Federated data storage evolution in HENP: data lakes and beyond

Andrey Zarochentsev, Xavier Espinal, Andrey Kiryanov, Alexei Klimentov, Jaroslava Schovancova.
LHC Run 3 and HL-LHC Run 4 Computing Challenges

- HL-LHC storage needs are above the expected technology evolution (15%/yr) and funding (flat).
- We need to optimize storage hardware usage and operational costs.
Why Data Lake?

- HL-LHC will be a (multi) Exabyte challenge.
- The WLCG community needs to evaluate LHC computing model to store and manage data efficiently.
- The technologies that will address the HL-LHC computing challenges may be applicable for other communities to manage large-scale data volume (SKA, DUNE, CTA, LSST, BELLE-II, JUNO, etc). Co-operation is in progress.
- We see the Data Lake model as an evolution of the current infrastructure bringing reduction of the storage costs.
Requirements for a future data storage infrastructure

WLCG has the following implementation requirements:
- Common namespace and interoperability
- Coexistence of different QoS
- Geo-awareness
- File transitioning based on namespace rules
- File layout flexibility
- Distributed redundancy
- Fast access to data, latency compensation
  - File R/W cache
  - Namespace cache
The prototype of Data Lake: EUlake

The prototype of Data Lake should allow to revise concepts of redundancy, caching, interoperability and reproducibility. This should give us some of the hints to address the future of data storage in scientific computing.
A short retrospective (1)

1. 2008: Distributed dCache for NDGF
2. 2010: AAA – CMS Federated Storage
3. 2010: FAX – ATLAS Federated Storage
4. 2013: CERN distributed T0 – 2 computing centres 1200 km apart with 50:50 distribution of EOS-managed disk resources
5. 2015: Russian Federated Data Storage prototype – 8 centres, 2 major locations (SPb and Moscow), alike resources, EOS & dCache
6. 2018: ATLAS/Google “Data Ocean” project – cloud computing can offer attractive solutions and we can learn from industry leaders.
7. 2018-now: EULake – many centres at different locations (CERN, Russia, Spain, Netherlands, Australia, UK), a spectrum of disk resources from 1TB to 5 PB
A short retrospective (2)

- CERN T0 successfully operated during LHC Run2
- Russian Federated Storage Project was one of the first realistic prototypes for geographically distributed sites with WAN interconnection
  - Revealed many small issues (reported and fixed)
  - Both EOS and dCache
  - Tested with synthetic tests and ATLAS/ALICE real-life applications
  - Testbed resources were somewhat limited
- ATLAS/Google “Data Ocean”
  - Main tracks of PoC phase: Integration with ATLAS data & workflow management stack.
  - Data: Google Cloud Storage as a standard Rucio Storage Element, 3rd party transfers through FTS, download & upload, accounting.
  - Jobs: schedule jobs to Google Compute Engine through Harvester.
  - End user analysis.
- EULake is currently at the level of Proof-of-Concept and is heavily tested with various synthetic and real-life tests based on existing experience
Requirements from the “Federated Data Infrastructure in Russian Academic Cloud” R&D project (2015):

- Single entry point
- Should be usable by major “players” (at least 2-3 LHC experiments)
  - And it should be interesting to the future experiments beyond LHC
- Scalability and integrity
  - It should be easy to add new sites
- Data transfer optimization
  - Transfers should be routed directly to the disk servers avoiding intermediate gateways and other bottlenecks
- Stability & fault tolerance
  - Core components redundancy
- Built-in namespace
  - No dependency on external catalogues

Most of these requirements overlap with WLCG requirements for future storage infrastructure.
File placement by QoS

- **Hot custodial file (2 fast copies+archive)**
- **Warm custodial file (disk copy+archive)**
- **Cold custodial file (archive)**
- **Hot ephemeral file (2 fast copies)**
- **Warm ephemeral file ("Rain")**
Possible software stack (not exclusive)

- **Site-local storage systems**
  - **dCache** – version 2 had a weak internal security model, which required VPNs for distributed scenarios. Version 3 fixes that.
  - **EOS** – based on xrootd, works well with remote storages.
  - **CEPH** – advanced storage, supports different redundancy models, can be mounted as a filesystem, but does not handle long distance well enough.

- **Federation-oriented storage systems**
  - **xrootd** – does not have built-in namespace, needs external catalogues, used as a basis for ATLAS FAX and CMS AAA federations.
  - **DynaFed** – designed as a dynamic federation layer on top of HTTP/WebDAV-based storages, mostly useful for read access.
Mapping of requirements to existing EOS features

Data Lake architecture requirements:
- Common namespace and interoperability
- Coexistence of different QoS
- Geo-awareness
- File transitioning based on namespace rules
- File layout flexibility
- Distributed redundancy
- Fast access to data, latency compensation
  - File R/W cache
  - Namespace cache

EOS features:
- Built-in namespace
- Storage groups and catalog attributes
- Geotags and Geo-scheduling
- Catalog attributes
- Layout types and geo-scheduling
- Layout types (replica, RAIN)
- xCache (xrootd proxy)
- Slave MGMs
Data Lake and HammerCloud

- We integrated the Data Lake Prototype with HammerCloud
- We can test real workflows and data access patterns of ATLAS and CMS

Initial focus on ATLAS
(Data is copied from storage to WN)

