Monitoring and characterizing GPU Usage

Le Mai Weakley
Research Technologies
Indiana University
Bloomington, IN, USA
llnguyen@iu.edu

Scott Michael
Research Technologies
Indiana University
Bloomington, IN, USA
scamicha@iu.edu

Laura Huber
Research Technologies
Indiana University
Bloomington, IN, USA
lamhuber@iu.edu

Abhinav Thota
Research Technologies
Indiana University
Bloomington, IN, USA
athota@iu.edu

Ben Fulton
Research Technologies
Indiana University
Bloomington, IN, USA
befulton@iu.edu

Matthew Kusz
Research Technologies
Indiana University
Bloomington, IN, USA
mkusz@iu.edu

***Abstract*—For systems with an accelerator component, it is important from an operational and planning perspective to understand how and to what extent the accelerators are being used. Having a framework for tracking the utilization of accelerator resources is important both for judging how efficiently used a system is, and for capacity and configuration planning of future systems. In addition to tracking total utilization and accelerator efficiency numbers, some attention should also be paid to the types of research and workflows that are being executed on the system. In the past, the demand for accelerator resources was largely driven by more traditional simulation codes, such as molecular dynamics. But with the growing popularity of deep learning and artificial intelligence workflows, accelerators have become even more highly sought after and are being used in new ways. Provisioning resources to researchers via an allocation system allows sites to track a project's usage and workflow as well as the scientific impact of the project. With such tools and data in hand, characterizing the GPU utilization of deep learning frameworks versus more traditional GPU-enabled applications becomes possible. In this paper we present a survey of GPU monitoring tools used in sites and a framework for tracking the utilization of NVIDIA GPUs on Slurm scheduled HPC systems used at Indiana University. We also present an analysis of accelerator utilization on multiple systems, including an HPE Apollo system targeting AI workflows and a Cray EX system.**

***Index Terms*—High Performance Computing, Monitoring**

I. Introduction

The Research Technologies (RT) division at Indiana University (IU) operates batch-scheduled HPC systems for a wide variety of researchers at the University. For nearly a decade, the systems operated by RT have had some substantial component that is equipped with accelerators. To date, the only accelerator technology that RT has deployed in large-scale production machines has been NVIDIA GPUs. When IU deployed its first large-scale GPU-based machine in 2013, the utilization of the GPUs was tracked to assess their effectiveness in facilitating research. Big Red II was a hybrid Cray system consisting of both CPU and GPU nodes, and initially GPU tracking was done using Cray-provided tools. As GPU accelerators were deployed in additional systems, and the support and ability to integrate the GPU tracking tools from Cray was unable to be extended to other systems, RT developed a system that directly leveraged APIs supported by NVIDIA GPUs.

Analysis of the data collected via this system has proven useful for various applications, from guiding users to more appropriate resources, to helping users improve applications and workflows and make more effective use of GPU accelerators, to informing decisions on the composition of future systems. While collecting the data can be a challenge in itself, analyzing and interpreting these data to provide actionable insights to end users, application support specialists, and division senior leadership can be even more of a challenge. By creating the tools to capture GPU utilization data and processes to analyze and interpret this data IU has been able to benefit end users in both the short term and long term.

This paper presents an overview of the framework that Indiana University has used over the past several years to collect GPU utilization data. The current process evolved from using the RUR reporting tool supported by Cray that provided GPU usage statistics on a per-job basis to a system that directly queries the GPUs via NVML. A data processing workflow was developed to aggregate data and match it with the relevant job logs (initially PBS Torque and later Slurm). We present the data collection method, which allows for the tracking of multiple simultaneous jobs running on a node or the correlation of GPU usage across many nodes in a single job. In addition to being used on an HPE Apollo system primarily used for deep learning workflows, we have recently deployed this system to IU's flagship Cray EX system, Big Red 200, and have been collecting data on that system. We present a comparison of this custom-written solution to other GPU tracking options, such as NVIDIA's Data Center GPU Manager (DCGM). We also discuss the nature of the metrics provided by NVML, what these numbers mean, and give context for how to evaluate the efficiency of GPU usage. We explore the differences between the metrics that can be provided by a custom solution vs. what DCGM gives out of the box, and how each approach can be integrated into a larger data gathering framework that incorporates data from the scheduler, systems that track application usage (e.g. XALT), software environments (e.g. Lmod or Modules), and systems for allocations. We also present an analysis of GPU utilization across both the Cray EX machine and the HPE Apollo system investigating the efficiency of different types of workloads. By combining project description data from our allocations process and application level data from XALT, we categorize workflows by the type of application (e.g. deep learning vs. simulation) and compare the overall efficiencies of GPU utilization for different types of workflows. We conclude with a discussion of what a reasonable expectation for accelerator utilization might be based on our analysis and experience.

This paper is structured as follows: in section II we outline the motivation behind the design of the system we created, and some of the base-level requirements for the system. Section III covers the implementation of the system and provides details on the infrastructure pieces that work together to track GPU usage. Some pitfalls we encountered and the workarounds for them are highlighted. In section IV we present some of the data and findings from the system, and in V, we survey what other sites are doing in this area. We finally present our conclusions in section VI.

