-node cluster, 19 of these nodes have a base configuration comprised of dual Intel Xeon E5-2650v2 i7 2.6 GHz processors with 64 GB DDR3 memory and 2 x 1 TB SATA2 disks. Each node is connected to both an Ethernet management network and a FDR infiniband (IB) fabric. A 1 Gb Ethernet adapter connects to the management network and a dual port FDR (56 Gb/s) IB adapter connects to the IB fabric. The remaining 12 nodes of the cluster include 8 with a combination of either Nvidia Tesla K20c or K40c GPU with a AMD Firepro 9000 6 GB GDDR5, 2 with Intel Xeon Phi co-processors, and 2 with an AMD A10-7850K APU processor instead of the Intel Xeons. Additionally, PowerInsight (PI) boards are attached to these nodes, and the corresponding PI controllers are installed on the nodes. The PI controller captures and reports power usage of the CPU, GPU, memory, and other peripherals.

IV. GPU Power Profiling Assessments

We ran the matrix-CUDA benchmark on the compute node having the Tesla K20c GPU with various arguments; for example, we changed the matrix size from 1000 to 5000, the sleep-time from 0.02 sec to 1 sec, and thus varied the frequency of high-low power pattern from 0.5 Hz to 20 Hz. In each run, we kept the number of repetitions $r = 15$. We used four methods to collect the resulting GPU power measurements: (i) NVML via Allinea MAP, (ii) NVML via direct reads, (iii) PI device, and (iv) PI via Allinea MAP. In the following subsections, we discuss in detail the quality and performance of each of these profiling mechanisms.

A. NVML via Allinea MAP

To record the output power patterns measured using NVML via Allinea MAP, we ran the matrix-CUDA program with four different parameter settings (m,s) = (4800,1000000), (4200,500000), (3200,250000), and (2400,125000), which respectively correspond to 0.5, 1.0, 2.0, and 4.0 Hz frequencies of the high-low power pattern; Figure 1 demonstrates two of such cases at frequencies 0.5 and 2.0 Hz. At the very first glance of Figure 1, we see that the matrix-CUDA program successfully creates the high-low power variations on GPU, which was not possible with the FIRESTARTER-CUDA.

In Figure 1(a), we had the biggest matrix size $m = 4800$ and longest sleep duration of 1 sec, and as a consequence we observed a very clean high-low pattern on the GPU power profile. However, it is also to be noticed that the transitions from high-load to low-load and vice versa were not sharp. For this particular parameter setting, during each high-load portion of the profile, we found that (time during the flat high-load) $>$ (time during the transition from low to high load) $\approx$ (time during the transition from high to low load). From Figure 1(a) we notice that the CPU power pattern also showed high-low variations, but the total system power pattern missed the high-low variations a few times.

Then, as we progressively reduced both the matrix-size and sleep-duration parameters to increase the frequency of high-low transitions, we noticed that (time during the flat high-load) $\approx$ (time during the transition from low to high load) $\approx$ (time during the transition from high to low load) during each high-load portion of the profile, as shown in Figure 1(b). Subsequently, at much smaller matrix-size and sleep-duration, (time during the flat high-load) became considerably smaller than (time during the transition from low to high load) or (time during the transition from high to low load). In such cases, it became really difficult to understand the timings corresponding to the onset of high and low loads. Therefore, it is reasonable to infer that we can reliably assess a high-low power pattern via Allinea MAP when the frequency of the high-low load profile is at 2.0 Hz or smaller.

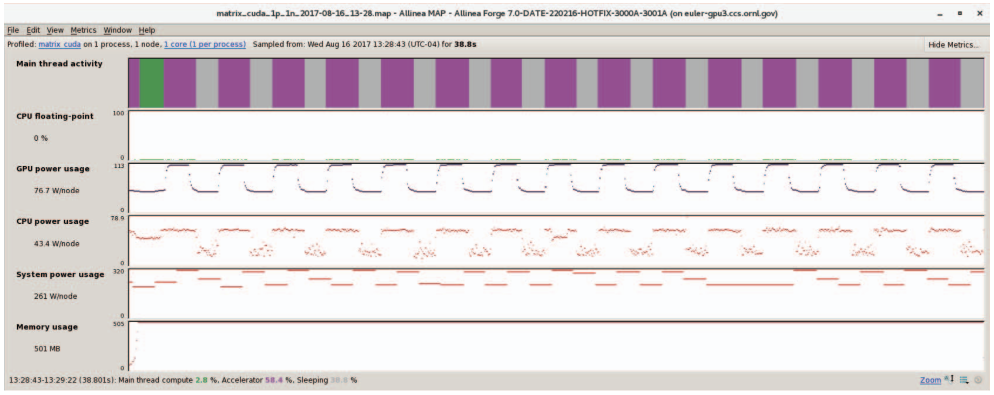
B. NVML via Direct Reads

We separately acquired power measurements from NVML directly, not via Allinea MAP, by executing *nvidia-smi* (Nvidia System Management Interface) program with - *-query-gpu=power.draw* and - *-loop-ms=1* arguments. In Figure 2, we show the collected high-low patterns of the GPU power profile due to four different executions of the matrix-CUDA program with (m,s) = (4200,500000), (3500,333333), (3200,250000), and (3000,200000), which respectively correspond to 1.0, 1.5, 2.0, and 2.5 Hz frequencies of the high-low power pattern.

