Results of Disk-Caching-On-The-Fly Benchmarks in Freiburg

Michael Böhler, Anton J. Gamel, Stefan Kroboth,
Dirk Sammel, Markus Schumacher

11.05.2021
ErUM-Data Collaboration Meeting

Albert-Ludwigs-Universität Freiburg

Introduction

- Goals:
  - Implementation of a caching solution in Freiburg ATLAS environment
    → Disk-Caching-on-the-Fly (DCOTF)
  - Performance benchmarks of DCOTF
    → comparison of caching setup to direct access
- Starting point:
  - Sandbox setup provided by Frankfurt/GSI
    (Serhat Atay, Kilian Schwarz, Paul-Niklas Kramp)
  - https://git.gsi.de/atay/xrootd-disk-caching-on-the-fly

Architecture of DCOTF

- DCOTF setup utilizes XRootD plugins
- Client:
  - Requests data transfer
  - XrdProxyPrefix plugin: redirects request to Datamanager
- Datamanager:
  - RedirLocal plugin: checks if data exists locally in Cache directory
  - Yes: points client to address in Local Storage
  - No: redirects request to ForwardProxy
- ForwardProxy:
  - Connects to External Data Server
  - Returns data to client
  - Writes data to Cache directory

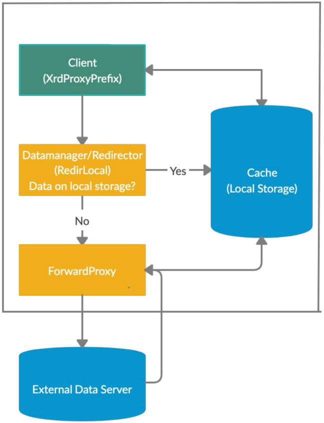
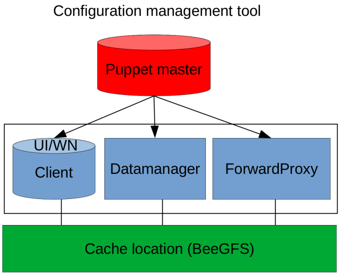


Figure by Serhat Atay

Freiburg setup at local HPC cluster NEMO

- Implementation with Puppet
- Deploys setup to 3 VMs:
  Client, Datamanager, ForwardProxy
- Client: typical worker node or user interface + XrdProxyPrefix plugin
  'typical' → same as machines in production
- Cache location: local cluster file system
  Mounted on all VMs



- Puppet: machines can be configured at any place
  → easy to deploy when ready for production

Benchmarking

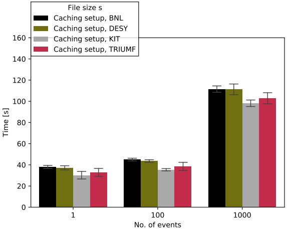
- Python script with PyROOT
- 'Pseudo analysis', implements features of typical user analysis:
    - Open file
    - Initialize xAOD tree structure
    - Loop over events
    - Read some branches
    - Loop over all electrons in event
- Output: information in JSON format
  → visualize with Matplotlib & pandas

Benchmarking

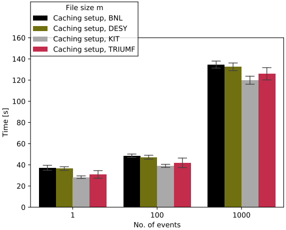
- Tested setups:
  - Caching without file in cache location
    → caching setup, file read and downloaded from external site
  - Caching with file in cache location
    → caching setup, ForwardProxy not used, file read from local cluster FS
  - Local access
    → caching setup not active, file read from local cluster FS
  - External access
    → caching setup not active, file read from external site
- For each setup test all combinations of
  - File size: s, m, l (400MB, 1GB, 6GB)
  - Number of events: 1, 100, 1000
- Repeat each test multiple times → mean value of elapsed time
- External sites:
  BNL (USA), TRIUMF (CAN), DESY-HH (GER), KIT (GER)

