EPJ Web of Conferences **214**, 03004 (2019) https://doi.org/10.1051/epjconf/201921403004
CHEP 2018

Producing Madgraph5_aMC@NLO gridpacks and using TensorFlow GPU resources in the CMS HTCondor Global Pool

Brian Paul Bockelman[1], *Edgar* Fajardo Hernandez[2], *Diego* Davila Foyo[3], *Kenyi* Hurtado Anampa[4], *Farrukh* Aftab Khan[5], *Krista* Larson[5], *James* Letts[2], *Marco* Mascheroni[5], *David* Mason[5], *Antonio* Perez-Calero Yzquierdo[6,7], and *Todor Trendafilovz* Ivanov[8]

[1]University of Nebraska-Lincoln, Lincoln, NE, USA
[2]University of California San Diego, La Jolla, CA, USA
[3]Autonomous University of Puebla, Puebla, Mexico
[4]University of Notre Dame, Notre Dame, IN, USA
[5]Fermi National Accelerator Laboratory, Batavia, IL, USA
[6]Port d'Informació Científica, Barcelona, Spain
[7]Centro de Investigaciones Energéticas Medioambientales y Tecnológicas, Madrid, Spain
[8]University of Sofia, Sofia, Bulgaria

Abstract. The CMS experiment has an HTCondor Global Pool, composed of more than 200K CPU cores available for Monte Carlo production and the analysis of data. The submission of user jobs to this pool is handled by either CRAB, the standard workflow management tool used by CMS users to submit analysis jobs requiring event processing of large amounts of data, or by CMS Connect, a service focused on final stage condor-like analysis jobs and applications that already have a workflow job manager in place. The latest scenario can bring cases in which workflows need further adjustments in order to efficiently work in a globally distributed pool of resources. For instance, the generation of matrix elements for high energy physics processes via Madgraph5_aMC@NLO and the usage of tools not (yet) fully supported by the CMS software, such as TensorFlow with GPU support, are tasks with particular requirements. A special adaption, either at the pool factory level (advertising GPU resources) or at the execute level (e.g: to handle special parameters that describe certain needs for the remote execute nodes during submission) is needed in order to adequately work in the CMS global pool. This contribution describes the challenges and efforts performed towards adapting such workflows so they can properly profit from the Global Pool via CMS Connect.

1 The submission system

While submission of CMS [1] user jobs to the Global Pool [2] is mostly managed by CRAB [3], the standard analysis workflow management tool, the generation of matrix elements for high energy physics processes via Madgraph5_aMC@NLO [4] and the usage of machine learning tools with GPU resources are independent use-cases that require special adaptation in order to take advantage of the Global Pool resources.

CMS Connect [5] provides a service where users can submit HTCondor jobs to the CMS Global Pool (a global HTCondor pool provisioned by GlideinWMS) with a submission interface similar to those provided by analysis facilities physicists are familiar with, such as the CERN Analysis Facility [6]. This service complements CRAB, as illustrated in Figure 1, dealing with a different set of analysis workflows, such as Madgraph gridpacks and the use of GPU resources with TensorFlow [7] jobs. The sections below describe the challenges and efforts performed towards adapting these two different workflow types in order to properly work with CMS Connect and the Global Pool resources.

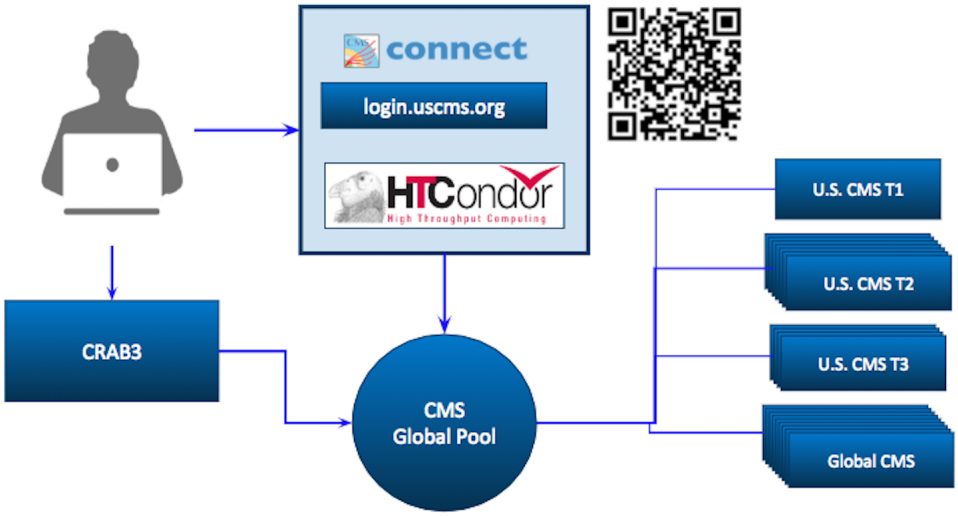

Figure 1. CMS analysis submission services.