II. Motivation and Design Considerations

The first large-scale GPU-based HPC system at IU was Big Red II, a Cray XE6/XK7 hybrid system, which went into production in the second half of 2013. Our initial objectives for tracking GPU usage were to track GPU usage on a per-job basis, ideally with measures of how much of the GPU had been utilized during the course of the job, to tie this data to other job reporting metrics, and be able to report GPU usage both holistically for the machine and individual users. To accomplish this, we needed a way to capture data for GPU usage, a way to store the data so that it could be correlated to other job data, and some way to analyze and package the data for reporting purposes. In an ideal situation, each of these components would be configured and connected together somewhat independently and would not require large changes from an existing scheduler configuration or other monitoring frameworks.

Over time, the tracking of GPU usage at IU has evolved; this evolution has been driven by several factors: a desire by center leadership for more precise and accurate accounting and utilization data, changing availability of monitoring tools, changing system hardware, and changing modes of GPU utilization by end users. The largest change came from moving from a large-scale Cray system with a single job and single GPU per node to a more commodity cluster with a non-Cray open-source HPC management stack, and both multiple GPUs and multiple users per node.

III. Implementation of GPU tracking

The tracking of GPU usage on the Big Red II system was accomplished by using the Cray-provided RUR data integrated with the scheduler output from PBS/Torque. The RUR system was not particularly well documented, making it difficult to know what each field in the log output represented. These data were largely used to determine whether or not a job requesting a GPU used a GPU. Each GPU node had a single K20x GPU and was scheduled job exclusive. However, a benefit of the RUR system was that it tied GPU utilization directly to jobs. Prior efforts to measure GPU utilization [1] had used periodic polling of GPUs via the `nvidia-smi` interface providing point-in-time utilization numbers causing additional challenges in correlating to specific jobs and workflows.

Big Red II was retired at the end of 2019 with a new Cray EX system on the horizon. In the interim, RT deployed a cluster-based GPU resource as part of the existing Carbonate cluster to support deep learning workflows. By mid-2020 RT had deployed a set of Lenovo and HPE Apollo nodes with a combined total of 16 P100 GPUs and 104 V100 GPUs. The Lenovo nodes were targeted specifically at deep learning workflows and contained two GPUs per node, and the Apollo nodes were made available to both deep learning and GPGPU workflows and had four GPUs per node. Both partitions are scheduled via Slurm.

Collecting GPU utilization data on these partitions required shifts on two fronts from how data was collected on Big Red II. The first being that a different scheduler was being used, and the second being that the RUR collection mechanism was not available on non-Cray systems. As all the machines at IU were moving to Slurm, RT looked into how other sites using Slurm were collecting their job accounting data. We used the `sacct` command from Slurm, which allows one to query job data from the Slurm accounting database. While NVIDIA's DCGM appeared to have many desirable features, on initial investigation, it appeared to have too intrusive of a footprint to be viable for deployment on RT's production systems.

A site report from the Swiss National Supercomputing Centre (CSCS) at the Slurm User Group Meeting in 2018 [2] detailed some custom tuning to Slurm to track resource usage and management. As part of this presentation, CSCS described how they were populating the AdminComment field in the Slurm accounting database with RUR data on their Slurm scheduled hybrid Cray systems. This approach by CSCS was also detailed in a paper at the Cray User Group meeting in 2018 [3], which provides details on the implementation and reporting abilities of the system that CSCS had implemented.

With this template in mind, we set out to design a system that leveraged the Slurm AdminComment field in a similar manner, but did not rely on the RUR system to track GPU usage. With Slurm, one can set a prolog or epilog script to run at the beginning or end, respectively, of a job. CSCS used this functionality to run RUR with the GPU accounting plugin. They then used a modified plugin for RUR to send the information to the SlurmDB to place the RUR data in JSON format in the AdminComment field in the Slurm accounting database. For the Carbonate GPU partitions, RT decided to use an epilog script to launch a Python script that utilizes the Python bindings to the NVIDIA Management Library (`pynvml`) [4] to acquire GPU utilization data from the GPUs reserved by a job and store the data in JSON format in the

AdminComment field in the Slurm database. Python scripts are then run on a daily cadence to get job accounting data and process the GPU accounting data using the `sacct` command and store the processed data in a MySQL database.

In 2022, Big Red 200, an HPE Cray EX system, was made available to IU researchers. Big Red 200 includes a Slurm scheduled partition of 64 GPU accelerated nodes, each with 4 NVIDIA A100 GPUs. The framework developed on Carbonate for collecting GPU utilization via `nvml` and storing the information in the Slurm accounting database was also implemented on Big Red 200 at the end of 2022.

The `nvml` functions provide access to accounting data stored on each GPU card and provide a variety of different types of data, which are well documented on NVIDIA's documentation pages for `nvml` [5]. The basic idea is that statistics are gathered for each process that is executed on a card, including the amount of time the GPU was active for that process, the memory utilized, the memory high water mark, and some other identifying data such as the GPU process ID and the GPU serial number. These data are stored in a ring buffer that can hold a given number of records (4,000 by default for the P100 and V100 cards in Carbonate), and once the buffer is full, the oldest records are replaced. The epilog script reads the records for each GPU process id from the buffer for each card in the Slurm job, adds these records to the AdminComment field, and then clears the records from the cards' buffers. An example of the record stored in the AdminComment field is provided in listing 1. A given job can have several hundred GPU processes recorded, though we only give a small example here. This approach has the advantage that it is already embedded in the Slurm record so correlating the GPU usage with the job is straightforward.

