Fingerprinting Anomalous Computation with RNN for GPU-Accelerated HPC Machines

Pengfei Zou, Rong Ge
{pzou,rge}@clemson.edu
Clemson University

Ang Li, Kevin Barker
{ang.li,kevin.barker}@pnnl.gov
Pacific Northwest National Laboratory

Keywords

HPC security, GPU accelerated systems, workload classification.

1 Introduction

The integration of GPGPUs complicates the security issues in HPC systems [4]. By exploiting GPU-accelerator based HPC systems, attackers get the needed high hash rate and avoid paying for the computing resource and energy bills. Attackers have exploited HPC systems for bitcoin mining [1], brute force password cracking [2] and GPU-inflated denial-of-service (DoS) attack [7]. Such HPC security incidents not only deprive mission-critical and scientific applications of execution cycles, but also increase the chance for attackers to steal data, damage systems, and leverage the high computation and network bandwidth for attacking other sites.

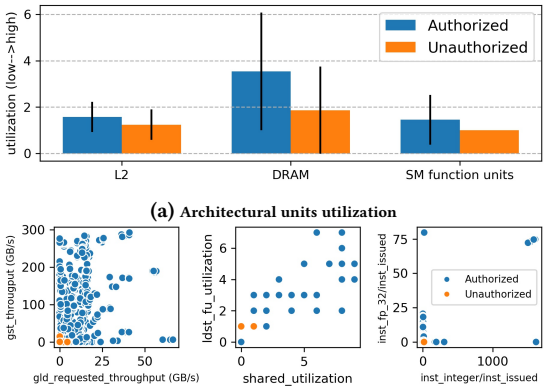
GPU accelerated HPC systems require different security measures from traditional IT and homogeneous HPC systems. Existing measures for homogeneous HPC systems detect illicit computation using CPU execution patterns [3]. Their applications to GPU accelerated systems are problematic because illicit computations are offloaded to GPUs without much CPU involvement [5, 6]; a GPU-side monitoring and detection approach is thus highly desired.

Fortunately, regarding HPC workloads, we can leverage their unique features and patterns to effectively mitigate risks for (open) systems and compute nodes. HPC systems often present a small set of programs with specific resource usage patterns that are more predictable [4]. Such workloads have a high chance to invoke certain functions such as linear algebra operations and fast Fourier transform. They have distinctive microarchitectural activities and system behaviors that can be monitored, collected, and identified.

In this paper, we present a machine learning framework for classifying GPGPU workloads based on their microarchitectural and system behaviors. We achieve over 95% accuracy to detect illicit workloads with the following contributions: **1.** We demonstrate the feasibility and accuracy of using workload behavioral data to fingerprint illicit computations and anomalous workloads in GPU-accelerated HPC systems. **2.** We investigate multiple online and offline machine learning methods for anomalous workload detection. **3.** We evaluate our workload analysis and classification framework with 83 applications and kernels over 3 generations of GPU architecture.

2 GPU Workload Profiling

To correctly classify the running applications, we collect workload profiles and behavioral data. As the characteristics of GPU-accelerated HPC workload have rarely been studied, features that can best identify the workloads are unknown. To address this issue, we collect as many features as possible with available profilers. Specifically, we profile workload execution at both system and microarchitectural levels, and collect multiple types of workload behavioral data from multiple sources.



(a) Architectural units utilization

(b) Architectural component throughput

Figure 1: Different patterns for measured performance metrics between authorized and unauthorized workloads.

2.1 Microarchitectural Events Behavior

Microarchitectural events are activities in hardware including processing units, memory, and caches, and can be monitored with Performance Monitor Counters (PMCs). Prior work shows that PMC measured events are strong indicators of workload characterization for CPU workloads [3]. Our initial study shows that PMC are also good indicators to classify GPU workloads. As shown in Figure 1, for unauthorized programs such as cryptocurrency and password cracking, hashing functions on integer units are intensively executed. Moreover, they tend to show a much smaller utilization of the devices in the memory hierarchy including L2 cache, load/store functional units, and DRAM. In addition, they demonstrate much smaller global load/store throughput compared to authorized workloads.

2.2 Data Movement Behavior

In addition to architectural events monitoring, we also collect time series data. Such data complement the cumulative event counts collected from PMCs, and are particularly suitable for online detection. We collect data movement between host and device over time. Unauthorized workloads periodically transfer data from host to device, and barely transfer data from device to host. In contrast, authorized programs are optimized, offloading larger amount of data to device, and transfer data back from device. Such differences in kernel execution time, data transfer direction and volume can be leveraged to identify illicit computations and anomalous programs.

2.3 Resources Utilization Behavior

Figure 2 shows an example of resource usage traces for linear algebra, data processing, and password cracking applications. Matrix-multiply consumes significantly higher and stable power compared to the other two. While radix-sort and password-crack show varying patterns with similar frequency for both power and memory

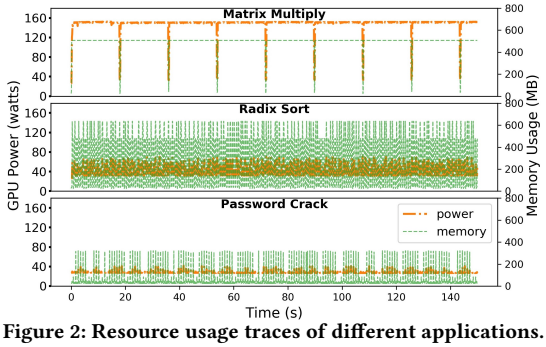


Figure 2: Resource usage traces of different applications.

