On-Grid GPU development

Via, interactive HT-Condor jobs and Analysis Facility style workflows

Albert Borbely[1,*], *David* Britton, *Emanuele* Simili, *Samuel* Skipsey, and *Gordon* Stewart

[1]SUPA-School of Physics and Astronomy, University of Glasgow, Glasgow, United Kingdom

Abstract. Over the last few years, an increasing number of sites have started to offer access to GPU accelerator cards but in many places they remain underutilised. The experiment collaborations are gradually increasing the fraction of their code that can exploit GPUs, driven in many case by developments of specific reconstruction algorithms to exploit the HLT farms when data is not being taken. However, there is no wide-spread usage of GPUs on the Grid and no mechanism, yet, to pledge GPU resources. Whilst the experiments gradually make progress porting their production code, and external projects such as Celeritas and AdePT tackle key common tasks such as the acceleration of E/M calorimeter simulation as a plug-in for GEANT4, there is no easy way for smaller groups or individual developers to develop GPU usage in a way that is easily transferred to the Grid environment. Currently, a user typically develops code on a local GPU in an interactive manner but there is significant overhead in subsequently containerising this work and moving it to the Grid environment. Indeed, many user jobs are not big enough to benefit from this last step and many sites must then maintain GPUs that are not integrated with the Grid infrastructure.

We have developed a proof-of-principle solution to enable interactive user access to Grid GPUs, enabling the initial development to take place on-Grid. This will ensure the development and production environments are identical and enable sites to move more GPUs to the Grid. An interactive development environment has been implemented with interactive HTCondor jobs and Apptainer containers. GPUs are split into MIG instances to allow for simultaneous multi-user utilisation. Users can install packages on the fly, giving them control over package versions as well as use what's available on CVMFS. Once development is done the sandbox container can be made imputable and submitted to either the local batch style GPU que or sent to the rest of the GPUs available on the Grid. The nature of interactive development means many hurdles had to be overcome such as: User authentication, security considerations, data replication to other sites, as well as management tools to allowing users to keep track of their environments and jobs.

1 Introduction

The primary project goal was to streamline the user development environment and reduce the overhead required to submit Grid jobs. This is primarily intended to be solved by bringing development onto the Grid and giving users flexibility. To achieve this users would be

*e-mail: albert.borbely@cern.ch

connected into a sandbox container to develop within. They would be allowed a degree of customization and the ability to install packages on the fly, via package managers such as spack/pip. By allowing the development to take place on-Grid hardware, the exact hardware specifications can be optimized for. It also allows for *in-situ* benchmarking to take place alongside development. It is especially beneficial for GPUs, as they exhibit a greater degree of variability compared to CPUs due to their large-scale differences. This variability arises from both the intended use cases (consumer vs. data-centre) and the recent AI boom, which has led to low-precision GPU hardware optimizations aimed at enhancing training speed and throughput. By allowing users to test and develop their software directly on-Grid hardware could allow for better optimised software, that is able to fully utilise the resources available i.e. memory/tensor cores. To further ease the user-experience a mechanism to convert the development environment (DE) to the submission environment (SE) seamlessly is desirable. This could then facilitate job submission to the wider Grid and promote on-Grid GPU utilisation by reducing the barrier of entry. Typical user-level problems currently are too small to substantially benefit from utilising on-Grid GPU resources whilst the barrier of entry is relatively high. The intention here was to lower the barrier of entry for the average user and to promote GPU development. With no wide-spread adoption of GPUs on the Grid and collaboration level projects, such as Geant4 GPU acceleration in development (Celeritas [2]/AdePT [3]). The intended outcome was to merge the development and batch hardware and promote expertise in GPU programming among user community

The key challenges identified that need to be overcome are: User Authentication; User Connection; Data management; Interactive development; Development and Submission environments; Integration of interactive and batch queues; Submission framework to wider Grid queues; and On-Grid Interactive GPU development. With the problems arising being categorized into two main themes: GPU problems and Interactive job development problems. Much of this work shares principles with Analysis Facilities which outlined the problems needed to be overcome in its white paper [1].

