Karlsruhe Institute of Technology

Belle II

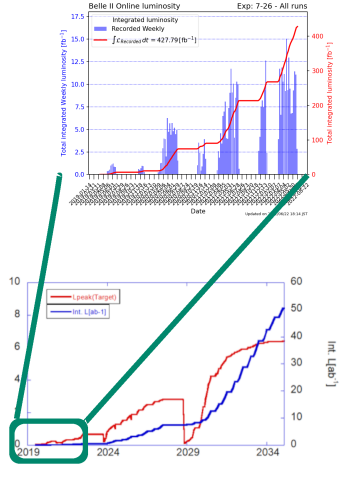
Belle II Caching Activities at KIT

Moritz Bauer, Günter Quast und Matthias Schnepf | 20. October 2022

Computing challenges for Belle II

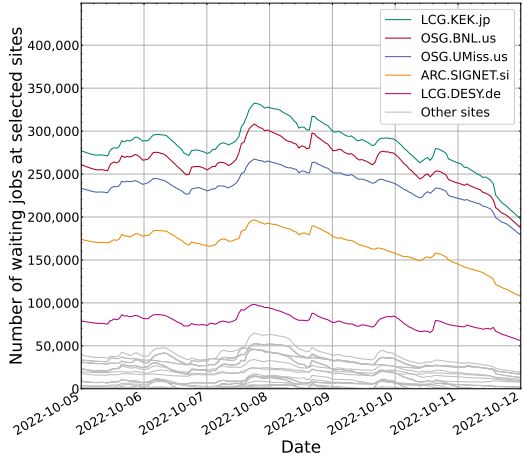
- Belle II will collect 50 ab^{-1} data $\rightarrow$ 50 PB raw data
  - + simulation samples, processed data, skims
- Many statistically-dominated analyses which process entire dataset
- MC generation and user analysis on all sites with `basf2`[a]
- Belle II computing infrastructure:
  - $\approx$ 30 sites of varying size
  - All provide Grid storage
  - DIRAC & Rucio for workload & data management
- High barrier to entry for smaller sites
  - Each site needs Grid storage and Grid middleware

[a] https://github.com/belle2/basf2

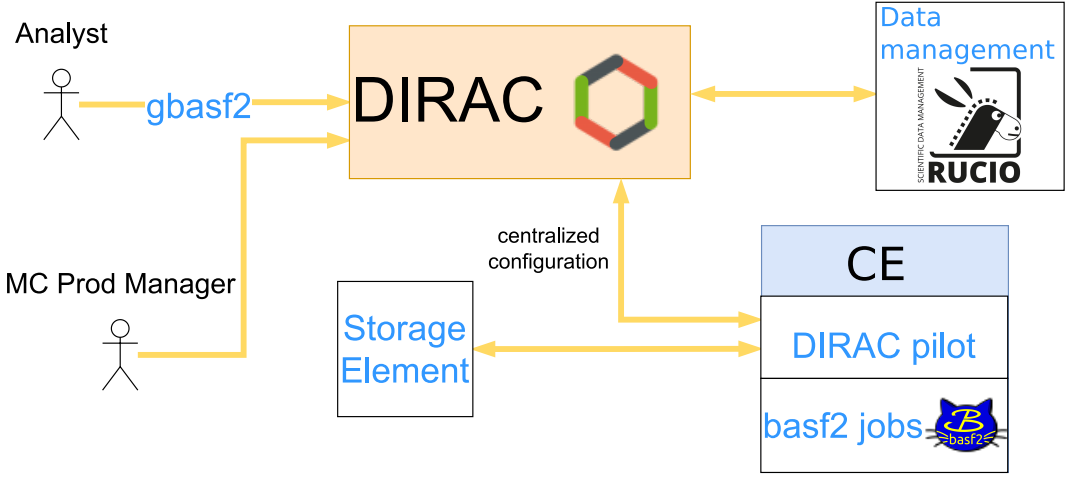


Why Caching?