Listing 1: Sample AdminComment Field

```
[{"pid": 99868, "startTime":
1626116546415972, "time": 11318,
"gpuUtilization": 0, "serial":
"0322418035782", "maxMemoryUsage":
4551.0, "memoryUtilization": 0},
{"pid": 99868, "startTime": ...}]
```

One challenge with this approach is determining a single number for GPU utilization for a given job. We have taken the approach of time-weighting each utilization measure for all the processes, summing these, and dividing by the total allocated GPU time for the job. For the memory utilization, we take the max of the `maxMemoryUsage` value. For a job using N GPUs that executes for T seconds of wall time, with p processes, per process utilization U_p and process time T_p the job utilization U_j is computed as:

$$U_j = \frac{\sum_p U_p T_p}{NT} \tag{1}$$

While this may not be entirely precise depending on how the GPU processes are being spawned, it seems to give a good indication overall. It is also important to note that the per process utilization number provided by `nvml` as `gpuUtilization` provides a measure of the percentage of time that the GPU was active *in any way* throughout the duration of the GPU process. In this sense, it does not fully represent the utilization percentage of the GPU in terms of the percentage of GPU cores that are utilized throughout the process lifetime. With potentially multiple overlapping GPU processes, it is difficult enough to determine the percentage of time the GPUs were active in a given job, much less to what extent the computational power of the GPUs were being used throughout the job.

There have been other minor issues with this approach, and we are continuing to refine and improve it over time. One of the issues encountered was caused by duplicate records of GPU processes appearing in multiple jobs. It was first noted that, in some cases, for two jobs requesting a single GPU apiece and being scheduled simultaneously on the same node to run at the same time, both jobs would have GPU utilization from both GPU cards attributed to them and have duplicate processes in their records. To solve this, the epilog script was updated to use the `CUDA_VISIBLE_DEVICES` environment variable to ensure that only data from the GPUs that were available to the job is being collected. During this investigation, it was conjectured that perhaps the duplication of the process records was happening across different jobs from different users and on different nodes, which was indeed the case, as process ids were determined by the node, not the GPU card. While a small fraction of jobs fit this pathology, it was important to track the frequency at which this was occurring and to diagnose the possible causes. To track the recycling of process ids and cross-reference it to the users running jobs and nodes they were running on, the startTime field was collected into the AdminComment as well. Additionally, recording the serial number of the GPU card on which the GPU process is running can help to disambiguate process IDs.

Another issue encountered is the character limit of the AdminComment field in the Slurm database. In the MariaDB Slurm database, the AdminComment field is defined to be a TEXT field, which has a 65,535 character limit. For jobs with AdminComments that exceeded this number of characters, the data entry would be truncated and not all of the GPU utilization data will be stored in the Slurm database. This is a potential issue for long-running multinode jobs using several GPUs that could have a lot of utilization data, though the number of jobs so far that are impacted by this has been very small. We are currently tracking the number of jobs affected by this limit and exploring various solutions to trim the data being recorded. Potential solutions are moving from a JSON format to a CSV format, shrinking the names of the keys in the JSON format, and reducing the number of `nvml` fields to place in the AdminComment.

A more interesting issue arose with the implementation of this tracking system on Big Red 200 where Slurm appeared to be handing out the GPUs in the opposite order of their device nodes resulting in a mismatch between what GPU card in use with what `CUDA_VISIBLE_DEVICES`. Reversing the order of the NVIDIA devices in Slurm's `gres.conf` was the workaround proposed by SchedMD which solved the mismatch issue.

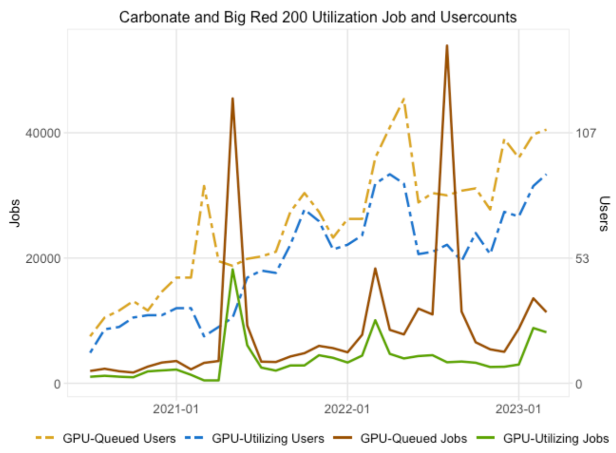


Fig. 1: Job and User Counts for GPU Utilization of Carbonate and Big Red 200

IV. Analysis of utilization

Measuring GPU utilization in an HPC center is important for optimizing resource allocation, reducing costs, planning for future capacity needs, and monitoring performance, making it a critical metric for data center administrators. For the `nvml` and Slurm-based data collection framework described in section III, we have been able to track multi-GPU usage on the hardware added to the Carbonate cluster in mid-2019 and again in mid-2020 as well as on Big Red 200 since the end of 2022. There are many uses for this data, including providing post-job reports to users, detecting highly inefficient usage patterns, and investigating system GPU utilization as a whole. In section IV-A, we present data collected from both the Carbonate Apollo nodes and from the Big Red 200 GPU-enabled nodes, then in section IV-B, we present data collected via the XALT application tracking framework [6] for the applications that were run in those GPU-enabled workflows.

