Russian National Data Lake Prototype

Aleksandr Alekseev, Andrey Kiryanov, Alexei Klimentov,
Tatiana Korchuganova, Andrey Zarochentsev

ISGC 2021, 22-26 March 2021

Background

HENP experiments are preparing for HL-LHC era, which will bring an unprecedented volume of scientific data. This data will need to be stored and processed by collaborations, but expected resources growth is nowhere near extrapolated requirements of existing models both in storage volume and compute power.

=> *Computing models need to evolve.*

This evolution includes multiple aspects:

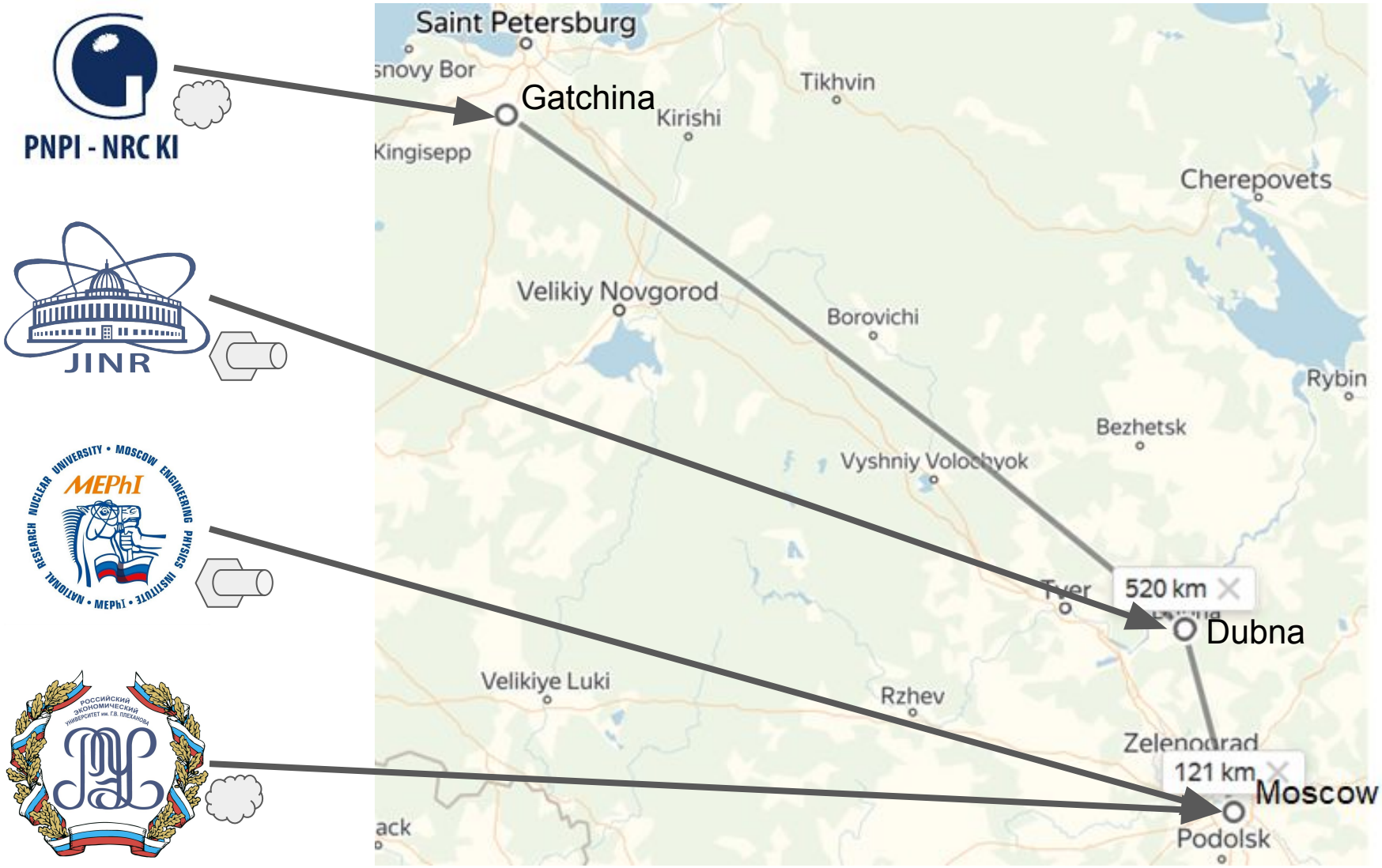
- Optimized data processing, squeezing the maximum from available CPU/GPGPU/FPGA resources
- Optimized data storage, reduction of the number of copies, different data access methods, full utilization of network resources
- Cost optimizations, no high-end expensive RAID setups, no underutilized CPUs on storage servers, no HDDs with 90% free space on the worker nodes
- Deployment optimizations, scalability and containerization with on-demand expansion into the cloud (both community and commercial)
- Operational cost optimization, more standardized solutions, lower requirements on unique Grid expertise

Data Lake in HL-LHC R&D Computing Projects

WLCG and experiments have launched R&D projects to address HL-LHC challenges:

- **Data Lake**. The aim is to consolidate geographically distributed data storage systems connected by fast network with low latency. The Data Lake model as an evolution of the current infrastructure bringing reduction of the storage and operational costs
- **Intelligent Data Delivery Service** (IDDS). Delivering events as opposed to delivering bytes. This allows an edge service to prepare data for production consumption, the on-disk data format to evolve independently of applications, and decrease the latency between the application and the storage.
- **Hot/Cold storage**. Data placement and data migration between “Hot” and “Cold” storages using data popularity information
- **Data Carousel**. Use tape more effectively and actively in distributed computing context
- **Data format and I/O**. Evaluating new formats (f.e. parquet) and I/O performance for HENP data
- **Third Party Copy**. Improve bulk data transfers between sites and find a viable replacement to the GridFTP protocol
- **Operations Intelligence**. Reduce computing operations effort by exploiting anomaly detection, time series and classification techniques to help the operators in their daily routines, and to improve the overall system efficiency and resource utilization

