Analysis Grand Challenge benchmarking tests on selected sites

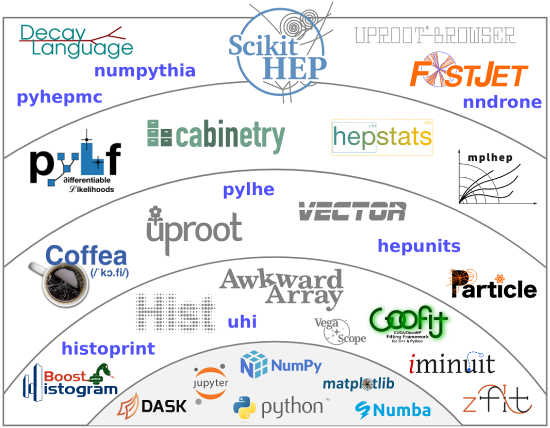
David Koch[1], Thomas Kuhr[1], Günter Duckeck[1], Nikolai Hartmann[1]

[1]Ludwig-Maximilians Universität München

WLCG Workshop 2024

Analysis Grand Challenge

- developed by the IRIS-HEP team
- effort to demonstrate feature-completeness and scalability of scikit-HEP tools
- main framework of the analysis: coffea, offers a high level interface for columnar analysis
- github, readthedocs



Scikit-HEP: Python ecosystem for HEP analyses

Analysis Grand Challenge

ttbar-Analysis includes

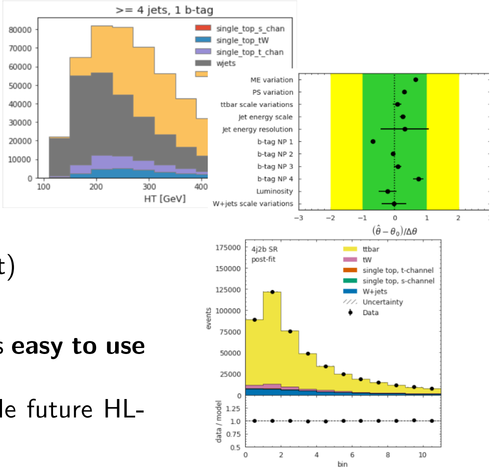
- 1-lepton event selection
- top reconstruction
- cross-section measurement
- on-the-fly evaluation of systematic uncertainties

Analysis Grand Challenge

ttbar-Analysis includes

- 1-lepton event selection
- top reconstruction
- cross-section measurement
- on-the-fly evaluation of systematic uncertainties
- total of 1.78 TB of CMS open data
- only ~75 GB are actually read (4% of the total dataset)
- 948 mio events and 10 variables

… all this sits in a single Jupyter notebook ⇒ analysis code is **easy to use**
should also be **scalable** and **fast**
The AGC analysis is meant as a showcase of how a possible future HL-LHC analysis could look like



Analysis Grand Challenge

the AGC supports a variety of **different setups**:

- different data sources: remote via https, xrootd, from /eos, servicex
- with or without xcache in between
- ML / no ML workflows
- configurable number of events to run over

bonus: export metrics like processed events, runtime, ...

Analysis Grand Challenge

the AGC supports a variety of **different setups**:

- different data sources: remote via https, xrootd, from `/eos`, servicex
- with or without xcache in between
- ML / no ML workflows
- configurable number of events to run over

bonus: export metrics like processed events, runtime, ...

→ great for a benchmark and "integration test" of computing resources
our test setup:

- basic workflow without ML
- data source: `LRZ-LMU_LOCALGROUPDISK` (authentication with certificate)
- distributed computation with `dask-jobqueue`

Analysis Grand Challenge Benchmarks

Benchmarks performed on two different sites:

LMU institute cluster at LMU Munich consisting of one very powerful node and desktop computers
job-scheduler: SLURM
reading of the data via xrootd from LRZ

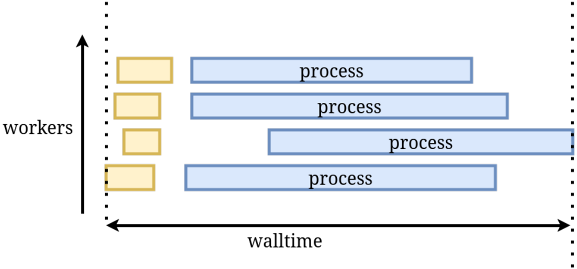
Analysis Grand Challenge Benchmarks

Benchmarks performed on two different sites:

- **LMU** institute cluster at LMU Munich consisting of one very powerful node and desktop computers
  job-scheduler: SLURM
  reading of the data via xrootd from LRZ
- **LRZ** WLCG Tier-2 site in Munich
  job-scheduler: SLURM
  data is stored on regular Grid storage (HDD) as well as on a XCache server (SSD)