2 User Authentication

When creating a large-scale decentralised resource that can be utilised by a wide user community, user authentication and identification is important should any misuse occur. In keeping with the theme of reusing the Grid tools x509 Grid certificates [5] are initially used, with the intention of transitioning to token based authentication in-step with the rest of the Grid community [6, 7]. This allows us to utilise the robust user identification already in place, which incorporates both physical ID requirement as well as trusted issuing authorities to manage users. In addition, CERN SSO [8, 9] and IRIS IAM [10] were also considered but deemed un-necessary for the initial scope of the project. Initially a minimal viable product was delivered for testing. This included supporting non WLCG/CERN collaborations and avoiding a lengthy web-based development cycle.

3 User Connection

In continuing with the theme of re-using already in-place Grid tools ARC [11, 12], email and SSH were utilized to facilitate user connections with various degrees of success. The ARC submission framework was used to facilitate the initial interactive job (IT) request. This was done by submitting a job with a specific format and providing a public ssh key. The users were then emailed a connection script, based on their x509 certificates, that would handle the multiple ssh hops required to connect securely into the container. This is illustrated in Figure 1.

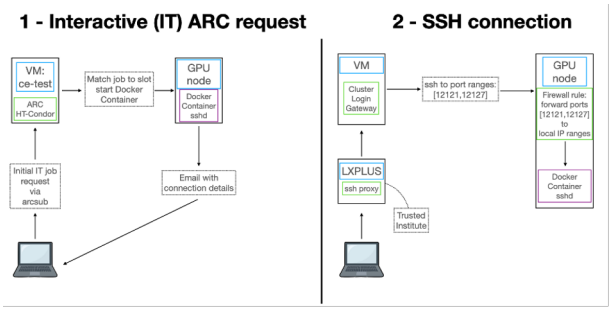

Figure 1. An illustration of: 1 the interactive (IT) ARC request, 2 the SSH connection.

The script would essentially install a duplicate ssh config, so as not to interfere with the users presumable heavily customized config. This step was deemed necessary to mitigate user error, as initially users were required to submit a bash script and then modify their ssh configs. During testing 3/5 users failed to correctly implement the bash script and successfully modify their ssh configs. However, this created a new host of problems as the setup script emailed to users was developed for UNIX systems so would only work with Mac/Linux users with Windows users being left out in the cold. This was deemed acceptable as most of the user community is familiar with UNIX systems, with the contingent using windows assumed to be familiar with scripting. The connection script contained all the required details for the connection and could be easily adapted for Windows if needed by the user.

The multiple ssh hops were a security requirement, as there is a delay between requesting the interactive job and then connecting/re-connecting to it. Jobs are initially kept alive for 20 minutes, until a live ssh connection is achieved otherwise they are put on hold to free up resources. Users may re-connect to pre-existing jobs with rudimentary mechanisms available for session management. As all ssh connections are entirely authenticated via the public ssh-key provided by the user at the time of interactive job request, which in turn was authenticated by the x509 proxy. Should the corresponding private ssh user key become compromised then said users' jobs could all be compromised. To minimise this attack vector, it was required that all incoming connections come from a list of trusted institutes which initially was implemented as: Glasgow, CERN, Nikhef and DESY. This would allow the tracking of malicious activities to the corresponding system admins and hopefully mitigate this potential vulnerability.

4 Data Management

Users have data management needs which are solved with existing Grid tools i.e. XrootD [13, 14] and the collaboration level wrappers (Rucio [15, 16], Dirac [17], StashCache [18], etc.). Storage at the the local Glasgow Tier-2 site where this is being implemented is provided via a CEPH cluster, with it's various services shown in Figure 2. The re-directors, gateways and caches are of particular importance as they dictate the speed at which I/O can be achieved. With on-site redundancy being provided a user may ship some data with their collaborations supported method to the site, which will be then stored in the CEPH cluster. If their collaboration supports CEPH-FS then their data set can directly be mounted with

