Cloud infrastructure @ CERN

By Laurens Ledeganck & Xander Dewulf

Introduction

Laurens Ledeganck

Laakdal (province of Antwerp)

Xander Dewulf

Heers (province of Limburg)

From collision to analysis

1. Tremendous amount of data
2. Trigger (99% of data is deleted)
3. 1 out of 1’000’000 collisions saved
4. Data = stored in the data centre + shared with institutions around the world

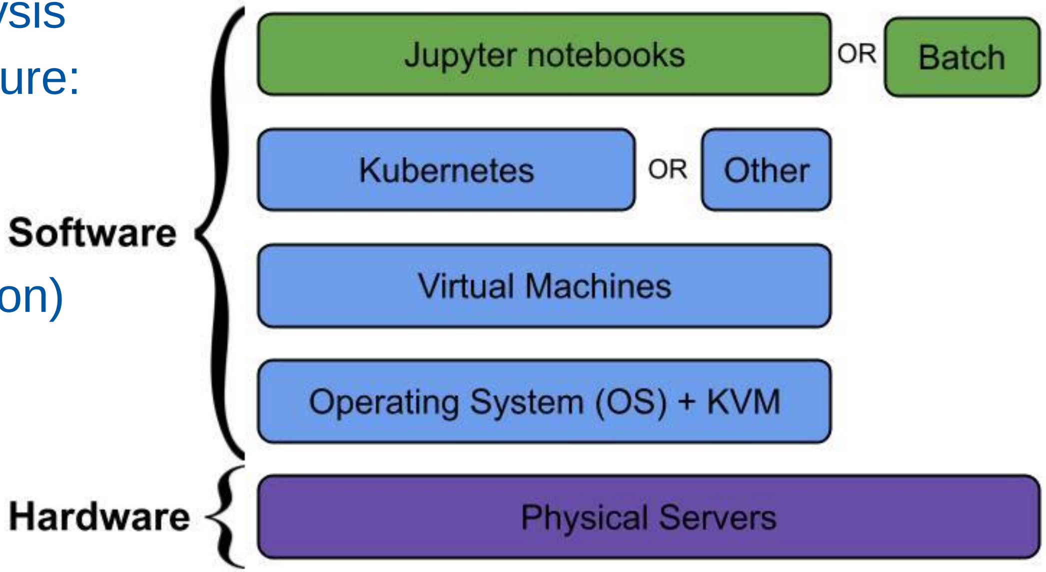
The Cloud: what?

- Sharing resources
- Connecting storage, servers and users
- Easy to use
- @ CERN: data from LHC (over 200 petabytes)
- Built in such a way that makes access to the data easy (for physicists)

CERN

The Cloud: how?

- Cloud is built in certain ‘layers’
- Physicists => top layer for analysis
- Layers below are used to structure:



- Public clouds, e.g. AWS (Amazon)
- Private clouds, e.g. CERN

Virtual machines

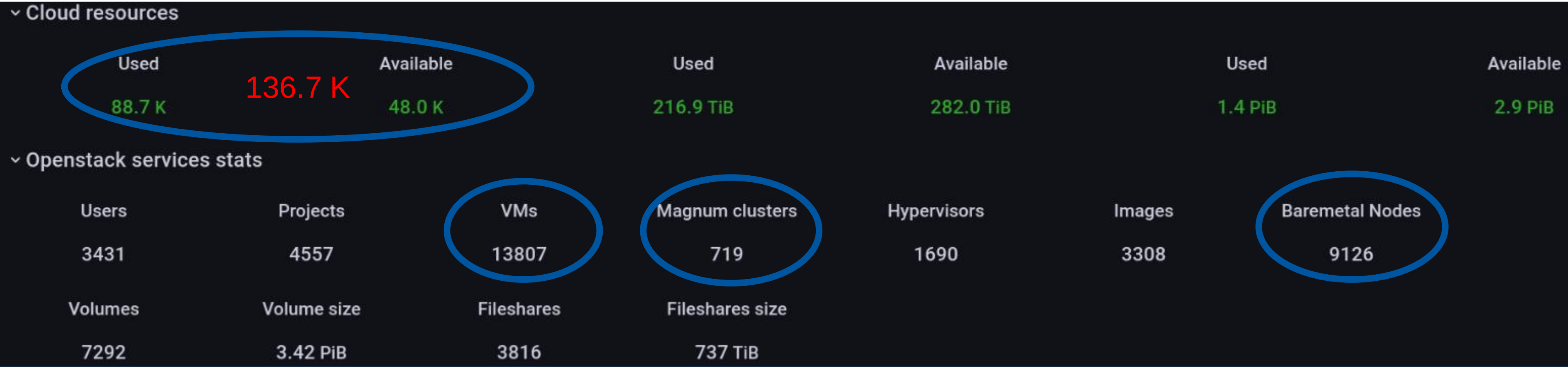
- Divide servers into smaller parts
- Allows you to work with another system on your pc
- Fair share of hardware
- Pros:
  - Fair sharing
  - Isolation
  - Ease of development
- Cons:
  - Small performance penalty (4%)

```
[lledegan@lxplus8s06 ~]$ openstack image list
| 214e130a-aecb-485a-89cb-3b6b85fd87b0 | CC7 - x86_64 [2021-11-01]                  | active |
...
[lledegan@lxplus8s06 ~]$ openstack flavor list
| ID    | Name      |  RAM | Disk | Ephemeral | VCPUs | Is Public |
| 12076 | m2.large  | 7500 |   40 |         0 |     4 | True      |
| 17895 | m2.small  | 1875 |   10 |         0 |     1 | True      |
| 38242 | m2.medium | 3750 |   20 |         0 |     2 | True      |
[lledegan@lxplus8s06 ~]$ openstack server create --image "CC7 - x86_64 [2021-11-01]" --flavor
"m2.small" --key-name lledegan-lxplus --property cern-waitdns=false VM-example
[lledegan@lxplus8s06 ~]$ openstack server list
| 5f40e28a-e01e-4a3b-8743-43f57fd863dd | VM-example                                 | ACTIVE |
CERN_NETWORK=188.185.90.42, 2001:1458:d00:41::100:3e5   | CC7 - x86_64 [2021-11-01] | m2.small  |
...
```

