EPJ Web of Conferences **214**, 03034 (2019) https://doi.org/10.1051/epjconf/201921403034
CHEP 2018

OSG and GPUs: A tale of two use cases

Edgar Fajardo[1,*], *Mats* Rynge[2,**], *Derek* Weitzel[3,***], *Gonzalo* Merino[4], *David* Schultz[4], *Vladimir* Brik[4], and *Heath* Skarlupka[4]

[1]University of California San Diego, 9500 Gilman Drive, La Jolla, CA, 92093
[2]Information Science Institute, 4676 Admiralty Way #1001, Marina Del Rey, CA 90292
[3]University of Nebraska – Lincoln, 1400 R Street, Lincoln, NE 68588
[4]University of Wisconsin-Madison, WIPAC, 222 W Washington Ave. Suite 500, Madison, WI 53703

Abstract. With the increase of power and reduction of cost of GPU accelerated processors a corresponding interest in their uses in the scientific domain has spurred. OSG users are no different and they have shown an interest in accessing GPU resources via their usual workload infrastructures. Grid sites that have these kinds of resources also want to make them grid available. In this talk, we discuss the software and infrastructure challenges and limitations of the OSG implementations to make GPU's widely accessible over the grid. Two use cases are considered for this. First: IceCube, a big VO with a well-curated software stack taking advantage of GPUs with OpenCL. Second, a more general approach to supporting the grid use of industry and academia maintained machine learning libraries like Tensorflow, and Keras on the grid using Singularity.

1 Introduction

In the recent years three trends have sparked the sudden increase of Graphic Processing Unit (GPU) accelerators use in the scientific community. First the cost of GPU has constantly decreased, second there is an increasing availability of data (either simulated or experimental) and third a proliferation of third party (Industry and Academia outside physics) Deep Learning libraries that can greatly profit from the parallel computing power of GPU while hiding from the scientist the interactions with the GPU. This demand has led to computing sites to add GPU enabled resources to their clusters and to more scientific communities or Virtual Organizations (VO) to want to seamlessly access these resources as they do with any other computing resource available.

The Open Science Grid (OSG) [1] in conjunction with GlideinWMS [2] and HTCondor [3] have sought to provide such grid access that is transparent from sites and VO perspective, effective (the science can be done) and accounted. In section 2 we describe the technical components to achieve transparent access on the grid. Then we follow up with the general use cases of GPU usage on the OSG VO in Section 2.1 and then specific IceCube GPU use case in Section 2.2. In order for effective scientific usage of the different deep learning libraries we leverage the use of containers and Singularity in Section 3. However maintaining

*e-mail: emfajard@ucsd.edu
**e-mail: rynge@isi.edu
***e-mail: dweitzel@unl.edu

a set of containers for outside libraries brings its own set of challenges looking into the future, which we discuss in our final Section 4.

2 How to access GPU

Transparent access to heterogeneous computing resources across differently owned and operated computing centers has been one of the main problems of Grid Computing. In the last five years the pilot job systems implementing a pull-based late-binding scheduling have shown great success in this area; for example the CMS Global pool [4]. One of such pilot systems that support multiple VO's is GlideinWMS [5]. GlideinWMS can now add several GPU resources to a preexisting VO pools. The general principle of GlideinWMS is to leverage several HTCondor capabilities like the ability to submit jobs to Grid Computing Elements, Clouds and HPC Centers (via Bosco) to create a virtual pool of resources that grows and shrinks on demand.

In a generic GlideinWMS use case, a user that wants to use a CPU to run a workload would submit some jobs requesting a certain number of cores per job from a HTCondor Submit host (Schedd) then the GlideinWMS frontend queries the jobs in the queue and requests the factories to submit pilots to different resource points based on the jobs needs. Once these pilots start running they will contact back to an HTCondor Central Manager. The Central Manager has a component (Negotiator) that matches resources (pilots) and jobs.

For the GPU case we leverage the fact that HTCondor supports GPU access and part of the work presented here is the integration of this feature into GlideinWMS. The differences on the GPU case are as follows: a user submits a job and request a number of CPUs and GPUs. The frontend realizes that jobs are requesting GPUs and will only submit pilots to sites who advertise having such resources. Once a pilot lands on a node it will advertise the number of cores and bind to the number of GPUs it was configured to. Some additions to standard HTCondor were added to the pilot, one of which is to use the HTCondor GPU discovery tool to advertise the CUDA [6] and OpenCL [7] versions available on a node, the GPU model, and driver version (in case a job is sensitive to those). Finally modifications were made to the GRACC System [8] to account for both CPU and GPU hours consumed. Monitoring probes installed on each submit host parse the HTCondor history files of each completed job and calculate both the CPU and GPU usage and send the records via RabbitMQ to the GRACC server. Finally a GPU based dashboard was created to track historic GPU usage among sites and VOs.