Overall, the characteristics of power profiles collected by directly reading NVML measurements are quantitatively very similar to what we obtained via Allinea MAP. When the frequency of the high-low power pattern was small, for example, 1.0 Hz, we clearly observed a load pattern having a transition phase from low-load to high-load before multiplication operation, followed by a flat high-load phase during multiplication, and finally another transition phase from high-load to low-load after multiplication. Then, as we progressively increased the frequency of the high-low power pattern, the transition phases from low-load to high-load and vice versa dominated the profiles over any flat high-load or low-load characteristics. Consequently, for frequencies more than 2.0 Hz, it became difficult to reliably assess the actual high-low power pattern with distinct understanding of the onsets of high and low loads. Therefore, we conclude that by directly reading NVML measurements, it is possible to reliably interpret a high-low load profile with frequency 2.0 Hz or smaller.

C. PowerInsight (PI) Device

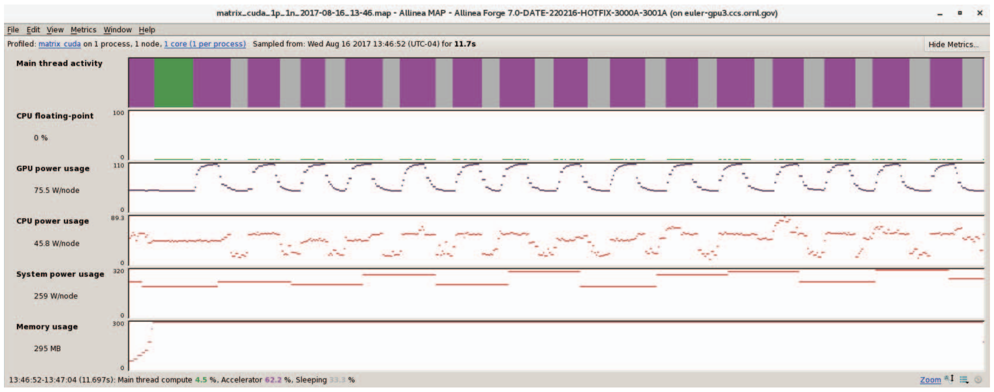
We repeated the above-mentioned executions of matrix-CUDA benchmark with the same specifications of (m,s) and collected the GPU power profiles via PI device. We used two vendor drivers, getRawPower() and powerInsight(), to acquire power readings at the PI device while executing the matrix-CUDA benchmark on the corresponding compute node. The results for getRawPower() driver are shown in Figure 3 when the frequencies of high-low load variations


matrix_cuda_1p_1n_2017-08-16_13-28.map - Allinea MAP - Allinea Forge 7.0-DATE-220216-HOTFIX-3000A-3001A (on euler-gpu3.ccs.ornl.gov)

Profiled: matrix cuda on 1 process, 1 node, 1 core (1 per process) Sampled from: Wed Aug 16 2017 13:28:43 (UTC-04) for **38.8s**

13:28:43-13:29:22 (38.801s): Main thread compute 2.8 %, Accelerator 58.4 %, Sleeping 38.8 %

(a) $m = 4800$, $s = 1000000$


matrix_cuda_1p_1n_2017-08-16_13-46.map - Allinea MAP - Allinea Forge 7.0-DATE-220216-HOTFIX-3000A-3001A (on euler-gpu3.ccs.ornl.gov)

Profiled: matrix cuda on 1 process, 1 node, 1 core (1 per process) Sampled from: Wed Aug 16 2017 13:46:52 (UTC-04) for **11.7s**

13:46:52-13:47:04 (11.697s): Main thread compute 4.5 %, Accelerator 62.2 %, Sleeping 33.3 %

(b) $m = 3200$, $s = 250000$

Figure 1: Power profiling via Allinea MAP.

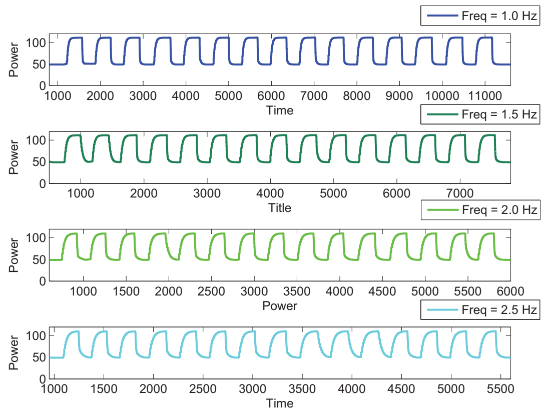


Figure 2: GPU power profiling via NVML direct reads.

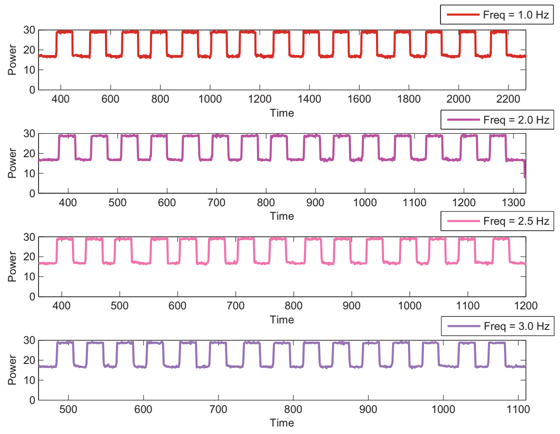
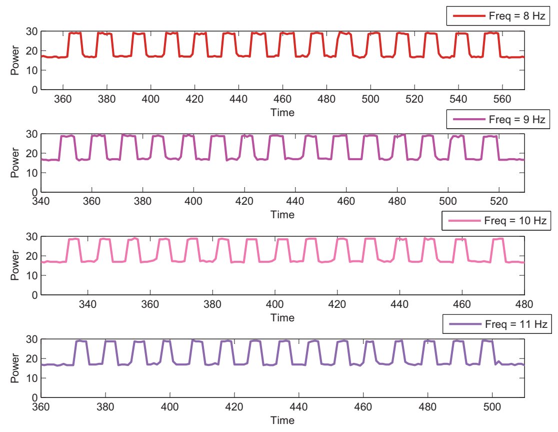


