Exploring GPU vulnerabilities in UMass Boston's Systems

Desmond Le

UMass Boston

CS410

Professor Deblois

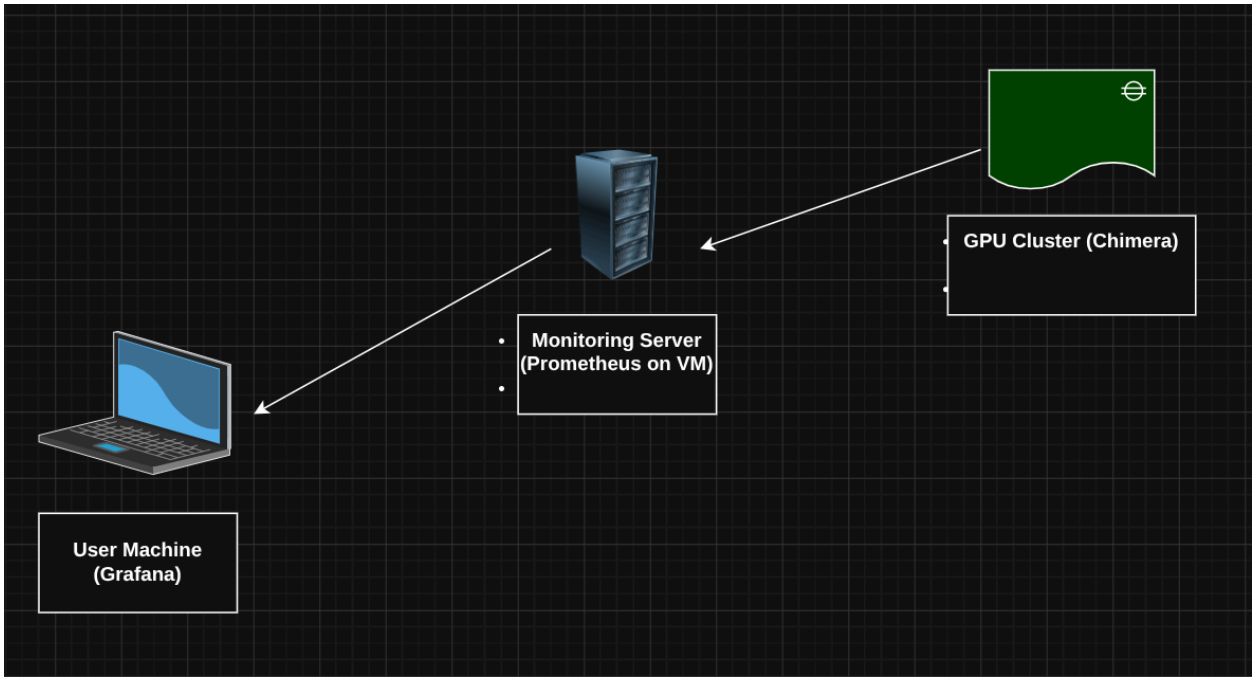
2026-04-23

Abstract

This paper explored the benefits and drawbacks of Team 6's implementation of revealing data metrics on Chimera, UMass Boston's GPU cluster. It was found that exposing such metrics could leave the system vulnerable to side-channel attacks. However, it could also serve as a monitoring system that could possibly increase the system's security. We find that there exist countermeasures such as self-correcting code, and masking techniques, that could reduce the drawbacks of each security risk. Finally, we evaluate these solutions to see how practical they are in UMass Boston's infrastructure.

