CMS e/g Level-1 Trigger: Parallelization and scheduling of a python-based analysis framework using Dask

Gintautas Svedas

Supervisor: Gianluca Cerminara

The contents of the talk

1

The project, explained

2

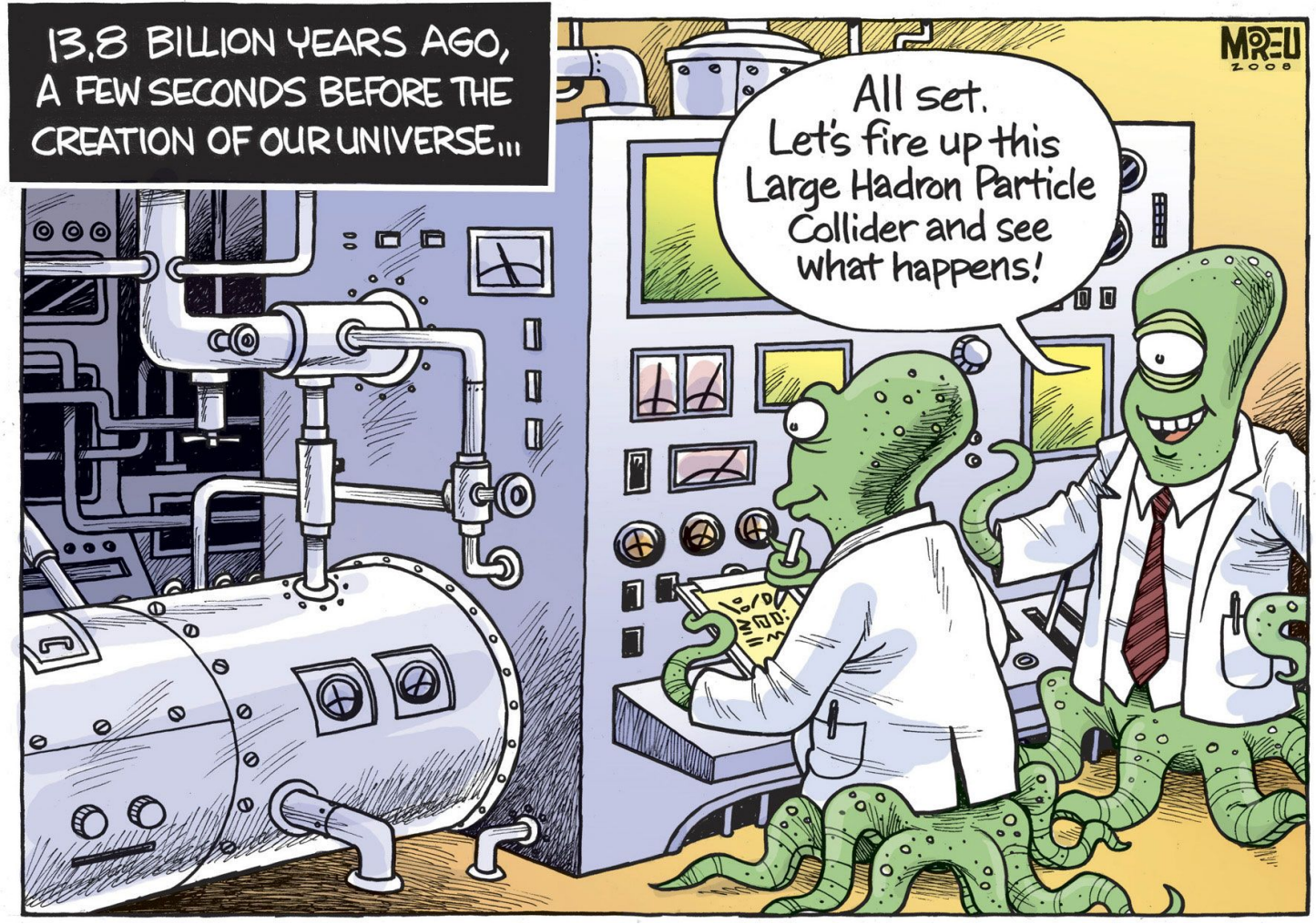
Work done & challenges encountered

3

Future implications

About me

- I am a recent bachelor's degree graduate in information technology from Vilnius University.
- Thesis: “Mystere: The Integration of Detective Elements in a First-Person 3D Puzzle Survival Game.”
- I have around 3 years of job experience working at banks, consultancies, and small companies.
- I came for a 3-month Erasmus+ internship (February 1st–April 30th).



This cartoon is pinned on the wall of the theory common room at CERN. Image credit: Mike Moreu

The ntuple-analysis package

