# Customizing RUCIO at LCLS

4th RUCIO Community Workshop
Kenny Lo, Wilko Kroeger, Andrew Hanushevsky, Wei Yang
September 28, 2021, CERN

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

# LCLS - Linac Coherent Light Source

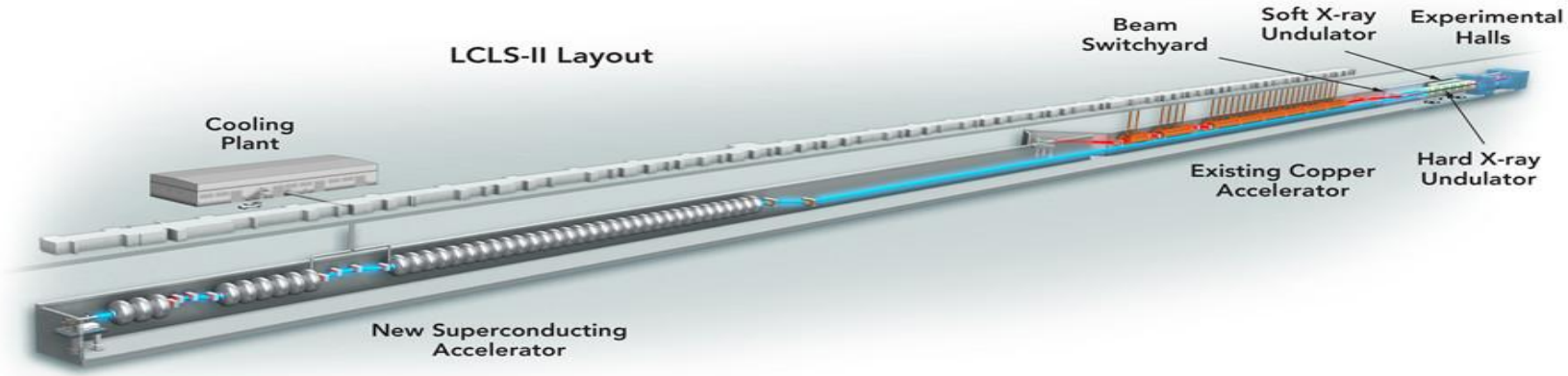
**The LCLS is a free electron laser that produces ultra fast X-ray pulses.**

**LCLS has already had a significant impact on many areas of science, including:**

- ➔ Resolving the structures of macromolecular protein complexes that were previously inaccessible
- ➔ Capturing bond formation in the elusive transition-state of a chemical reaction
- ➔ Revealing the behavior of atoms and molecules in the presence of strong fields
- ➔ Probing extreme states of matter
- ➔ Covid-19 research, e.g. imaging of SAR-CoV2 spike protein

# LCLS-II

- LCLS has been operating since 2009 using SLAC's warm copper linear accelerator
- LCLS has provided a wide spectrum of exiting scientific results
- For LCLS-II a new superconducting accelerator is being build and the science program is scheduled for 2022
- LCLS-II will provide a jump in capabilities allowing experiments that were not possible so far.
- The pulse rate will jump to about 1MHz from the current 120Hz of the copper linac.
- Data rates will jump from a few GB/s to TB/s  (data written out will be smaller)
- Both accelerators will be operated in parallel