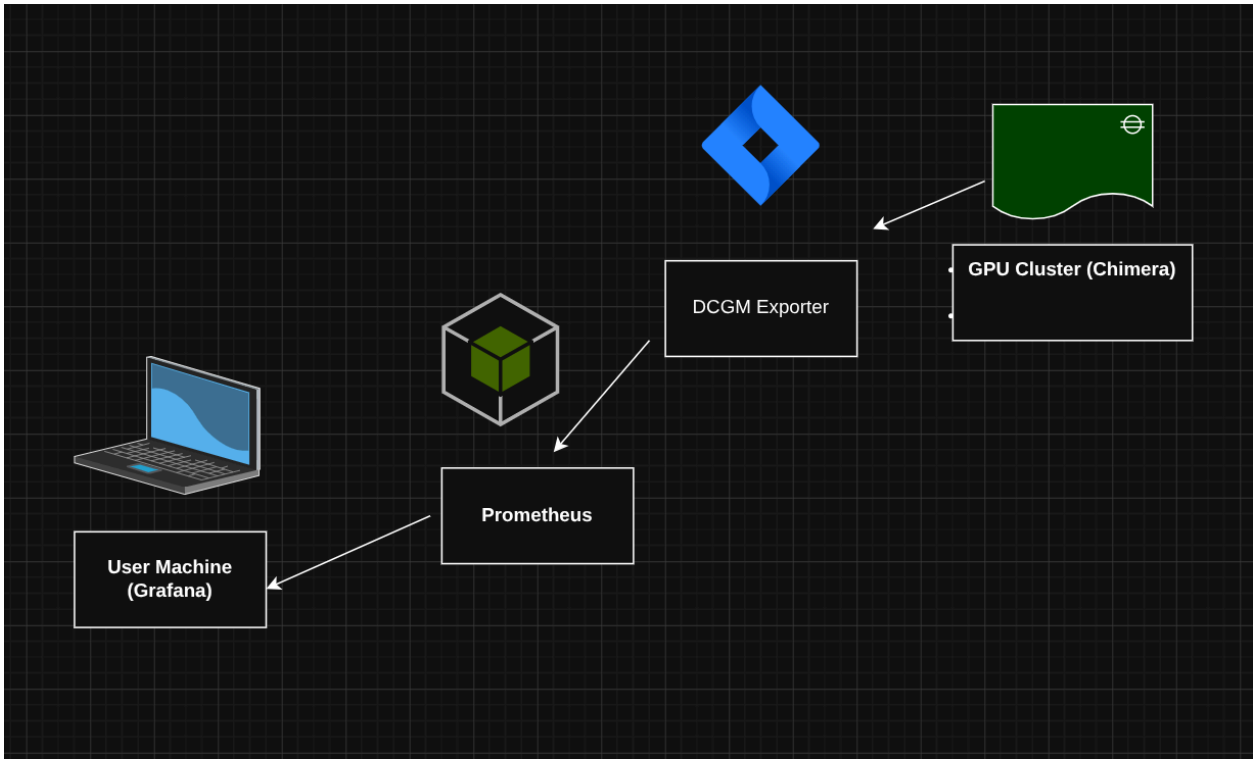
Chapter 1 - Introduction

For our semester-long group project, our team was tasked with extracting performance metrics from UMass Boston's GPU cluster, then visualizing them.



Hardware Diagram

To better illustrate, from a hardware perspective, there are three machines to consider. First is Chimera, the GPU cluster that we want to extract metrics from. Second is the monitoring server, specifically named B6. It's a virtual machine on the CS servers whose job is to scrape and store data using Prometheus. Finally, there is the local machine that takes data from the server and displays it in a user-friendly way using Grafana.



Software Diagram

From a software perspective, we have the DCGM Exporter, a program that reveals all the performance metrics of the Chimera cluster. Prometheus is the software that interacts with the DCGM exporter, scrapes the data, and then stores it on the virtual machine. Finally, Grafana is used on a local machine to take metrics from Prometheus and display them in a neat, user-friendly way.

Having access to Chimera's metrics could leave it susceptible to security vulnerabilities, namely, side-channel and Rowhammer attacks. Both of these attacks will be discussed in detail in the following chapters. This paper aims to weigh the benefits and drawbacks of the Prometheus and Grafana stack regarding the CS server's security. I suspect that the act of exposing such data metrics will introduce vulnerabilities in UMass Boston's systems. However, I also think that our tech stack will be able to detect or at least imply that an attack is taking place.

Chapter 2 - Side-Channel Attacks

On the Grafana interface, I was able to find out the metrics that Prometheus scraped, which were a total of 4666 unique metrics[1]. Among all the metrics shown, there was power usage. Lungu et al. (2024) found that just analyzing our power usage can leak information about what a GPU is computing (p. 4). Their finding connects to my research because we can use our tech stack to see the power usage of the GPU, therefore offering attackers enough information to deduce what a GPU is computing. It's common knowledge that GPUs can be used in encryption algorithms. Suppose that a user was encrypting a key while someone was mointoring the power usage metrics of the GPU. This information is theoretically enough to deduce what the key is by means of statistical analysis.

Furthermore, on the Grafana interface, we are able to see Chimera's memory usage and multiprocessor clock speed. Lungu et al. (2024) go on to say that "timing analysis" can be used to infer what kind of information is being processed by the GPU. Timing analysis is basically seeing how long something takes to execute on a GPU. We can use the memory usage and clockspeed metrics to carry out timing analysis. Essentially, when we see spikes and dips in these

[1] Data metrics can be quiried under Grafana's Explore tab. This action is done under the assumption that Prometheus is already scraping data from Chimera's DCGM, and Grafana is already connected to Prometheus.

two metrics, we can infer how long a process takes to execute on a GPU, which in turn allows us to perform timing analysis.

