ePIC

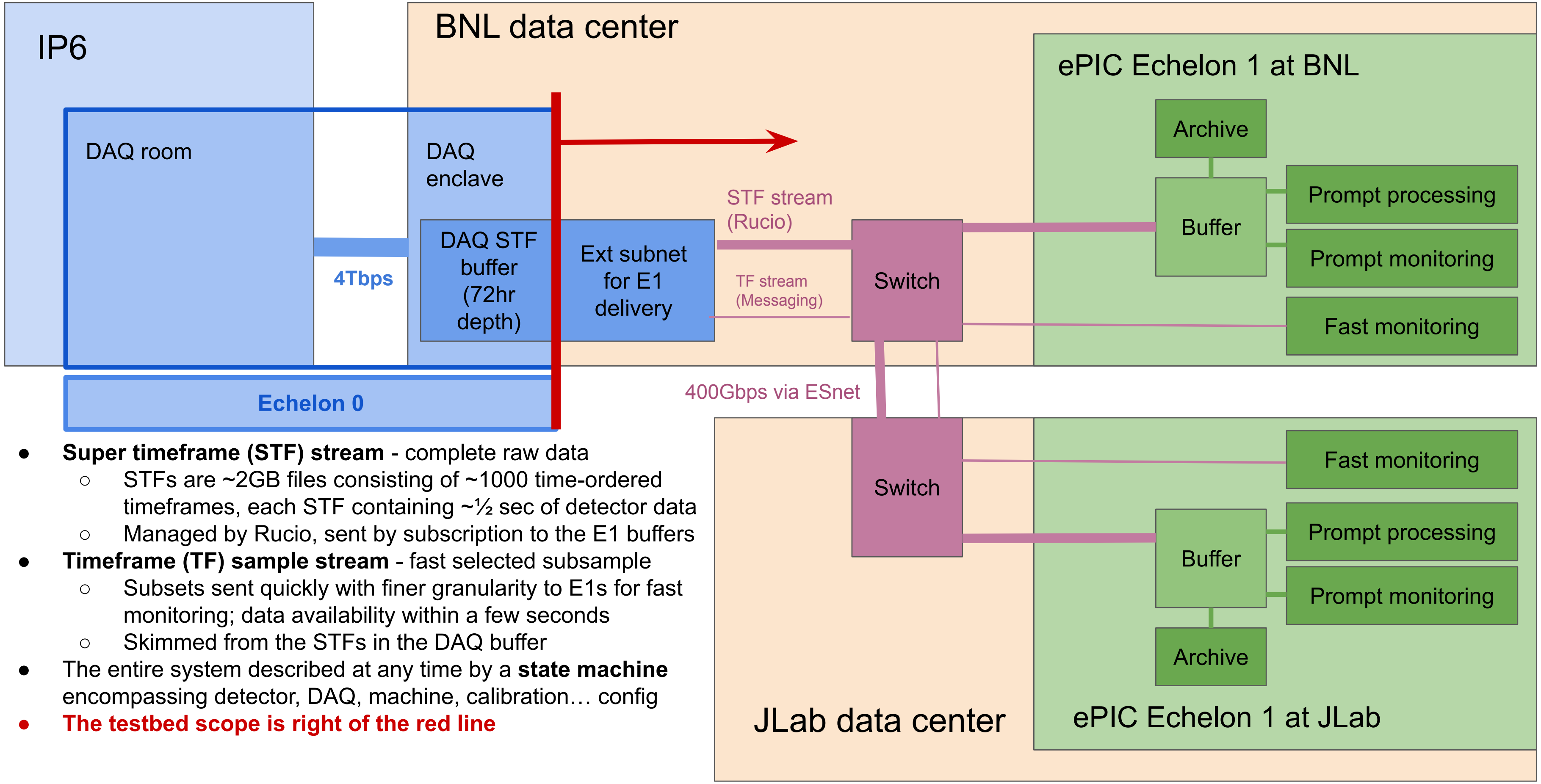
Orchestration of TF Processing with Panda and Rucio

Wen Guan, Dmitry Kalinkin, Maxim Potekhin, Michel Villanueva, Torre Wenaus, Zhaoyu Yang, Xin Zhao

BNL Nuclear and Particle Physics Software group (NPPS)