2 Generating Madgraph5_aMC@NLO gridpacks

Monte Carlo (MC) event generators, such as Madgraph5_aMC@NLO, are used to model physics processes in the high energy physics field. The information generated, including for example, the computation of the differential cross sections and final state particles involved in these processes, is stored in a compressed tarball package called gridpack. This is one of the very first steps in the simulation chain that produces the MC samples used in physics analyses, as shown in Figure 2.

A generator's package that automates the production of these gridpacks by setting up the CMS software environment and providing Madgraph5_aMC@NLO is used in the experiment. From a computational point of view, this can be achieved in two different ways, by using all cores available on a single machine, or by having Madgraph5_aMC@NLO create and submit multiple jobs to a batch manager (e.g.: HTCondor [8]).

The second method is preferred for complex processes, due to the high demand of CPU power. Furthermore, while local resources, such as the CERN Analysis Facility or local Tier 3s, where users have login access to the resource batch submission system (in contrast to grid-enabled resources, where a grid middleware manages the submission to the batch

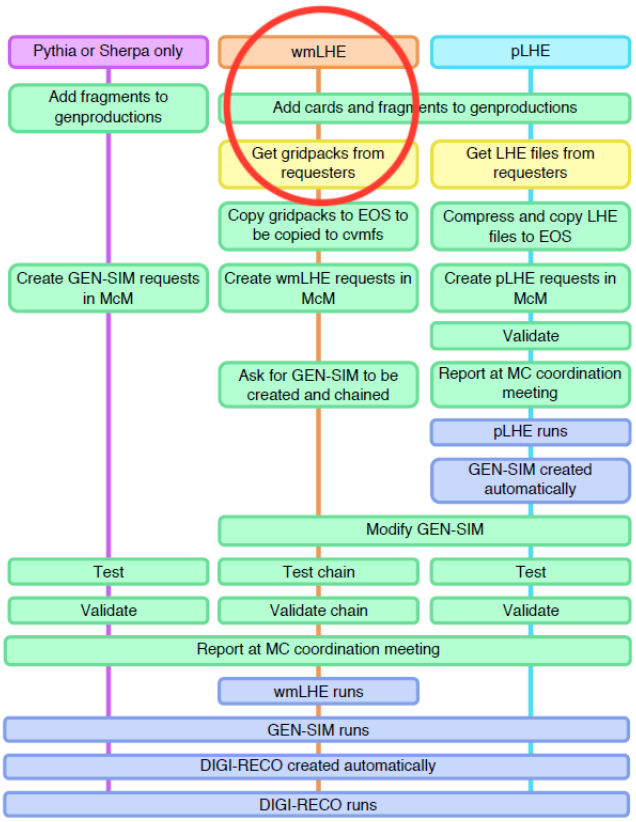

Figure 2. Monte Carlo request submission chain. Red circle in the diagram shows where gridpacks are requested in this chain.

system), can in principle be used for this goal, not all CMS users have access to the same local resources and the submission methods can vary, depending on the batch submission manager available. Also, long term running jobs might need special requirements, such as renewing AFS tokens periodically. The submission of jobs to the Global Pool offers many advantages:

- Higher computing power distributed across all grid site resources available in the CMS Global Pool.
- Better accounting and monitoring of jobs.
- A central submission node for all CMS users with a grid proxy certificate registered in the CMS Virtual Organization.
- A single batch submission manager (HTCondor) to deal with.

However, the Global Pool infrastructure expects certain parameters that characterize the job that are not set by default, such as the maximum executable wall time estimated, or a list of the CMS sites to submit the jobs to. Additionally, jobs that were not able to finish running because of an error that might require further action and are put on "hold" state in the system

(meaning, these jobs will not match to any resource until they are released) are treated as a general failure in Madgraph5_aMC@NLO, aborting the whole submission, but transient errors leading to held jobs are not uncommon when submitting to several different sites globally. The cluster manager in Madgraph5_aMC@NLO was adapted in order to account for these factors.