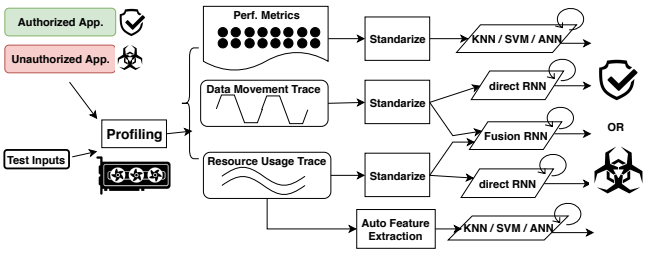


Figure 3: Workload classification framework using ML models.

usage, the magnitude for radix-sort are notably larger. These traces are good features for discriminating unauthorized applications.

3 ML-Based Illicit Workload Detection

The presented profiles suggest that unauthorized workloads are different from authorized ones and can be discriminated. Here we explore a data-driven approach that leverages advanced feature extraction and machine learning methods to automatically detect illicit workloads. We build a framework illustrated in Figure 3. It comprises data acquisition, data processing, and workload classification with multiple machine learning (ML) models. We test different ML models as well as their fusion including SVM, KNN, ANN (i.e., MLP), and (fused) RNN. On each architecture, the framework retrains all the models with input samples on the same architecture.

As discussed, we collect three types of feature per workload: performance metrics measured by PMCs, data movement trace, and resource utilization trace. We use NVIDIA profiler (i.e., `nvprof`) to collect target PMCs. When an application contains multiple kernels, each kernel is treated as an independent data entry. In other words, we classify execution based on the granularity of kernel rather than application. We also use `nvprof` to collect data movement trace, including the starting time, duration, throughput, source and destination for each data movement transaction. We rely on `nvidia-smi` to track four types of running signals including power consumption, GPU core utilization, memory footprint, and device memory bandwidth.

4 Evaluation

To evaluate the classification accuracy, we split the data samples into three groups: training (50%), validation (25%) and test (25%). With the selected 16 PMCs, ANN achieves the highest accuracy (e.g., 98% on P100), while 76% and 88% for KNN and SVM, respectively. Overall, memory access patterns show the most significant difference among workloads. The execution-related factors such as SM occupancy, instruction-replay, etc. are also good indicators for application discrimination.

Table 1: Classification Accuracy with time-series traces

Data source	ML model	Accuracy		
		K40	P100	V100
Data movement trace	RNN	90%	93%	92%
Resource utilization trace	ANN*	89%	88%	89%
	RNN	89%	90%	88%
Data movement & resource utilization trace	RNN	93%	97%	93%

*: Input features extracted by *tsfresh* from trace for the ANN

Table 1 shows the accuracy of multiple ML methods using time-series traces as inputs. The models trained with data-movement is slightly more accurate than the other three over all of the three GPU platforms, implying that data movement implicitly encodes more application-specific features.

Furthermore, when models are trained using multiple time-series signals, superior classification accuracy can be achieved. This implies that different signals (e.g., data-movement and resources utilization) can encode different application-specific patterns thus can complement with each other.

5 Conclusion

In this work we present a machine learning based automatic illicit workload detection framework for GPU accelerated HPC systems such as Summit. It uses multi-physics data sources for model training, feature extraction, and online/offline detection. Various ML-based classification techniques are applied and evaluated. Their deployment depend on targeted classification accuracy, cost of data collection, and whether being online or offline. In addition, we have several observations: First, detecting illicit workloads based on performance profiles is feasible. Second, both accumulative and time-series profiling data can be discriminative workload signatures. Third, various machine learning methods can offer decent classification accuracy. Comparatively, MLP based ANN demonstrates the best accuracy over accumulative profiles, while RNNs show the optimal accuracy over time-series profiles. As the future work, we will evaluate the proposed detection framework in real GPU-accelerated HPC systems.

Acknowledgement

This work is supported in part by the U.S. National Science Foundation under Grants CCF-1551511, CNS-1551262 and the U.S. DOE Office of Science, Office of Advanced Scientific Computing Research, under award 66150: "CENATE - Center for Advanced Architecture Evaluation"

References

[1] Esea relese malware into public client, forcing users to farm bitcoins. https://goo.gl/sLDMw8, 2013.
[2] Hashcat advanced password recovery. https://hashcat.net/hashcat/, 2018.
[3] M. Abbas, S. Kadiyala, A. Prakash, T. Srikanthan, and Y. Aung. Hardware performance counters based runtime anomaly detection using svm. In *2017 TRON Symposium*, pages 1–9. IEEE, 2017.
[4] Sean Peisert. Security in high-performance computing environments. *Commun. ACM*, 60(9):72–80, August 2017.
[5] G. Vasiliadis, E. Athanasopoulos, M. Polychronakis, and S. Ioannidis. Pixelvault: Using gpus for securing cryptographic operations. In *Proceedings of CCS*, pages 1131–1142. ACM, 2014.
[6] G. Vasiliadis, M. Polychronakis, and S. Ioannidis. Gpu-assisted malware. *International Journal of Information Security*, 14(3):289–297, 2015.
[7] Y. Wu, Z. Zhao, F. Bao, and R. Deng. Software puzzle: A countermeasure to resource-inflated denial-of-service attacks. *IEEE Transactions on Information Forensics and security*, 10(1):168–177, 2014.