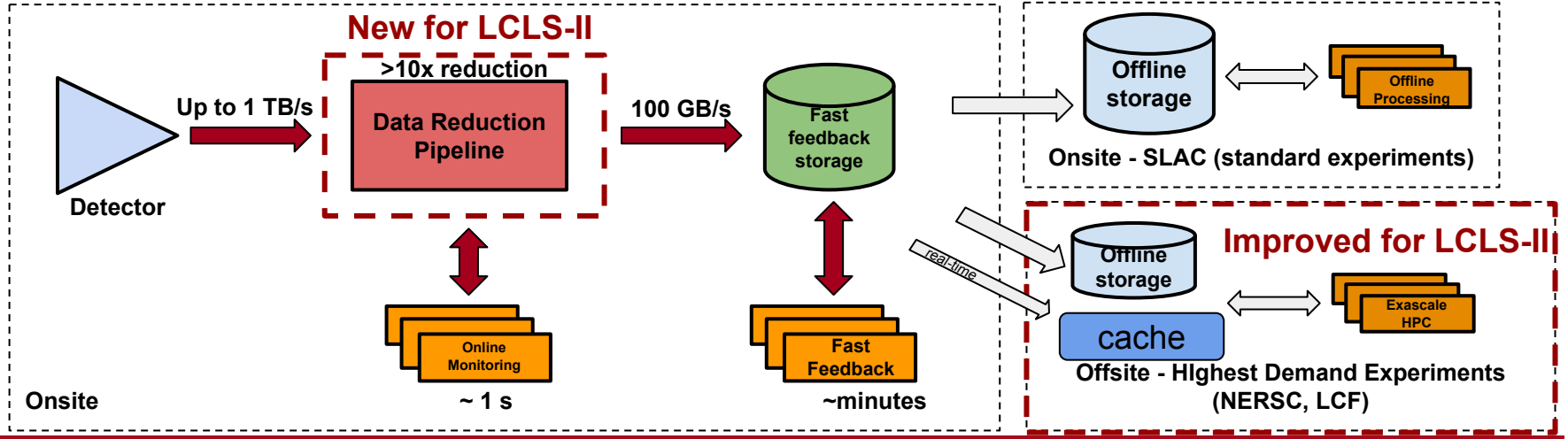


# LCLS-I in Numbers

- about 2000 registered LCLS users
- about 25% of proposed experiments are accepted
- 150-200 Experiments per year
- 2-3 PB and 350-450K files per year
- 20 PB of archived data in 2.5 million files collected since 2009
- 250K Runs in total
- ~50M key/value pairs for all Runs
- 4.5 PB storage for raw data
- 1.2 PB  storage for user writable files (scratch, results)
- 120 compute nodes with 1600 cores
- For LCLS-II most of these numbers will go up by x10 - x100

# LCLS-II Data System

- Similar to LCLS-I but:
  - Data rates will increase from few GB/s to up to 1 TB/s
  - **Data needs to be reduced as we can not write everything to disk**
- Events are not built before being written to disk. The detector contributions to an event are distributed over many files.
- Streaming to remote HPC sites
- Expect single file write rate to increase from 200MB/s to ~1-2GB/s

