Federations of grid and cloud storage solutions with Dynafed

Albert-Ludwigs-Universität Freiburg

Benjamin Rottler, Frank Berghaus, Anton Gamel, Markus Schumacher

ErUM-Data IDT

ATLAS EXPERIMENT

Masse und Symmetrien nach der
GRK 2044
Entdeckung des Higgs-Teilchens am LHC

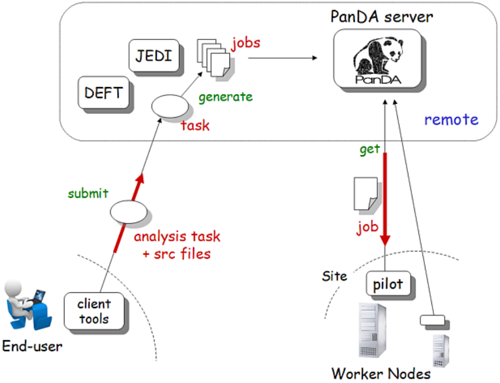
UNI FREIBURG

Analyzing datasets on the WLCG in ATLAS

Current workflow

- users send analysis job to the production system (JEDI/PanDA)
  - only specify which dataset to analyze
  - no need to know where dataset is stored
- distributed system
  - two replicas stored on two different sites
  - jobs follow data
- production system sends analysis job to one of the sites where the dataset is stored

https://twiki.cern.ch/twiki/bin/view/PanDA/PandaJediAnalysis

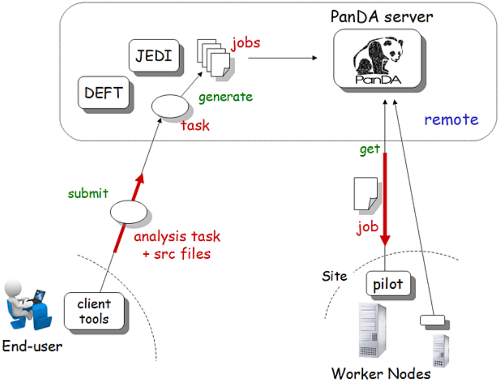


Analyzing datasets on the WLCG in ATLAS

Disadvantages

- system is inflexible
  - on failure of storage endpoint
    → job fails
    → dataset cannot be read from other sites
  - hot datasets
    → replicate dataset to new site
    → submitted jobs are not aware of this
- most WLCG sites provide both storage and computing power
  - probably not feasible anymore at HL-LHC
  - alternative storage solutions: “data lakes”, . . .
- cloud object stores are not supported (yet)

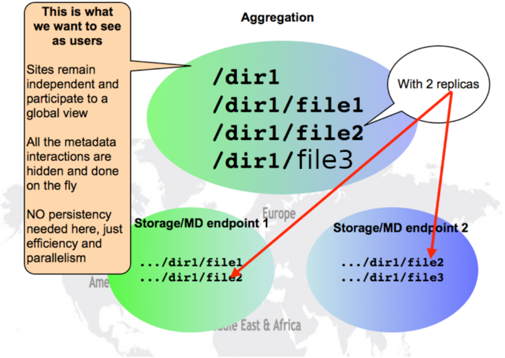
https://twiki.cern.ch/twiki/bin/view/PanDA/PandaJediAnalysis



Dynafed

http://lcgdm.web.cern.ch/dynafed-dynamic-federation-project

- Dynafed = “dynamic federation”
  → developed by CERN’s Lcg-DM group
- protocol: WebDAV
  → extension of HTTP, open industry standard
- combine multiple storage endpoints
- merging and caching (in memory) of metadata
- redirect GET/PUT requests
- support for multiple file replicas
  → select closest endpoint via GeoIP
- focus on performance, scalability and realtime fault resilience



This is what we want to see as users

Sites remain independent and participate to a global view

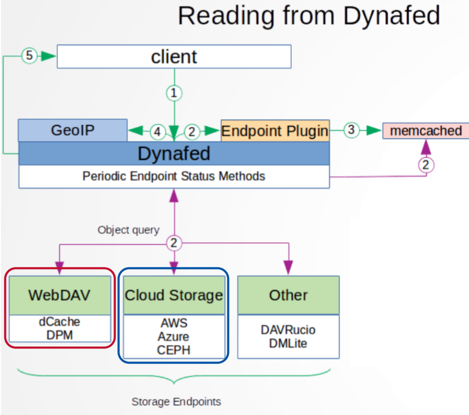
All the metadata interactions are hidden and done on the fly

