# Optimized GPU usage in high energy physics applications

Journal of Physics: Conference Series **3206** (2026) 012005 doi:10.1088/1742-6596/3206/1/012005

# Optimized GPU usage in high energy physics applications

**Tim Voigtlaender, Manuel Giffels, Günter Quast, Matthias Schnepf and Roger Wolf**

KIT – Karlsruhe Institute of Technology (DE)

E-mail: tim.voigtlaender@kit.edu, Manuel.Giffels@kit.edu, guenter.quast@kit.edu, matthias.schnepf@kit.edu, roger.wolf@kit.edu

**Abstract.** Machine learning (ML) applications, which have become quite common tools for many high energy physics (HEP) applications, benefit significantly from GPU resources. To fulfill the rapidly increasing demand for GPU resources, the efficient utilization of GPU clusters is vital. The Karlsruhe Institute of Technology (KIT) provides a GPU cluster, accessible via a traditional batch system as well as via grid compute elements. Because the exact hardware needs of such applications heavily depend on the ML hyperparameters, a flexible resource setup is necessary to utilize the available resources as efficiently as possible. Therefore, the multi-instance GPU (MIG) feature of the NVIDIA A100 GPUs was studied. Several neural network (NN) training scenarios performed on the GPU cluster at KIT are discussed to illustrate the setup that has been used and possible performance gains.

## 1. Usage of GPU resources by the CMS Collaboration

High energy physics (HEP) computing is a resource-intensive field due to the large amounts of data provided by the CERN LHC [1] and its experiments, the Compact Muon Solenoid (CMS) [2] being one of them. In addition to the workload needed to reconstruct the underlying particle physics events from this data, additional resources are necessary to simulate events based on theory. Finally, a wide range of end-user analyses require further resources to utilize the resulting event data and obtain physically relevant results. The hardware necessary for this is distributed in the Worldwide LHC Computing Grid (WLCG) [3] and is separated into four layers. The Tier 0, Tier 1 and Tier 2 sites provide resources that are available to the entire CMS Collaboration. These resources are utilized for a number of tasks, including event reconstruction, event simulation, long term storage of the data sets and end-user analyses. A number of Tier 3 sites exist, that are not officially part of the WLCG but still contribute to the computing resources accessible to HEP physicists. These sites are usually only used by local groups. Overall, this creates a heterogeneous mix of resources that has to be carefully designed and managed to satisfy the needs of the LHC and its experiments.

An overview of the plans that the CMS Collaboration has with its computing budget is given in the 2022 CMS Note [4], including a forecast with regard to GPU resources. It is estimated that 20% of the simulation workload and 25% of the reconstruction workload could be offloaded to GPU resources. The CMS Collaboration assumes that GPU computing resources are cheaper than CPU resources by a factor of 2.8, leading to an estimated increase in available computing resources of 20-26% by the end of the LHC Run 4 in 2030. Another factor in the rising importance

of GPU resources is their use in ML-based end-user analyses. While there are many tasks that can offload a significant fraction of their work to GPU, ML trainings can be performed almost completely on GPU.
Many groups in the CMS Collaboration already utilize machines with some GPUs to accelerate their end-user analyses, but in order to more efficiently use resources like these, advanced scheduling and wider availability are of importance. For this purpose, the Tier 3 Throughput Optimized Analysis System (TOpAS) [5] was deployed close to the Tier 1 GridKa site. It provides opportunistically resources to the CMS Collaboration and contains a significant amount of GPU resources. The GPUs present are 24 V100S, 8 V100 [6] and 24 A100 [7] NVIDIA GPUs.
Due to the wide usage of GPUs for ML applications outside HEP computing, manufacturers like NVIDIA have created a number of GPU models with unique features suited for the use in computing centers, and physicists stand to benefit from this development. One of these features, the NVIDIA A100's Multi-Instance GPU (MIG), is the focus of this paper and will be discussed in detail below.