Comparison of external sites

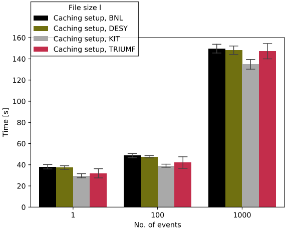
- Compare access to external sites (with caching setup)



small file



medium file



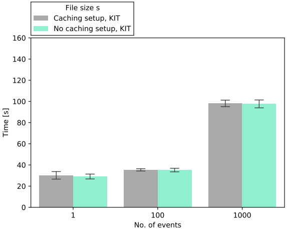
large file

Setup (large file, 1000 events)	Time [s]	Uncertainty [s]
Caching setup, BNL	150	5
Caching setup, DESY	148	5
Caching setup, KIT	135	6
Caching setup, TRIUMF	147	9

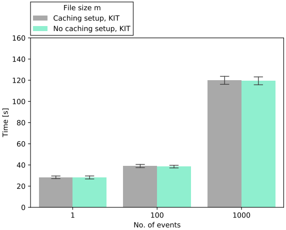
- Fastest: KIT (as expected → fast connection)
- BNL/DESY/TRIUMF comparable

Overhead check KIT

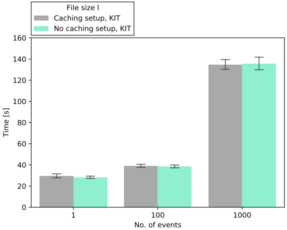
- Is the caching setup causing overhead when reading from external sites?
- Comparison for KIT



small file



medium file



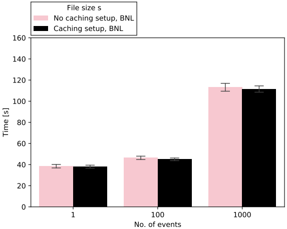
large file

Setup (large file, 1000 events)	Time [s]	Uncertainty [s]
Caching setup, KIT	135	6
No caching setup, KIT	136	7

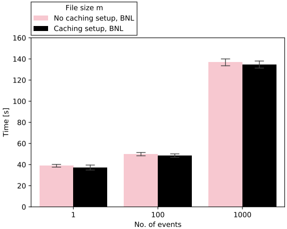
- No overhead from caching setup

Overhead check BNL

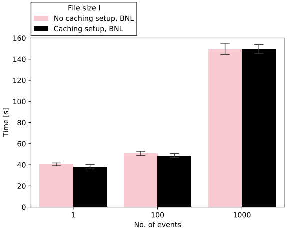
- Is the caching setup causing overhead when reading from external sites?
- Comparison for BNL



small file



medium file



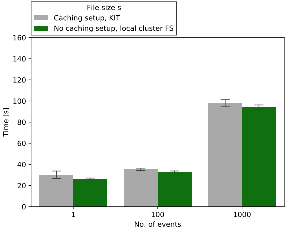
large file

Setup (large file, 1000 events)	Time [s]	Uncertainty [s]
No caching setup, BNL	149	6
Caching setup, BNL	150	5

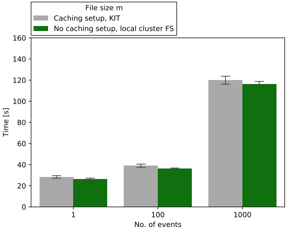
- No overhead from caching setup

Comparison to local access

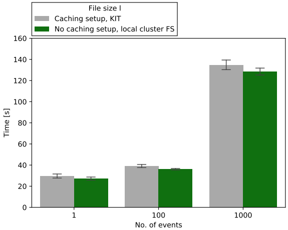
- Compare access to external sites to local access
- Comparison for KIT



small file



medium file



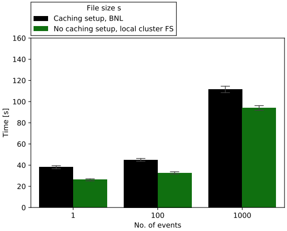
large file

Setup (large file, 1000 events)	Time [s]	Uncertainty [s]
Caching setup, KIT	135	6
No caching setup, local cluster FS	129	4

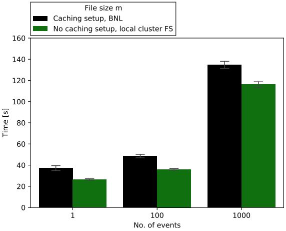
- Comparable within uncertainties
- Fast network connection between UNI-FR and KIT