NO persistency needed here, just efficiency and parallelism

Dynafed

Application to the WLCG

- intermediate layer between client and storage endpoint
- support for different types of storage endpoints
  → grid solutions
  → cloud storage
- change in philosophy
  - jobs do not need to know where dataset is stored
    → data follow jobs
  - integrates well into remote access of datasets
  - replicas can be added while job is already defined
- trade-off: effective usage of storage vs. network traffic



Benchmarks

Requirements

- comparison between WebDAV and currently used protocols (XRootD, . . .)
- comparison between WebDAV and Dynafed
  - WebDAV: plain WebDAV from dCache/cloud storage
  - Dynafed: WebDAV, but request file over Dynafed server
- benchmarks
  - reading data
  - writing data

Benchmarks

Solutions

- two different benchmarks
  - **IO benchmark** (reading/writing)
    → use `gfal2` library for copying data
  - **ROOT benchmark** (reading only)
    → use `ROOT.TFile.Open()` and loop over events
- results are stored in elastic search instance
- data can be visualized with kibana dashboards/plotting scripts

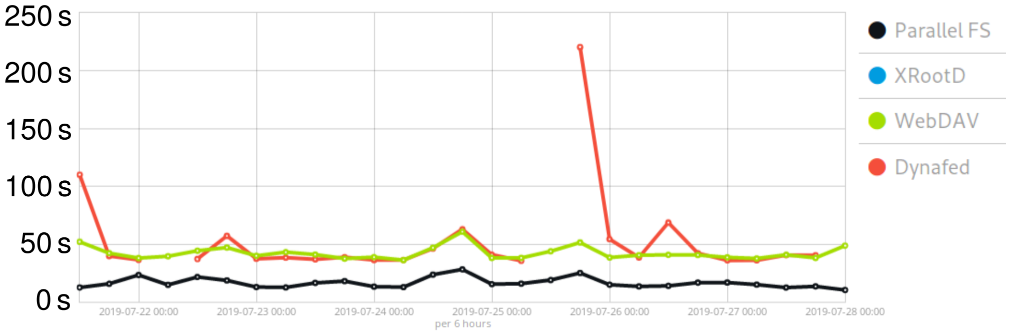
IO benchmark

How it works

- copy 1GB file with random content to storage endpoint (writing benchmark)
- copy file back from storage endpoint (reading benchmark)
- time both copy operations
- use `gfal2` python library for copying
- endpoints
  - local (Freiburg):
    parallel filesystem (BeeGFS, mounted), WebDAV, Dynafed
    → XRootD unfortunately not working because configured as read only
  - remote (CERN, provided by Frank Berghaus):
    XRootD, GridFTP, WebDAV, Dynafed

IO benchmark

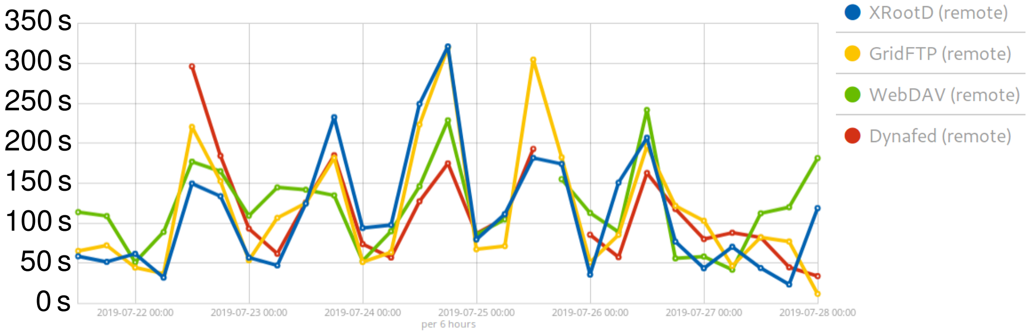
Writing benchmarks – local endpoints



- parallel filesystem faster
- similar benchmark results for WebDAV/Dynafed
- WebDAV more stable than Dynafed
- useful for detecting anomalies