Jan 21, 2026
ePIC Collaboration Meeting

ePIC Echelon 0 - Echelon 1 workflows



- **Super timeframe (STF) stream** - complete raw data
  - STFs are ~2GB files consisting of ~1000 time-ordered timeframes, each STF containing ~½ sec of detector data
  - Managed by Rucio, sent by subscription to the E1 buffers
- **Timeframe (TF) sample stream** - fast selected subsample
  - Subsets sent quickly with finer granularity to E1s for fast monitoring; data availability within a few seconds
  - Skimmed from the STFs in the DAQ buffer
- The entire system described at any time by a **state machine** encompassing detector, DAQ, machine, calibration… config
- **The testbed scope is right of the red line**

Technology downselect for distributed data processing

1. We are leveraging the proven data distribution and management system – Rucio, which has been successfully used in major experiments (ATLAS, Belle II) over a long period of time.

2. PanDA is another proven component of the infrastructure being used, handling the distributed workload management. It’s a Rucio-aware system which opens up implementation patterns that are robust and require less code that would be needed otherwise.

3. BNL Scientific Computing and Data Facilities (SCDF) are hosting instances of both systems (Rucio and PanDA) which provide an efficient platform for development and testing for the ePIC Software and Computing Organization. Importantly, many of the SCDF and NPPS personnel have experience in these systems and their integration.

STF processing orchestration: an agent-based system

1. The principal design choice in the development of the STF processing orchestration was the use of agents, which are loosely coupled and utilize a neutral communication layer — currently based on ActiveMQ – to become an end-to-end processing framework.

2. For STF processing, the system relies on the following two agents:
   a. The Data Agent, whose role is to form datasets corresponding to relevant run periods, and to ship the data from the DAQ buffer to both parts of the Echelon 1. The operation of the Data Agent is controlled by the MQ messages received from the DAQ, reflecting the various states of the system.

   b. The Processing Agent, whose role is to create processing tasks within the PanDA system. It’s operation is controlled by the MQ messages received from the Data Agent.

STF processing orchestration: the testbed

To validate the design of processing orchestration, and evaluate its scalability, a project was started at BNL with the aim to develop a comprehensive testbed. The testbed leverages the existing test instances of Rucio, the PanDA server and ActiveMQ (in its Artemis version) deployed at the SCDF at BNL.

DAQSIM

1. Since the actual DAQ is not in place yet, it needs to be emulated to make the testbed functionally complete. For this reason, we included another agent – the “daqsim” – to perform the role of the data source emulation, creating files in its local storage – serving as a proxy for the DAQ buffer – and notifying other components of the system about the state of the detector and DAQ (simulated at this point), according to a predefined schedule/timeline, which is supplied in the YAML format which is human readable and easy to create and modify (see an example in the next slide).
2. The DAQSIM is based on the popular SimPy package which is well suited for simulation of time-dependent systems in Python.
3. The current state/substate of the emulator is set based on the “clock” that comes as a part of SimPy and the time points set in the schedule.