## 2. Efficient sharing of GPU resources

An important challenge in HEP computing is the maximal utilization of the available hardware resources. For most users, it is not feasible to actively use their hardware at all times, leading to periods of inactivity where other users could utilize the hardware with little additional costs. It is therefore reasonable to share resources through some sort of scheduling that ensures a fair and efficient distribution to the individual users. HTCondor [8] is commonly used in HEP computing and is also installed on the TOpAS cluster mention above. The application and its requirements set by the user are referred to as job, and the assigned hardware resources are referred to as one or more job slots.
Multiple jobs are often executed on the same machine simultaneously to achieve maximum throughput. In order to hold users accountable and protect different jobs from each other, a high degree of isolation between the jobs is necessary. Due to the wide range of different analyses in HEP, the hardware requirements of the jobs also vary significantly. As a result, the boundaries between jobs have to be flexible and stable at the same time. Most hardware resources, like RAM, scratch space or the amount of assigned CPU cores, have a fine enough granularity to be distributed without problems. On the other hand, GPUs are typically not dividable and have to be assigned as a whole.
A whole GPU might be assigned to a job, even if the job is only capable of using a fraction of the GPUs computational capacity. This can lead to significant inefficiencies in the use of costly hardware resources. There are multiple options to approach this issue, the first being to assign the same GPU to more than one job. Due to recent advancements, such jobs are unlikely to interfere with each other as long as none of them exceed their assigned GPU usage. Unfortunately, it is unrealistic to assume that there will be no failing jobs. An insufficient isolation between jobs could then lead to multiple jobs failing together because one of them exhausted the GPUs capacity. In addition, it is difficult to monitor the resource usage of a specific job on a shared GPU, making it difficult to investigate issue.
A second option is to rely on features like NVIDIA's MIG service [9]. The service requires minimal setup and allows one GPU to be separated into multiple instances that each possess a fraction of the original resources and otherwise behave like a separate GPU. Figure 1 shows the minimal building blocks from which the MIG instances of an NVIDIA A100 GPU can be created. A GPU is somewhat limited in how it can be divided, and a full list of the possible setups for the available GPUs with MIG support can be found at [10]. The resulting MIG instances are isolated on a hardware layer, providing strong protection between the processes on different MIG instances of a GPU. HTCondor can register each of these instances as individual GPUs, each providing a slot for a potential GPU job. For an NVIDIA A100 GPU this means,

that with the help of the MIG service, one GPU can be turned into up to seven instances that each act like a smaller GPU.
In an optimal case, this approach could lead to an increase in GPU utilization by a factor of seven.

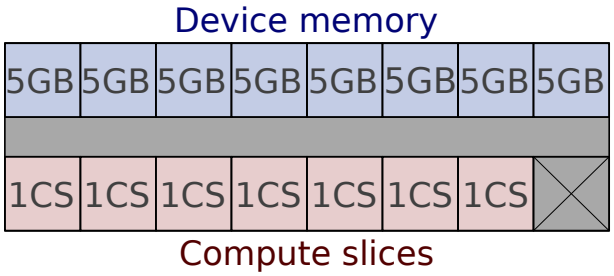


**Figure 1.** Sketch of the MIG building blocks of an NVIDIA A100 GPU. Each MIG instance consists of one or more compute slices (CS) and one or more slices of device memory with a size of 5 GB. One CS is reserved for the MIG overhead.

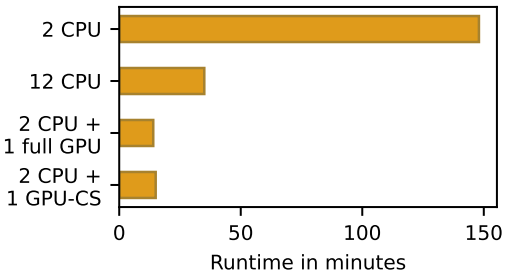


**Figure 2.** Benchmark of an ML task with varying amounts of hardware resources available to the training.

## 3. Viability check and performance benchmarks

To test the viability of the MIG service in combination with HTCondor two benchmarks were performed. The first one was to determine how the reduction in available GPU resources would impact an ML job with an otherwise low GPU utilization. The second benchmark was intended to simulate the impact of this approach on a filled queue of suitable ML jobs. The workload for both of these benchmarks consists of a sequential training of a feed forward fully connected neural networks (NN) with four hidden layers and 512 nodes per hidden layer. The trainings utilized $\mathcal{O}(10^6)$ training samples and a batch size of 512. All trainings were performed on the same machine, with varying degrees of the machine's hardware available to an individual training. The CPUs used were two AMD EPYC 7662 64-Core and there were eight NVIDIA A100 GPUs installed in the machine. Available scratch space and RAM were set high enough to not inhibit the training tasks. All trainings were performed using TensorFlow 2 [11] and the code was adapted to allow the utilization of GPU if such a resource was available to the training.