3 test scenarios, stage-in from
1. Base: Local access (no data lake)
2. A: DLP, data @CERN, WN @CERN
3. B: DLP, data @RU, WN @CERN
Synthetic file locality tests

• The following combinations for layouts and placement policies were put in place and tested from a single client with geotag RU::PNPI:
  – Layouts: Plain, Replica (2 stripes), RAIN (4+2 stripes)
  – Placement policies: Gathered, Hybrid, Simple (based on client geotag)

• Expected results
  – Availability of geo-local replicas should improve file read (stageing) speed
  – An ability to tie directories to local storages (FSTs) should improve write speed for files in such directories
100 MB file read/write tests with different layouts and placement policies (1)

- sys.forced.placementpolicy="gathered:RU":
  - sys.forced.layout=plain
  - Write
    - Replica counts
      - CERN::HU 1
      - RU::Dubna 55
      - RU::PNPI 44
  - Read

- no sys.forced.placementpolicy (based on client geotag)
  - sys.forced.layout=plain
  - Write
    - Replica counts
      - RU::PNPI 100
  - Read

- sys.forced.placementpolicy="gathered:RU":
  - sys.forced.layout=raid6
  - Write
    - Replica counts
      - RU::Dubna 300
      - RU::PNPI 300
  - Read

- no sys.forced.placementpolicy (based on client geotag)
  - sys.forced.layout=raid6
  - Write
    - Replica counts
      - CERN::HU 67
      - ES::PIC 69
      - RU::Dubna 229
      - RU::PNPI 235
  - Read
100 MB file read/write tests with different layouts and placement policies (2)

no sys.forced.placementpolicy (based on client geotag)
  sys.forced.layout=replica

  Write
  Read
  Replica counts
  CERN::HU 28
  ES::PIC 39
  RU::Dubna 16
  RU::PNPI 117

sys.forced.placementpolicy="gathered:RU":
  sys.forced.layout=replica

  Write
  Read
  Replica counts
  RU::Dubna 100
  RU::PNPI 100

sys.forced.placementpolicy="hybrid:RU":
  sys.forced.layout=replica

  Write
  Read
  Replica counts
  CERN::HU 22
  ES::PIC 44
  RU::Dubna 64
  RU::PNPI 70

Observed replica scattering in a couple of days.
Read is always redirected to the closest server.
RAIN impacts I/O performance the most.
Subtleties of EOS QoS features

- Geo-scheduling overrides Geotags on write (placement policy can cause a file to be placed on a random storage even if geotags are defined)
- Geo-scheduling and Geotags are respected on write but not during data storage (files can be reshuffled by the balancers in the background)
- One option is to disable group balancing, create multiple groups with alike resources and balance data only within a group if needed
File placement by QoS

- Orange: Hot custodial file (2 fast copies + archive)
- Red: Warm custodial file (disk copy + archive)
- Green: Cold custodial file (archive)
- Green: Hot ephemeral file (2 fast copies)
- Magenta: Warm ephemeral file (“Rain”)

In EOS:

- Group X replica 2 + CTA
- Group Y replica 3 + CTA
- Group Z plain + CTA
- Group W replica 2
- Group U RAIN

11.03.2019

ACAT 2019, Saas Fee
Summary

- Data Lake is currently operational as a proof-of-concept
- Real-life (production) tests revealed some problems in such a complicated system
- We need to have instrumentation that pins data to specific storages in the long term
Future plans

- Extensive testing of different types of QoS (possibly simulated) with different storage groups
- Exploit different caching schemes
- Deploy and test slave replicated MGMs
- Test automatic data migration
- Evolve the infrastructure from a simple Proof-of-Concept to an infrastructure capable of measuring performance of future possible distributed storage models
Thank you for attention
100 MB file read/write tests with different layouts and placement policies (1000 repetitions) statistic(1).

sys.forced.placementpolicy="gathered:RU":
sys.forced.layout=plain

no sys.forced.placementpolicy (based on client geotag)
sys.forced.layout=plain

sys.forced.placementpolicy="gathered:RU":
sys.forced.layout=raid6

no sys.forced.placementpolicy (based on client geotag)
sys.forced.layout=raid6

ACAT 2019, Saas Fee. Andrey Zarochentsev
100 MB file read/write tests with different layouts and placement policies (1000 repetitions)

Statistic(2)

Observed replica scattering in a couple of days

ACAT 2019, Saas Fee. Andrey Zarochentsev
100 MB file read/write tests with different layouts and placement policies (1000 repetitions) statistic(3)

- sys.forced.placementpolicy="hybrid:RU"
- sys.forced.layout=plain

**Write**
- CERN::9918 39
- ES::PIC 31
- NL::NIKHEF 1
- NL::SARA 2
- RU::Dubna 190
- RU::PNPI 737

**Read**

**Write**
- CERN::9918 560
- ES::PIC 721
- RU::Dubna 2718
- RU::PNPI 1995

**Read**
Data Access on a Data Lake straw model (WLCG-DOMA)

AC - Archive center Defined as Tape or tape-equivalent-QoS enabled center able to archive custodial data.

DCC - Data and computer center providing disk-equivalent QoS storage

CCC - Compute center with cache

CCDA - Compute center without cache: relies on accessing all data via the network from either a CCC or a