IO benchmark

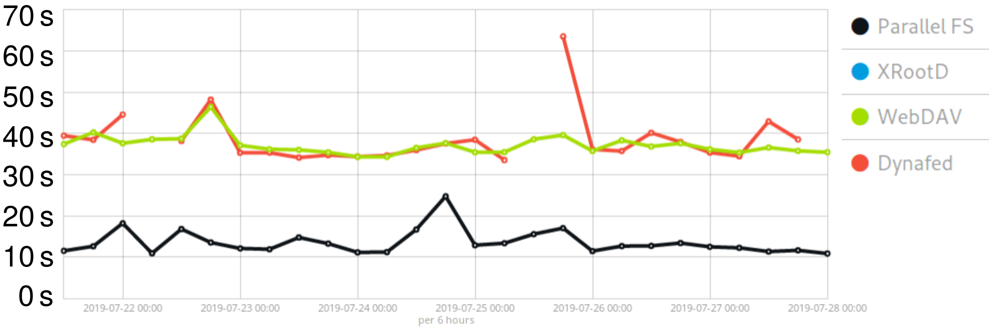
Writing benchmarks – remote endpoints



- a lot of fluctuation
- sometimes several times slower than local access
- similar performance of different protocols

IO benchmark

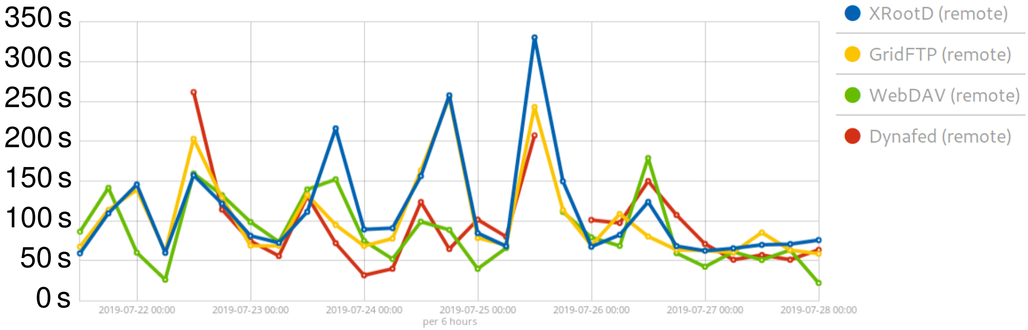
Reading benchmarks – local endpoints



- parallel filesystem faster
- similar benchmark results for WebDAV/Dynafed
- WebDAV more stable than Dynafed

IO benchmark

Reading benchmarks – remote endpoints

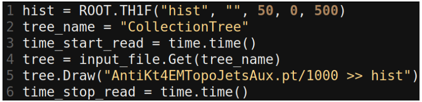


- a lot of fluctuation
- sometimes several times slower than local access
- similar performance of different protocols

ROOT benchmark

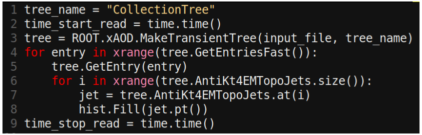
How it works

- ATLAS software release is used (`AnalysisBase,21.2.56`)
- open file with `TFile::Open(path, "READ")` (streaming, file is not copied first to computing node)
- two benchmarks:
  - time to open ROOT file
  - time to read transverse momentum (p_T) of all jets
- two ways to read ROOT file
  - “plain ROOT mode” (faster, but limited functionality)
  - “xAOD mode” (slower, but access to ATLAS EDM)
- caveats
  - problem with some SL6/dCache combinations when reading via `WebDAV`
  - run benchmarks on CentOS 7 (VM in Freiburg)
  - still some error when reading from Freiburg dCache with “xAOD mode” via `WebDAV`
  - not working anymore with `AnalysisBase,21.2.81`



```python
hist = ROOT.TH1F("hist", "", 50, 0, 500)
tree_name = "CollectionTree"
time_start_read = time.time()
tree = input_file.Get(tree_name)
tree.Draw("AntiKt4EMTopoJetsAux.pt/1000 >> hist")
time_stop_read = time.time()
```