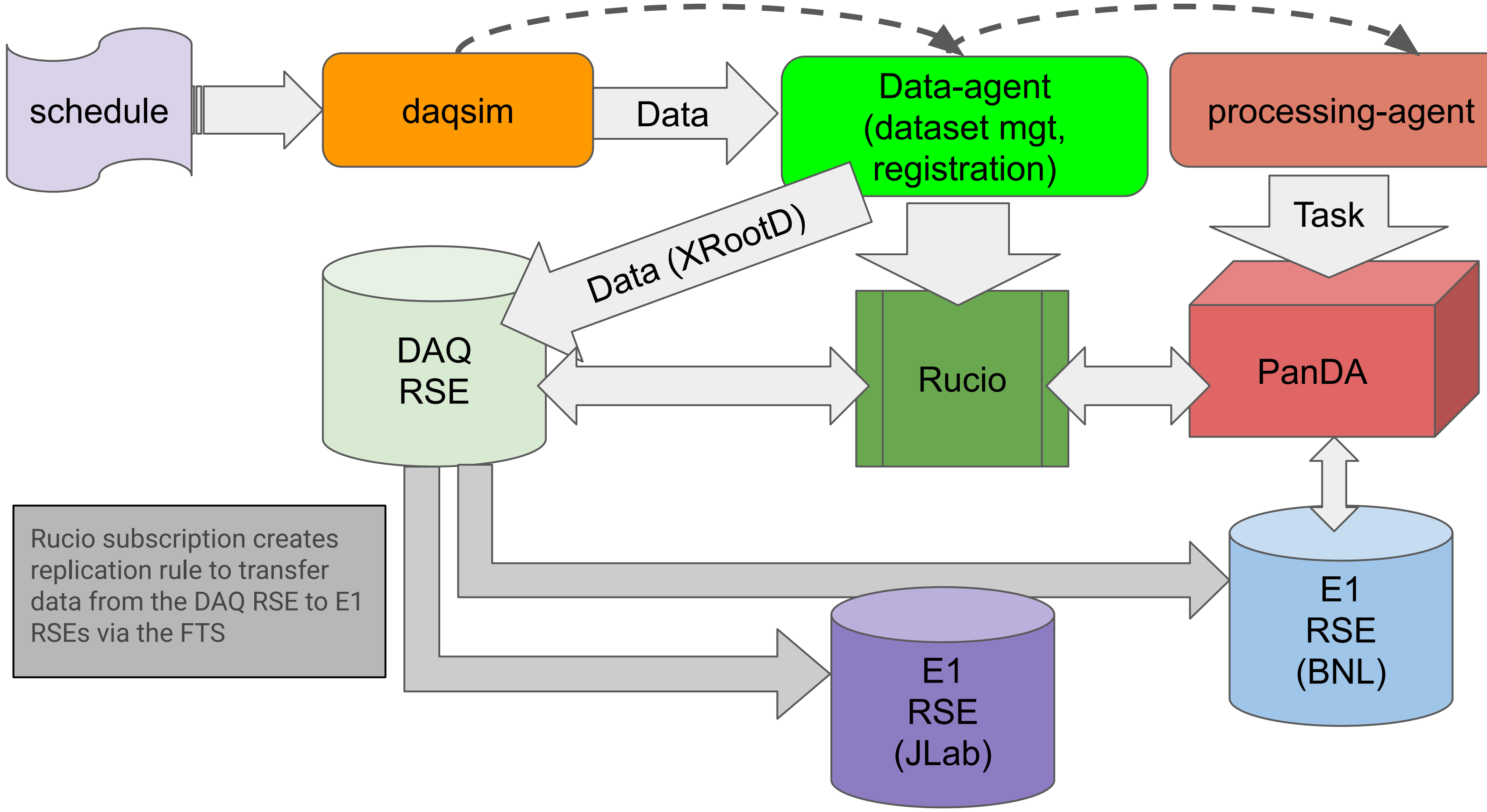
STF processing orchestration: the testbed schedule example

```
# The daqsim-agent schedule defines the states and substates of the DAQ system,
# the time spans for each state and substate, and the transitions between them.
# The schedule is used to control the simulation of the DAQ system, allowing it to move
# through different states and substates, simulating the data acquisition process.
# Time span (duration) format is tuple with no spaces: weeks,days,hours,minutes,seconds

- state:    no_beam
  substate: calib
  span:     0,0,0,0,10

- state:    beam
  substate: not_ready
  span:     0,0,0,0,10

- state:    beam
  substate: ready
  span:     0,0,0,0,10

- state:    run
  substate: standby
  span:     0,0,0,0,10

- state:    run
  substate: physics
  span:     0,0,0,1,0
```

STF processing orchestration: the testbed (cont’d)

The data transfer mechanism from the DAQ buffer to the two E1 endpoints has not being designed yet. Currently, it’s emulated by combining XRootD transfer with registration of the data in Rucio. This actions are performed using the Python APIs for XRootD and Rucio, respectively. A conceptual diagram of the testbed components is presented in the next slide. Dashed lines represents MQ messages.