Russian Data Lake Prototype



Rationale

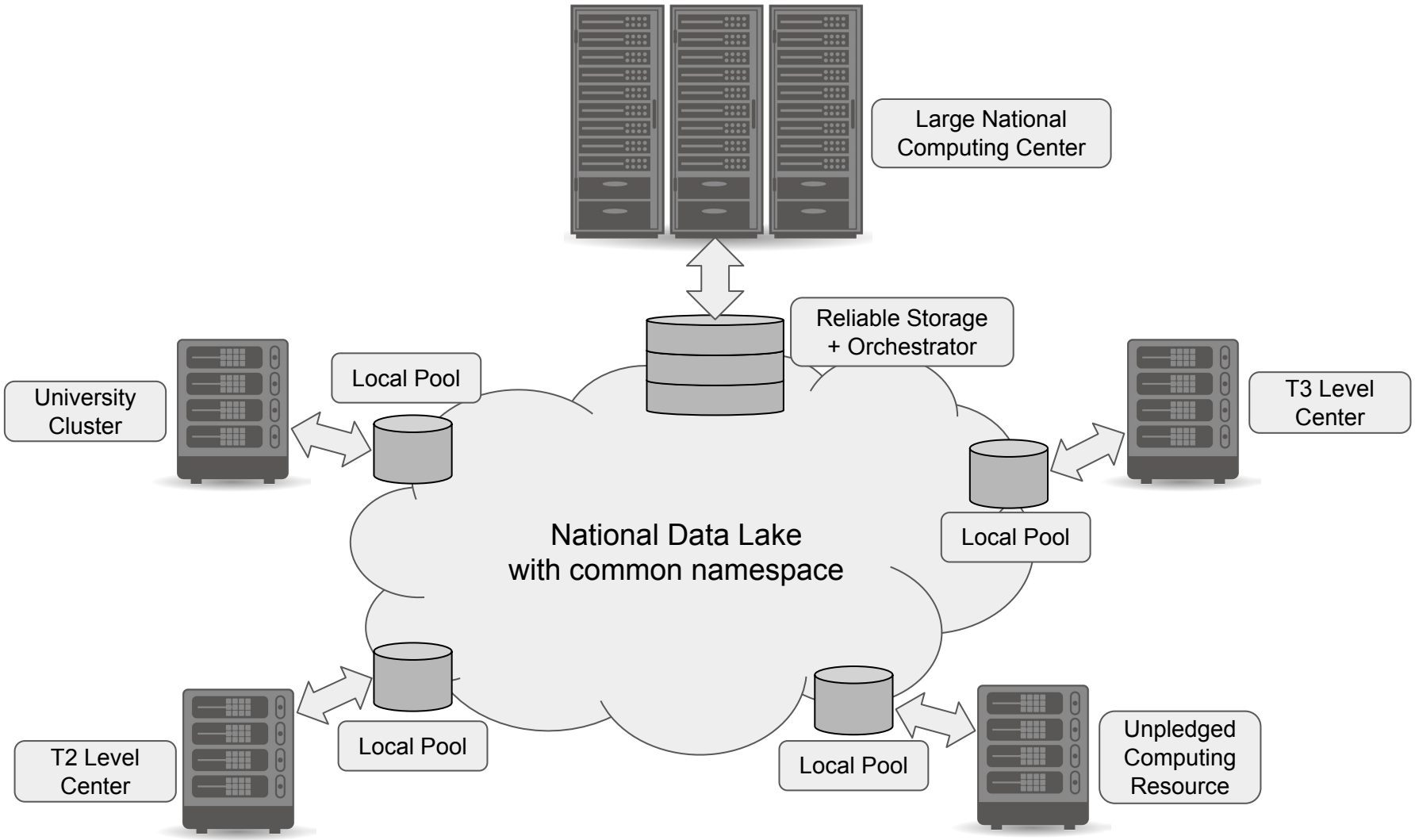
In the LHC Run 1-2 computing model experiment's payloads were reading and writing data to the site-local SE. Networks were (considered) not fast enough at the moment and this approach seemed reasonable. There were a couple of buts though:

1) Local SE reliability was crucial for any data processing
2) It was impossible to have a CPU-only site without any storage except for very special cases like large HPC facilities
3) This kind of storage policy may not be suitable for non-LHC workloads

Now when the network throughput is growing faster than CPUs and storage, payloads are no longer bound to local disk resources even for software distribution (CVMFS), remote I/O became standard practice, computational and storage resources are almost completely decoupled.

In the Data Lake scenario we want to use fast local disks (not a full-fledged SE) to minimize data transfer time for both read- and write-oriented payloads. The payload still accesses the Data Lake directly with local disk resources being transparent and optional, but real data transfers are routed in a fastest way possible. Most of the data I/O with remote storages is happening in the background freeing the CPU time on the worker nodes.