For instance, a dynamic adjustment of the requested maximum wall time per job is performed, as well as specifying the the remote sites for submission through environment variables in the system (while in most cases, matching to all sites is desired, selecting particular sites can be especially useful to e.g: exclude sites known to have transient issues at the time, test submission of jobs with special dependencies not yet distributed via the CernVM File System [9] to a specific site, etc). The cluster manager in Madgraph5_aMC@NLO was modified to use the HTCondor python bindings in order to check for the status of the jobs and release held jobs with common transient errors for retrial. Also, the environment in the worker nodes were adjusted in order to propagate library dependency paths that are lost when using Singularity [10] containers (the default behavior for remote resources in the Global Pool). Figure 3 shows a diagram with the changes described above. Additionally, gridpack jobs set special HTCondor classads that are later used to track the activity of each on CMS monitoring dashboards. For example, Figure 4 shows the gridpack activity in the Global Pool divided by name. The name for each gridpack was stored as an HTCondor classad that is later used at the monitoring side in order to make this classification.

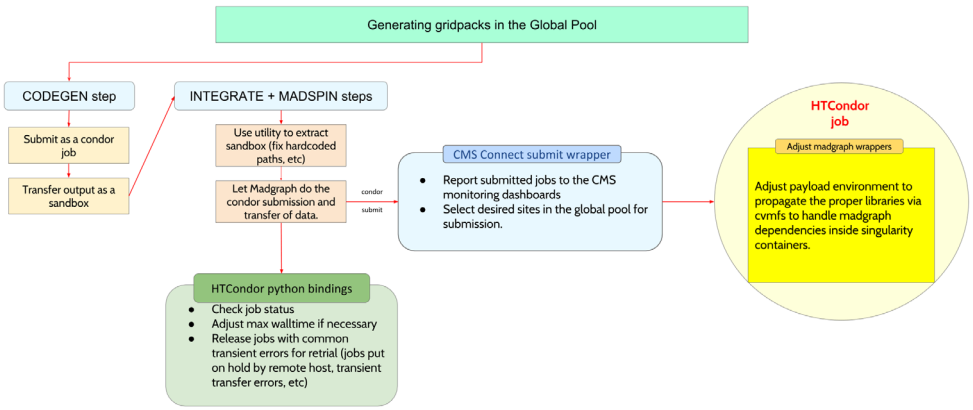

Figure 3. Adaptations applied to Madgraph5_aMC@NLO in order to make it compatible with the Global Pool infrastructure.

3 Deep learning and GPU resources

Machine Learning algorithms, such as boosted decision trees, random forest or artificial neural networks, have been successfully used within the high energy physics field for decades, but the rise in terms of demand of GPU resources started just a few years ago, with the training of deep neural networks, a subset of Machine Learning inspired in artificial neural networks (see Figure 5[1]).

[1]Source: NVIDIA

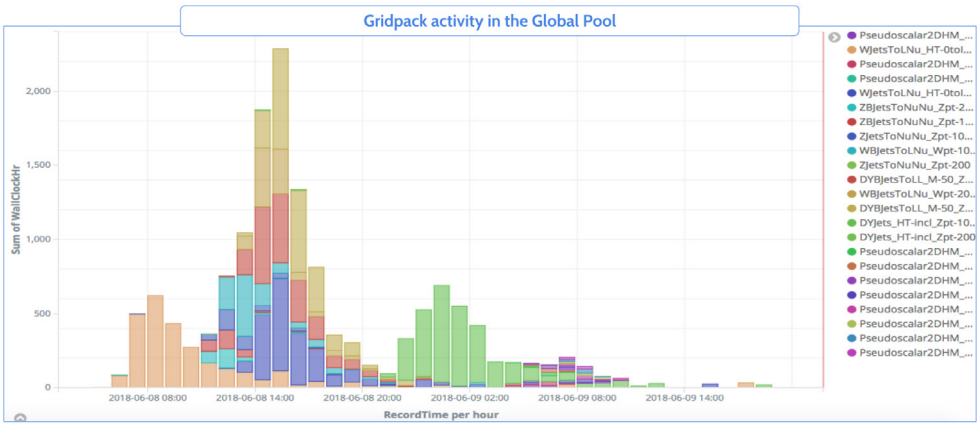

Figure 4. Gridpack production activity in the Global Pool during a day.