plain ROOT mode



```python
tree_name = "CollectionTree"
time_start_read = time.time()
tree = ROOT.xAOD.MakeTransientTree(input_file, tree_name)
for entry in xrange(tree.GetEntriesFast()):
    tree.GetEntry(entry)
    for i in xrange(tree.AntiKt4EMTopoJets.size()):
        jet = tree.AntiKt4EMTopoJets.at(i)
        hist.Fill(jet.pt())
time_stop_read = time.time()
```

xAOD mode

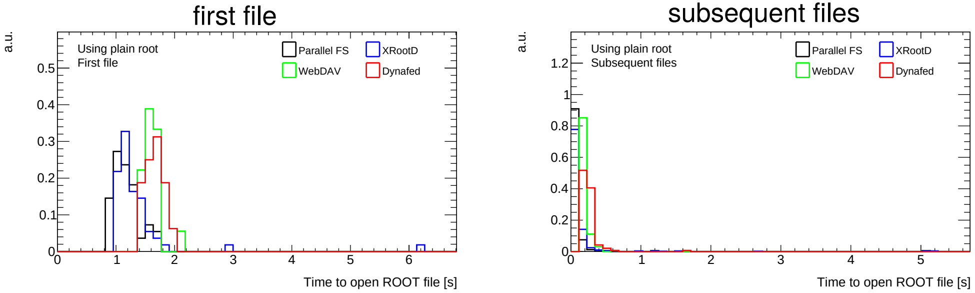
ROOT benchmark

Input files and visualization

- file with simulated $t\bar{t}$ events
  $\rightarrow 1.5$ GB and ~ 30500 events per file
- local input files:
  - 10 exact copies on local parallel file system and local dCache
  - local Dynafed server in Freiburg
  - protocols: local parallel file system (mounted), XRootD, WebDAV, Dynfed
- remote input files:
  - 10 exact copies on dCache and cloud storage
  - remote Dynafed server at CERN
  - protocols: XRootD (dCache), WebDAV (dCache), Dynafed (cloud storage)

ROOT benchmark

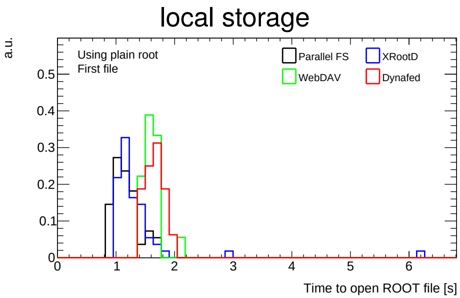
Opening files (local storage)

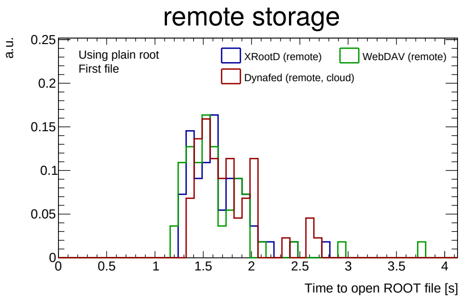


- opening the first file takes longer than opening the subsequent ones → caused by initialization
- similar performance for parallel FS and XRootD
- similar performance for WebDAV and Dynafed
- parallel FS/XRootD faster than WebDAV/Dynafed

ROOT benchmark

Opening files (first file)



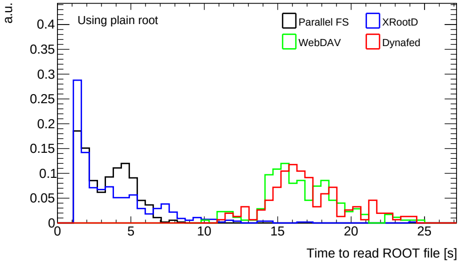


- similar performance of different protocols for remote storage
- comparable results for local and remote storage