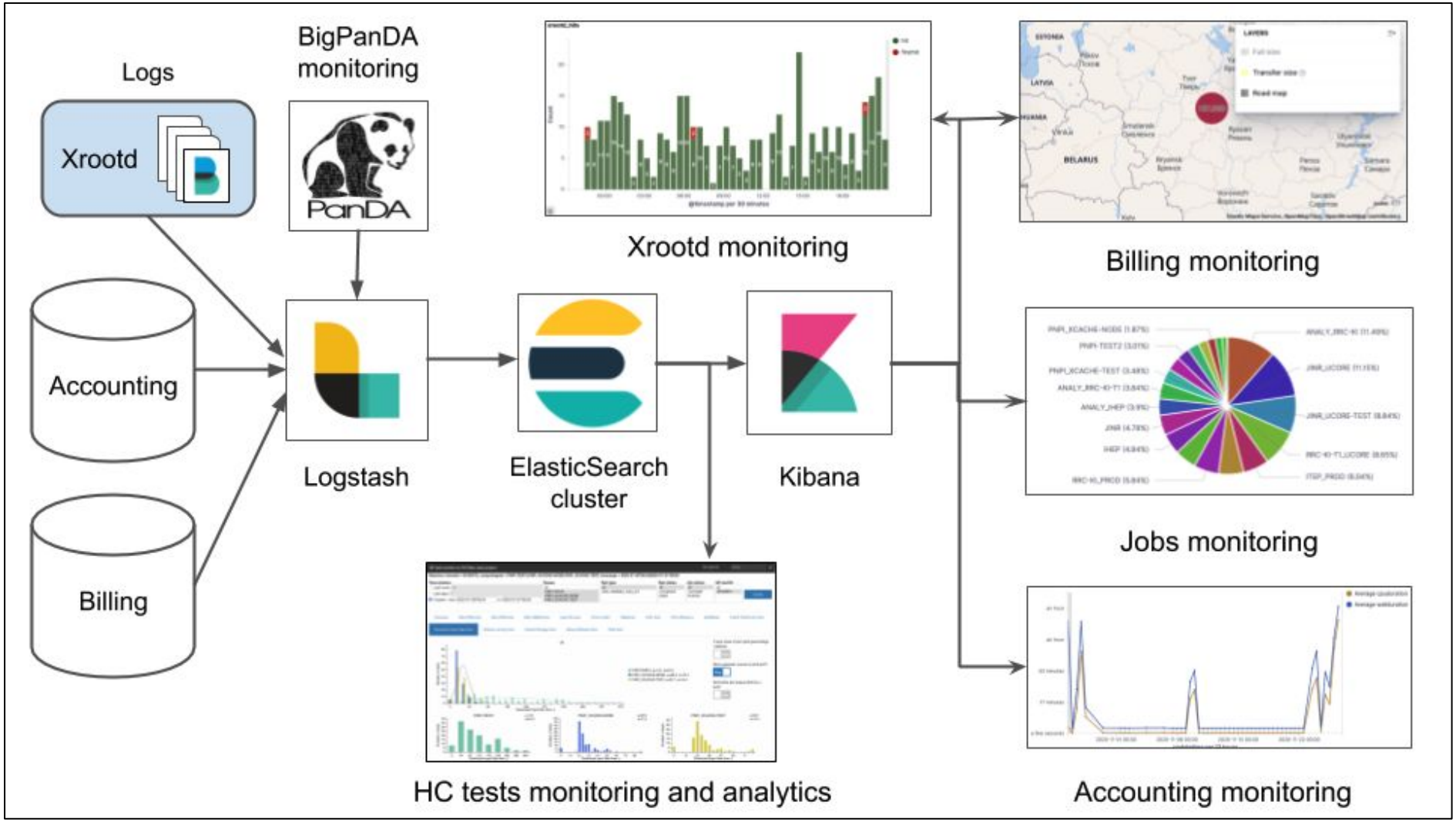
Data Lake with CPU-oriented smaller sites



Russian Data Lake Prototype Building Blocks

- Resources
  - Bare-metal at JINR and MEPhI
  - Virtualized at PNPI and REU
- Storage systems
  - EOS
  - dCache
  - XCache (Xrootd based)
- Payloads configuration, submission and testing
  - Custom synthetic tests
  - PanDA and ProdSys2 (ATLAS)
  - HammerCloud
  - CRIC (former AGIS)
- Monitoring infrastructure
  - perfSONAR
  - Logstash
  - ElasticSearch
  - Kibana
  - Custom web apps

Russian Data Lake Prototype Monitoring



- Four primary data sources: BigPanDA, Xrootd logs, Accounting and Billing databases (dCache)
- Five monitoring views: Xrootd, Billing, Jobs, Accounting , HammerCloud

Web interface for HammerCloud monitoring

- Due to some limitations of Kibana it was decided to create a custom web application for HammerCloud monitoring
- Monitoring interface allows the user to interactively select test jobs by a rich spectrum of criterias: execution time, queue, test type, test status, job status and HC test id

Read-oriented payloads: Caching

We will show some of the possible ways of optimizing remote data access from the worker nodes in somewhat small T2/T3 setups or dynamically scaled containerized deployments for physics analysis payloads.

This kind of deployment implies the necessity of heavy site-remote read-biased data I/O and time slot (t) allocated for analysis job is normally split into three phases (disregard some overhead):

1. input read (t_1)
2. compute (t_2)
3. output write (t_3)

Sometimes analysis payloads can read and write data while performing computation which makes it hard to separate t_1 from t_2 and t_2 from t_3, but in any case at least some data needs to be preloaded before computation can start.

Here we will focus on optimizing t_1 and thus improving CPU utilization of a compute resource with the help of XCache.

ATLAS Data Popularity in Users Analysis

DAOD metrics 2/2

Users Analysis

data DAOD datasets (deriv only)

- max used : #181
- #Tasks used DAOD datasets as input

Not used	1	2	3	4+
25122	11952	6573	5635	52998
< 7 days	**1-2 weeks**	**2 -4 weeks**	**1-3 months**	**3+ months**
11318	22400	65969	376831	751417

MC16 DAOD datasets (deriv only)

- max used : #1170
- #Tasks used DAOD datasets as input

Not used	1	2	3	4+
73721	25602	16762	14019	111403
< 7 days	**1-2 weeks**	**2 -4 weeks**	**1-3 months**	**3+ months**
400329	68483	151691	581119	957728

- Delta = [dataset used as input] – [dataset

7

8/06/19

ALL Numbers for prun/pathena (credit to T.Maeno)

There's at least one dataset that was accessed 1170 times.