The availability of thousands of cores and a faster bandwidth to memory than conventional CPU resources are some of the main features in GPU resources. On the other hand, the low memory, low clock speeds and the fact data has to be transferred to the GPU card makes it challenging for several applications to take advantage of this. However, deep learning algorithms involve several matrix multiplication and other operations that can be massively parallelized and are not tied to high memory requirements or large data transfers, making it suitable for the GPU architecture.

The usage of deep learning algorithms in industry has lead to the development of powerful machine learning frameworks. For instance, Tensorflow [7] is an open source software library, originally developed by Google, providing strong support for machine learning and deep learning with several APIs for programming languages, such as Python and C++, two popular languages in the high energy physics community.

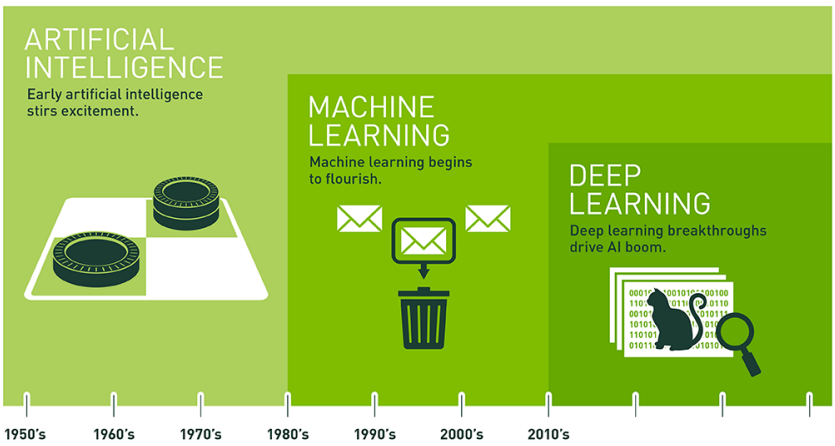

Figure 5. Artificial Intelligence algorithms over the years. Source: NVIDIA [11].

3.1 Using TensorFlow and GPU resources in the Global Pool

Even though many grid sites with GPU resources are available in the CMS Global Pool, meeting the software dependencies needed in order to use deep learning algorithms in them can become a challenge due to the lack of support of TensorFlow and other related frameworks in the base Operating Systems commonly used by CMS (Red Hat 6 and 7).

To help with this, CMS provides such dependencies through CVMFS, but its support is only available at the CPU level. The integration with GPU resources can easily fall into potential conflicts with GPU library dependencies. For instance, different TensorFlow versions can require specific versions of cuDNN (the Nvidia Deep Learning SDK) or the CUDA [12] toolkit to work.

To overcome this issue on a wider scale, Singularity containers [10, 13] based on Ubuntu with TensorFlow installed with GPU support are built, maintained and distributed via CVMFS by the Open Science Grid (OSG) [14, 15]. Figure 6 illustrates the different components involved in the provisioning of such software dependencies handled by the OSG. These Singularity images are used with the CMS resources in a transparent way, due to the full support for Singularity in the CMS Global Pool infrastructure.