A. Job-level utilization

We analyzed job data from July 2020 through March of 2023 for the Carbonate system and from December 2022 through March of 2023 for the Big Red 200 system. In all this encompassed 295,470 jobs that users ran where GPU's were requested. Overall we found that the mean GPU utilization as calculated via equation 1 was 11%. Of these jobs, approximately 156,785 jobs, or 53%, did not use GPUs at all. If we discount these jobs, the mean utilization increases to 26%. Mean memory utilization across all jobs was 9%, rising to 16% when excluding jobs that did not use a GPU. There was no correlation between the length of a job and the GPU utilization, nor between the number of jobs run by a particular user and their average utilization. Figure 1 shows the number of jobs run in the GPU partitions and number of users running those jobs on a monthly basis. The subset of jobs that used GPUs are shown as a separate line. These data contain counts of users utilizing GPUs for the Big Red 200 GPU queue from December 2022 through March 2023.

From figure 1, one may note periods where the number of users and jobs requesting GPUs and are utilizing them are closer together, followed by periods where those numbers are very far apart, with many jobs and users requesting GPUs but not using them in their work. Of note is the overall increase of users requesting GPUs in their jobs as time goes by, which can be attributed to the growing popularity and interest in AI workflows as well as to the increase in the availability of GPU-enabled nodes with the introduction of Big Red 200. The large gaps between the line of users and jobs requesting GPUs and those that actually use them could be due to new users familiarizing themselves with the system and how to run GPU-enabled workflows. Several of the spikes in job submission and in users submitting jobs align in time with the start of a new semester, so this may be a reflection of new students attempting to use GPUs for research. Further reasoning for this possible new user behavior is that while the number of users increases for some of these stretches of time, the baseline for the number of users and jobs using the GPUs remains quite steady, which may be evidence that the base level of users utilizing the GPUs have continued to utilize the GPUs for their ongoing projects while new users are getting started.

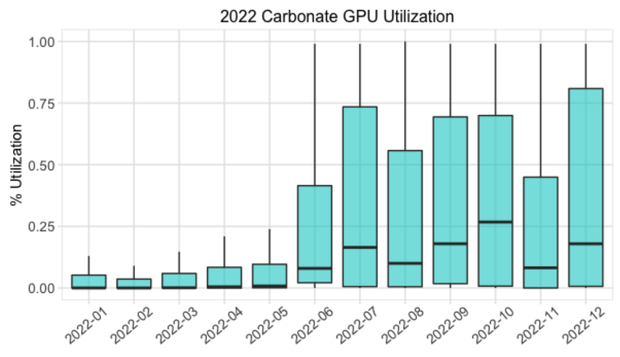
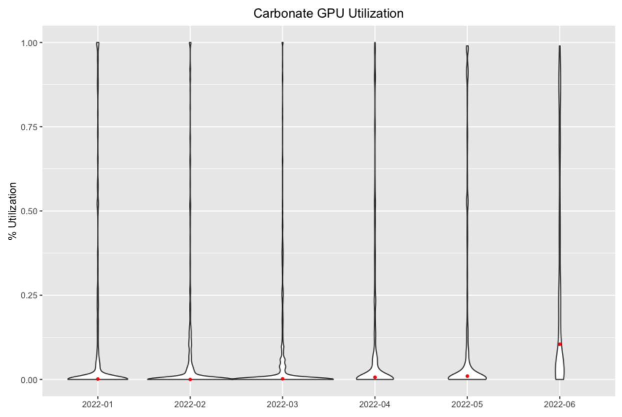


Fig. 2: Monthly distribution of GPU utilization percentages for jobs on Carbonate that used the GPUs