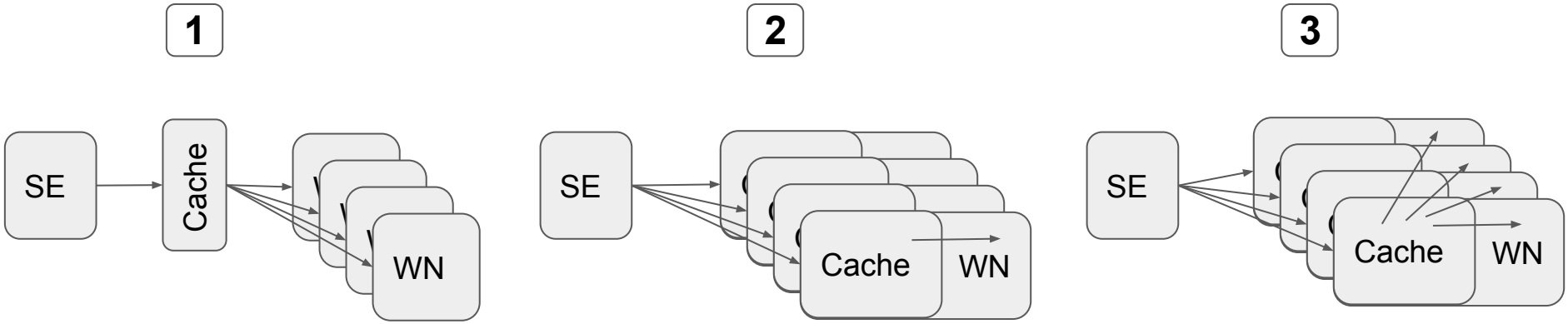
At the average ATLAS datasets consist of 50 files.

=> Each file in this dataset was accessed at least 20 times if data popularity is evenly split between these files.

Possible caching configurations

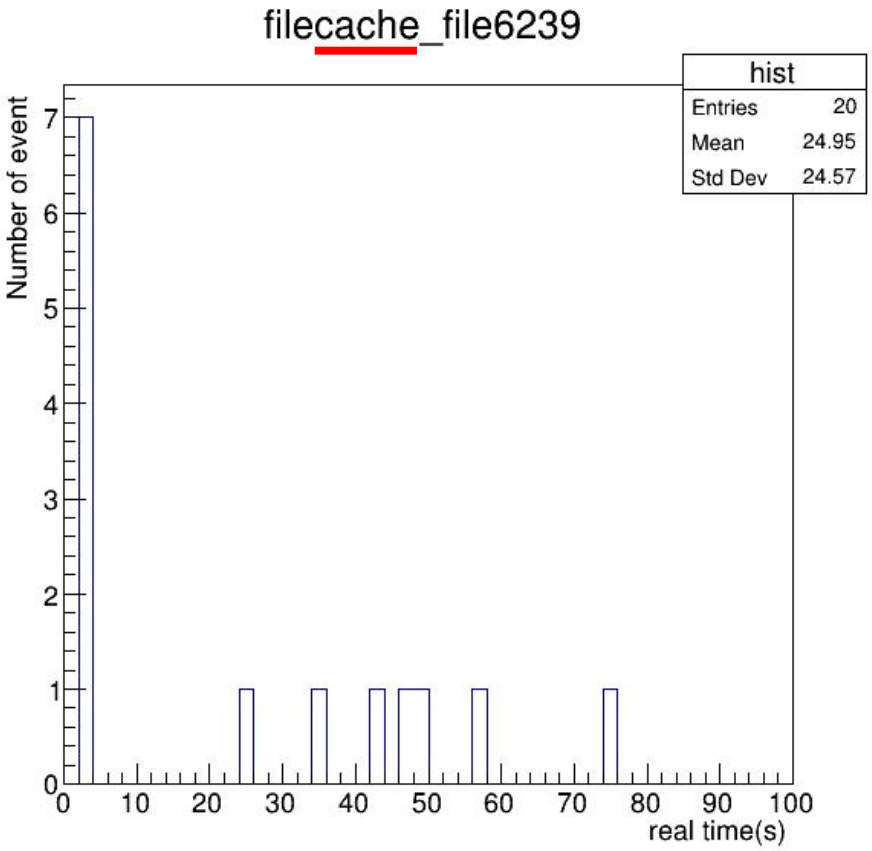
There's no one-size-fits-all solution because of hardware (especially network throughput and configuration) differences on different sites. We were testing three pretty obvious scenarios:

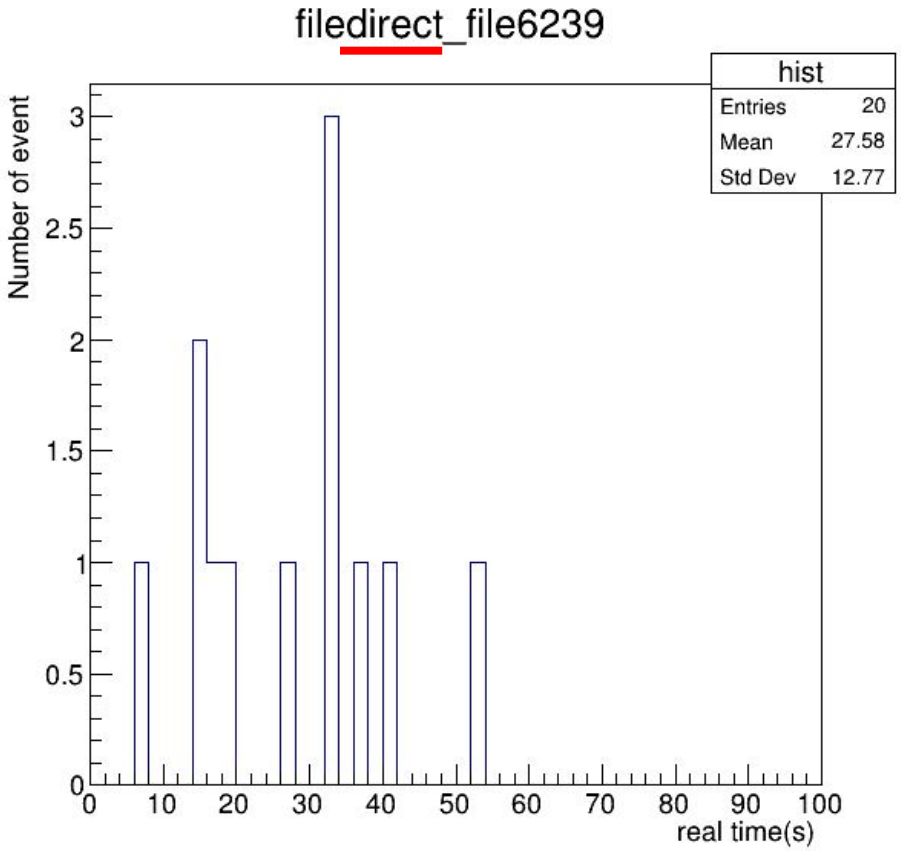
1. A single dedicated cache server (poor external network, good internal)
2. A local isolated cache on every worker node (good external network, poor internal)
3. A shared cache between worker nodes (external and internal networks of the same quality) – *requires some sort of service discovery*