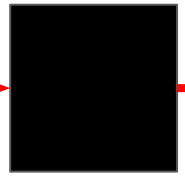
cerminar/**ntuple-analysis**

4 Contributors | 0 Issues | 3 Stars | 2 Forks

1. PYTHON framework for the analysis of ROOT TTree data using uproot for the IO and awkward-array for the columnar data analysis.
2. The tool was originally developed for the analysis of L1T ntuples for Phase-2 e/g, but it should work with any kind of flat ntuples.
3. Can be run on LXPLUS service

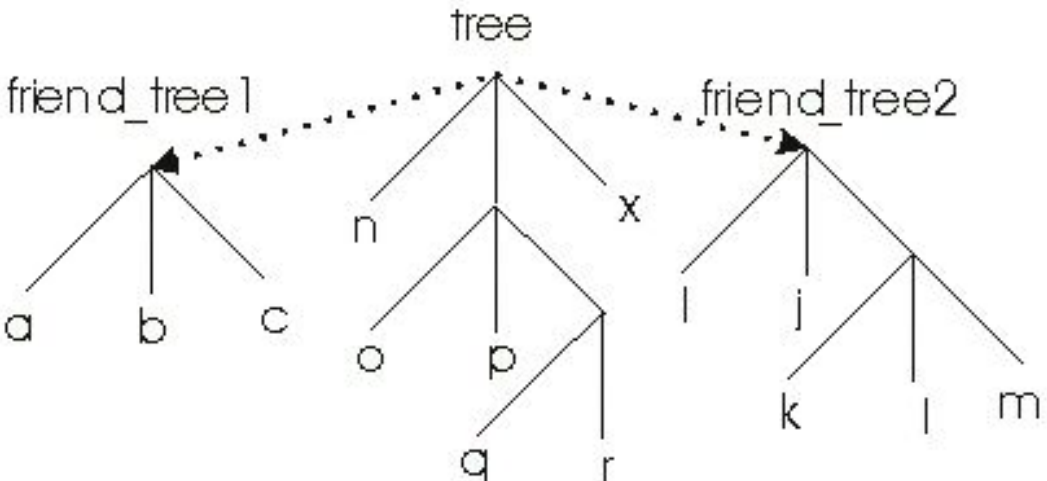
In other words…

The high-level view



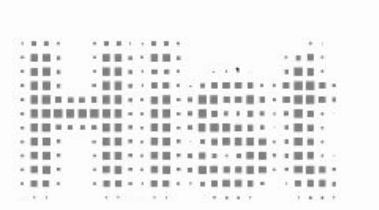
Level-1
Trigger

YES NO



"Triggering" = filtering events to reduce data rates to manageable levels

Technical stack



Histograms

I/O

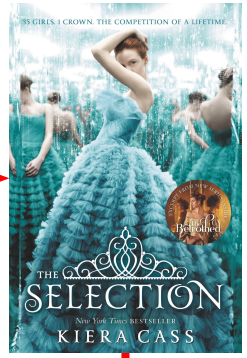
Selections

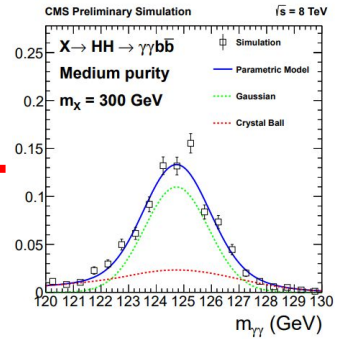
The existing set-up (summarized)