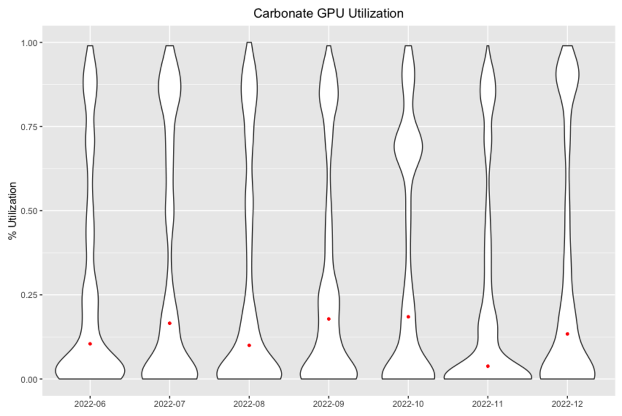
While the mean values can be informative, we decided to look a bit closer at the distribution of utilization across a number of jobs. Figure 2 shows the monthly breakdown of Carbonate jobs that utilized GPUs and the distribution of their utilization percentages throughout 2022. This time period encompasses 156,439 jobs in the GPU queues on the Carbonate machine. The boxplots are equally weighted for each job, and there is no consideration of resources allocated (i.e. there is no weighting by walltime or GPUs requested). It is interesting to note how not only the median percentage utilization goes up but also how the distribution of the GPU utilization spreads out between the months of July and December of 2022. There are a number of possible causes for this shift. It could perhaps be due to a confluence of changing workloads on the system that coincide with the mid-year boundary and required GPUs, or it could suggest that over time users have improved their workflows due to a combination of learning from testing and education on the part of center staff. Another consideration is that a large number of jobs may have been short-lived jobs with low utilization, which would skew these distributions toward lower utilization, while the system utilization could be dominated by a small number of long-running, highly efficient, multi-GPU jobs. In figure 3, we re-plot the distributions from the boxplots as density distributions (aka violin plots) for the first and last halves of the year. Clearly, these distributions are changing over time, likely due to different user populations, types of workloads, and the overall mix of jobs, but it appears that over the course of the year, the distribution of utilization has improved. The improvement is perhaps more striking than one might assume based on the average values or even the boxplots. Clearly, in the latter half of 2022, jobs with much higher utilization percentages have been added to the distribution. There appears to be an elongation of the distribution at lower utilization percentages, with jobs rising to utilization numbers as high as 30% followed by a desert of utilization until 65-70%. This bifurcated distribution could be explained by the difference between novice and experienced users, experimental versus more highly optimized codebases, or fundamental limitations on a class of workflows or algorithms. There are other dimensions to the utilization data to explore, such as utilization weighted by GPU hours, by user, or by project. This initial investigation provides a baseline understanding of what range of utilization a typical job mix has with IU's users on the Carbonate machine. Further discussion of future investigations is provided in section VI.

Although the Big Red 200 machine entered production in the middle of 2022, we first started fully capturing utilization data in December of 2022. Big Red 200 is managed by HPCM and scheduled via Slurm, and some modest effort was required to deploy the implementation of the data-gathering framework. With Big Red 200 being a Cray EX machine, having newer A100 GPUs, and fewer users with larger resource needs overall, we decided to compare the utilization distributions between the two machines. The user population does have some overlap, but is generally quite different with Carbonate seeing much more use for AI workloads, high throughput workloads, genomic and bioinformatic workloads, and student use, and Big Red 200 seeing larger scale simulation workloads like climate, materials, and astrophysical simulations. Figure 4 compares the percentage of GPU utilization for jobs on Carbonate and Big Red 200 that used the GPUs from December 2022 to March 2023. Of note is that while the median utilization for both Carbonate and Big Red 200 are both fairly low, for the upper half of the distributions, the percent of GPU utilization for Big Red 200 spreads out further into higher utilization percentages. The part of the distribution at higher utilization is much more substantial and spreads over a larger range. The higher end of the utilization distribution could be attributed to the size of the GPU partitions on each cluster, with Carbonate having significantly fewer GPU cards as well as older GPU cards compared to Big Red 200. Having

Fig. 3: Comparison of GPU percentage utilization over time



(a) Monthly distribution of GPU utilization percentages for jobs on Carbonate that used the GPU in the first half of 2022. The red dot indicates the median value of the distribution.

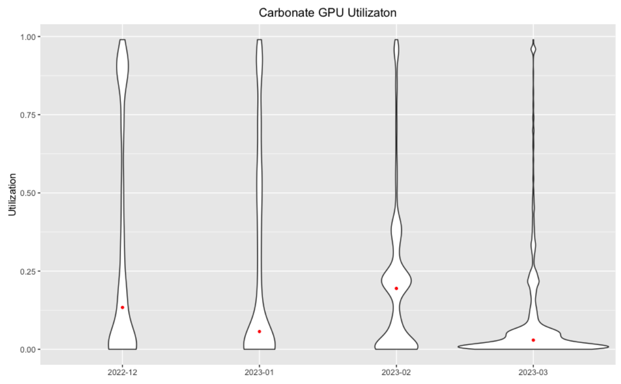


(b) Monthly distribution of GPU utilization percentages for jobs on Carbonate that used the GPU in the second half of 2022. The red dot indicates the median value of the distribution.

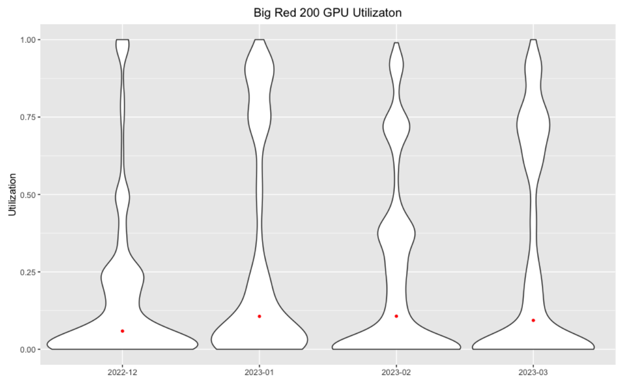
more GPU-enabled nodes on Big Red 200 could also lead to having a wider variety of workflows that could explain the larger variance in utilization. It is also possible that users running GPU-enabled jobs on Big Red 200 may have already tuned their workflows on Carbonate and then made a move to Big Red 200 where the larger number of GPU-enabled nodes allows them to submit more jobs running with higher utilization numbers on Big Red 200. In March 2023 we announced that the Carbonate cluster would be retired at the end of 2023. This announcement may have encouraged users to move their GPU jobs to Big Red 200, which could explain the distribution seen in March 2023 in Carbonate.