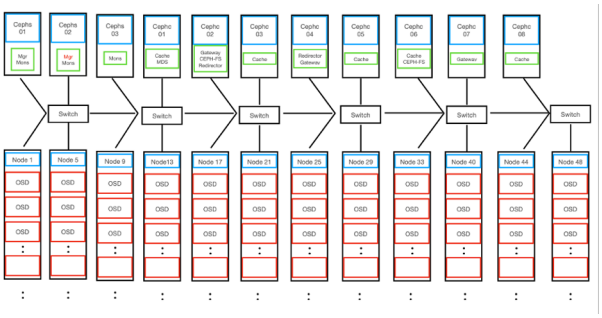

Figure 2. An illustration of the local CEPH cluster and the various services, with the gateways being of particular relevance for the project as the run the XrootD services. The primary gateway is Cephc04 which hosts the redirector which balances the load between the 3 gateway nodes Cephc(02,04,07) for the object store portion of the data. The primary gateway for the CEPH-FS portion of the data is Cephc02 which redirects half the traffic to Cephc06. Additionally Cephc01 hosts the Meta Data Server (MDS) which is a cache for the posix file system layer for the CEPH-FS portion of the data. The other Cephc nodes serve as read/write buffers. The Cephs(01,02,03) nodes run the mangers and the monitors. The managers can fail over from one to another (01,02) and are responsible for load balancing and provide metrics for monitoring the cluster. The monitors maintain the maps of the cluster including data placement. In depth details of a Ceph cluster can be found [4].

XrootD into the DE container. This allows them to access their files (read/write) in a POSIX file system and works for any type of file e.g. HDF5, root, JSON etc.

The primary drawback is in I/O access directly from the cluster, which can be particularly painful when reprocessing a batch of data. To alleviate this issue a read-only XCache proxy was implemented to cache recently read data on the node for subsequent re-reads. Users must take care to direct their write requests directly to the cluster and not via the proxy, where it will fail. An illustration of this can be seen in Figure 3.

5 Container Environments

The development environment for user was chosen to be docker [21] as it supports sandbox containers and GPUs where changes can be committed. It also supports an out of the box container registry that is easy to manage and can provide redundancy via CEPH-FS. In order to manage their sessions, users must be able to run certain commands on the host machine without having widespread admin access. A rudimentary mechanism was implemented using cron jobs [19] and the SUID bit [20], to elevate user permissions, for specific tasks. The container would run a ssh daemon directly and users would connect as a non-privileged UID, that exists only within the container. This is illustrated in Figure 4. Users can install packages into their home directory using spack/pip package managers and so don't require elevated privileges inside the container.

Apptainer [22] was chosen as the container solution for the submission environment as it is the standard Grid tool. Converting Docker images to Apptainer is straightforward with it ideally stripping out the unnecessary bits required for interactive development i.e. the ssh daemon. Submission to the wider Grid was not thoroughly tested, beyond a few pilot jobs to Manchester, with the standard arc framework being used.

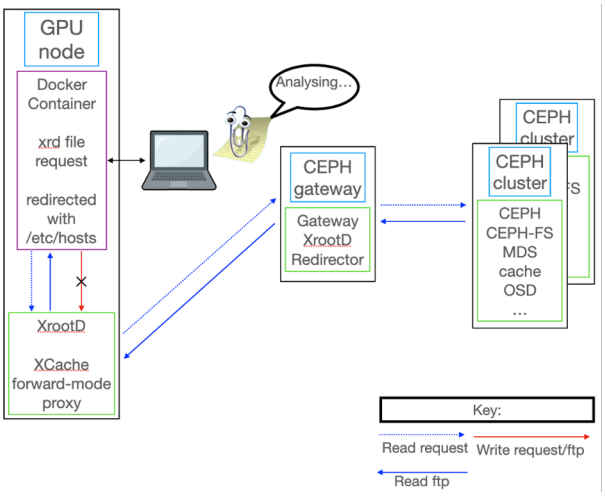

Figure 3. An illustration of the XCache implementation on the host. Traffic is redirected via the host file in the container to the XCache, which authenticates the users X509 proxy and applies the appropriate permissions. The XCache uses a shared secret to authenticates it's self as root against the CEPH cluster. The XCache is setup as a read-only proxy should a user wish to write to the cluster they must choose an alternative endpoint.