### *3.1. Hardware based performance comparison*

In the first benchmark, all jobs had the exact same workload, including random seeds, to ensure that the results would only depend on the hardware available to the job. As a baseline, the training was first performed, using two of the CPU cores and no GPU resources. Next, the training was performed using 12 of the CPU cores instead, still with no GPU resources. In the third run, the number of CPU cores was again reduced to two and one NVIDIA A100 GPU was assigned to the training. In the final run of this benchmark, the NVIDIA A100 GPU was set up as seven MIG instances, each with the smallest possible size. Each MIG instance consisted of one compute slice (CS) with roughly one eighth of the GPU's processing power and one 5 GB slice of device memory. The number of CPU cores was kept at two per job, and one of the seven instances was assigned to the training. The results of this benchmark can be seen in figure 2.
The benchmark shows that even a fraction of an NVIDIA A100 GPU provides more resources to the training than 12 CPU cores do, and that these additional resources can be utilized, even for a relatively simple NN. The runtime with the full GPU is about 5% faster than the runtime with

**Table 1.** Summary of the scaling benchmark results. "1g.5gb" is the name set by MIG for a MIG instance with one compute slice and 5 GB of VRAM. Total runtime is the time it took from the start of the first job until the completion of all 414 ML jobs.

| Configuration | # slots | VRAM per GPU | Total runtime in minutes |
|---|---|---|---|
| Full NVIDIA A100 GPUs | 8 | 40 GB | 594 |
| 1g.5gb MIG instances | 56 | 5 GB | 77 |

the MIG instance. This is a small difference considering that the training with the full GPU had access to eight times the GPU resources compared to the training with the MIG instance. This result is understandable, considering that the training with the full GPU only utilized it to about 15% of its maximum compute capacity.

### *3.2. Scaling benchmark*

After the first benchmark confirmed, that the general approach of using the MIG service to divide a GPU into smaller pieces can be used in conjunction with HTCondor, the second benchmark was intended to test its scalability. By providing a large amount of similar, but not identical, ML jobs, a filled batch job queue has been simulated. While all trainings in this benchmark used the same NN architecture, there were differences in the amount and distribution of the input data. A total of 414 jobs entered into the HTCondor job queue. All eight NVIDIA A100 GPUs of the machine were used in this benchmark. In the first iteration, the GPUs were used as is, providing eight job slots. In the second, each GPU was split into seven MIG instances with one CS and 5 GB of device memory each, resulting in a total of 56 job slots. In both cases, each job was assigned two CPU cores. The result of this benchmark can be seen in table 1.

During the runtime of the trainings, an average of seven times as many jobs were running with the MIG setup than with the whole GPU setup. This is also visible in the speedup of the total runtime. While the overall runtime depends on the length of each individual job and the number of simultaneous jobs, it also depends on the efficiency of the job scheduler, which is difficult to quantify exactly in this benchmark, due to a high variety in the runtime of the individual jobs. Nonetheless, the benchmark provides a clear picture of the possible benefit that can be gained from utilizing a technology like the MIG service.

## 4. Caveats and outlook

The MIG service is a technology with the potential to greatly benefit the sharing of GPU resources in HEP computing. It is new and in active development. While this means that there is hope for improvements to certain issues, it also means that it is not stable, yet, and changes to some core aspects have to be expected. As of writing, there are three GPU models that support MIG, the A100, the A30 and the H100 NVIDIA [12] GPUs. Older hardware and hardware from other manufacturers are not supported. This means that most current GPUs utilized in HEP computing are unable to use MIG at all.

There has been considerable development of the MIG service and its supporting software between the initial benchmarks and the writing of this paper, and a number of issues have been reduced. It is now possible to obtain the monitoring data of specific MIG instances by using the NVIDIA Data Center GPU Manager (DCGM) [13]. This is a crucial step towards the usage of the MIG service in cluster environments.