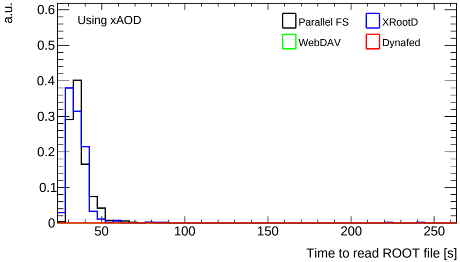
ROOT benchmark

Reading files (local storage)

plain ROOT mode



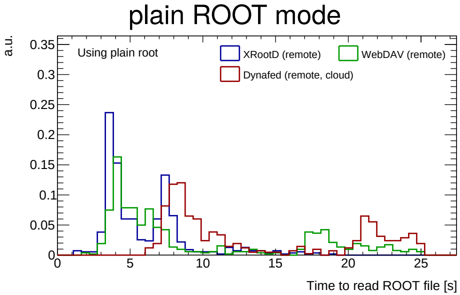
xAOD mode



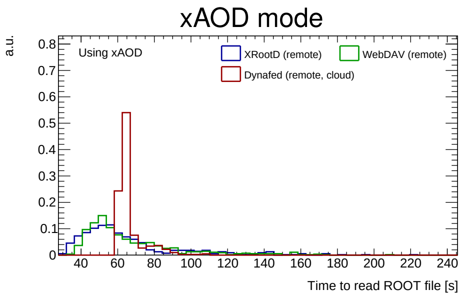
- similar performance for parallel FS and XRootD
- similar performance for WebDAV and Dynafed
- parallel FS/XRootD faster than WebDAV/Dynafed
  (reminder: “xAOD mode” not working with Freiburg dCache)

ROOT benchmark

Reading files (remote storage)



plain ROOT mode



xAOD mode

- similar performance for XRootD and WebDAV
- Dynafed slower (remember: cloud storage is used for this)
- “plain ROOT mode”: two peaks for each protocol
- “xAOD mode” slower due to loading EDM objects

Conclusion & Outlook

Conclusion

- Dynafed is a flexible solution to federate multiple storage endpoints
  - WebDAV protocol is used
  - easy remote access to datasets
  - datasets can be added/removed on the fly without jobs failing
- two benchmarks for comparing WebDAV/Dynafed with existing protocols
- similar performance for WebDAV/Dynafed
- WebDAV/Dynafed slower than other protocols during local access
- WebDAV/Dynafed comparable to other protocols during remote access
- using ROOT with WebDAV access doesn't work always

Outlook

- integrate benchmarks into HammerCloud to run (automated) tests on other sites

Backup

IO benchmark

Remote endpoints

- XRootD
  `root://eosatlas.cern.ch:1094/eos/atlas/atlascerngroupdisk/proj-dynafed/`
- GridFTP
  `gsiftp://eosatlassftp.cern.ch:2811/eos/atlas/atlascerngroupdisk/proj-dynafed/`
- WebDAV
  `https://eosatlashttp.cern.ch/eos/atlas/atlascerngroupdisk/proj-dynafed/`
- Dynafed
  `davs://dynafed-atlas.cern.ch/data/heos/`

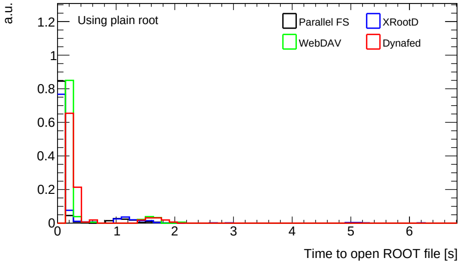
ROOT benchmark

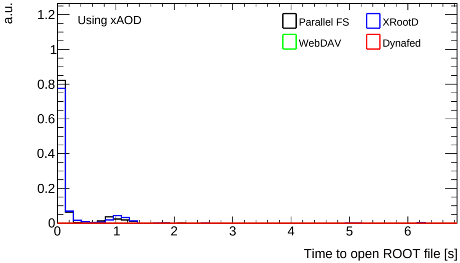
Remote endpoints

- XRootD
  `root://eosatlas.cern.ch:1094//eos/atlas/atlascerngroupdisk/proj-dynafed/`
- WebDAV
  `https://eosatlashttp.cern.ch//eos/atlas/atlascerngroupdisk/proj-dynafed/`
- Dynafed
  `davs://dynafed-atlas.cern.ch:443/data/cloud/`