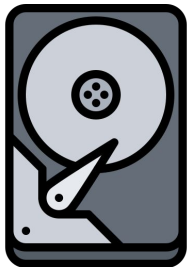
1.49 < |#eta^{TOBJ}|

.ROOT

uproot





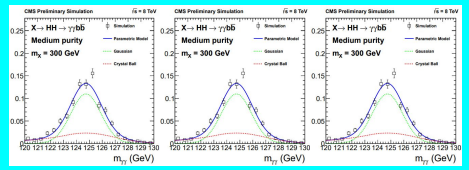


Nikita Golubev, flaticon.com

The idea was something like this.

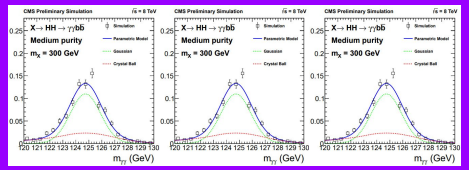
Worker 1

.ROOT



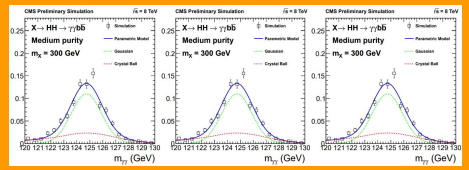
Worker 2

.ROOT



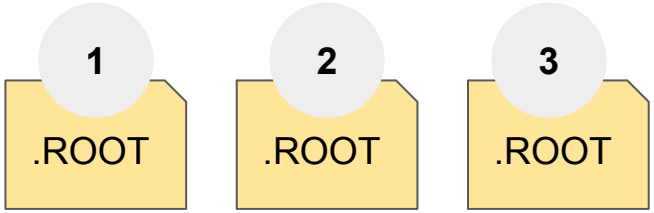
Worker 3

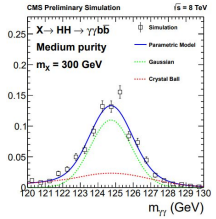
.ROOT



However, we achieved the following architecture:

The new set-up (summarized)





Sequential

Parallel

Sequential

It turned out to be even harder than I thought.

The challenges

Understanding the existing code base

Integrating the dask functionality

Verifying the results (Histograms)

Finding tutorials and resources online.

Profiling the new solution or benchmarking it against the old one.

Discussing problems with the experts (Fermilab, Princeton).

Conclusions

1. The new set-up is still slower and incomplete.

How can we optimize it further?

How can we make it produce the same output as the uproot version?

Tests are needed to verify histograms (otherwise doing it manually).

Type	Events	Time
UPROOT	1	1.6 (1.6 s/ev)
	1000	1.89 (0.00 s/ev)
	50000	12.2 s (0.00 s/ev)
	n -1 (99103)	18.92 s (0.00 s/ev)
DASK v1	1	116.27 s
	1000	23.46 s
	50000	200 s
	n - 1 (99103)	388.12 s
DASK v2	1	6 s
	1000	6 s
	50000	20 s
	n - 1 (99103)	31 s

2. More investigation is needed into the Dask ecosystem and the problem.

Understanding the differences and pros and cons between different packages.

Hist vs dask-histogram vs hist.dask.Hist

Python libraries vs Dask

Uproot vs Coffea

Is the Dask appropriate to the problem? Does it scale?

Are there existing solutions at CERN/Fermilab?

3. Is it possible to integrate the dask functionality into the existing package?

The current package is complex. 20+ .py scripts, 8000+ lines of code

Input is taken from .yml files. Is it a good approach?

How is the current package is being used vs the actual needs for the scientists?