2.1 OSG Use Case

The OSG VO is used for individual researchers and small collaborations which are too small and outside the scope of the large VOs. Because of the diverse user base and the different codes and requirements those users bring with them, it is difficult to narrow down the use case. In general, the OSG VO has seen a lot of interest from users, especially in the Machine Learning and the Geographic Information System communities. The challenge has been mostly on the supply side, and the chicken or the egg problem of having resources available to attract users. The VO has interacted with sites who want to make resources available, but at that point we have not had users ready. Similarly, we have had users requiring a number of resources which have not been available, and thus been disappointed by the capabilities provided at that particular point of time. This is not a problem specific to GPUs. The resources available to OSG VO are almost entirely opportunistic access to other VOs' resources. Availability swings of generic CPUs is not very noticeable for the individual users, but availability

of scarce resources like GPUs can be very noticeable. We expect that this problem will solve itself over time as GPUs become more and more common on the resources contributing to the OSG VO.

2.2 IceCube Use Case

The virtual organization that consumes the most GPU hours in the OSG realm is the IceCube collaboration. The IceCube detector [9] is located at the geographic South Pole and was completed at the end of 2010. It consists of 5160 optical sensors buried between 1450 and 2450 meters below the surface of the South Pole ice sheet and is designed to detect interactions of neutrinos of astrophysical origin. Eight years after detector completion, IceCube continues improving its uptime and taking lots of data. As the data set grows, it becomes essential to have an excellent understanding of experimental effects that are potential sources of systematic errors, which requires very large amounts of simulation. One of the experimental effects that is crucial to model to aim for precision measurements are the properties of the light propagation inside the instrumented ice. This is a difficult process to model, due to the optical complexity of the ice in the Antarctic glacier, but developments in IceCube's simulation over the last decade include a powerful approach for simulating direct propagation of photons [10] by using GPUs. This simulation module is much faster than a CPU-based implementation and, most importantly, is able to account for some of the ice properties such as the tilt of the ice layers [11] and the ice anisotropy [12] that are crucial for performing precision measurements and would be extremely difficult to implement with a table-based simulation that parametrizes the ice properties in finite volume bins [13].

To support this GPU-based simulation in production, IceCube uses a variety of software to setup up a clean environment and enable GPUs. IceCube leverages a pilot job system similar to OSG, with a global pool central server located in Madison, WI, that aggregates heterogeneous resources. For IceCube sites connected to Grid infrastructures in Canada, Europe or the US, IceCube uses those resources using regular GlideinWMS OSG-managed pilots. For other sites, IceCube developed a light weight HTCondor pilot job system called Pyglidein [14] that works similarly to GlideinWMS. Regardless of which pilot system is used, available resources are advertised in the global pool as standard HTCondor resources and IceCube jobs requesting GPUs eventually run on a GPU-enabled node independently of its location. The first thing every job does is load the IceCube software environment, located on CVMFS [15]. This does some basic GPU and driver detection, and helps enable OpenCL support. OpenCL is used by IceCube to avoid driver dependencies in compiled software when running across multiple computer clusters in different sites.

This system has been running for more than two years now, growing in size every year. It is not without its challenges, as there are multiple configurable parameters that can make the system unstable, or performing suboptimally, under some circumstances. 95% of the time things work perfectly, but as each new site gets added a new variation can cause problems and needs to be fixed. Overall, the advantages of providing a simple uniform access to heterogeneous GPU resources greatly compensates the operational hurdles. In order to overcome some of these problems, IceCube is considering increasing the use of containers as others in OSG are doing, as described in the next section.

3 Singularity Containers in OSG

In the grid heterogeneous world a strict need to decouple the application operating system, libraries and namespace from the compute nodes has spurred the usage of containers. The

Open Science Grid uses a combination of CVMFS [15] and Singularity [16] to run applications inside containers that the user specifies [17]. OSG maintains some catch all singularity images out of Docker containers. Then if a users needs something that it is not inside those images then they can build on top of those centrally maintained and then request them to be added to the OSG singularity catalog. Once in there an OSG application downloads the Docker container and extracts the image into a file system based layout. This is then distributed to the compute nodes around the world using CVMFS.

In order for OSG to support several Deep Learning libraries like Tensorflow [18], Keras [19] that do not have "out of the box" installation on Red Hat system (which are the ones OSG supports) Singularity was a must. We created GPU specific images with Keras and Tensorflow with GPU support that are totally transparent to the scientist who just need to select them in their jobs.