To further investigate the differences in the utilization distributions between the two machines, we looked at how many cards were being requested per job on each machine. On both Big Red 200 and Carbonate, the distribution of card hours by number of cards skews heavily in favor of jobs requesting four or fewer cards, with a large portion of those card hours being devoted to jobs requesting a single card. Figure 5 shows the GPU card hours requested by jobs for the

Fig. 4: Comparison of GPU percentage utilization on Carbonate and Big Red 200



(a) Monthly distribution of GPU utilization percentages for jobs on Carbonate that used the GPU in the first quarter of 2023. The red dot indicates the median value of the distribution.



(b) Monthly distribution of GPU utilization percentages for jobs on Big Red 200 that used the GPU in the first quarter of 2023. The red dot indicates the median value of the distribution.

same time period as the distributions in Figure 4, categorized by the number of cards requested. On Carbonate there is a limit imposed on the GPU queues such that jobs can not request more than 16 GPUs. For Big Red 200 there is no limit on the number of GPUs a job can request so the full 256 GPUs could be used in a single job. The peaks in GPU card hours requested are for a single card and 3-4 cards. With each node in Carbonate and Big Red 200 having 4 GPUs, the 3-4 card peak aligns with a full node request. There is also a peak on Big Red 200 for jobs requesting 32 cards or more. Although the amount of resources requested beyond a single node is not massive, we wanted to investigate the differences in utilization based on the number of GPUs requested. Figure 6 shows the distributions of the percentage of GPU utilization broken out by the number of GPU cards requested for the job. We divided the groups into jobs that requested a single node's worth of GPUs, jobs requesting more than 4 GPUs, but fewer than the limit on Carbonate, and jobs requesting more than the maximum on Carbonate. Somewhat unsurprisingly, the largest utilization numbers are for the single-node jobs with the median being somewhat higher, and the upper quartile and upper bounds being much higher than for multi-node jobs. What is a bit surprising is that there is little difference in the utilization numbers for the 5-16 and 17+ distributions. This suggests that the differences seen in the monthly distributions between Carbonate and Big Red 200 in figure 4 is largely due to a difference in the utilization for single-node jobs.

B. Characterization of workflows

In addition to investigating the GPU utilization of jobs on the GPU-enabled partitions, we wanted to understand how jobs are utilizing the GPUs in these partitions. We sought to find answers to questions such as the size of the jobs and types of applications being run in the GPU partitions; whether there were differences in utilization between software that was user-installed or installed by RT staff; and whether the workflows utilizing the GPUs were involved in Machine Learning or

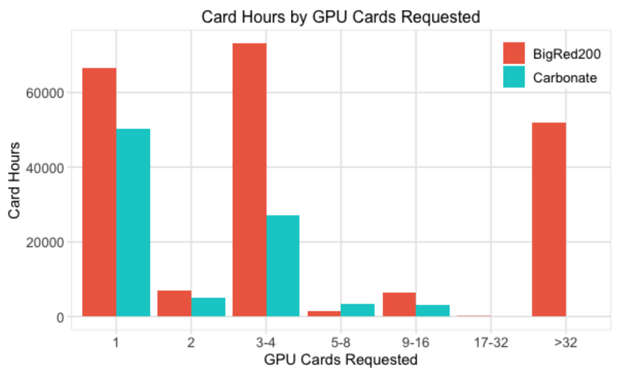


Fig. 5: Distribution of GPU card hours requested by number of cards requested per job.

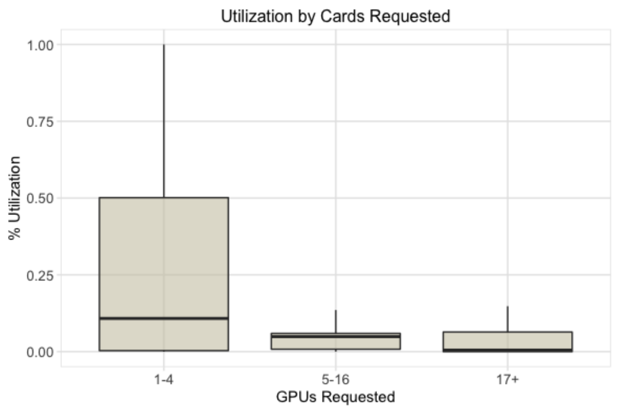


Fig. 6: Percent GPU Utilization distribution by number of cards requested

Artificial Intelligence, or were being used in more general purpose modeling and simulation.

