Hardware Accelerated ATLAS Workloads on the WLCG

A. C. Forti[1], L. Heinrich[2], M. Guth[3]

1. University of Manchester, 2. CERN, 3. Albert Ludwigs Universitaet Freiburg

Summary

In recent years the usage of machine learning techniques within data-intensive sciences in general and high-energy physics in particular has rapidly increased, in part due to the availability of large datasets on which such algorithms can be trained as well as suitable hardware, such as graphics or tensor processing units which greatly accelerate the training and execution of such algorithms. Within the HEP domain, the development of these techniques has so far relied on resources external to the primary computing infrastructure of the WLCG. In this paper we present an integration of hardware-accelerated workloads into the Grid through the declaration of dedicated queues with access to hardware accelerators and the use of linux container images holding a modern data science software stack. A frequent use-case in the development of machine learning algorithms is the optimization of neural networks through the tuning of their hyper parameters. For this often a large range of network variations must be trained and compared, which for some optimization schemes can be performed in parallel -- a workload well suited for grid computing. An example of such a hyper-parameter scan on Grid resources for the case of Flavor Tagging within ATLAS is presented.

Enabling GPUs on the WLCG

The WLCG distributed resources have been built around the HTC (High Throughput Computing) paradigm that “focuses on the efficient execution of a large number of loosely-coupled tasks”. The environment has been traditionally uniform, not only at OS and software level but also at hardware level with all sites using Intel CPUs. Some sites have recently started to enable GPUs on grid queues and made them accessible to users for loosely coupled workflows.

Machine Learning

In HEP ML has been around for several years, until recently though it was confined to end user analysis mostly for signal to background discrimination. Recently however the increase in dataset sizes, the improved performance of algorithms, the availability of vast ML python libraries and jupyter notebooks made it easier testing and training machine learning algorithms in a variety of experiment data processing applications like reconstruction and real-time analysis to replace the most computing intensive traditional algorithms. This makes the case for making GPUs available on the grid on a bigger scale.

User containers

One of the advantages from the users point of view to use containers is that they can build the images containing all the software the application needs without having to rely either on the system administrators to install it or the experiment to distribute it. ATLAS has worked in the past few months to allow this type of users containers to run on the grid. ML jobs use custom software like python3, keras, tensorflow, jupyter. These jobs are a good candidates to run in standalone containers [1].

```
FROM tensorflow/tensorflow:latest-gpu-py3

## ensure locale is set during build
ENV LANG C.UTF-8

RUN pip install keras && \
    pip install uproot && \
    pip install jupyter && \
    pip install matplotlib && \
    pip install papermill && \
    mkdir /afs

RUN apt-get update && apt-get install -y nano

COPY . /btagging
WORKDIR /btagging
```

GPUs drivers and libraries

In addition to run on GPUs the container has to be able to access the devices and load the drivers. Singularity, the runtime of choice on the grid, can be configured to do this automatically. This works particularly well with nvidia cards and CUDA libraries, which, are the most popular hardware. At some sites that run on HPC like resources as well as configuring singularity the libraries have to be explicitly preloaded before the container can see them. To simplify the setup procedure we created some very simple test containers that can be used either by the system administrators or by us to verify the nodes are correctly configured and the application runs on the GPUs and not accidentally on the CPUs.

Queues setup and Brokering

The goal is to be able to submit a variety of GPU workloads passing job requirements to grid, HPC or cloud resources via Panda, the ATLAS WMS. Depending on how sites are configured they can have multi-GPU nodes and MPI enabled. This happens more typically at HPC. On the more traditional grid sites the resources are likely to be configured in a more straightforward way with 1 GPU per host or in any case configured to be accessible only by one job. With this setup we could broker jobs to GPU resources at two sites and run the HP scan payload without changes to the brokering system other than:

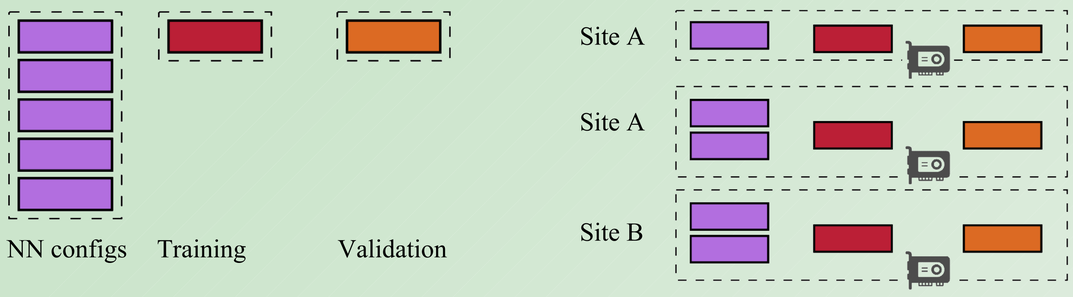
- Adding a selectable generic vendor “nvidia-gpu” label to select the queues. Jobs could run on any K20/.../VT100 GPUs that are accessible, there is no point in allowing users to select based on the model even though it is one of the parameters usually allowed. We will learn more by not doing that for the moment.
- Removing all the release tags to avoid standard jobs being brokered to GPU resources. This is important because sites with the current pilot setup per VO have no way of distinguishing which payload lands on which resource.

We currently have 2 grid queues setup in this way and we are working on 3 HPC resources.

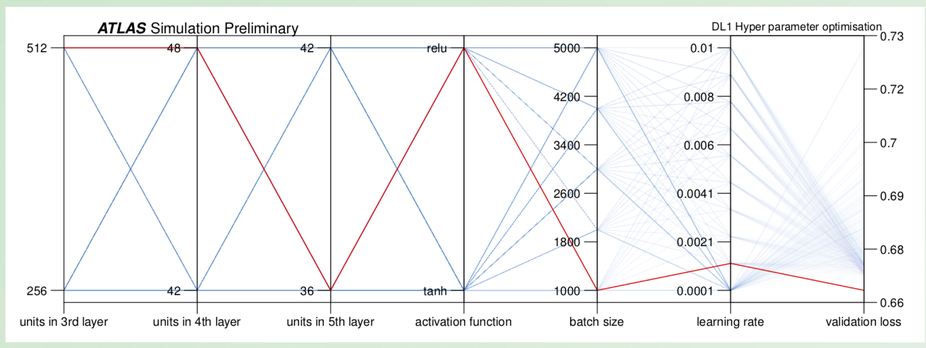
b-tagging Hyper Parameter optimization

Some workloads are more suited than others to run on HTC environment, in particular the HP(Hyper Parameter) scan is an embarrassingly parallel workload and can be split in several independent jobs each running on a GPU.

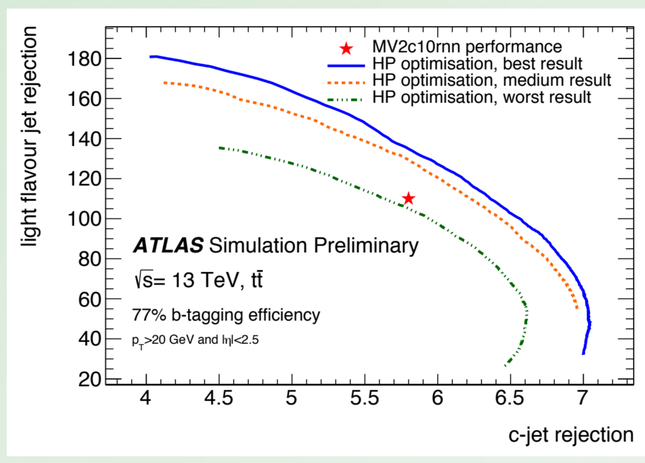
The optimisation presented here is setup to scan 800 combinations spanning 6 HP dimensions (3 layers, learning rate, batch size and activation functions). The workload has been split in 10 jobs each with 80 combinations. Each job run on the same training and validation data. The input files, small json files containing the configuration for each combination, were replicated to the sites with GPUs using rucio.



The output were also small json files containing the optmization values for the validation loss for each HP combination. The smaller the validation loss the better the combination.



Results were then verified by running again the training on best, medium and worst validation loss combinations of HPs to produce a ROC (Receiver operating characteristic) plot.



Conclusions

The use of GPUs in ATLAS and more in general in WLCG may increase due to the introduction of ML and resources coming online at sites, particularly at HPC centres, but not only. For loosely coupled workloads like the b-tagging HP scan presented here users can use them at standard grid sites already with minimal changes to their client commands. In addition there is ongoing work to run this payload on larger resources at 3 HPC sites and to implement a more sophisticated brokering to run workloads via the production system.

[1] https://indico.cern.ch/event/708041/contributions/3276174