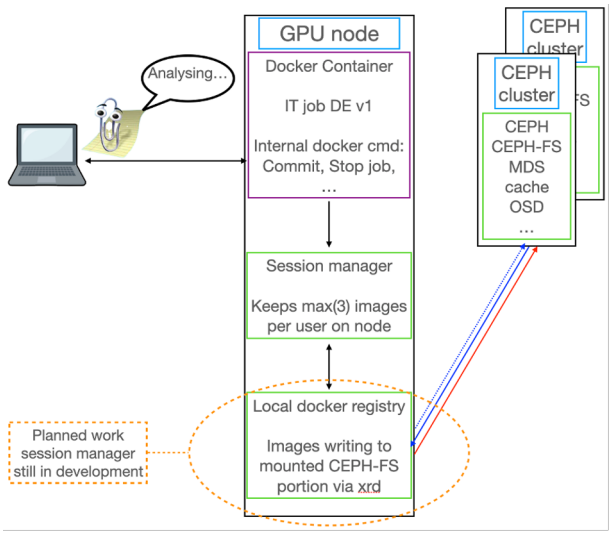

Figure 4. An illustration of the planned session manager infrastructure. It depicts how a user could issue commands on the host from within the jobs container environment, via the session manager. The users images are kept on a local docker registry with redundancy provided through the sites CEPH cluster. This is implemented by mounting the CEPH-FS portion with XrootD into the registry container.

6 Integration of interactive and batch queues

Having both interactive and batch queues utilising the same hardware represents numerous challenges. By nature, they are quite different as the batch queue can simply run one job after the other with the users not having to actively wait in the meantime. In comparison an interactive queue has the user actively waiting for a job slot. Once a user is connected for development there needs to be a way to monitor activity and free up the slot should the user become inactive.

Resource utilisation is also markedly different as resources are only sporadically used during development. Thus it would be ideal for interactive users to share resources to better maximise their utilisation. This can either be achieved simply by attaching i.e. one GPU to multiple user sessions, or by actively partitioning the GPU into separate Multi-Instance GPU (MIG) slices [23]. The former is more efficient and gives users access to the whole card, allowing for potential optimisation to take place. It has the issue that users would then be able to access memory segments in each others' code / workflow and relies on the users following a code of conduct. It could work on a collaboration level where there are other mechanisms in place to resolve disputes. The latter MIG partitions the cards into memory and compute slices giving users private access to a segment of the card. This has a few drawbacks one of which is it adds overhead which in turn has a performance hit. Additionally, the driver implementation is not robust and can require full power-cycles of the system to restore the GPUs operation. Finally, it somewhat defeats the purpose of giving developers access to the same hardware, as using MIG instances to partition the card in segments gives developers different amounts of memory and compute compared to what will be available in the batch system. Therefore, the first simple solution is preferable.

7 Problems encountered during development

This proof of concept was developed at an active Tier-2 site at Glasgow, which added numerous challenges. The test ARC submission entry point for interactive jobs was previously used to for ARM validation by several VOs. Because of this it was occasionally flooded with validation jobs, even after it had been repurposed, and would effectively DDoS the interactive queue. The CEPH cluster would also be in active use by jobs and would highlight several I/O bottlenecks that are noticeable in an interactive environment but negligible in a batch setting. This is hopefully solved with an on node XCache but would require users to manage it.

8 Summary

Overall, this was intended as a proof of concept and had the intention of trying to solve the interim issues and promote wide scale GPU adoption on the Grid. The trade-off between giving users the flexibility to customise their development environment and handle an off-site active connection created too many security concerns requiring the use of the SUID bit. No significant penetration testing was performed which was always a concern. Significantly more development time would be required to make a proper web-based session manager which was beyond the scope of this project, with the rudimentary implementations being cumbersome to use. As software that offloads significant processing to GPUs becomes available at the collaboration level and the subsequent investment into infrastructure to run it, the benefit of merging interactive and batch queues will decrease due to simple statistics. In other words, only a handful of GPUs are required for development while the demand for GPUs in batch systems will scale proportionally with the job requirements. Projects such as Celeritas/AdePT, which currently aim to offload EM calorimeter simulation in Geant4 [24] to

GPUs, already target the most demanding CPU requirements for several collaborations, and could provide a good case for widespread GPU adoption by the Grid.

References

[1] D. Ciangottini, A. Forti, L. Heinrich, N. Skidmore, C. Alpigiani, M. Aly, D. Benjamin, B. Bockelman, L. Bryant and J. Catmore, *et al.* "Analysis Facilities White Paper," [arXiv:2404.02100 [hep-ex], https://arxiv.org/abs/2404.02100].