Classification	Total Runtime (Hours)	Total Runs
AI Applications	1,162,885	289,154
Non-AI Applications	694,475	251,788
System Applications	496,857	182,760
User Applications	1,360,504	358,182

TABLE I

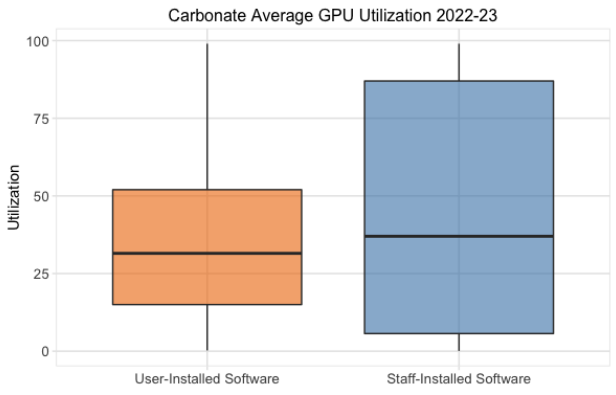


Fig. 7: Average Application GPU Utilization by Software Installation Type

To address the question of what kinds of GPU-enabled workflows are being run on Carbonate and Big Red 200, we looked at XALT data. XALT is a tool that instruments individual jobs to generate a picture of the compilers, libraries, and other software that users need to run their jobs successfully [6]. Since XALT data includes the executable and path of a subset of all applications that run in a job, we examined the applications that ran in jobs that utilized the GPUs on Carbonate and Big Red 200 from 2022-2023 to understand which applications are being run as part of GPU-enabled workflows. We identified 645 unique applications pulled from Carbonate and Big Red 200 XALT data and determined if each application had been installed by a user or by RT staff for general use. We did this by examining the location of the application to see whether it was located in one of the staff-managed application areas for common use, or in a user-owned location. Figure 7 expresses a higher median utilization in staff-installed software and an elongated distribution towards both ends of utilization percentages. In addition, a panel of applications experts examined the applications used along with the project descriptions in our allocations system and attempted to identify those that belonged to an AI workflow. From this data, it was determined that 53% of the 645 applications that were run in jobs that utilized the GPUs were part of AI workflows, and 34% were installed by RT staff. Though the number of unique applications is an interesting metric, examining the resources used by different classes of applications is more instructive. Table I gives the values for the total runtimes for this application classification. This metric comes from the XALT data and is the total walltime where these applications were running. To determine the prevalence of popular programming languages in AI research, we divided the AI applications into those that were Python-based, and others. Less than 1% of the applications run in AI workflows were not Python scripts (Table II).

AI Applications	Total Runtime (Hours)	Total Runs
Python Applications	1,156,424	286,335
Other Applications	6,461	2819

TABLE II

V. Survey of tracking tools

In order to have a more complete picture of the landscape of GPU monitoring in HPC centers, a survey was conducted to build a list of the various tools sites have been using and how these tools are being used. The survey was shared with the HPC community via multiple channels, including the Campus Champions mailing list, HPC Social Slack organization, and other venues. Overall, we estimate that the survey reached over 300 institutions with varying levels of HPC resources on-site, but we only received 11 total responses. Based on the responses we received, it appears that it is standard practice to measure and bill for GPU time that is consumed by jobs, but that does not confirm whether the jobs utilized the GPUs or how much the GPUs were utilized during the job.

Out of the eleven responses we received, eight respondents said that they are tracking GPU utilization in some manner, and three said they are not tracking this at the moment (two of these three respondents said that they were tracking the job accounting via XDMoD [7], without tracking GPU utilization). The majority of the respondents tracking GPU utilization are doing it using in-house scripts that depend on `nvml` [5] and the `nvidia-smi` utility. Only one respondent is using DCGM [8]. One respondent mentioned using NVTOP [9], which is a `top` like task monitor and has support for AMD, Intel, and NVIDIA GPUs.

We briefly explored DCGM but found it hard to fit it into a traditional HPC system monitoring model. At the time of writing this paper, unlike `nvidia-smi`, DCGM is unable to write output in a machine-readable format. NVIDIA is providing a tool to read the DCGM output called `DCGM-Exporter` [10], but according to NVIDIA, "`dcgm-exporter` can only be run as a standalone container or be deployed as a daemonset on GPU nodes in a Kubernetes cluster" [10].

In February 2023, SchedMD released Slurm 23.02. The release notes state that they have "added usage gathering for gpu/nvml (NVIDIA) and gpu/rsmi (AMD) plugins" [11]. We are still working on gathering more information about what this entails at the time of writing.

We should note that while we only focused on NVIDIA GPUs in our discussion, overall, we have observed limited adoption of GPU utilization monitoring at HPC sites. The sites deploying AMD GPUs are still in the early stages of their deployments, but we expect to see their efforts on GPU monitoring in the coming days.

VI. Conclusions and Future Work

Using the tools described here has given us a preliminary understanding of how GPUs are being used and how well they are utilized by researchers at IU. This is important for both individual research groups to make the most of the HPC resources they are allocated and for operators of HPC resources to help guide researchers so that a system is optimally utilized.