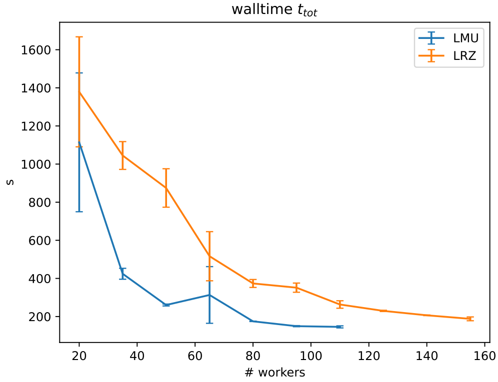
Measurements

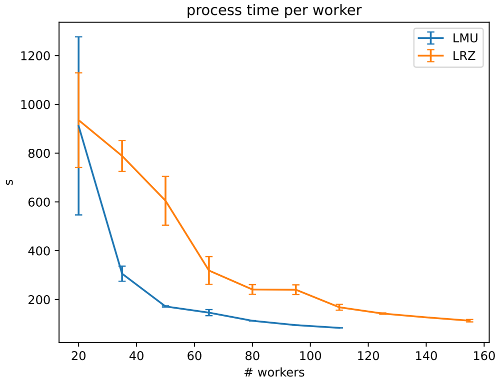
For our benchmarks, only the distributed part of the analysis is considered: no startup time, no plots



measure walltime, processing time, amount of data (in bytes) requested via the AGC's tooling

Measurements
Runtime

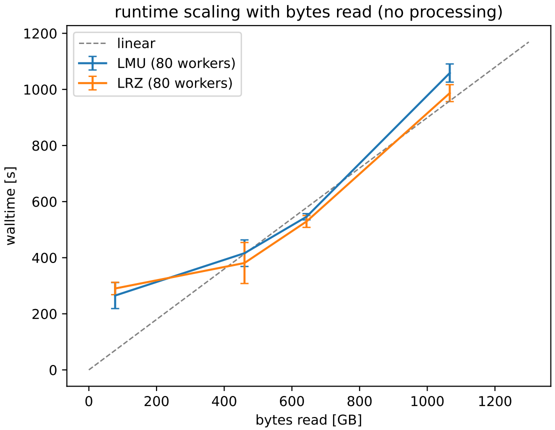


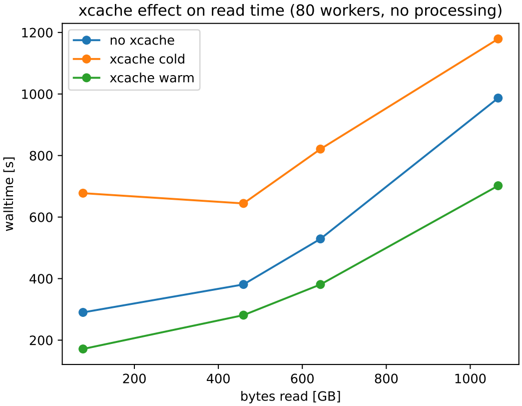


total walltime and average process time per worker

How does runtime scale with amount of data?

- focus on I/O → all processing steps bypassed, only "reading": read, transfer, decompress, load as awkward arrays
- request growing number of branches to increase I/O load





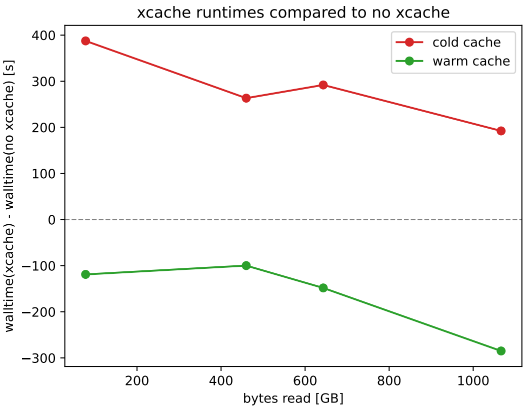
How does using an SSD-based XCache affect runtimes?

- repeat walltime measurements with varying amount of data
- cold cache: run AGC with XCache enabled and no files present on the XCache server
- warm cache: repeat same run directly after

$\rightarrow$ using XCache introduces some overhead when the cache is empty but a noticeable advantage kicks in during consecutive runs

Measurements

XCache



How does the benefit of XCache scale?

$$\text{time(xcache)} - \text{time(no xcache)}$$

$\rightarrow$ trend: the more data is read, the greater the potential benefit of using XCache

Outlook

- more data!
- test ML workflow
- optimize XCache settings with the help of further benchmark tests

Questions?