Synthetic tests (local xcache vs direct access) at PNPI

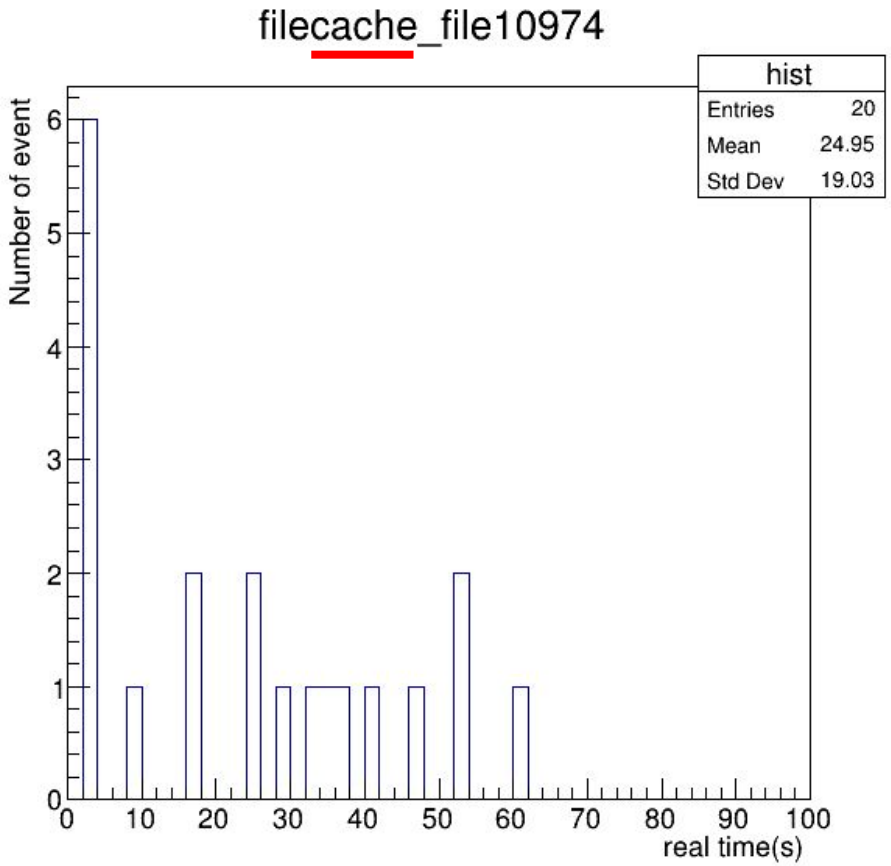
We submit 20 jobs into the queue, each of which reads one (always the same) file via **local cache**. 12 jobs run in parallel on 4 nodes with independent local caches, each node downloads a file from external storage. This caching scheme shows minimal performance gain in this test.