As the landscape of research computing continues to evolve with an enhanced focus on machine learning and continues to grow in terms of diversity of fields and data size, the importance of tracking how GPU acceleration is utilized in these fields and how that can be supported and improved also continues to grow. With deep learning workflows and frameworks becoming more prevalent in HPC, teams that support GPU-based HPC systems must adapt to a new kind of user and use case that may be less focused on optimally leveraging the available hardware. Works cited in section III and results from the survey outlined in section V provide evidence that sites with such resources have an interest in gathering GPU utilization data of this nature but perhaps have not had the time to devote to collecting these metrics and publishing in-depth analysis of what these data tell us about the actual utilization on the systems we support.

A key question in this analysis is, "What is the expected or desired level of GPU utilization for a job?" Over the past several decades, the research computing community has developed a very good intuition and expectation for utilization in CPU-based workloads, but that intuition is not necessarily directly transferrable to GPU workloads. In addition, as our analysis has shown, there is a broad spectrum of GPU utilization when looking at large number of jobs. How much of this is due to inefficient deployment or unoptimized code versus being inherent to the workload and data structures themselves is very hard to say without further study. One thing to remember is that applications with very low GPU utilization can still be many times faster in execution time than their CPU-only counterpart. Whether an application with 20% GPU utilization that runs three times faster than its CPU-only counterpart justifies the "low" utilization is a question open for debate. Ultimately, it is probably best to give as much data to the research teams using the resources and let them decide. Something that we did not find in the survey or a search of the literature available to us is a standard definition of job-level GPU utilization at different sites, how these data are collected, and how best to interpret these metrics to understand and address the needs of our user base.

To that end, along with current plans to tune and continue to improve the current implementation of the GPU data collection framework, IU plans to conduct further individualized analysis on the data coming out of these tools. The analysis made in section IV offers some examples of what insights can be garnered from such data collection frameworks. In addition to trying to correlate job-level data in IV-A and with application-level data IV-B and furthering the analysis of the observations made in those sections, a future goal is to provide individual researchers and/or research teams a comprehensive view into their overall usage and efficiency to help them make the best choices for optimal deployment of their workflows. The data that have been collected to date will continue to inform our planning for GPU capacity and deployments of future systems. It would also be informative to have finer-grained application-level utilization metrics and to be able to untangle or detect that from other applications that are run as part of these GPU-enabled workflows.

As we look toward future GPU platforms, the framework that has been built may need to be adapted to accommodate new features arriving on current or next-generation GPU cards (e.g. MIG modes on the NVIDIA GPUs). To date, IU has only ever deployed large-scale GPU systems based on NVIDIA accelerators; however, in the future other accelerators may be used, which would require additional adaptation of the framework to accommodate data collection from different architectures.

Acknowledgment

We would like to thank Bret Hammond, Adrian Hosey, Matthew Allen, and David Hancock for their contribution in implementing this data collection framework Carbonate and Big Red 200. This research was supported in part by Lilly Endowment, Inc., through its support for the Indiana University Pervasive Technology Institute.

References

[1] T. K. Samuel, S. McNally, and J. Wynkoop, "An analysis of gpu utilization trends on the keeneland initial delivery system," in *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the EXtreme to the Campus and Beyond*, ser. XSEDE '12. New York, NY, USA: Association for Computing Machinery, 2012. [Online]. Available: https://doi.org/10.1145/2335755.2335793

[2] M. Gila. (2018, Sep.) Tuning slurm the cscs way. [Online]. Available: https://pdfs.semanticscholar.org/e07a/be7f0a3a05514f72cd238af693836536ad80.pdf

[3] N. P. Cardo, M. Gila, and M. Klein, "Gpu usage reporting," presented at the Cray User Group, 2018. [Online]. Available: https://cug.org/proceedings/cug2018_proceedings/includes/files/pap152s2-file1.pdf

[4] NVIDIA. (2021, Jul.) Python bindings to the nvidia management library. [Online]. Available: https://github.com/gpuopenanalytics/pynvml

[5] ——. (2021, Jul.) Nvml api reference guide. [Online]. Available: https://docs.nvidia.com/deploy/nvml-api/index.html

[6] K. Agrawal, M. R. Fahey, R. McLay, and D. James, "User environment tracking and problem detection with xalt," in *2014 First International Workshop on HPC User Support Tools*, 2014, pp. 32–40.

[7] J. T. Palmer, S. M. Gallo, T. R. Furlani, M. D. Jones, R. L. DeLeon, J. P. White, N. Simakov, A. K. Patra, J. Sperhac, T. Yearke, R. Rathsam, M. Innus, C. D. Cornelius, J. C. Browne, W. L. Barth, and R. T. Evans, "Open xdmod: A tool for the comprehensive management of high-performance computing resources," *Computing in Science Engineering*, vol. 17, no. 4, pp. 52–62, 2015.

[8] NVIDIA. (2023, Apr.) Nvidia dcgm. [Online]. Available: https://developer.nvidia.com/dcgm

[9] NVTOP. (2023, Apr.) Nvtop. [Online]. Available: https://github.com/Syllo/nvtop

[10] NVIDIA. (2023, Apr.) Nvidia dcgm-exporter. [Online]. Available: https://docs.nvidia.com/datacenter/dcgm/latest/gpu-telemetry/dcgm-exporter.html

[11] SchedMD. (2023, Apr.) Slurm 23.02. [Online]. Available: https://www.schedmd.com/news.php