Providing the computing and data to the physicists

Overview of the ATLAS distributed computing system

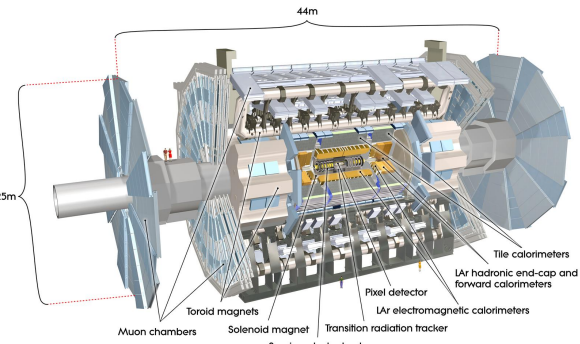
M. Svatoš on behalf of the ATLAS Collaboration

ICHEP 2020, 28.7.2020-6.8.2020

Introduction

The ATLAS Experiment

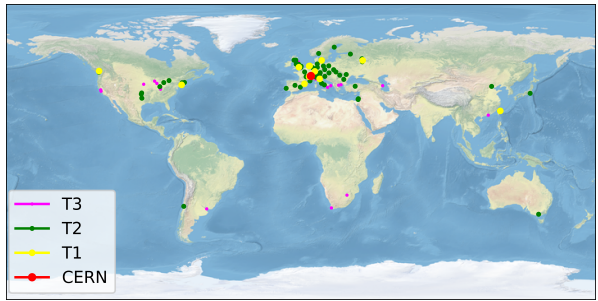
- located at the Large Hadron Collider (LHC) at CERN near Geneva
- the detector is cylindric, 44m long, 25m in diameter, weighting 7,000 tonnes
- the collaboration comprises about 3000 scientific authors from 183 institutions, representing 38 countries



Introduction

The ATLAS Distributed Computing (ADC)

- manages resources on more than 150 sites located around the world
  - about half an exabyte of detector and simulation data
  - more than 400 thousand CPU cores
- runs 24 hours/day, 365 days/year
- sites:



T0 , i.e. CERN - the largest computing resource; detector data archived on tape

T1 - largest computing centres; second copy of the detector data on tape

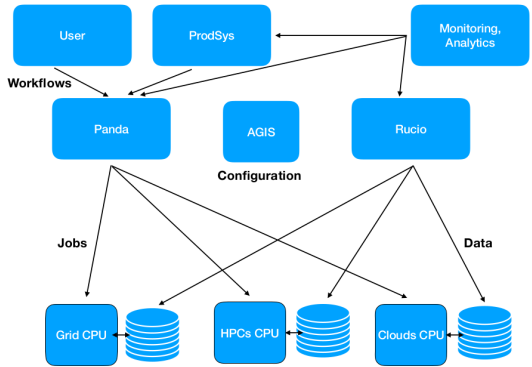
T2 - computing centres usually (not always) smaller than T1 sites; no tape

T3 - small sites (sometimes, only for local users)

Introduction

ADC components

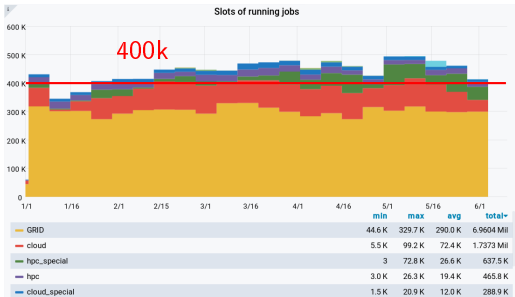
- Workflow management
  - ProdSys
    - organizes the workflow of tasks (i.e. group of jobs) and requests (i.e. group of tasks)
  - PanDA/JEDI
    - deals with job submission to heterogeneous resources
- Data management
  - Rucio
    - data storage, access, replication, deletion, ...
    - scientific data management standard in the HEP community
- additional components
  - information system (AGIS/CRIC), monitoring and analytics, ...