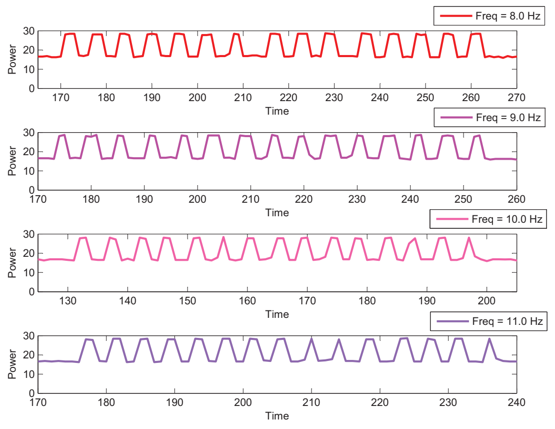
Figure 3: GPU power profiling via PI device at 1-3 Hz.

were small, for example, 1-3 Hz; the powerInsight() driver had very similar results. It is clearly evident from these power profiles that the PI device can reliably produce square-wave like high-low load pattern in this frequency range. We also observed that the profiles collected via PI device have sharper transitions, both from high-load to low-load and vice versa, compared to those measured using NVML direct reads and Allinea MAP.

As PI device could reliably characterize GPU power profiles at small frequencies, we continued to increase the frequency of high-low load pattern by progressively reducing the values of m and s. In Figures 4(a) and 4(b), we respectively depict the power profiles collected via the getRawPower() and powerInsight() drivers at four higher frequencies, 8, 9, 10 and 11 Hz, which were obtained by

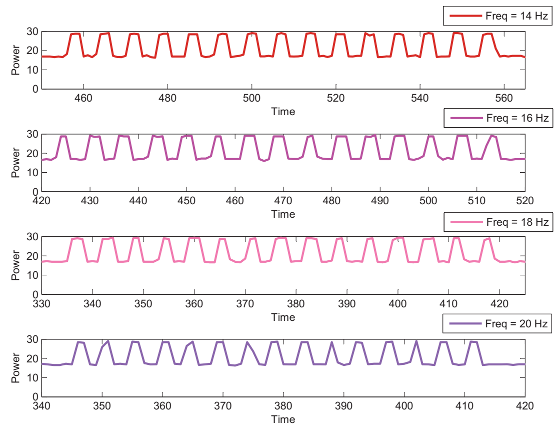


(a) via getRawPower() driver

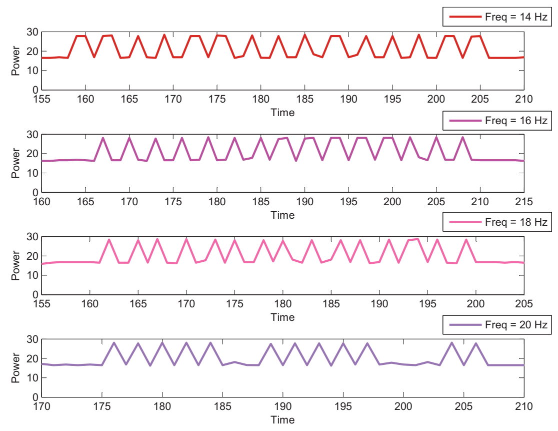


(b) via powerInsight() driver

Figure 4: GPU power profiling via PI device at 8-11 Hz.



(a) via getRawPower() driver



(b) via powerInsight() driver

Figure 5: GPU power profiling via PI device at 14-20 Hz.

setting (m,s) = (1800,62500), (1700,55556), (1600,50000), and (1500,45455). We notice from Figure 4(b) that, as the frequency of the high-low load pattern was increased, particularly at or more than 10 Hz, the power profiles measured by powerInsight() driver started to drift away from an idealistic square-wave behavior to a more trapezoidal/triangular shape, in which the transition phases from low-load to high-load and vice versa took more time than the flat portions with constant high or low load. Therefore, it seems that by using the powerInsight() driver of PI device we can reliably assess a high-low power profile with frequency of approximately 10 Hz on GPUs. On the other hand, the getRawPower() profiles were still able to maintain approximated square-wave patterns with distinctively flat portions indicating the constant high or low load phases at this frequency range.

To understand the limit of getRawPower() driver, we further increased the frequency of high-low load pattern to 14, 16, 18, and 20 Hz, respectively by setting (m,s) = (1400,35714), (1350,31250), (1300,27778), and (1200,25000). In Figures 5(a) and 5(b), we show the corresponding results obtained respectively via the getRawPower() and powerInsight() drivers. From Figure 5(a), we observe that at frequencies about 18 Hz or more the responses of getRawPower() driver started to become more trapezoidal/triangular in shape than the idealistic square-wave pattern. This indicates that via the getRawPower() driver of PI device it is possible to reliably assess a high-low power pattern with frequency up to 18 Hz on GPUs. On the other hand, profiles in Figure 5(b) due to the powerInsight() driver clearly show that it is not possible to assess a high-low load pattern at these higher frequency (higher than 10 Hz) via the powerInsight() driver of PI device.