- Some sites and some datasets are in higher demand than others
- Sometimes sites fail while SE is still functional, leading to waiting jobs at remaining sites
  - Manually adding replicas in Rucio doesn't scale well
- Plot shows: Not all sites can provide all datasets so individual sites have very high number of waiting jobs



Distributed Computing at Belle II



DIRAC & Rucio

- Belle II chose DIRAC, originally for LHCb, as basis of our Grid concept
  - Belle II implemented an extension on top of DIRAC (“BelleDIRAC”) to customize to our needs
- DIRAC’s pilot concept uses central configuration to perform e.g. sandbox setup
  - Caching not easily implemented by site-level config
- Belle II adopted Rucio in 2021 instead of existing custom file catalog
  - Provides caching functionality but again only with central config

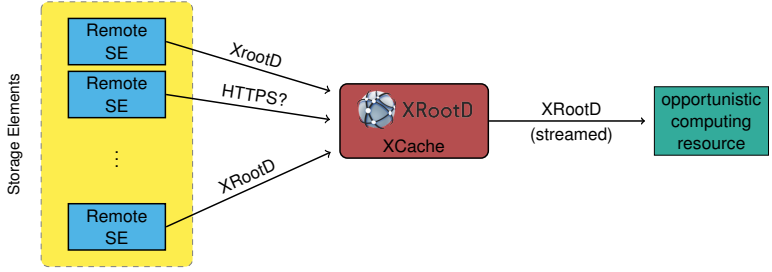
→ **Different solution needed for decentralized caches!**

XRootD

Implemented Caching Setup

XRootD to the rescue!

- XRootD request is transparently forwarded to cache using redirector plugin
  - Dynamically loaded library on worker node, loaded by XRootD binary
- XRootD cache server retrieves data from disk or streams it from SE
  - Almost no time lost since job doesn't have to wait for file to download from SE to cache
  - Completely transparent to analysis users



A simple XRootD proxy server setup

- Currently running on single machine with 500TB disk backend
- A few lines of configuration are enough to start the XRootD cache
- XRootD handles purging of old files from cache but allows adding plugins for dynamic decisions
- If cache breaks or is overloaded, job will access original SE

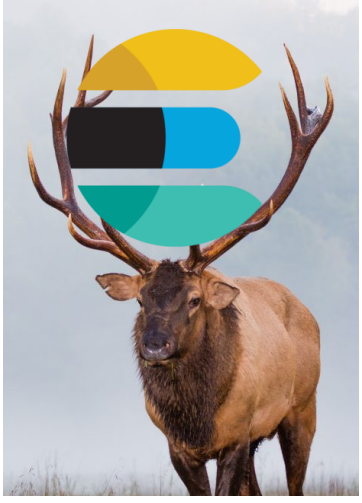
```
xrd.port 1094
all.export /xroot:/ nostage
all.export /root:/ nostage

xrootd.chksum max 4 adler32
ofs.osslib  libXrdPss.so
pss.cachelib default
ofs.ckslib * libXrdPss.so
pss.origin =
oss.localroot [your storage]
pfc.ram 32g
pfc.diskusage 0.9 0.95
```

Monitoring setup

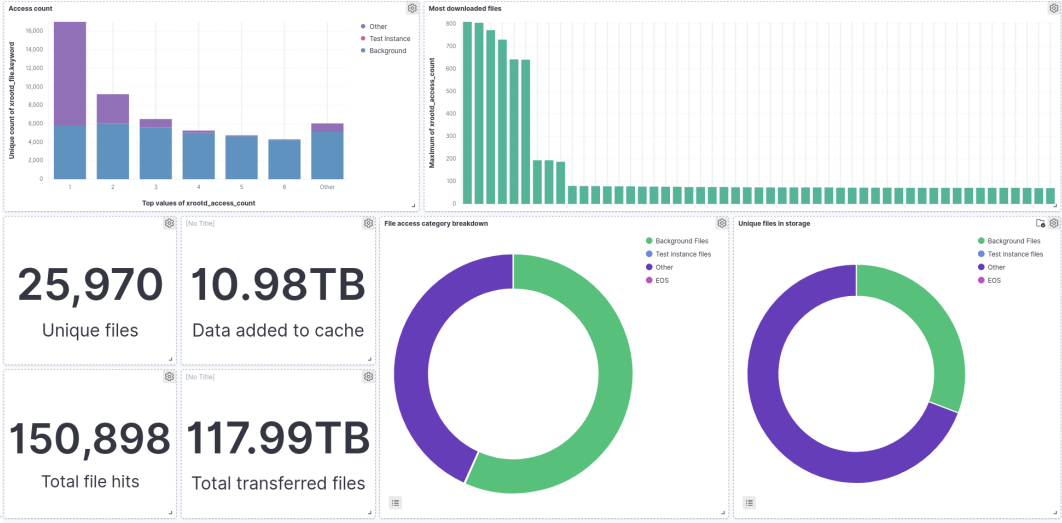
- Capture XRootD monitoring streams with `xrootdlib`[a]
- Captured attributes include file size, path and access count
- Feed data into Elasticsearch DB and visualize/analyze with Kibana (E(L)K stack)
- Elasticsearchs' query tools allow easy visualization of performance