Current status

The ATLAS Distributed Computing (ADC) uses various CPU resources to run jobs:

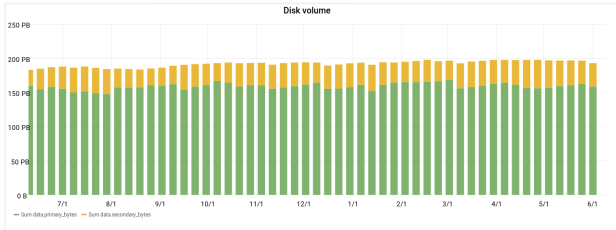
- Worldwide LHC Computing Grid (WLCG) sites,
- Cloud resources,
  - including opportunistic usage of ∼90k cores of High Level Trigger farm (so-called P1 farm)
- HPCs,
- Volunteer computing (BOINC), etc.

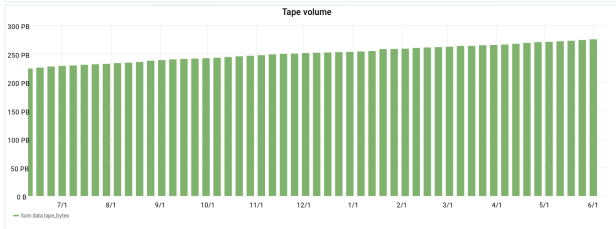


Current status

Storage

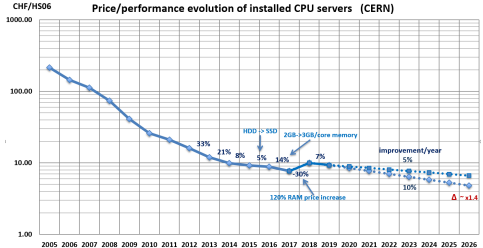
- about half of available space is on tape, half on disk
- majority of data on disk are analysis formats
- disks are always full
  - most of data are primary copies (they need to stay) with only limited amount of secondary data (cached data, can be deleted if necessary)
  - older versions of analysis formats are removed as well as unused data



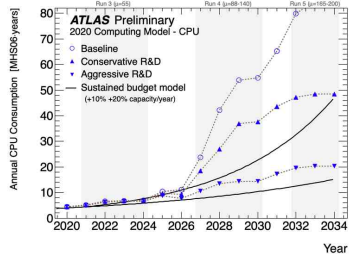


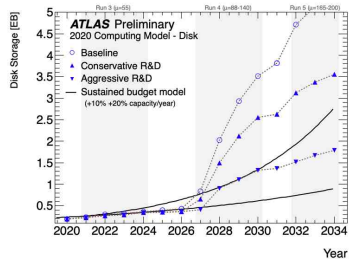
Run 3 /HL-LHC outlook

- with improving performance of the LHC, more data is expected to come
  - an order of magnitude increase of volumes of data due to increasing event sizes and rates
  - but the computing budget is assumed to stay flat and performance gains from technology advancements are decreasing



- Run 3 would be manageable without changes, further Runs would exceed available resources
- significant amount of R&D is needed to fit the constraints





Run 3 /HL-LHC outlook

The R&D involves, for example:

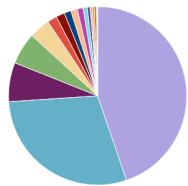
- new analysis model
- improvements in software, both internal (simulation, digitization, reconstruction) and external (event generators)
- improvements in tape usage (data carousel)
- unification of access to compute
- containerization
- interactive analysis
- ...

Analysis model

Current analysis model

- there is a centralized data reduction system using the output of the reconstruction (AODs)
  - the DAOD (i.e. Derived AODs) content is created from AODs by slimming, thinning, skimming, or adding new variables or objects
  - analysis teams can define formats tailored for their specific analysis
- there is a significant overlap in the output formats produced by the various analysis groups
  - causing heavy disk footprint

data formats on disk



	current	percentage
DAOD	101.6 PB	45%
AOD	66.3 PB	29%
HITS	16.1 PB	7%
user	13.09 PB	6%
EVNT	8.58 PB	4%
log	3.97 PB	2%
RAW	3.55 PB	2%
NTUP	2.822 PB	1%
DRAW	2.676 PB	1%
group	2.435 PB	1%