D. PowerInsight via Allinea MAP

We also analyzed the performance of GPU power profiles using a custom designed profiling scheme, namely *map-pi* tool, that combines the PI device measurements into Allinea MAP. The operating principle of the map-pi approach is depicted in Figure 6. In this approach, the Allinea MAP tool runs on the compute node along with the matrix-CUDA benchmark. We initiate the power measurement process by

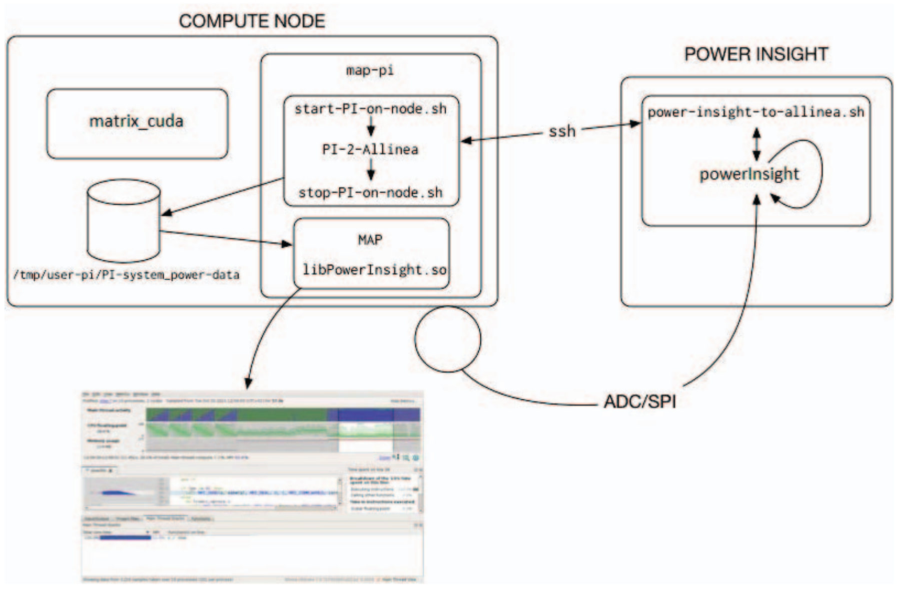


Figure 6: A schematic diagram of the workflow involved in the map-pi tool.

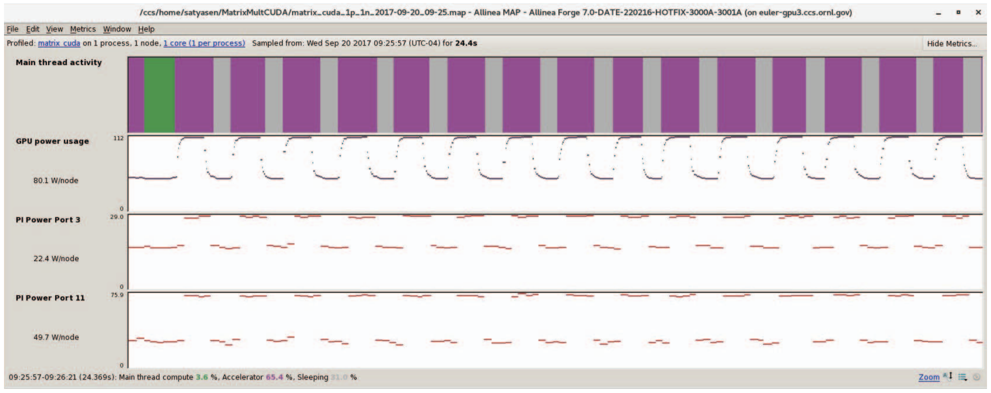
running the map-pi script, which first creates a secure shell (ssh) connection to the PI device (using *start-PI-on-node.sh*), and then repeatedly calls the powerInsight() driver to read the measurements from the power sensors (using *power-insight-to-allinea.sh*). The measured data are formatted and sent back via the ssh connection to the compute node, where they are saved in a temporary file (called *PI-system-power-data*). The Allinea MAP tool uses a custom plugin called *libPowerInsight.so* to read this temporary file as if it was a source of instrumentation data from a running application. Once a particular run is complete, the map-pi script terminates the data collection (using *stop-PI-on-node.sh*). Resulting PI data are finally merged with the obtained NVML measurements via other plugins.

We ran the matrix-CUDA program with four different parameter settings (m,s) = (4200,500000), (3500,333333), (3200,250000), and (2800,166667), respectively corresponding to 1.0, 1.5, 2.0 and 3.0 Hz frequencies of the high-low load pattern, to record the output power patterns collected via map-pi tool. Figure 7 demonstrates two of such cases at 1.0 and 2.0 Hz frequencies. In each of these plots, we show the power profiles collected by the PI device at port number 3 and 11, along with the NVML power profile, which just serves the purpose of a baseline profile as it is not measured by the PI device. Similar to the performances observed via Allinea MAP (see Figure 1) and NVML direct read (see Figure 2), we observed that the high-low power profiles were reliably produced via the map-pi tool when the frequency of the high-low pattern was small, for example, 1.0 Hz. However, with the increasing frequency of the high-low power pattern, we noticed several irregular periods in the profiles measured by the PI device. In these irregular periods, the extents of high and low load phases (i.e., duty cycle) were not similar to what observed in other periods. This irregular behavior incorrectly implies that the frequency and duty-cycle of the high-low square-wave pattern are not constant, which is not true. Therefore, we conclude that, to reliably assess a high-low power profile via the map-pi tool, it is advisable to keep the frequency of the high-low load profile at 2.0 Hz or smaller.