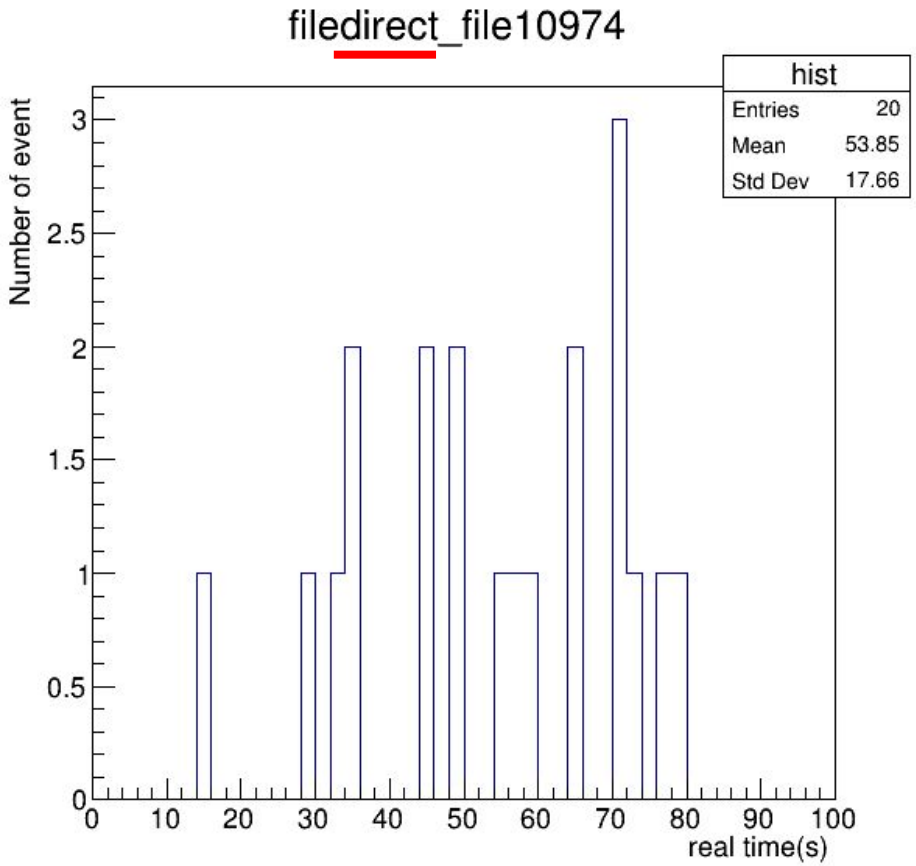




Synthetic tests (dedicated xcache vs direct access) at PNPI

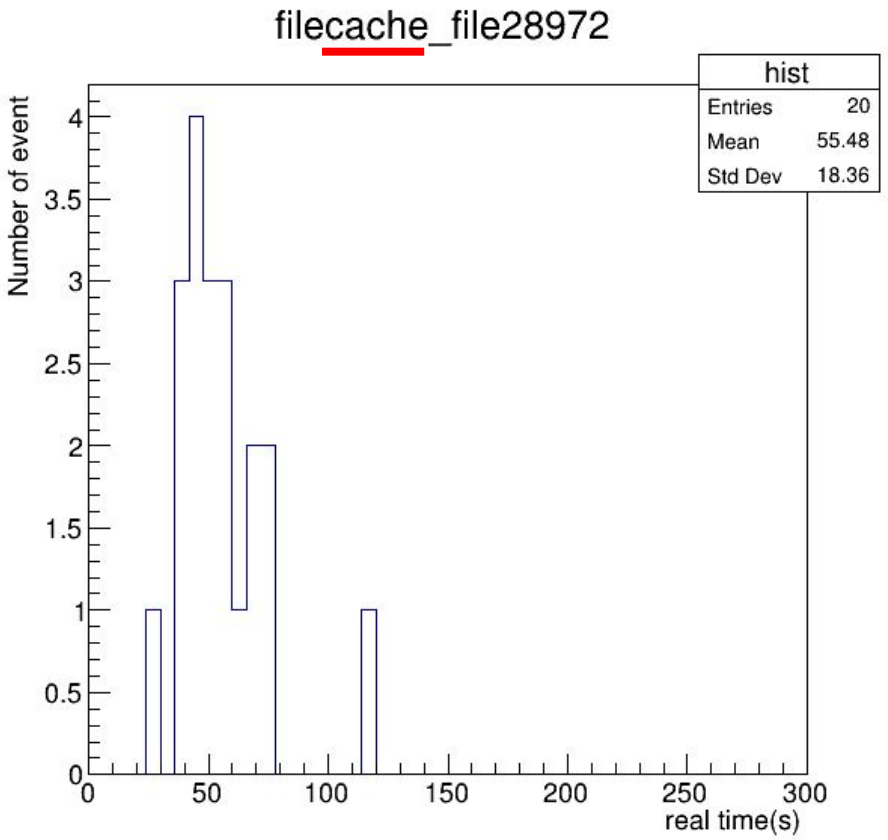
We submit 20 jobs into the queue, each of which reads one (always the same) file via **dedicated common cache**. 12 jobs run in parallel on 4 nodes, the file is downloaded from external storage only once. This time performance gain is more visible.