Given the two examples above, we can see how the information exposed by our tech stack could be used to perform an attack on the Chimera cluster. If this information were to get into the wrong hands, what would happen? My findings raise the question: who will have access to these metrics at the end of our implementation? What safeguards can be put in place to prevent this sensitive information from falling into the wrong hands? The information being exposed by our tech stack could lead to vulnerabilities in the UMass Boston system. We need to be extra careful about whom we trust with the final graphical interface because this information could be used to harm in the wrong hands.

Chapter 3 - Rowhammer Attacks

Essentially, Rowhammer attacks exploit electro-physics. The attack works by repeatedly (and quickly) accessing a row of capacitors in memory. When a row is accessed in this way, the practice is called hammering. The result of hammering rows is that electrical charges seep out into neighboring capacitors, flipping bits. When bits are flipped, memory corruption can occur, leading to privilege escalation. According to the research by Plin et al. (2025), GPUs are particularly vulnerable to rowhammer attacks due to their parallel processing capabilities (p. 1). This finding relates to our group's research because the chimera cluster consists of GPUs.

While our team could monitor the GPU's metrics, we did not have access to use the GPU. This detail meant that, given our tech stack, we could not execute the Row Hammer attack. However, based on the metrics we scraped, I identified two that could relate to the Row Hammer attack. Namely, those metrics were linked to the amount of memory consumption and the clock speed of memory access. Given the nature of the Row Hammer attack, there should be an

immediate spike in these two metrics when it occurs. However, it is worth acknowledging that a spike in memory consumption and clock speed could be due to different GPU uses, such as rendering high-definition videos. Even so, a spike in the aforementioned method could suggest a Row Hammer attack. From this theory, arises the question: Could we can set up Grafana alerts so that when these metrics spike, we can alert a system administrator to investigate the incident and determine whether the server is under attack? From our findings in chapter two and our current chapter, we can see how our tech stack can both reduce and enhance Chimera's security.

Chapter 4 - How Side-Channel and Row Hammer attacks tie into the Grand Scheme

To recap, we found that the metrics we obtained from our tech stack could be used to both attack and defend our CS server system simultaneously. To protect computer systems against power analysis attacks, Lungu et al. (2024) recommend using masking techniques for GPU programs (p. 6). Masking techniques are essentially randomizing intermediate values in a program. The end goal of this technique is to remove the correlation between power consumption and the exact quantity being computed. A practical way to implement this technique on the CS server is to create software that reads the program a user wants to run and applies the masking techniques while it runs. Think of it like an interpreter for a program whose sole purpose is to improve its security. The downside of this solution is the increased computational load it instills on the system.

Another protection against side-channel attacks that Lungu et al. (2024) propose is that users are only able to run constant-time algorithms on the GPU (p. 6). The logic behind this proposal is that if all programs run in constant time, they would be identical to each other, therefore removing the possibility of a timing analysis attack. However, this proposal is not

plausible to implement on the CS server because computer scientists need to run a variety of programs and test on the Chimera. Sometimes, the programs they run could theoretically be impossible to implement in constant time. Therefore, while Lungu et al.'s proposal is effective in theory, it is not practical.

To combat Row Hammer attacks, Plin et al. (2025) recommend using ECC memory (p. 3), which is hardware that essentially self-corrects its own memory. This solution only works with hardware, which cannot be directly adapted to Chimera cluster. However, software can be implemented to self-correct. For example, when we implement Hamming codes[2] in our database, the software will automatically correct itself during Row Hammer attacks. The downside to Hamming codes is that they increase the size of our files. One way to maximize the benefits of Hamming codes would be to only use them for sensitive information. For example, we could implement Hamming codes for certain sections of data related to privilege escalation. That way, even if a Row Hammer attack were executed, the code would automatically revert the privileges once escalated.

Chapter 5 - Conclusion

Throughout the semester, Team 6 has implemented a tech stack that monitors and displays metrics for Chimera, the GPU cluster the CS department uses for research. Its implementation, while being counterintuitive, can be seen as both a security risk and a defense for the CS servers. Through exposing the metrics, we allow for the data used in side-channel attacks to be accessed. However, by exposing these metrics, we also introduce a new avenue for detecting attacks, such as the Row Hammer attack. In each case, there is a measure that can be taken to maximize the security of CS servers while minimizing their risk.

[2] Hamming codes were covered in CS444 (Operating Systems) and are assumed to be general knowledge for this paper.

References

Lungu, N., Patra, S. S., Mishra, M. R., Dash, B. B., Sasmal, G. C., Pattnayak, P., Singh, S., & Gourisaria, M. K. (2024). GPU side-channel attack classification for targeted secure shader mitigation. SN Computer Science, 5, 1148. https://doi.org/10.1007/s42979-024-03514-9

Plin, A., Fauberteau, F., & Nguyen, N. (2025). OpenGL GPU-based Rowhammer attack (work in progress). arXiv. https://arxiv.org/abs/2509.19959