[2] The Celeritas software project: https://celeritas-project.github.io/celeritas/, accessed on 1 May 2025.

[3] The AdePT software project: https://geant4.web.cern.ch/collaboration/working_groups/task_force_rd/g4rd14, accessed on 1 May 2025.

[4] S. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," *Proceedings of the 7th Conference on Operating Systems Design and Implementation (OSDI'06)*, pp. 307–320, 2006.

[5] X509 Grid Certificates https://www.nordugrid.org/documents/certificate_howto.html, accessed on 1 May 2025.

[6] SciTokens https://scitokens.org/, accessed on 1 May 2025.

[7] T. Dack, F. Agostini, J. Basney, L. Cornwall, J. S. De Stefano, Jr., D. Dykstra, F. Giacomini, M. Litmaath, R. Miccoli and M. Sallé, *et al.* "WLCG Transition from X.509 to Tokens. Status, Plans, and Timeline," EPJ Web Conf. **295** (2024), 04054 doi:10.1051/epjconf/202429504054

[8] Cern Single Sign On https://auth.docs.cern.ch, accessed on 1 May 2025.

[9] E. Ormancey, "CERN single sign on solution," *Journal of Physics: Conference Series*, vol. 119, no. 8, p. 082008, 2008. doi: 10.1088/1742-6596/119/8/082008.

[10] D. Crooks, I. Neilson, D. P. Kelsey, and I. Collier, "Building an IRIS Trust Framework," *EPJ Web Conf.*, vol. 245, p. 03018, 2020. doi: 10.1051/epjconf/202024503018.

[11] M. Ellert, M. Grønager, A. Konstantinov, B. Kónya, J. Lindemann, I. Livenson, J. L. Nielsen, M. Niinimäki, O. Smirnova, and A. Wäänänen, "Advanced Resource Connector middleware for lightweight computational Grids," *Future Generation Computer Systems*, vol. 23, no. 2, pp. 219–240, 2007.

[12] NorduGrid Advanced Resource Connector https://www.nordugrid.org/arc/arc6/, accessed on 1 May 2025.

[13] A. Dorigo, P. Elmer, F. Furano, and A. Hanushevsky, "XROOTD—A Highly scalable architecture for data access," *WSEAS Transactions on Computers*, vol. 1, no. 4.3, pp. 348–353, 2005.

[14] XrootD https://xrootd.org/, accessed on 1 May 2025.

[15] M. Barisits, T. Beermann, F. Berghaus, B. Bockelman, J. Bogado, D. Cameron, D. Christidis, D. Ciangottini, G. Dimitrov, M. Elsing, et al., "Rucio: Scientific data management," *Computing and Software for Big Science*, vol. 3, pp. 1–19, 2019.

[16] Rucio https://rucio.cern.ch/, accessed on 1 May 2025.

[17] F. Stagni, A. Tsaregorodtsev, L. Arrabito, A. Sailer, T. Hara, X. Zhang, et al., "DIRAC in large particle physics experiments," in *Journal of Physics: Conference Series*, vol. 898, no. 9, p. 092020, 2017.

[18] D. Weitzel, M. Zvada, I. Vukotic, R. Gardner, B. Bockelman, M. Rynge, E. F. Hernandez, B. Lin, and M. Selmeci, "StashCache: a distributed caching federation for the open science grid," in *Practice and Experience in Advanced Research Computing 2019: Rise of the Machines (learning)*, pp. 1–7, 2019.

[19] M. S. Keller, "Take command: cron: Job scheduler," *Linux Journal*, vol. 1999, no. 65es, pp. 15–es, 1999.
[20] W. Von Hagen, *Ubuntu Linux Bible*, vol. 352, John Wiley & Sons, 2007.
[21] Docker https://www.docker.com, accessed on 1 May 2025.
[22] Apptainer https://apptainer.org/, accessed on 1 May 2025.
[23] Nvidia Multi-Instance GPU https://www.nvidia.com/en-us/technologies/multi-instance-gpu/, accessed on 1 May 2025.
[24] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand, et al., "GEANT4 a simulation toolkit," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 506, no. 3, pp. 250–303, 2003.