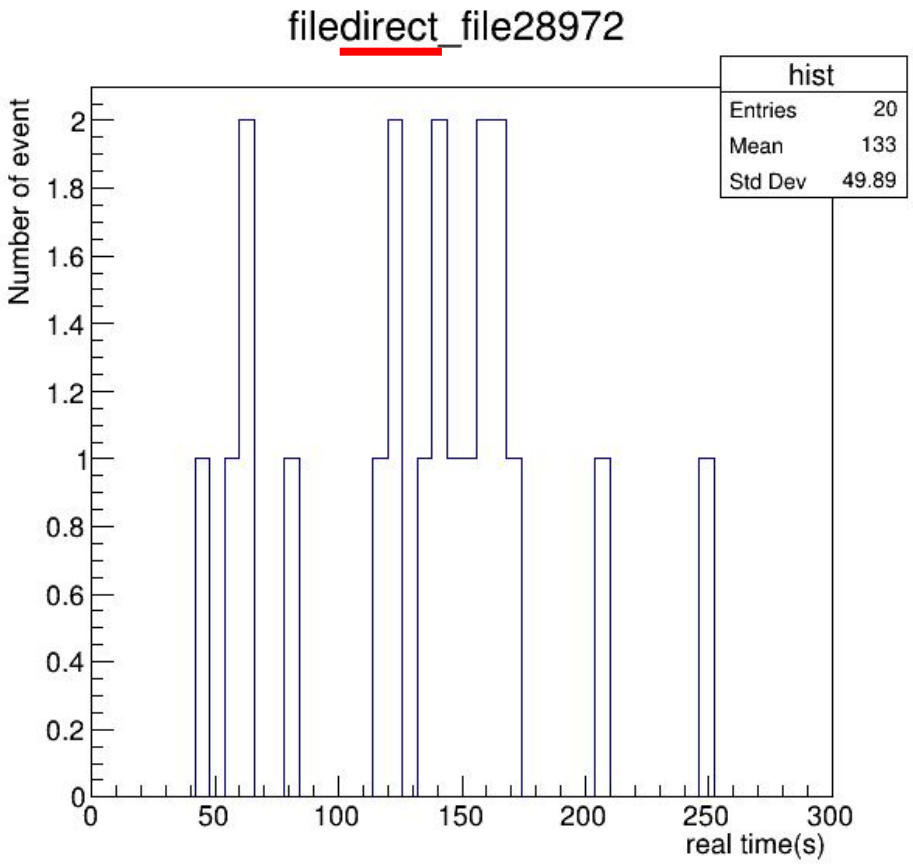




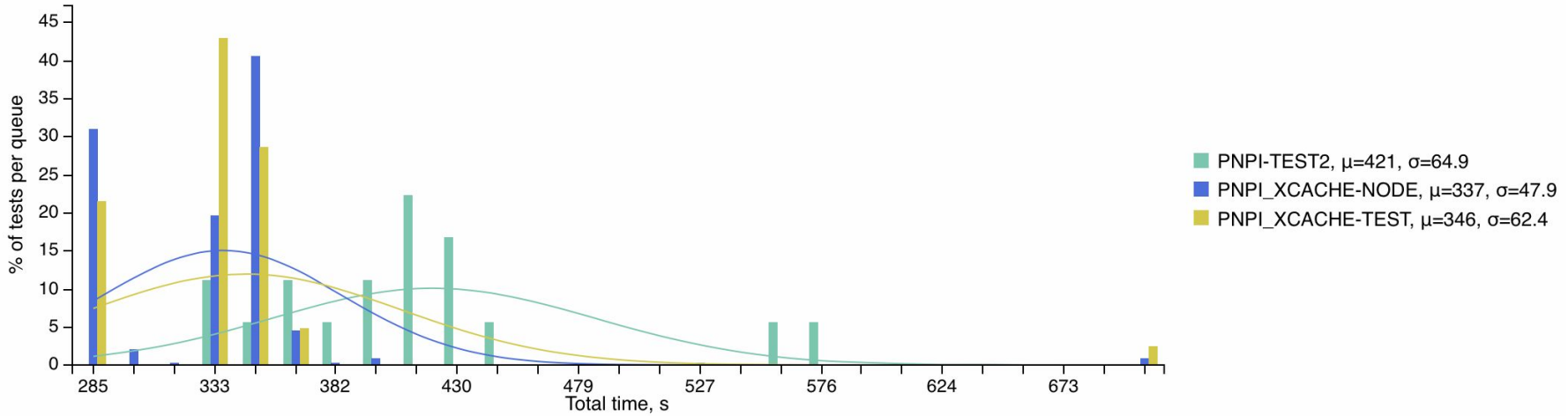
Synthetic tests (dedicated xcache vs direct access) at MEPhI

We submit 20 jobs into the queue, each of which reads one (always the same) file via **dedicated common cache**. 12 jobs run in parallel on 4 nodes, the file is downloaded from external storage only once. This time performance gain is more visible.





Total time distribution of read tests at PNPI using HammerCloud



PNPI-TEST2 – no cache

PNPI_XCache-NODE – distributed cache

PNPI_XCache-TEST – dedicated cache

Cached read provides 1.2x speedup

Write-oriented payloads: Buffering

Here we will focus on optimizing t_3 (see slide 7). In order to save CPU time we need to offload files to the remote storage in the background, and we've found a way to achieve this using a combination of two built-in EOS features:

- LRU Engine: https://eos-docs.web.cern.ch/configuration/lru.html
- Converter Engine: https://eos-docs.web.cern.ch/configuration/converter.html

The docs say that one can create an LRU policy for a directory which will change the placement policy for all files with a given properties. So, we've come up with the following scenario:

- We create a dedicated directory in the global namespace
- Default placement policy for this directory is to store all files on the site-local pool using GeoTag
- LRU will pick up files that were created at least 10 seconds ago and change their placement policy to a new one
- New placement policy will force flles to migrate to the remote pool

Configuration details

In order to fulfill our scenario we've defined two groups in EOS:

1) Group one contains both site-local and remote pools with different GeoTags
2) Group two contains only remote pools

Baseline test were performed on a directory with the following attributes:

```
sys.forced.group="2"
```

Then we've switched to a different directory with the following attributes:

```
sys.conversion.*="gathered:RU::JINR::LITDVL"
sys.forced.group="1"
sys.forced.space="default"
sys.lru.convert.match="*:10"
```