It is possible to create MIG instances as long as there is space on the host GPU, and they can be destroyed as long as no process is using them at a given moment. This provides a great degree

of freedom to dynamically manage GPU resources. It appears realistic, that the MIG service could be set up in a way that creates MIG instances of specific sizes, depending on the job requirements of the queued jobs in a batch system. Such a setup would reduce wasted computing resources for GPU jobs that do not fully occupy a whole GPU by placing multiple jobs on the same GPU well isolated from each other. After such a job is completed, the used instance could be removed and reused for the following job.
Due to the way that the MIG instance slots are set up, certain combinations are only possible in specific order. E.g., three instances with 4/2/1 CS can exist at the same time, but only if the instance with four CS is set up first. This could limit how well GPU jobs, that require more than the minimal amount of GPU resources, fit together with already existing jobs.

## 5. Conclusion

The Multi Instance GPU (MIG) service allows a GPU to be divided into multiple instances. The standout feature of this service is its high isolation between the individual instances. Monitoring of the instances is possible using the NVIDIA Data Center GPU Manager (DCGM) and creation as well as destruction of the instances can be performed dynamically.
The compatibility of this service with the software HTCondor was benchmarked using the TOpAS Tier 3 cluster, and an ideal usage scenario was explored. In this scenario, a filled batch queue of ideal jobs was simulated. A throughput increase of up to seven was observed by reaching a higher number of simultaneously running jobs.
The implementation of an HTCondor setup that is able to dynamically allocate GPU resources in the form of MIG instances of a requested size has been discussed and seems feasible.

## References

[1] Evans L, Bryant P. *LHC Machine*. Journal of Instrumentation. 2008 August;3(08):S08001. Available from: https://dx.doi.org/10.1088/1748-0221/3/08/S08001.
[2] Chatrchyan S, et al. *The CMS Experiment at the CERN LHC*. JINST. 2008;3:S08004.
[3] Shiers J. *The Worldwide LHC Computing Grid (worldwide LCG)*. Computer Physics Communications. 2007;177(1):219-23. Proceedings of the Conference on Computational Physics 2006. Available from: https://www.sciencedirect.com/science/article/pii/S001046550700077X.
[4] CMS Offline Software and Computing. *CMS Phase-2 Computing Model: Update Document*. Geneva: CERN; 2022. Available from: https://cds.cern.ch/record/2815292.
[5] Caspart, René, Fischer, Max, Giffels, Manuel, von Cube, Ralf Florian, Heidecker, Christoph, Kuehn, Eileen, et al. *Setup and commissioning of a high-throughput analysis cluster*. EPJ Web Conf. 2020;245:07007. Available from: https://doi.org/10.1051/epjconf/202024507007.
[6] NVIDIA Corporation. *NVIDIA Tesla V100 GPU Architecture* [Internet]; 2017. WP-08608-001. [Accessed 27th February 2023]. Available from: https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf.
[7] NVIDIA Corporation. *NVIDIA A100 Tensor Core GPU Architecture* [Internet]; 2020. [Accessed 27th February 2023]. Available from: https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf.
[8] HTCondor Team. *HTCondor*. Zenodo; 2023.
[9] NVIDIA Corporation. *NVIDIA Multi-Instance GPU*. https://www.nvidia.com/en-us/technologies/multi-instance-gpu;. [Accessed 20th February 2023].
[10] NVIDIA Corporation. *NVIDIA Multi-Instance GPU User Guide, Supported MIG Profiles*. https://docs.nvidia.com/datacenter/tesla/mig-user-guide/#supported-profiles;. [Accessed 20th February 2023].
[11] TensorFlow Developers. *TensorFlow*. Zenodo; 2023.
[12] NVIDIA Corporation. *NVIDIA H100 Tensor Core GPU Architecture* [Internet]; 2022. [Accessed 27th February 2023]. Available from: https://resources.nvidia.com/en-us-tensor-core/gtc22-whitepaper-hopper.
[13] NVIDIA Corporation. *NVIDIA Data Center GPU Manager*. https://github.com/NVIDIA/DCGM;. [Accessed 20th February 2023].