V. Discussions and Further Analyses

As discussed in the previous section, different profiling mechanisms offer different degrees of reliability in producing the square-wave pattern generated by the matrix-CUDA program. In general, the mechanisms involving NVML measurements, either by direct reads or using Allinea MAP, and via map-pi tool reliably assess a high-low power profile when frequency is approximately 2.0 Hz or smaller. On the other hand, collecting power measurements using the PI device, we can interpret high-low power pattern with higher frequency, for example, up to 18 Hz using getRawPower().

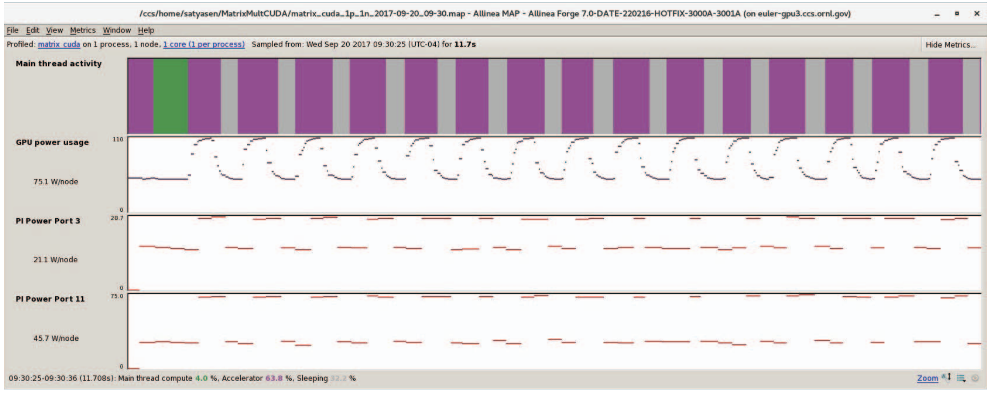
Now, in addition to understanding the maximum frequency of the high-low power profiles that can be reliably assessed by these four profiling schemes, there is another noticeable aspect in these power profiles: non-sharp or slow variations at the leading and trailing edges of the square-wave patterns when measured via NVML, compared to very sharp leading and trailing edges measured by PI device. We illustrate this effect further in Figures 8(a) and 8(b) respectively for NVML read and PI device measurements. In each figure, the top panel shows the variations around the leading edge (i.e., when power is changing from low to high load) for three different periods, and the bottom panel shows the same for the trailing edge (i.e., when power is changing


/ccs/home/satyasen/MatrixMultCUDA/matrix_cuda_1p_1n_2017-09-20_09-25.map - Allinea MAP - Allinea Forge 7.0-DATE-220216-HOTFIX-3000A-3001A (on euler-gpu3.ccs.ornl.gov)

Profiled: matrix cuda on 1 process, 1 node, 1 core (1 per process) Sampled from: Wed Sep 20 2017 09:25:57 (UTC-04) for **24.4s**

09:25:57-09:26:21 (24.369s): Main thread compute 3.6 %, Accelerator 65.4 %, Sleeping 31.0 %

(a) $m = 4200$, $s = 500000$


/ccs/home/satyasen/MatrixMultCUDA/matrix_cuda_1p_1n_2017-09-20_09-30.map - Allinea MAP - Allinea Forge 7.0-DATE-220216-HOTFIX-3000A-3001A (on euler-gpu3.ccs.ornl.gov)

Profiled: matrix cuda on 1 process, 1 node, 1 core (1 per process) Sampled from: Wed Sep 20 2017 09:30:25 (UTC-04) for **11.7s**

09:30:25-09:30:36 (11.708s): Main thread compute 4.0 %, Accelerator 63.8 %, Sleeping 32.2 %

(b) $m = 3200$, $s = 250000$

Figure 7: Power profiling via map-pi tool.

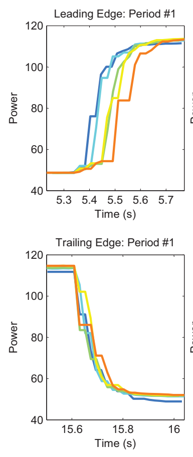
from high to low load). We use five different matrix-sizes (m) to generate these results.

The non-sharp behavior at the leading edges for NVML measurements is clearly evident from the top panel of Figure 8(a). Additionally, as expected, a bigger matrix-size m results into a slower leading edge profile. On contrary, from the top panels of Figure 8(b), we observe that the power profiles measured via PI device rise from the low-load value to a slightly higher value (approximately 2 Watts higher than the low-load level), before sharply transitioning to the high-load value. The magnitude of this intermediate power value does not depend on the matrix-size m; however, the extent of this intermediate pedestal increases with m. Plots of the trailing edges (bottom panels in Figures 8(a) and 8(b)) suggest that the matrix-size m does not have any noticeable effect when high-load changes to low-load; however, we again observe the non-sharp profiles in the NVML data compared to the PI device measurements.

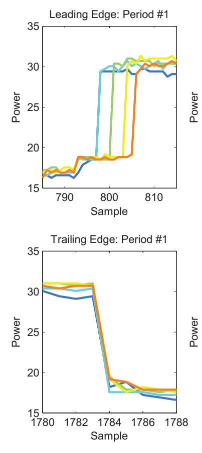
To characterize these leading and trailing edge behaviors, we present below a discussion on the effects of moving average filter on the measured power profiles. Later, we discuss the power profiling performances with respect to two variants of the matrix-CUDA benchmark.