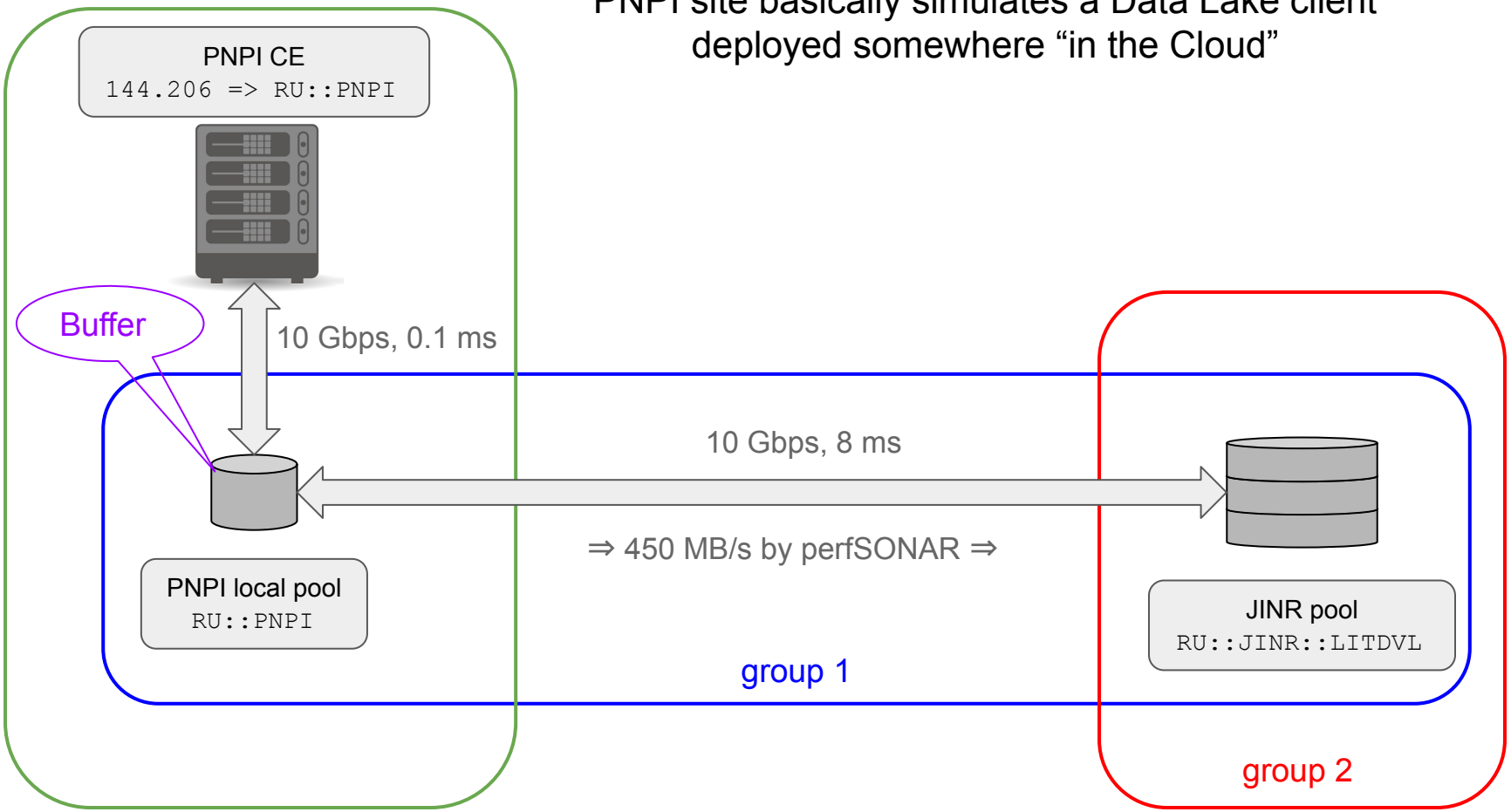
← *This is needed for Converter*

LRU was configured to run every minute

```
lru.interval := 60
```

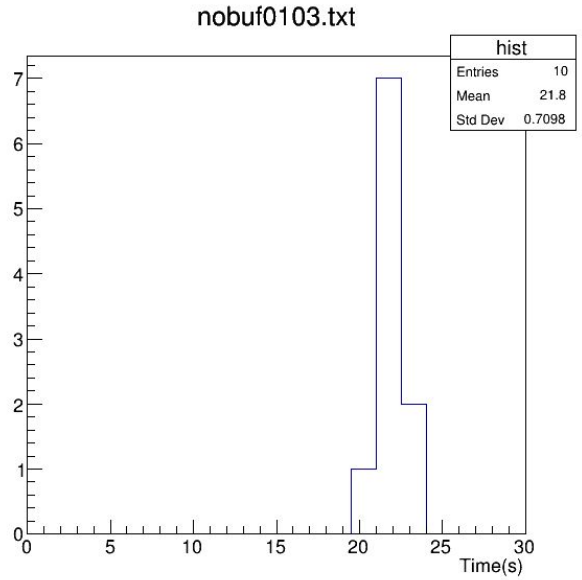
Testbed Configuration for Buffering

PNPI site basically simulates a Data Lake client deployed somewhere “in the Cloud”

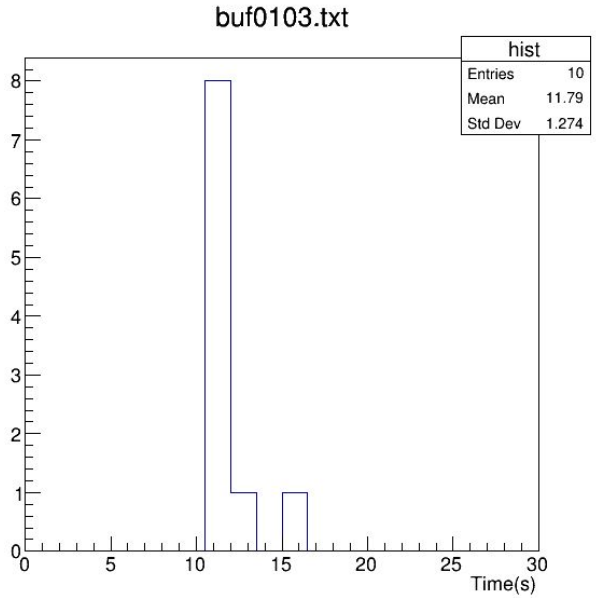


Tests and Results

Our synthetic test job generates ten 100MB files and successively uploads them into specified location using xrdcp. We were running 10 of these on different WNs at PNPI site.



Direct upload to JINR (left plot) takes 22 seconds



Upload to PNPI Buffer (right plot) takes 12 seconds

Buffered write provides 1.8x speedup

Issues

- HammerCloud tests are mostly functional tests now, they ought to be reconfigured to check the site efficiency in the scope of this research. HammerCloud have all the necessary tools and interfaces.
- XCache VOMS authorization requires some external packages and libraries, which are not included in the official Xrootd distribution. This may cause incompatibilities with the future versions of Xrootd.
- Our tests have demonstrated some issues with EOS LRU, e.g. ~10% of the files were not properly migrated from the buffer. More testing and tuning seems necessary.
- HammerCloud requires registration of sites and storages in CRIC (former AGIS). So far this has caused us quite a lot of headache (imagine ATLAS production suddenly hammering your resources in the middle of testing). There's definitely a room for improvement here.
- So far caching and buffering are implemented using completely different configurations and even storage systems.

Conclusions

- A functional Data Lake prototype is built with Russian sites
- A monitoring system is setup and functioning. The most important Data Lake metrics are monitored and controlled
- At this stage several different architectures have been configured and tested:
  - Two types of read cache with XCache
    - Dedicated cache
    - Distributed cache
    - Caching performance benefits have been demonstrated for
      - Synthetic tests
      - Real-life ATLAS analysis workloads
  - EOS-based write buffering
    - Buffering performance have been demonstrated for synthetic tests
    - Tuning and tests are still ongoing

Near-term plans

- Extend our prototype with more sites
- Run multiple workflows and production style tasks in addition to HammerCloud tests
- Evaluate TPC and non-x509 authentication to Data Lake access mode
- Provide a unified deployment recipes
- Demonstrate Data Lake beyond HENP data and applications

Acknowledgements

This research is conducted in collaboration with European Data Lake Project, which is part of DOMA initiative.

This project in Russia is supported by Russian Science Foundation award No. 19-71-30008 (research is conducted in the Plekhanov Russian University of Economics).

Thank you!