[a] https://github.com/maxfischer2781/xrootdlib



Monitoring results

The last 9 months



Monitoring results

- Stored 26k files / 11TB
  - Actually 17TB in cache, started monitoring some time after launching cache
- Provided 150k files / 118TB
- 60% of cached files are MC background overlay
  - Many (mostly run-independent) accessed >80 times
  - Could also be pre-cached manually but this again requires admin intervention and maintenance
- $\approx$ 55% of files only accessed once

$\Rightarrow$ Available cache is $\approx$ 30x larger, we need more jobs to fill the cache

- Already integrated all 6 TARDIS subsites so expanding to more SEs is the next step

Summary and Next Steps

Shown today

- Belle II is a great opportunity to experiment with new computing setups
- We at KIT are testing a caching solution with XRootD
- Cache has been operating for 9 months locally

Next steps

- Cache data at remote sites, preferable far away
  - Testing this with SIGNET.si SE at “Jozef Stefan” Institute in Ljubljana, Slovenia
- Adapt the DIRAC pilot to redirect HTTPS requests as not all sites provide XRootD protocol access
- Each LCG.KIT-TARDIS.de subsite can provide local XRootD cache

Backup

Further considerations

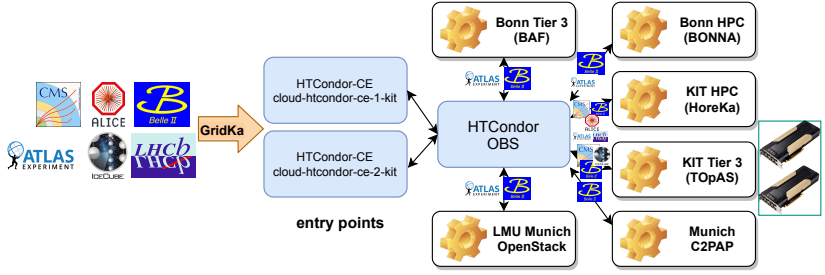
Possible pitfalls

- Make sure the Space Token (json file with information about used/free space) is excluded from caching to avoid misreporting storage
- Make sure fallback to upstream SE works if cache unavailable

Additional planned features

- Management interface to pre-fetch heavily used datasets before first job runs
  - Here Cache RSEs could be useful
- Monitoring of the cache to identify bottlenecks etc
- VOMS proxy forwarding to avoid need for robot certificate

Developments at KIT
COBaID & TARDIS



- Integrate unused, external resources via overlay batch system (OBS)
- Load balancing with COBaID and life-cycle management with TARDIS

Accessing the Cache

How do users select sites with cache?

- In DIRAC: Cached XRootD/HTTPS SEs are added to TARDIS site in DIRAC configuration
  - → Jobs with datasets at e.g. CNAF can also run at LCG.KIT-TARDIS.de
- At site: XRootD client-side redirector plugin (`libXrdClProxyPlugin`) deployed on caching-enabled worker node
  - Redirects e.g. `root://[SE]//[path]` to `root://[XCache]//root://[SE]//[path]`
- HTTPS-only sites: Prefetch these datasets and register them as Cache RSE in Rucio

Cache Rucio Storage Elements (RSEs)

- Usually, Rucio manages replicas on RSEs with rule-based system
- Marking RSE as Cache excludes it from this and allows external management of entries
- Cache RSEs can expire after some time, allowing the cache to be easily removed (if needed)