Analysis model

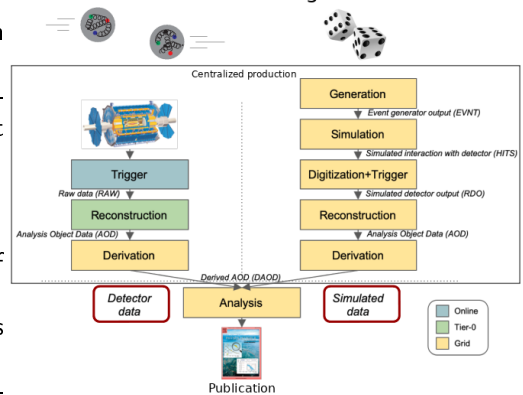
A new analysis model is being prepared in order to fix issues of the current analysis model:

- two new common unskimmed data formats and will be introduced:
  - DAOD_PHYS (about 50kB/event)
  - DAOD_PHYSLITE (about 10 kB/event)
- the goal is to cover needs of up to 80% of ATLAS analyses
- with smaller size, ATLAS can keep more copies, i.e. availability of data for analysers will improve
- event data model:
  - flat representation should allow for better integration with the growing Python-based analysis ecosystem
- appropriate application of lossy compression can help save space

Event generation

- it is expected that NLO and NNLO level of precision will be needed
- event generators are not product of the collaboration, i.e. there is a limited influence on development and optimizations
- there are few ways to decrease resource usage:
  - by a careful optimisation of the physics choices
  - by biasing the event generation (as a function of a kinematic quantity of interest)
  - by computing uncertainties from scales and PDFs through a re-weighting technique
  - by sharing of samples with other LHC experiments (mainly relevant for ATLAS and CMS)

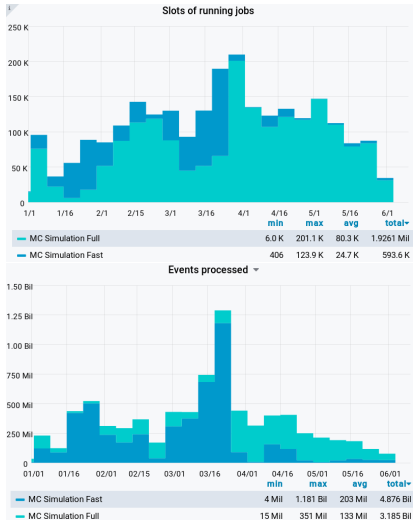
The Data Processing Chain



Software

Simulation (modelling interaction particles with the detector)

- there are many R&D projects in ATLAS and GEANT4 dedicated to resource usage reduction
- there are few ways to decrease resource usage:
  - fraction of events simulated with FullSim (based on GEANT4) need to decrease
  - fraction of fast simulation (primarily parametrized calorimeter response) needs to increase
- this year
  - FullSim used majority of simulation resources
  - majority of events were simulated by fast simulation



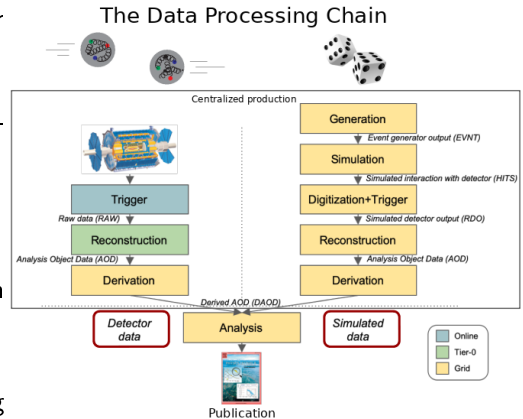
Software

Digitization (modelling the output of the detector readout)

- the plan is to use pre-mixed pile-up datasets
- hard-scatter events will be digitized and then "over-laid" on top of a pre-mixed pile-up event
  - considerably faster
  - reduced I/O requirements
  - scales much less steeply with pile-up luminosity
  - but the pre-mixed pile-up event need to stay on the disk

Reconstruction (creating high-level objects)

- ATLAS initiated the ACTS (A Common Tracking Software) open source project to develop the next generation tracking software in a common cross experiment project



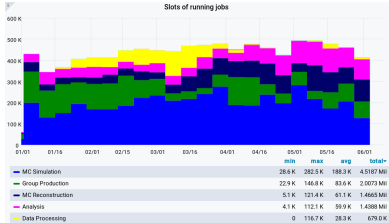
Improvements in compute

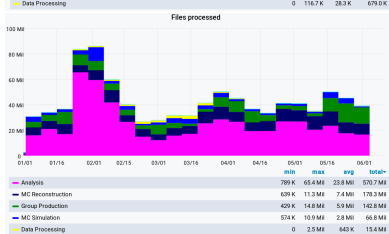
From user point of view, jobs can be split into two categories