A. Analyses in terms of Moving-Average Filter

We simulate a few numerical examples considering an ideal square-wave pattern with 0 and 1 respectively representing low and high load values. We feed such square-wave to a moving-average (MA) filter of length L, which produces output as $y_L(t) = \left(\boldsymbol{w}_L^T\, \boldsymbol{x}(t)\right) / \left(\boldsymbol{w}_L^T\, \mathbf{1}\right)$, where $\boldsymbol{x}(t) = [x(t-L+1), \ldots, x(t-1), x(t)]^T$ represents a vector of length L containing the last L values (including the current time instant t) of the ideal square wave; $\boldsymbol{w}$ is another L-dimensional vector representing the MA filter coefficients; and $\mathbf{1}$ is a vector of L ones. In general, the MA filter coefficient-vector takes the following form $\boldsymbol{w}_L = [\alpha^{L-1}, \ldots, \alpha, 1]^T$, where $\alpha \in [0,1]$ is a scalar parameter describing the exponential weights of MA filter. The MA filter with $\alpha = 1$ corresponds to the standard MA filter, whereas for any other values of α we have the weighted-MA filter. In Figure 9, we depict the leading and trailing edge performances of the MA-filter outputs along with the ideal square wave. We use three different values of $L = 5$, 10, 15, and two values of $\alpha = 1$, 0.85. The curvy profiles of the weighted-MA filter (i.e., $\alpha < 1$) outputs, during both the leading and trailing edges, imply that the NVML mechanism must be using a weighted-MA type filter



(a) via NVML read



(b) via getRawPower() driver of PI device

Figure 8: Variations of leading and trailing edges in the GPU power profiles measured via NVML read and PI device for matrix-CUDA program.

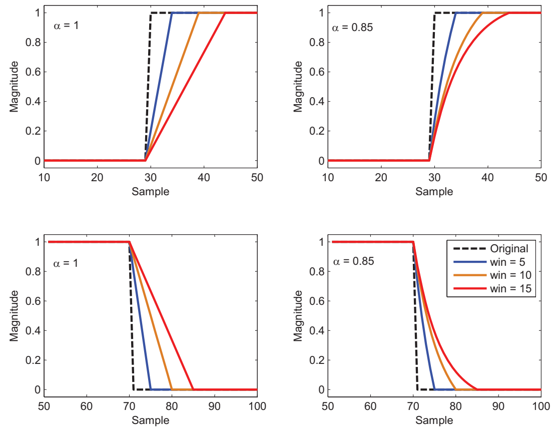


Figure 9: Representations of moving-average method at the leading and trailing edges of a square-wave pattern.

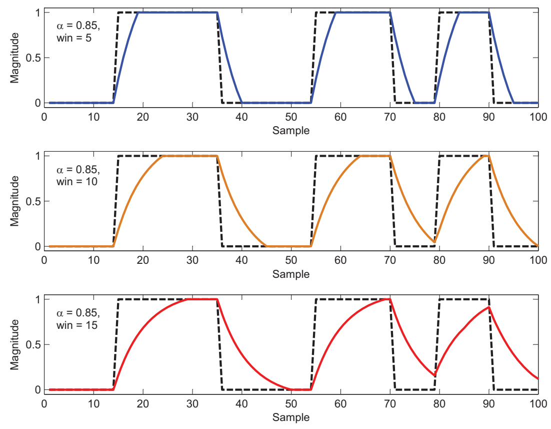


Figure 10: Effect of moving-average method on the output of a square-wave pattern.

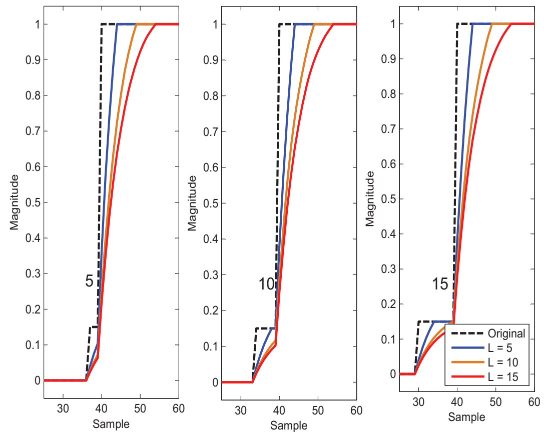


Figure 11: Effect of moving-average method at the leading edges of a square-wave pattern having an intermediate pedestal value.

before reporting its measurements.

A straightforward consequence of the weighted-MA filter is that when the duty-cycle or period of the high-low power pattern becomes small, comparable to the length L of MA filter, the output looks distorted with no clear indications of the leading and trailing edges of square-wave patterns. We illustrate this effect in Figure 10 with decreasing values of period and duty cycle (precisely, the extent of high-load phase). It is evident that the outputs of weighted-MA filter progressively become triangular as the MA-filter length is increased, ultimately even not being able to reproduce the actual high-load and low-load values. Now, decreasing the duty cycle is equivalent to decreasing the matrix-size m in our matrix-CUDA program. Therefore, as m is decreased, implying as the frequency of the high-low power pattern is increased, the profiles measured via NVML (shown in Figures 1, 2, and 7) resemble high similarity with the weighted-MA filter output in the bottom panel of Figure 10.

Finally, to characterize the effect of an intermediate pedestal value at the leading edge (as observed in Figure 8(b)), we simulate a three-valued square-wave pattern, in which the low and high loads are respectively represented by 0 and 1 as earlier, and an intermediate load value is considered as 0.15. The duration of this intermediate load value is varied for 5, 10, and 15 samples (as demonstrated in Figure 11), representing the effect of increasingly bigger matrix-size m. Figure 11 shows the corresponding outputs of weighted-MA filter ($\alpha = 0.85$) with three different length $L = 5$, 10, and 15. We observe that when the duration of the intermediate load value is small, for example, 5, the output of the MA filter seems to always have an increasing profile (leftmost plot in Figure 11); in this setting, it is difficult to distinguish any change in profiles during and after the intermediate pedestal value. However, for a longer duration of intermediate load value, for example, 15, we notice that the MA-filter output first increases to the intermediate value, stays there for a while, before increasing again to the high-load value (rightmost plot in Figure 11). A careful observation of the NVML profiles in Figure 8(a) reveals that for larger matrix-sizes, for example, $m = 8750$, 10000, the NVML power profiles settle at an intermediate level for some tiny duration before continuing their increasing trend towards the high-load value. Certainly, the effect of a weighted-MA filter in NVML measurements makes it harder to recognize the presence of this intermediate load value compared to the responses obtained via PI device.