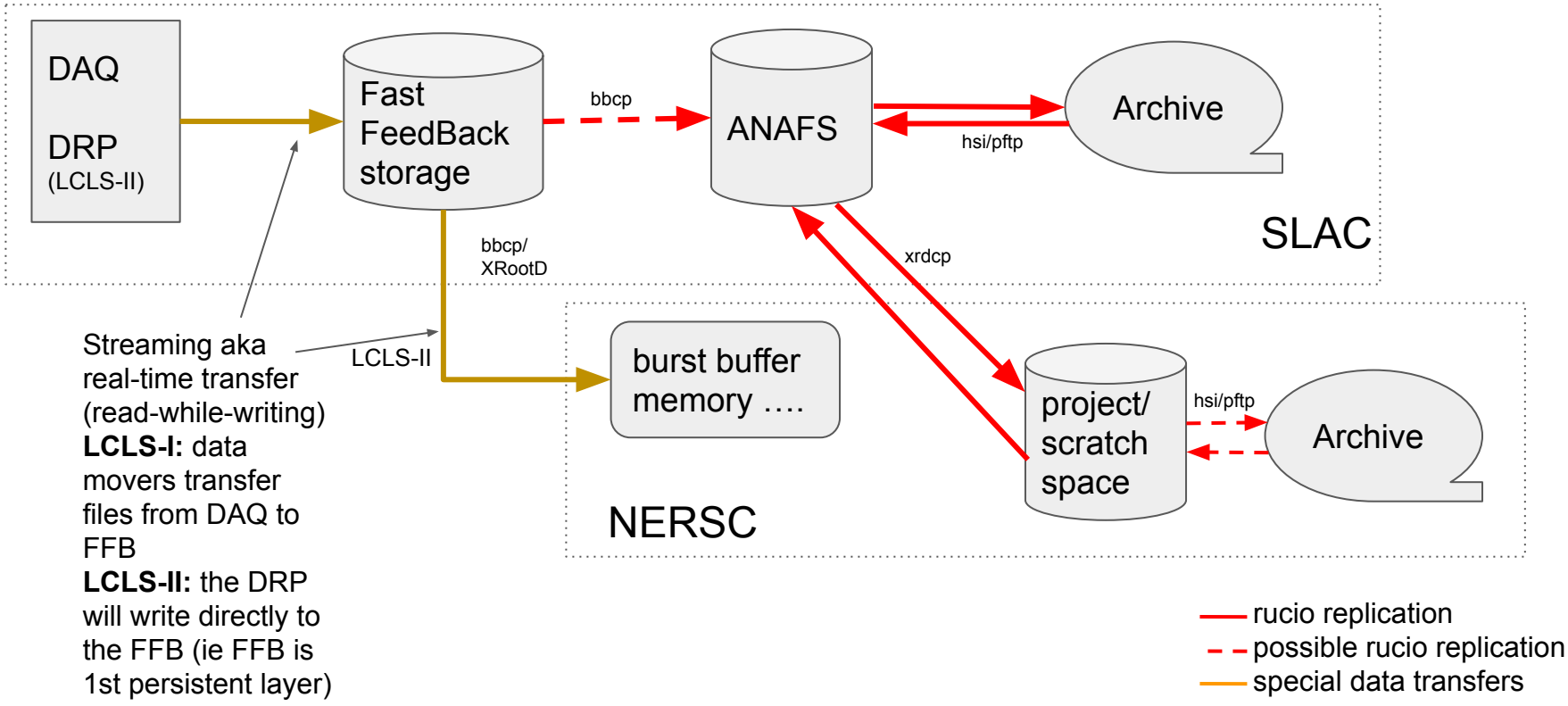



**Data reduction mitigates storage, networking, and processing requirements**

# LCLS Data Management

- The Data Management handles three different data types:
  - raw data in xtc format (grouped into runs, a.k.a **datasets** in Rucio)
    - most files are in this group
  - user daq data (not associated with a run)
    - only collected by a few experiments, files size typically small O(MB)
  - Secondary processed data (could be run associated)
    - example: process and converting raw data to hdf5 format.
- Users access files/runs using a parallel file system (Lustre), files are found by using a predefined path e.g.: /cds/data/psdm/<instr>/<exper>/xtc.
  A user is not directly accessing the file-catalog.
- User created files (during processing/analysis) are not managed by us we just provide disk space

# LCLS - Data Transfers/Replication




**LCLS-I:** data movers transfer files from DAQ to FFB

**LCLS-II:** the DRP will write directly to the FFB (ie FFB is 1st persistent layer)

# Rucio Role in LCLS

- Data Catalog (File Manager)
- File Replication
  - Archive and restore from tape
- Purging files from disk
- Only be used for raw data
- Users don't have direct access to RUCIO
- File count and size
  - total size ~20PB, 2.5 million files
  - ~3PB/year with 300-400K files /year
  - the yearly rates will increase by x20 (2022) - x100 (2028)

| | |
|---|---|
| experiments | ~10K |
| runs (data sets) | millions |
| number of raw data files | 100's of millions over the next 5-10 yr. |

# Configuring Rucio Service Account

- Install git, docker, and docker-compose as pre-requisite tools in development machine
- Clone rucio repos in GitHub: rucio/rucio and rucio/containers
- Prefer SSH over ‘username/password’ authentication method
- As noted in the Policy packages section of Rucio Documentation:
  - rucio/lib/rucio/common/schema/lcls.py (atlas.py as template)
  - rucio/lib/rucio/core/permission/lcls.py (generic.py as template)
    - Set up SSH access for user (ssh-keygen credentials)
      - Add and define perm_get_auth_token_ssh(issuer, kwargs)
      - In has_permission(issuer, action, kwargs), add:
        - 'get_auth_token_ssh': perm_get_auth_token_ssh,
      - In the [client] section of user’s rucio.cfg :
        - auth_type=ssh
        - ssh_private_key=/root/.ssh/id_rsa,
  - rucio/lib/rucio/common/utils.py (implement **non-deterministic** SURL algorithm for LCLS)

# Non-deterministic SURL for LCLS

```
    def construct_surl_LCLS(dsn, filename):
        """
        Defines relative SURL for replicas. This method uses the LCLS convention
        for xtc files. To be used for non-deterministic sites.

        @param: filename of format <instrument>.<experiment><fld>.<remain>
        @return: relative SURL for new replica.
        @rtype: str
        """

        instr, expt, fld, remain = filename.split('.', 3)

        if fld == 'xtc' and filename.endswith('smd.xtc'):
            return '/%s/%s/xtc/smalldata/%s' % (instr, expt, remain)

        return '/%s/%s/%s/%s' % (instr, expt, fld, remain)

register_surl_algorithm(construct_surl_LCLS, 'LCLS')
```

Could have been easier if the scope of the raw data file's DID was made available in /lib/rucio/common/utils.py. (feature request submitted to Rucio team)

# Running RUCIO in Docker Containers

- **# docker build -t rucio/rucio-dev .** (in containers/dev) ⇒ ~900MB rucio/rucio-dev:latest image
- **# docker-compose --file docker-compose.yml up -d** (in rucio/etc/docker/dev)