- production
  - run by central team
  - contains chain starting at raw data from the experiment (or product of event generation) to common format used for analysis
- analysis
  - jobs from individual analysers or analysis groups

Grand-unification

- user analysis takes only fraction of CPU resources but dominates in number of files it reads
- currently, there is ongoing campaign (grand-unification) to unify access to site's compute through one queue instead of having separate queues for production and analysis
  - it makes it easier to tune amount of analysis in the system (putting more analysis where more relevant input is located)





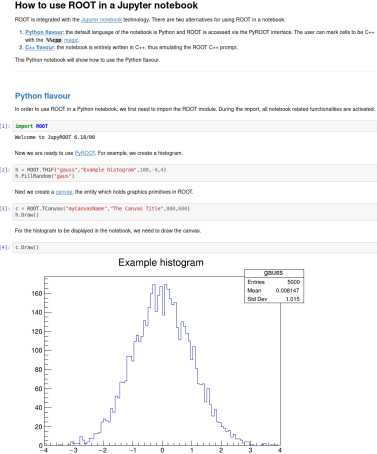
Improvements in compute

Containerization

- almost all jobs are running inside of generic (singularity) container
- the containerization can be also used for users (user specific containers) and data preservation

Interactive analysis

- Jupyter Notebooks (connected to horizontally scalable compute clusters such as Spark, Dask or Ray or batch systems) seem promising



How to use ROOT in a Jupyter notebook

ROOT is integrated with the Jupyter notebook technology. There are two alternatives for using ROOT in a notebook:

1. Python flavour: the default language of the notebook is Python and ROOT is accessed via the PyROOT interface. The user can mark cells to be C++ with the %%cpp magic.
2. C++ flavour: the notebook is entirely written in C++, thus emulating the ROOT C++ prompt.

This Python notebook will show how to use the Python flavour.

Python flavour

In order to use ROOT in a Python notebook, we first need to import the ROOT module. During the import, all notebook related functionalities are activated.

```
In [1]: import ROOT
```

```
Welcome to JupyROOT 6.18/00
```

Now we are ready to use PyROOT. For example, we create a histogram.

```
In [2]: h = ROOT.TH1F("gauss","Example histogram",100,-4,4)
        h.FillRandom("gaus")
```

Next we create a canvas, the entity which holds graphics primitives in ROOT.

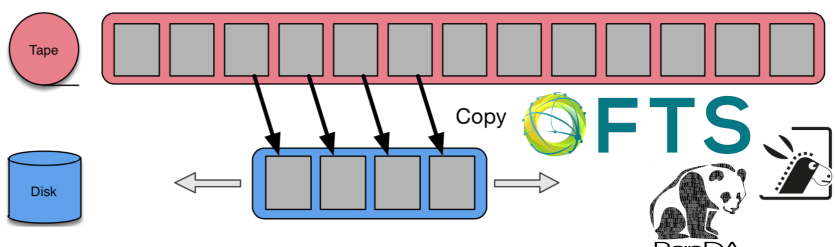
```
In [3]: c = ROOT.TCanvas("myCanvasName","The Canvas Title",800,600)
        h.Draw()
```

For the histogram to be displayed in the notebook, we need to draw the canvas.

```
In [4]: c.Draw()
```

Data Carousel

- is a sliding window approach to orchestrate data processing with the majority of data resident on tape storage
- The processing is executed by staging the data onto disk storage and promptly processing them
  - only the minimum required input data are located on disk at any time
  - tested on full Run2 RAW data reprocessing (18 PB staged over several weeks rather than all at once)



Summary and Conclusions

- The upcoming Run 3/HL-LHC brings many challenges for the ATLAS Distributed Computing.
- The current model is not sustainable for the HL-LHC.
- Many improvements are need for ADC to be able to make LHC data available to ATLAS physicists and to provide them with means to analyse them, such as:
  - new analysis model introducing new small data formats
  - internal and external software improvements
  - improvements in compute access for analysers
  - improvements in disk/tape usage