Comparison to local access

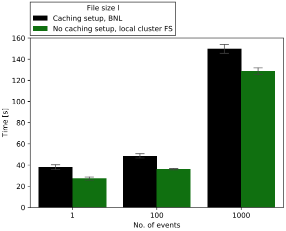
- Compare access to external sites to local access
- Comparison for BNL



small file



medium file



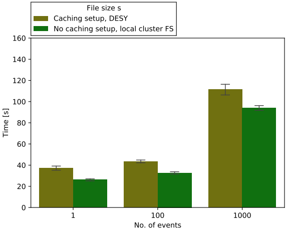
large file

Setup (large file, 1000 events)	Time [s]	Uncertainty [s]
Caching setup, BNL	150	5
No caching setup, local cluster FS	129	4

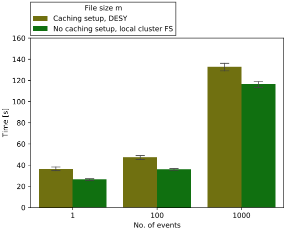
- Local access faster
- Caching setup improves time of event loop

Comparison to local access

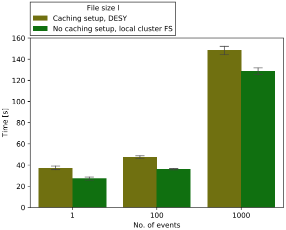
- Compare access to external sites to local access
- Comparison for DESY



small file



medium file



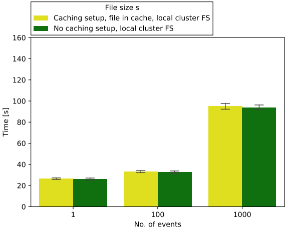
large file

Setup (large file, 1000 events)	Time [s]	Uncertainty [s]
Caching setup, DESY	148	5
No caching setup, local cluster FS	129	4

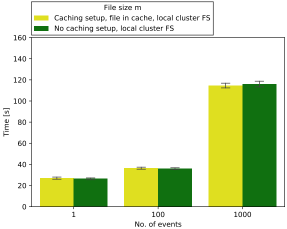
- Local access faster
- Caching setup improves time of event loop

Overhead check local access

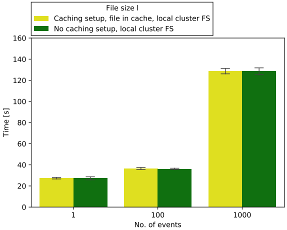
- Is the caching setup causing overhead when reading from local cluster FS?
- File exists in cache location



small file



medium file



large file

Setup (large file, 1000 events)	Time [s]	Uncertainty [s]
Caching setup, file in cache, local cluster FS	129	3
No caching setup, local cluster FS	129	4

- No overhead from caching setup

Experiment-independent benchmarks

- Benchmarks in this talk: ATLAS-specific ROOT files and event loop
- As discussed in previous meeting: comparisons with other caching setups at other sites/experiments are important
  → provide experiment-independent, plain-ROOT benchmarks
- Input files: generated $t\bar{t}$ events with PYTHIA8
  - 50k events, 1.3 GB
  - 200k events, 4.9 GB
  - 500k events, 13 GB
- Event-loop script and files are ready → currently collecting results
- Will be distributed to the community

Conclusion & Outlook

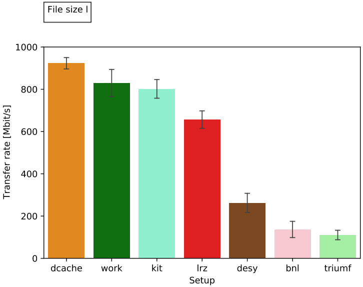
- Benchmarks results
  - Accessing the locally cached files is faster than accessing files from external site
  - Exception: KIT → fast network connection
  - Caching setup is not causing significant overhead
- Performance comparison with other caching setups at other sites
  - Created experiment-independent event-loop script
  - Generated events with PYTHIA8 → plain-ROOT input files
  - Script and input files will be distributed to the community
- Planned benchmarks
  - Test complete user physics analysis (many files with different sizes, ...)
- Caching setup is in 'proof-of-concept' state
- Still some open issues to solve before it's ready for production
  → in close contact with Frankfurt/GSI