B. Analyses with respect to matrix-CUDA Variants

The analyses and performance results presented thus far are based on the matrix-CUDA program shown in Algorithm 1. We modified this benchmark code to create two variants of it, which are presented in Algorithms 2 and 3 in terms of pseudo codes. When we used matrix-CUDA-variant-1 to profile the GPU power, we did not notice any high-low load variation in the GPU power profile. The CPU power profile also did not reproduce any expected high-low profile. Therefore, this variant of the matrix-CUDA program is certainly not suitable for the power profiling purpose.

Algorithm 2 matrix-CUDA-variant-1 program

```
1: Input: matrix-size $m$, repetition no. $r$, sleep duration $s$
2: Initialize: Matrix $A(m,m)$, $B(m,m)$
3: cudaMalloc():    to allocate device memory space
4: cudaMemcpy():    to copy from host to device memory
5: for ($ii = 1$; $ii <=$ itr; $ii++$) do
6:     gpu-matrix-multiplication($A$,$B$,$C$,$m$)    for high load
7:     waitFor(usleep(),s)    for low-load
8: cudaMemcpy():    to transfer results from device to host
9: cudaFree():    to free up device memory
```

Algorithm 3 matrix-CUDA-variant-2 program

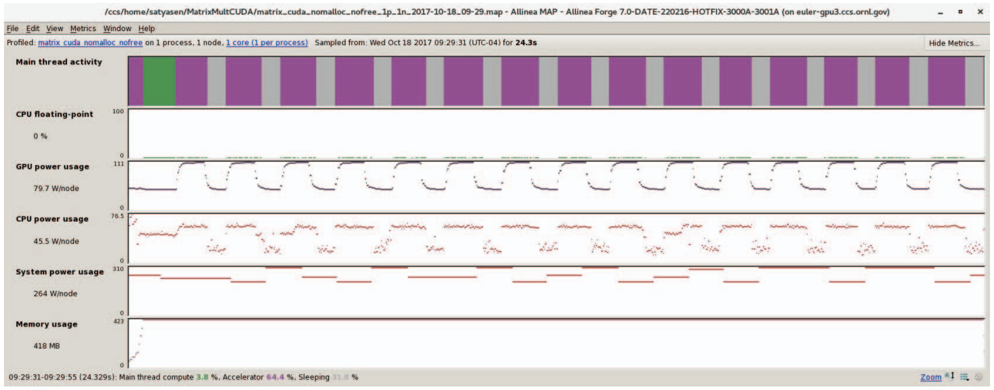
```
1: Input: matrix-size $m$, repetition no. $r$, sleep duration $s$
2: Initialize: Matrix $A(m,m)$, $B(m,m)$
3: cudaMalloc():    to allocate device memory space
4: for ($ii = 1$; $ii <=$ itr; $ii++$) do
5:     cudaMemcpy():    to copy from host to device memory
6:     gpu-matrix-multiplication($A$,$B$,$C$,$m$)    for high load
7:     cudaMemcpy():    to copy from device to host memory
8:     waitFor(usleep(),s)    for low-load
9: cudaFree():    to free up device memory
```

In Figure 12, we depict the GPU power profiling using NVML via Allinea MAP while executing the matrix-CUDA-variant-2 program (pseudo-code in Algorithm 3), with parameter (m,s) = (4200,500000) for 1 Hz frequency of the high-low load pattern. We noticed that this variant of the matrix-CUDA benchmark reproduces the expected high-low power variations in GPU similar to the original matrix-CUDA program. Therefore, we collected two more power profiles at parameter settings: (m,s) = (3200,250000) and (2800,166667), respectively for 2 and 3 Hz frequencies of the high-low load pattern; however, we did not observe any noticeable difference when we compared the overall performances of NVML via Allinea MAP scheme for the matrix-CUDA and matrix-CUDA-variant-2 programs. The GPU power profiling performance of the NVML via Allinea MAP tool using matrix-CUDA-variant-2 program also seems to be limited to a high-low load frequency of 2 Hz or smaller.

VI. Conclusions

In this paper, we presented a detailed assessment of four GPU-power profiling mechanisms, NVML via Allinea MAP, NVML via direct read, PowerInsight device, and PowerInsight via Allinea MAP (namely, map-pi tool), while utilizing a custom-designed GPU stress-test benchmark matrix-CUDA. Our approach was to generate a high-low, square-wave like power profile using the matrix-CUDA benchmark, and to assess the profiling mechanisms in terms of their reproduced power patterns. In comparison to other schemes, the PI device based GPU power profiling mechanism seemed to be the best because it could reliably generate the expected power patterns at higher frequencies of high-low load variations. Additionally, the PI device reported the instantaneously measured power values, and therefore the measured profiles showed square-wave patterns having very sharp transitions from low to high load and vice versa.

