Resources at CERN: a user experience

Maurizio Pierini
CERN

General remarks

This talk was supposed to happen during the iML workshop

- Prepared collecting feedback from students and post-docs working on my ERC projects in the last 4 years
- Eventually I could not give it, and nobody felt comfortable about coming here to say what I am about to say

The rest of this talk will be about

- A historical recollection
- A collection of different workflows that we followed in years

GPU for DL at CERN how it started

Back in 2015, Jean-Roch and I walked uphill to ask IT for some GPU to perform training at CERN

- We were already using some machine @Caltech
- We wanted to have something local, integrated with local resources (EOS, etc.)

It didn’t go very well

- Can’t you use CPUs?
- How do you know that GPUs are better?
- Prove us that you cannot do on CPUs what you do on GPUs?

Inner answer: “Did you guys look at nVidia stock values recently?”

Instead, we calmly negotiated 3 GTX1080, for the astronomical investment of 2400 CHF

Nobody at CERN can keep a secret, so after 6 months there were 50 users on those three GPUs (you can see evidence of that on some of the acknowledgements of the early DL works in our community)

GPU for DL at CERN how it is going

Very badly…

The main message

To seriously support the growth of Deep Learning research for LHC physics, CERN needs to invest on a cluster of at least a few hundred GPUs

Rumors say (since years) that this is coming but

- Did you see it? I didn’t
- Is it going to look like what we need?

Meanwhile, people managed to arrange something here and there and some central support came

The rest of this talk is a collection of these efforts

Hardware Infrastructures

GPU for NN training at CERN

We worked with several options across the years

- GPUs provided w/o software stack
  - TechLab gaming cards (the first GPUs@CERN, now decommissioned)
  - Condor GPUs: 8 V100 cards accessible via condor
  - lxplusGPU: lxplus nodes mounting T4 nvidia cards
- mPP private GPUs
  - Our own GPUs, similar to lxplusGPU but with limited access list
- Swan GPU server
- Kubeflow
  - Interactive training environment with workflows and jupyter notebook service

Our experience with private machines

- We were forced this way when mPP started, due to lack of alternatives
  - We had money to build a cluster back then, and nobody willing to host it so that the money was refused
- This is certainly NOT the way to go nowadays
  - It requires time to set it up and maintain it
  - It is a local duplication of lxplusGPU (same limitations as listed on previous slide)
  - As of today, the only use case is for very long trainings, for which no AFS/Kerberos workflow on local disk and exclusive GPU access can be set up

Our experience with lxplusGPU

- This is the most easy solution at this moment
  - One uses ssh for access, as for other workflows
  - One takes the software stack from cvmfs (e.g., lcg) or from containers/local installations
- Limitations we found (attached to inheritance from the classic CPU infrastucture)
  - No guarantee of exclusive access to a node
  - No access to multiple GPUs at once (problem for advanced work with big models)
  - Some issue with Singularity containers (which are what HPC sites ask us to develop to use their GPUs).
  - Memory-usage limited (jobs killed when threshold exceeded)
  - Need of some ssh-keep-alive setup, that has to be custom installed (e.g. screen + keytabs + custom scripts, not a friendly entry point and quite annoying)
- Conclusions
  - For us, this was the winner for small projects so far
  - It removed our need to buy workstations with GPUs, since we use these the way we used those
  - The main concern here is on how to move to the next step and make the service more AI friendly
    - Multiple GPUs
    - Interactive access
    - Exclusive access to GPUs
    - Token issue

Our experience with Condor GPUs

- Certainly the principle works, as much as lxplusGPU
  - Software from lcg software stack, containers or private AFS installation
  - Jobs go on condor like traditional CPU jobs
- But the “cluster” was ridiculously small when we tried (8 V100 GPUs)
- GPUs don’t talk to each other (limitation for large-scale training)
- Too many users for too few GPUs
  - One ended up waiting forever in the queue
  - We abandoned this possibility very early, and stayed with our private resources

Our experience at clusters outside of CERN

- We used different external sites
  - CSCS, Summit, Marconi (ongoing) and Flatiron
  - Ups & downs depending on local support
- So far, Flatiron cluster at Simons Foundation and Marconi in Bologna are the dream solution
  - Has **HUNDREDS** of very powerful NVIDIA V100 and A100 GPUs
  - Based on **SLURM**
  - Can see cvmfs
  - Work very well with Singularity containers
  - User submits a job and specifies the time limit to run the training (max time limit is 7 days)
  - User can also request an interactive node for development purposes where one can login with **ssh**
  - **Exclusive multiple GPU access** (easy to access up to four GPUs at once), with very high amount of RAM
  - Disk access is also very fast and without errors (better than EOS and AFS, which are built for other purposes)
  - No need to care for tokens to run trainings longer than 24 hours as is needed for Kerberos
- **WHY CAN'T WE HAVE THIS AT CERN?**