ROOT benchmark

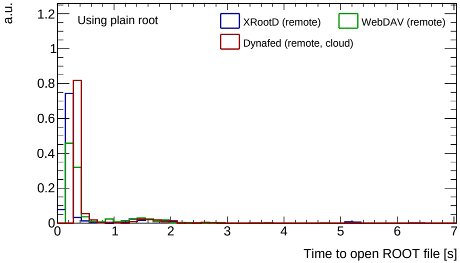
Opening files – local storage

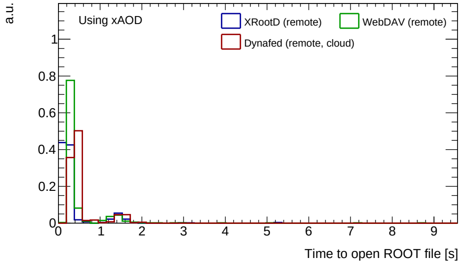




ROOT benchmark

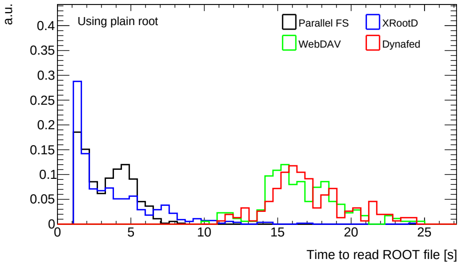
Opening files – remote storage

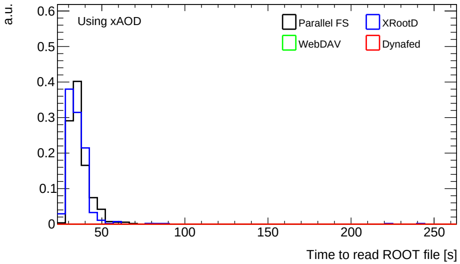




ROOT benchmark

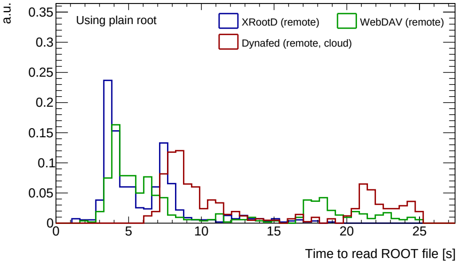
Reading files – local storage

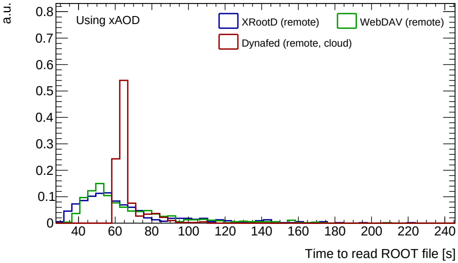




ROOT benchmark

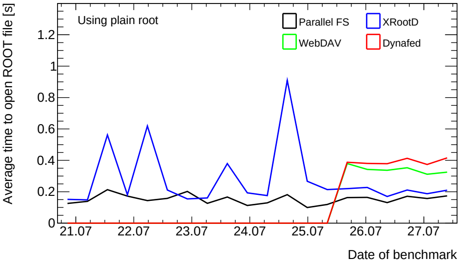
Reading files – remote storage

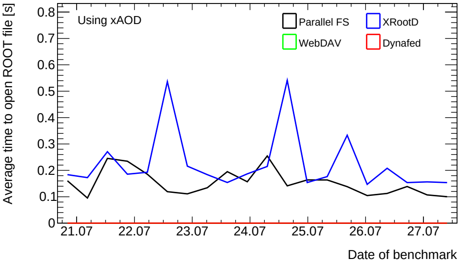




ROOT benchmark

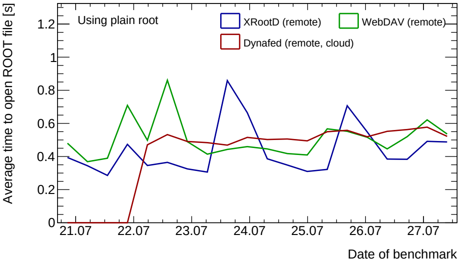
Opening files – local storage

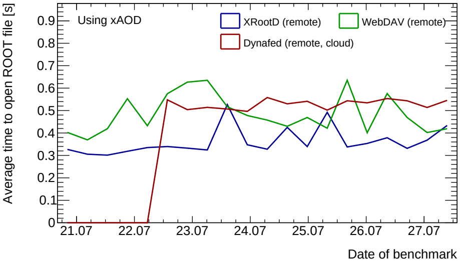




ROOT benchmark

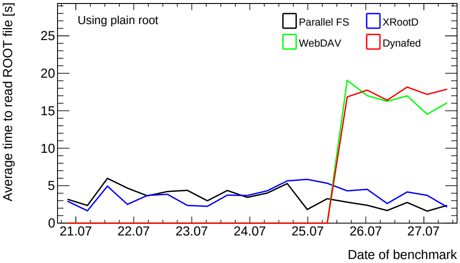
Opening files – remote storage

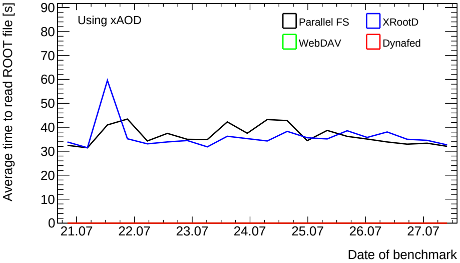




ROOT benchmark

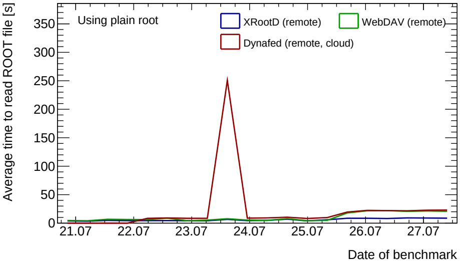
Reading files – local storage

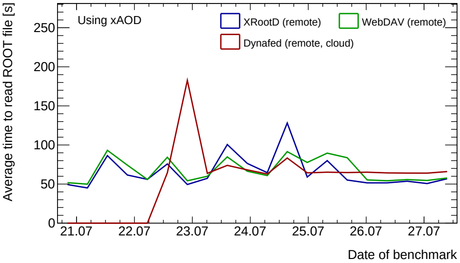