It would be of future interest to assess the performance of the profiling mechanisms with respect to a benchmark that produces more than two levels of power values on the GPUs. More generally, it would lead to a pattern-matching type of assessment technique for the power measurement schemes. In addition, future work may include the development of a statistics-based assessment methodology to provide more quantitative performance characterizations, for example, in terms of the false positive and false negative measures.



/ccs/home/satyasen/MatrixMultCUDA/matrix_cuda_nomalloc_nofree_1p_1n_2017-10-18_09-29.map - Allinea MAP - Allinea Forge 7.0-DATE-220216-HOTFIX-3000A-3001A (on euler-gpu3.ccs.ornl.gov)

Profiled: matrix cuda nomalloc nofree on 1 process, 1 node, 1 core (1 per process) Sampled from: Wed Oct 18 2017 09:29:31 (UTC-04) for **24.3s**

09:29:31-09:29:55 (24.329s): Main thread compute 3.8 %, Accelerator 64.4 %, Sleeping 31.8 %

Figure 12: GPU power profiling via Allinea MAP for matrix-CUDA-variant-2 program.

ACKNOWLEDGMENT

This work is supported by the United States Department of Defense, using resources of the Computational Research and Development Programs, at the Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U.S. Department of Energy, under contract DE-AC05-00OR22725. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

REFERENCES

[1] D. Göddeke, D. Komatitsch, M. Geveler, D. Ribbrock, N. Rajovic, N. Puzovic, and A. Ramirez, "Energy efficiency vs. performance of the numerical solution of PDEs: An application study on a low-power ARM-based cluster," *Journal of Computational Physics*, vol. 237, pp. 132 – 150, Mar. 2013.

[2] Oak Ridge Leadership Computing Facility. Summit. [Online]. Available: https://www.olcf.ornl.gov/summit/

[3] R. A. Bridges, N. Imam, and T. M. Mintz, "Understanding GPU power: A survey of profiling, modeling, and simulation methods," *ACM Comput. Surv.*, vol. 49, no. 3, pp. 41:1–41:27, Sep. 2016.

[4] PAPI. Performance Application Programming Interface. [Online]. Available: http://icl.cs.utk.edu/papi/index.html

[5] P. Mucci, S. Browne, C. Deane, and G. Ho, "PAPI: A portable interface to hardware performance counters," in *Proc. Department of Defense HPCMP Users Group Conf.*, Monterey, CA, Jun. 7–10, 1999.

[6] NVIDIA. (2011, Oct.) CUDA Tools SDK CUPTI User's Guide. [Online]. Available: https://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUPTI_Users_Guide.pdf

[7] NVIDIA Developer. Performance Analysis Tools. [Online]. Available: https://developer.nvidia.com/performance-analysis-tools

[8] ——. NVIDIA Management Library (NVML). [Online]. Available: https://developer.nvidia.com/nvidia-management-library-nvml

[9] M. Burtscher, I. Zecena, and Z. Zong, "Measuring GPU power with the K20 built-in sensor," in *Proc. of Workshop on General Purpose Processing Using GPUs*, Salt Lake City, UT, Mar. 1, 2014, pp. 28:28–28:36.

[10] ARM Information Center. (2009-2010) Mali GPU Shader Development Studio User Guide. [Online]. Available: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0504b/index.html

[11] AMD. (2015) GPU Performance API User Guide. [Online]. Available: http://developer.amd.com/wordpress/media/2013/12/GPUPerfAPI-UserGuide-2-15.pdf

[12] Allinea. Allinea MAP. [Online]. Available: https://www.allinea.com/products/map

[13] C.-H. Hsu and S. W. Poole, "Power measurement for high performance computing: State of the art," in *Intl Green Computing Conf. and Workshops (IGCC)*, Orlando, FL, Jul. 25–28, 2011, pp. 1–6.

[14] X. Feng, R. Ge, and K. W. Cameron, "Power and energy profiling of scientific applications on distributed systems," in *IEEE Intl Parallel and Distributed Processing Symp.*, Denver, CO, Apr. 4–8, 2005, pp. 34–34.

[15] R. Ge, X. Feng, S. Song, H. C. Chang, D. Li, and K. W. Cameron, "PowerPack: Energy profiling and analysis of high-performance systems and applications," *IEEE Trans. Parallel and Distributed Systems*, vol. 21, no. 5, pp. 658–671, May 2010.

[16] D. Bedard, M. Y. Lim, R. Fowler, and A. Porterfield, "PowerMon: Fine-grained and integrated power monitoring for commodity computer systems," in *Proc. IEEE SoutheastCon*, Concord, NC, Mar. 18–21, 2010, pp. 479–484.

[17] J. H. Laros, P. Pokorny, and D. DeBonis, "PowerInsight - A commodity power measurement capability," in *Intl Green Computing Conf. Proc.*, Arlington, VA, Jun. 27–29, 2013.

[18] D. DeBonis, J. H. Laros, and K. Pedretti, "Qualification for PowerInsight accuracy of power measurements," Sandia National Laboratories, Tech. Rep. SAND2013-9639, Nov. 2013.

[19] Penguin Computing. (2012, Nov.) Penguin Computing Releases New Power Monitoring Device. [Online]. Available: https://www.penguincomputing.com/company/press-releases/penguin-computing-releases-new-power-monitoring-device/

[20] TU Dresden. FIRESTARTER: A Processor Stress Test Utility. [Online]. Available: https://tu-dresden.de/zih/forschung/projekte/firestarter/

[21] D. Hackenberg, R. Oldenburg, D. Molka, and R. Schne, "Introducing FIRESTARTER: A processor stress test utility," in *Intl Green Computing Conf. Proc.*, Arlington, VA, Jun. 27–29, 2013, pp. 1–9.