| IMAGE | COMMAND | PORTS | NAMES |
|---|---|---|---|
| gitlab-registry.cern.ch/fts/fts-monitoring | "/usr/sbin/apachectl…" | 0.0.0.0:8449->8449/tcp | fts_webmon_1 |
| gitlab-registry.cern.ch/fts/fts-rest | "/usr/sbin/apachectl…" | 0.0.0.0:8446->8446/tcp | fts_rest_1 |
| gitlab-registry.cern.ch/fts/fts3:x509-scitokens-issuer-client | "/usr/bin/supervisor…" | 2170/tcp | fts_server_1 |
| mysql:5 | "docker-entrypoint.s…" | 3306/tcp, 33060/tcp | fts_mysql_1 |
| rucio/rucio-dev | "httpd -D FOREGROUND" | 0.0.0.0:80->80/tcp, 0.0.0.0:8443->8443/tcp, 443/tcp | dev_rucio_1 |
| rucio/rucio-ui | "/docker-entrypoint.…" | 0.0.0.0:8444->443/tcp | dev_rucioui_1 |
| postgres:11 | "docker-entrypoint.s…" | 0.0.0.0:5432->5432/tcp | dev_ruciodb_1 |
| graphiteapp/graphite-statsd | "/entrypoint" | 80/tcp, 2003-2004/tcp, 2013-2014/tcp, 2023-2024/tcp, 8125-8126/tcp, 0.0.0.0:8080->8080/tcp, 8125/udp | dev_graphite_1 |

- FTS service configured with xrootd service endpoints for replication
- PostgreSQL (conscious choice for open source DB) configured for the Rucio database backend.

# Validating and Debugging Rucio Installation

- # rucio --version ==> **1.26.1 (Stable)**
- User-oriented Web UI container
- The actual changes for enabling SSH are discovered through debugging the provided unit tests inside the Rucio container:
  - tools/run_tests_docker.sh (take time to run all tests; inundated the DB with test data for validating initial installation! Consider -i option to pick individual unit tests.)
- Run through the examples in the Rucio documentation
- [ToDo] Increase visibility into Rucio Server operation:
  - Run Flask with Rucio app in Debug mode
  - Remote debugger attached to Rucio server in container
    - Visual Studio Code (Free)
    - PyCharm Professional Edition ($)

# Replicating a File from Rucio

1. # rucio list-rse-attributes LCLS_DATA
2. LCLS_DATA: True
3. fts: https://<SLAC IP address>:8446
4. istape: False
5. naming_convention: LCLS
6. # rucio **upload** --rse LCLS_DATA --scope rte01 --name xtc.file.rte01-r0001-s02-c00.xtc --pfn <SLAC pfn> /tmp/rte01-r0001-s02-c00.xtc
7. # rucio **add-rule** psdm:rte.rte01.xtc.rte01-r0001-s02-c00.xtc 1 LCLS_NERSC (similar to LCLS_DATA)
8. Run the rucio daemons --run-once: **judge-evaluator**, **conveyor-[submitter**, **poller**, **finisher]**
9. # rucio **list-file-replicas** --pfns psdm:rte.rte01.xtc.rte01-r0001-s02-c00.xtc
    - root://<SLAC xrootd endpoint>//psdm/rucio//rte/rte01/xtc/rte01-r0002-s02-c00.xtc
    - root://<NERSC xrootd endpoint>//psdm/rucio//rte/rte01/xtc/rte01-r0002-s02-c00.xtc

- Reported BUG: **double slashes** needed in storage **prefix,** highlighted in RED above.
- Question/Issue: why **pfn** necessary (error otherwise) in rucio **upload** when LCLS_DATA already attributed LCLS as non-deterministic naming convention ?

# Testing RUCIO with FTS3 Transfers

- We were able to use FTS 3 with Rucio to drive Third Party Copy (TPC)
  - LCLS will use xrootd TPC with “sss” security module (Simple Shared Secret)
  - We use rucio-conveyor-poller
- FTS3 came in container form from Gitlab @ CERN, including:
  - FTS server (set up is a bit more generic than LCLS need)
    - support xrootd TPC and http TPC: sss and x509 (w/ macaroon for HTTP TPC)
    - The “sss’ key is bind mounted to the container.
  - FTS rest
  - FTS web monitoring
  - MySQL 5 backend
- One issue
  - Rucio uses an X509 proxy to submit to FTS3. This step doesn’t include delegation
  - We need to **manually** delegate a X509 proxy to FTS3, outside of Rucio.
  - Can we merge the two steps to one? Or better, can we submit without using X509?

# More Open Questions

- Better way (rather than using htar out of loop?) in Rucio to transfer large quantity of small files with tracking, especially to tape storage?
- Besides MD5 (option), ADLER32 (default) in Rucio, how about support for CRC32C (for hardware efficiency with Intel SSE 4.2)?
- How best to interface with HPSS tape systems,  e.g. commanding CLI tools?

# Next Steps

1. Test against the 3rd type of docker-compose w/ActiveMQ full monitoring stack.
2. Integration with HPSS tape archives
3. Provision and operate a much larger Rucio installation beyond the pilot stage, with sizable backend DB (and DBA support) mimicking production.

- Onboarding Notes: **https://github.com/slaclab/onboard-rucio.git**