Conclusion & Outlook

- Benchmarks results
  - Accessing the locally cached files is faster than accessing files from external site
  - Exception: KIT → fast network connection
  - Caching setup is not causing significant overhead
- Performance comparison with other caching setups at other sites
  - Created experiment-independent event-loop script
  - Generated events with PYTHIA8 → plain-ROOT input files
  - Script and input files will be distributed to the community
- Planned benchmarks
  - Test complete user physics analysis (many files with different sizes, ...)
- Caching setup is in 'proof-of-concept' state
- Still some open issues to solve before it's ready for production
  → in close contact with Frankfurt/GSI

Thanks!

Transfer rates

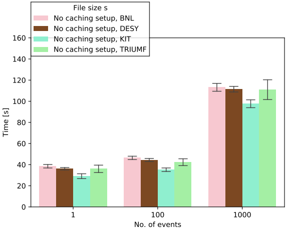


Setup	Transfer rate [Mbit/s]	Uncertainty [Mbit/s]
dcache	923	33
work	830	79
kit	802	54
lrz	656	51
desy	262	55
bnl	137	47
triumf	110	27

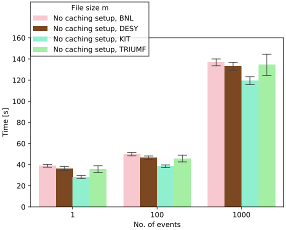
- Fast: local cluster FS, Freiburg dCache, KIT
- Close to KIT: LRZ (Munich)
- Slow: BNL, TRIUMF
- DESY closer to BNL and TRIUMF

Comparison of external sites

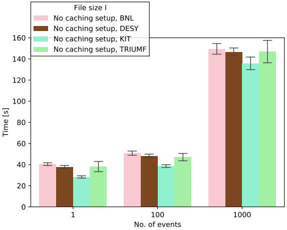
- Compare access to external sites



small file



medium file



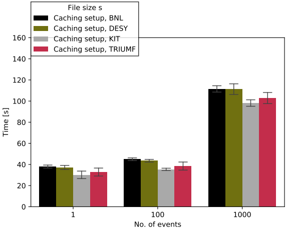
large file

Setup (large file, 1000 events)	Time [s]	Uncertainty [s]
No caching setup, BNL	149	6
No caching setup, DESY	147	4
No caching setup, KIT	136	7
No caching setup, TRIUMF	147	13

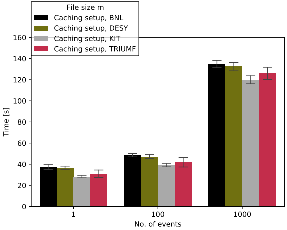
- Fastest: KIT (as expected → fast connection)
- BNL/DESY/TRIUMF comparable

Comparison of external sites

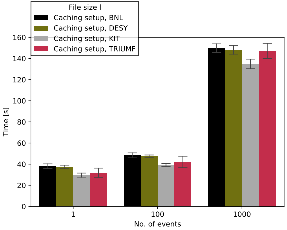
- Compare access to external sites (with caching setup)



small file



medium file



large file

Setup (large file, 1000 events)	Time [s]	Uncertainty [s]
Caching setup, BNL	150	5
Caching setup, DESY	148	5
Caching setup, KIT	135	6
Caching setup, TRIUMF	147	9

- Fastest: KIT (as expected → fast connection)
- BNL/DESY/TRIUMF comparable

Open issues

- Caching of files from external sites:
  - Client credentials (VOMS) are not forwarded to external site → authentication fails
  - Current fix: create credential on ForwardProxy
- Reading of cached files by a given user:
  - Files are written by user 'xrootd'
  - Default permissions are too strict (600)
  - Known issue: https://github.com/xrootd/xrootd/issues/649
  - Current fix: Multi-user plugin (but user needs to be hardcoded)
- XrdProxyPrefix is not working:
  - Site URL is not forwarded
  - Current fix: hardcore site URL
- Expiration of cached files? Lifetime mechanism?