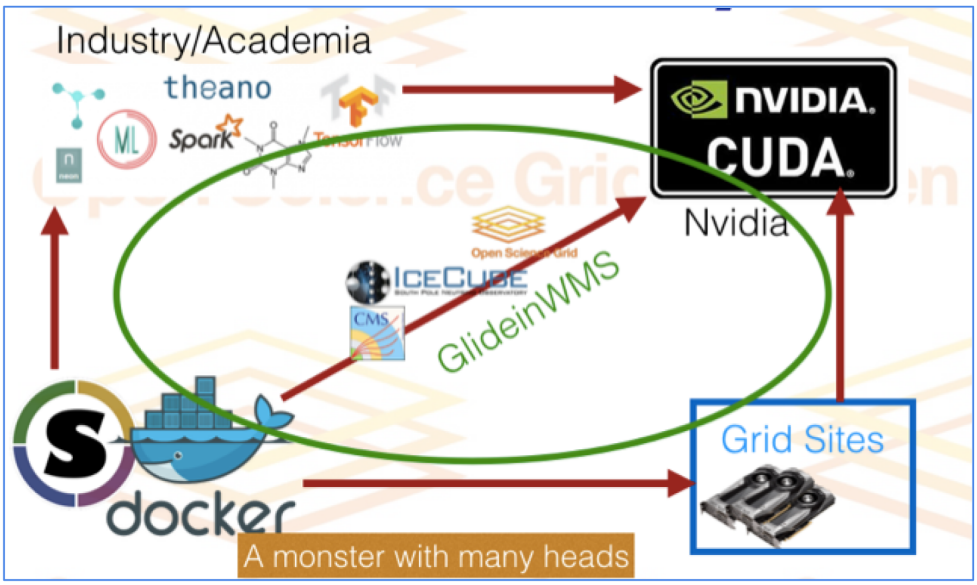

Figure 6. Schematic representing the link between GPU resources, CUDA drivers, machine learning frameworks and the experiments and organizations using this infrastructure.

4 Conclusions

This work has presented the needs and challenges present in two different types of workflows existing in the CMS collaboration, as well as the solutions provided in order to make them compatible with Global Pool resources via CMS Connect. While the production of gridpacks required adaptations at the Madgraph5_aMC@NLO code level, in order to set a group of job

variables expected by the Global Pool and to handle common transient errors for release and resubmission, the support of machine learning tools working with GPU resources is handled by integrating the solutions provided by the OSG in the infrastructure.

The Open Science Grid and University of Chicago Team not only provide and maintain the CI-Connect based submission hardware and networking infrastructure, but also support the service on different levels, including hosting, registration, software updates and maintenance notifications, within other tasks. We are especially grateful to Robert Gardner, Lincoln Bryant, Suchandra Thapa and Balamurugan Desinghu who assisted closely in the integration of the CI-Connect platform with the CMS Global Pool and continue supporting the service operability. The present work is partially funded under grants from the U.S Department of Energy, the National Science Foundation and Spain Ministry of Economy and Competitiveness grant FPA2013-48082-C2-1/2-R. The Port d' Informació Científica (PIC) is maintained through a collaboration between the Generalitat de Catalunya, CIEMAT, IFAE and the Universitat Autonoma de Barcelona.

References

[1] S Chatrchyan *et al.* (The CMS Collaboration), The CMS experiment at the CERN LHC JINST **3**, S08004 (2008)

[2] B Bockelman *et al.* Exploring GlideinWMS and HTCondor scalability frontiers for an expanding CMS Global Pool, to be published in these proceedings (CHEP2018)

[3] M Cinquilli, D Spiga, C Grandi, J M Hernandez, P Konstantinov, M Mascheroni, H Riahi and E Vaandering, CRAB: Establishing a new generation of services for distributed analysis at CMS, J. Phys. Conf. Ser., **396(3)**, 032026 (2012)

[4] J Alwall *et al.*, The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations, arXiv:1405.0301 [hep-ph] (2014)

[5] J Balcas *et al.*, CMS Connect, J. Phys. Conf. Ser., **898**, 082032 (2017)

[6] O Buchmueller *et al.*, The CMS CERN Analysis Facility (CAF), J. Phys. Conf. Ser., **219**, 052022 (2010)

[7] M Abadi *et al.*, TensorFlow: Large-scale machine learning on heterogeneous systems, arXiv:1603.04467 [cs.DC] (2016)

[8] D Thain, T Tannenbaum and M Livny, Condor and the Grid, *Grid Computing: Making The Global Infrastructure a Reality*, ISBN: 0-470-85319-0 (John Wiley & Sons, 2003)

[9] J Blomer, C Aguado-Sánchez, P Buncic and A Harutyunyan, Distributing LHC application software and conditions databases using the CernVM file system, J. Phys. Conf. Ser., **331**, 042003 (2011)

[10] GM Kurtzer, V Sochat, MW Bauer, Singularity: Scientific containers for mobility of compute, PLoS ONE 12(5): e0177459. (2017)

[11] NVIDIA Deep Learning: https://developer.nvidia.com/deep-learning

[12] J Nickolls, I Buck, M Garland, and K Skadron. Scalable Parallel Programming with CUDA. Queue **6**, 2 , 40-53 (2008)

[13] Singularity Software: https://doi.org/10.5281/zenodo.1308868

[14] E Fajardo Hernandez, OSG and GPUs: A tale of two use cases, to be published in these proceedings (CHEP2018)

[15] CVMFS-Singularity-Sync OSG scripts to maintain Singularity images in CVMFS: https://doi.org/10.5281/zenodo.1469012