The CERN Cloud in numbers

These are some of the data centre's impressive statistics!

For reference: TiB = 2^{40} B, PiB = 2^{50} B



GPU’s

mnist-kale-katib.ipynb | Terminal 2

```
@gpu1-0:~$ nvidia-smi
Thu Oct 28 13:20:52 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.64       Driver Version: 440.64       CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100S-PCI...  Off  | 00000000:00:07.0 Off |                    0 |
| N/A   38C    P0    38W / 250W |  31459MiB / 32510MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

Examples

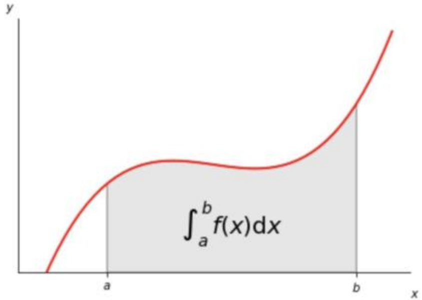
SWAN

- Service for Web-based ANalysis
- Plot functions
- Calculate integrals

```
ax.spines.right.set_visible(False)
ax.spines.top.set_visible(False)
ax.xaxis.set_ticks_position('bottom')

ax.set_xticks((a, b))
ax.set_xticklabels(('$a$', '$b$'))
ax.set_yticks([])

plt.show()
```



Minima and roots of a function

Demos finding minima and roots of a function.

Define the function

#

Find minima

#

```
Global minima found [-1.30641113]
Local minimum found 3.8374671194983834
```

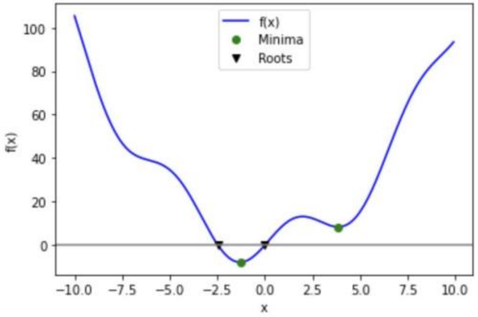
Root finding

#