4 Challenges moving forward

One of the biggest challenges supporting the new deep learning libraries is the explosion of the dependency tree. Each library has specific requirements for supported CUDA library versions. Moreover the CUDA library versions on a Singularity image cannot be "sufficiently" far way from the ones in the bare metal in order for the GPU acceleration to take place. In the usual grid environment the first ones are chosen by users or VO software managers and the latter ones by site administrators. In future versions of Singularity this should be addressed and CUDA libraries will bind from whatever the host has. Then, the challenge will be to make sure that all OSG computing resources have the latest version of Singularity.

5 Conclusions

Open Science Grid has integrated into its main pilot job system the capabilities to help scientific communities transparently access heterogeneous GPU resources geographically distant while correctly accounting for site and VO usage. New Deep Learning technologies and the eagerness of the HEP community to use them into their scientific workflows produced an opportunity to create multi VO support that OSG leveraged through Singularity. However it also created a future looking challenge in the form of tighter and deeper dependency trees.

References

[1] R. Pordes, D. Petravick, B. Kramer, D. Olson, M. Livny, A. Roy, P. Avery, K. Blackburn, T. Wenaus, F. Würthwein et al., Journal of Physics: Conference Series **78**, 012057 (2007)

[2] I. Sfiligoi, D.C. Bradley, B. Holzman, P. Mhashilkar, S. Padhi, F. Wurthwein, *The Pilot Way to Grid Resources Using glideinWMS*, in *2009 WRI World Congress on Computer Science and Information Engineering* (2009), Vol. 2, pp. 428–432

[3] D. Thain, T. Tannenbaum, M. Livny, Concurrency - Practice and Experience **17**, 323 (2005)

[4] J. Balcas, S. Belforte, B. Bockelman, D. Colling, O. Gutsche, D. Hufnagel, F. Khan, K. Larson, J. Letts, M. Mascheroni et al., Journal of Physics: Conference Series **664**, 062031 (2015)

[5] P. Mhashilkar, M. Mambelli, I. Sfiligoi, B. Holzman, K. Larson, J. Dost, ddbox, M. Mascheroni, J. Weigand, L. Lobato et al., *glideinwms/glideinwms: v3.4* (2018), `https://doi.org/10.5281/zenodo.1309679`

[6] J. Nickolls, I. Buck, M. Garland, K. Skadron, Queue **6**, 40 (2008)
[7] J.E. Stone, D. Gohara, G. Shi, IEEE Des. Test **12**, 66 (2010)
[8] K. Retzke, D. Weitzel, S. Bhat, T. Levshina, B. Bockelman, B. Jayatilaka, C. Sehgal, R. Quick, F. Wuerthwein, Journal of Physics: Conference Series **898**, 092044 (2017)
[9] M.G. Aartsen et al. (IceCube), JINST **12**, P03012 (2017), 1612.05093
[10] D. Chirkin, Proceedings of the 32nd International Cosmic Ray Conference, ICRC 2011 **4**, 161 (2011)
[11] M. Aartsen et al. (IceCube Collaboration), Nuclear Inst. and Meth. in Phys. Res., A **A711**, 73 (2013)
[12] D. Chirkin (ICECUBE), *Evidence of optical anisotropy of the South Pole ice*, in *Proceedings, 33rd International Cosmic Ray Conference (ICRC2013): Rio de Janeiro, Brazil, July 2-9* (2013), p. 0580, http://www.cbpf.br/%7Eicrc2013/papers/icrc2013-0580.pdf
[13] D. Chirkin, Nuclear Inst. and Meth. in Phys. Res., A **725**, 141 (2013)
[14] D. Schultz, B. Riedel, H. Skarlupka, J. van Santen, V. Brik, T. Ehrhardt, renereimann, G. Merino, P. Eller, C. Kopper et al., *Wipacrepo/pyglidein: 1.1.9* (2018), https://doi.org/10.5281/zenodo.1469023
[15] J. Blomer, B. Bockelman, P. Buncic, B. Couturier, D. Dykstra, G. Ganis, M. Giffels, H. Nikola, S. Heikkila, R. Meusel et al., *The cernvm file system: v2.5.0* (2018), https://doi.org/10.5281/zenodo.1254930
[16] G.M. Kurtzer, *Singularity 2.5.2 - Linux application and environment containers for science* (2018), https://doi.org/10.5281/zenodo.1308868
[17] M. Rynge, B.P. Bockelman, D. Weitzel, jthiltges, R. Jones, T. Downes, E. Fajardo, D. Blyth, H. Skarlupka, B. Riedel et al., *opensciencegrid/cvmfs-singularity-sync: Singularity-Sync First Release* (2018), https://doi.org/10.5281/zenodo.1469012
[18] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard et al., *TensorFlow: A System for Large-scale Machine Learning*, in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation* (USENIX Association, 2016), OSDI'16, pp. 265–283, ISBN 978-1-931971-33-1, http://dl.acm.org/citation.cfm?id=3026877.3026899
[19] F. Chollet et al., *Keras*, https://keras.io (2015)