ROOT benchmark

Reading files – remote storage



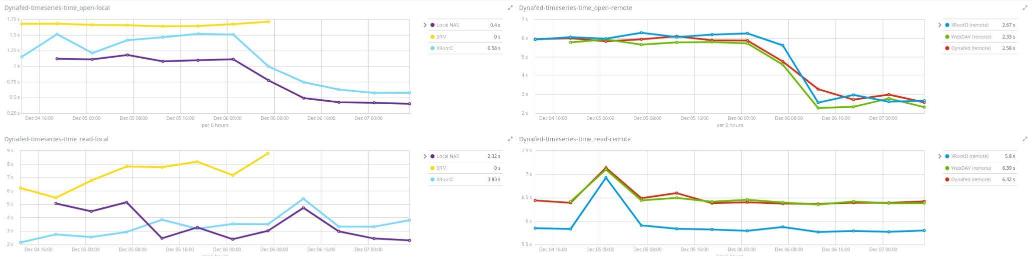


Change of analysis release

- previously used `AthAnalysis` (full Athena setup)
- switch to `AnalysisBase`
  - no Athena, but still allowes for xAOD access mode
- "Athena mode" → "xAOD mode"
- `srm` does not work with `AnalysisBase`
  (gfal library is not included in release)
- use ROOT benchmark
- done on SL6 ⇒ local WebDAV/Dynafed not working

Changes of analysis release

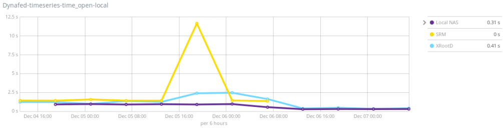
Results for plain-root mode

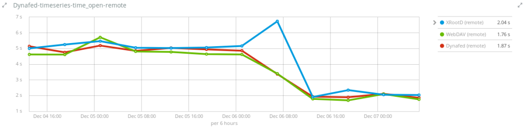


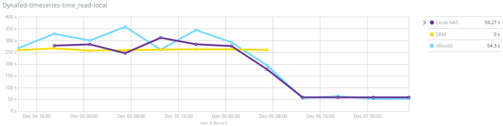
- opening files is faster
- no change for reading files
- (local NAS = parallel FS)

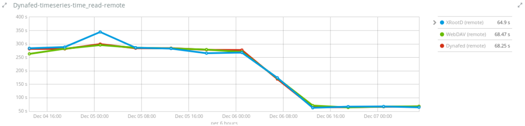
Changes of analysis release

Results for athena/xAOD mode









- opening files is faster (less visible)
- reading files is faster
- (local NAS = parallel FS)