```
First root found [0.]
Second root found [-2.47948183]
```

Plot function, minima, and roots

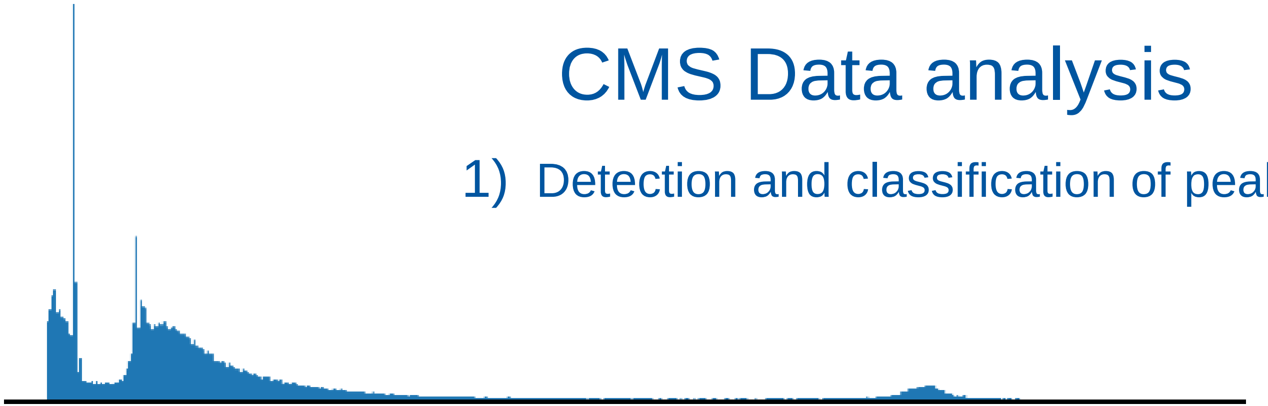
#



CMS Data analysis

- Data = public (after short period)
  - Avoid military use
  - https://opendata.cern.ch
- Analysis on laptop: 2 years
- Analysis in cloud: 5 minutes

=> 2 examples



CMS Data analysis

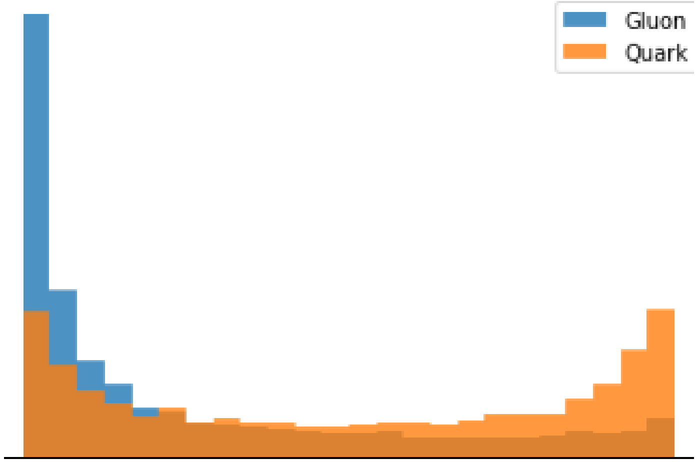
1) Detection and classification of peaks

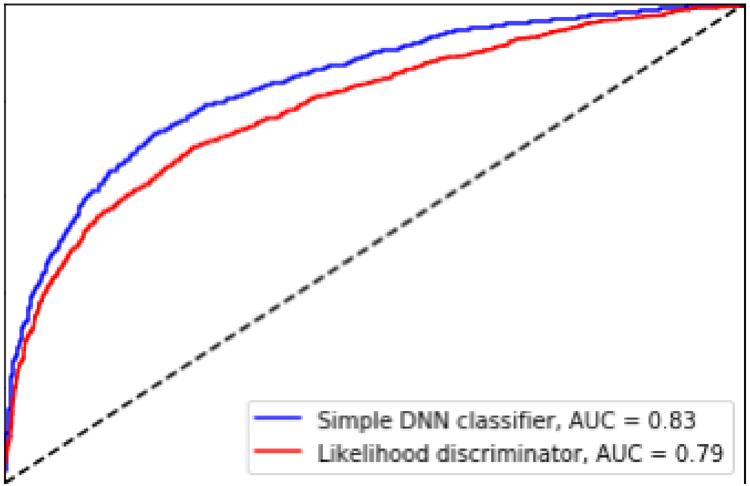
Particle	Tag	Mass	Particle	With		Composition	
η	0		0.548	η	0.00131		complicated
ω (782)	1	0.783	ω (782)	8.49000		complicated	
ρ (770)	2	0.775	ρ (770)	150.00000		complicated	
Kº	3		0.498	Kº	NaN		ds^-
Dº	4	1.860	Dº	NaN		$c\bar{u}$	
...							

These are the peaks we get:
[0.545 0.77   1.008 1.222 2.462 3.106 3.716 5.323 6.748 9.454 19.516 56.523 91.139]

CMS Data analysis

2) Identification of QCD-jets with machine learning





Links and references

CERN-cloud documentation: https://clouddocs.web.cern.ch/
Docker: https://www.docker.com/
Tutorial for Jupyter Notebooks: https://jupyter-tutorial.readthedocs.io/en/latest/first-steps/install.html
Python course: https://www.coursera.org/specializations/python#courses
Scipy Lectures: https://scipy-lectures.org/
Matplotlib tutorial: https://matplotlib.org/stable/tutorials/index.html

CERN training: https://clouddocs.web.cern.ch/containers/training.html#external
Aviator application: https://aviator.web.cern.ch/aviator/#/

Thank you!

Many thanks to Spyros Trigazis, José Castro León and Daniel Rickard Holmberg

Thank you!

… and also to:

François Briard, Feza Tankut, Marguerita Boselli and all the other coördinators

Cédric Vanhoolandt and Marijke Keupers

All the other students