Our experience with Kubeflow

- Integration with other CERN services
  - People use CERN computing infrastructure (Swan, EOS, condor, etc) while Kuberflow comes with Google solutions to many of the issues the CERN tools address. It seems that moving to Kubeflow implies buying much more than Kubeflow
  - Not clear how many GPUs will be available on Kubeflow eventually. Now access limited to one or two GPUs per (few) users. **How will this scale to hundreds of users?**
  - Not clear how development of pipeline beyond Google-provided solutions will happen. We do custom things for custom problems, and we need the possibility to customize
  - Software stack not up to date (e.g., old TF and Pytorch)
  - Again, token issue
- Bottom line:
  - Idea is interesting and promising
  - (imo) Service is not (yet) production ready. Not sure if this is a delivered service or R&D
  - Not clear (to us) the long-term transition to large-scale service and how many resources IT plans to put on it (people and hardware)
- Conclusion:
  - We gave it a try but we don’t use it to train models regularly, waiting for substantial development (which might be ongoing)
  - We are not convinced of the whole infrastructure (we want something like Flatiron, and this is not that)

Our experience with Swan

- Swan is way more than a ML training platform
  - It's a data analysis facility, designed when DL training was not a thing for HEP
  - It's main flow is on physics analysis, ROOT integration etc.
  - This is why it was for long running just on CPUs (more than enough for bread & butter ML, e.g., BDTs)
- Work happened to allow DL training
  - GPU server now available, relying on lcg stack.
  - Integrated to Spark server
  - …
- Questions/concerns:
  - What is the plan in terms of hardware support in the future: how many GPUs one will eventually see from Swan? **We are talking about supporting a community of hundreds of people**
  - Multiple GPU access for large-scale training (several tens of GPUs at once)?
  - Integration to Kubeflow? A lot of overlap there but often different directions taken (e.g., EOS integration)
  - Long trainings are an issue: now relies on Spark, which most of us are unfamiliar with (and those who are see it as something that DL community is drifting away from https://towardsdatascience.com/stop-using-spark-for-ml-59496927ef93)
- Conclusion:
  - Potentially interesting, particularly for interactive work, for a newcomer to DL training at CERN
  - Certainly the most user-friendly entry point (people use it already in other context)
  - We don't use it much: GPU support came late in the game, so our team was already otherwise set for small-case trainings

Software Stack

Our experience with local installation

AFS

- Provided one has space, a local installation (via anaconda/miniconda) is an effective solution
- Highly customizable, seen across the entire CERN infrastructure

But AFS is going to be dismissed

EOS instead was more problematic

- Too many small files to handle in the installation, not ideal for EOS
- This might have been solved in the meanwhile. Seeing the problem, we used AFS instead

Our experience with lcg

- No need for local installations now that lcg offers a GPU stack
  - Most of the packages under use are there
  - Very efficient service to integrate new packages, based on jira tickets
    - We obtain support for python geometric in 48h
- At the moment, the easy-to-access solution on most of CERN machines

```
source /cvmfs/sft.cern.ch/lcg/views/LCG_100cuda/x86_64-centos7-gcc8-opt/setup.sh
```

Our experience with singularity containers

- In more advanced applications, we tend to use containers (e.g., when working at CERN and outside CERN)
- Pros
  - Easy-to-setup container with all the required software packages
  - Very useful for complex workflows with lots of software requirements. That is frequently the case for HEP, such as for CMSSW, Geant4 or root.
  - You can base it on OS such as Ubuntu which is most widely used in the open source community
  - Within the scope of the container, packages can be installed which generally require root privileges
  - Supports GPUs
  - You can upload the code on dockerhub so anyone can freely use the container
- Cons
  - We need a machine with root access to build the container (but not to run it)
  - At lxplus machines, the performance is very unreliable. Sometimes, it takes ~30+ minutes to run the container (EOS issue most likely)
  - Compatibility with singularity containers at CERN not established. Effort needed there

Conclusions

Since JR and I walked uphill, many steps were taken to improve the user experience with training DL models on GPUs @CERN

Still, we are far from an ideal solution

- A lot of fragmented efforts (lxplusGPU, Condor, KubeFlow, Swan)
- Serious limitations: multiple GPU access, relatively short token duration, custom software stack support

We have successful stories to take as inspiration

- It is our opinion that something like the Flatiron/Marconi clusters would be what we would need here
- Certainly, it has to come with an easy-access terminal solution more than multiple GUIs
- Whatever are the solutions, there should be one BIG pool of GPUs that these solutions pull from