STF processing orchestration: the testbed (cont'd)



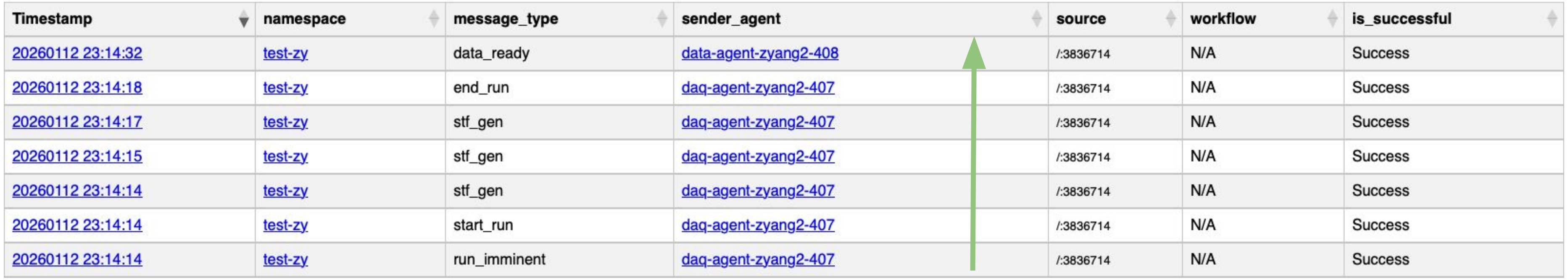
Rucio subscription creates replication rule to transfer data from the DAQ RSE to E1 RSEs via the FTS

STF processing orchestration: an example of the testbed run

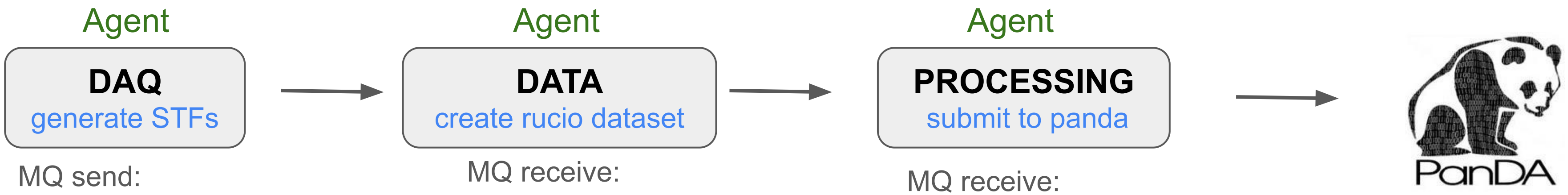
1. The **daqsim** agent – reads in the simulation schedule and initiates the timeline. Issues the run_imminent, start_run, end_run and other messages reflecting the state as defined in the schedule.
2. The data-agent — creates a Rucio dataset consistent with the announced run, upon receiving the “run_imminent” message. When receiving “stf_gen” messages from the daqsim (at each STF generation), initiates a XRootD-based upload to endpoints, and performs the registration of each file into the respective dataset. Generates the “data_ready” message, principally for the processing agent. Upon receiving “end_run” message, closes the dataset, which is important in the design of the processing-agent (next bullet).
3. The processing-agent: receives the “data_ready” message for the specific run, and actuates the PanDA client, instructing it to generate a PanDA task which is configured in such a way that it will keep processing STF files being continuously added to the given dataset, until such dataset is closed in Rucio by the data-agent.
4. Agents update their status in the monitoring web application.
5. See next slide for more information.

STF processing: full chain as reflected in the monitor

- The full chain is driven by ActiveMQ messages broadcasted to topic “epictopic”, consumed by all subscribers.



Timestamp	namespace	message_type	sender_agent	source	workflow	is_successful
20260112 23:14:32	test-zy	data_ready	data-agent-zyang2-408	/:3836714	N/A	Success
20260112 23:14:18	test-zy	end_run	daq-agent-zyang2-407	/:3836714	N/A	Success
20260112 23:14:17	test-zy	stf_gen	daq-agent-zyang2-407	/:3836714	N/A	Success
20260112 23:14:15	test-zy	stf_gen	daq-agent-zyang2-407	/:3836714	N/A	Success
20260112 23:14:14	test-zy	stf_gen	daq-agent-zyang2-407	/:3836714	N/A	Success
20260112 23:14:14	test-zy	start_run	daq-agent-zyang2-407	/:3836714	N/A	Success
20260112 23:14:14	test-zy	run_imminent	daq-agent-zyang2-407	/:3836714	N/A	Success



MQ send:
- run_imminent
- start_run
- stf_gen
- end_run

MQ receive:
- run_imminent
- start_run
- stf_gen
- end_run

MQ send: data_ready

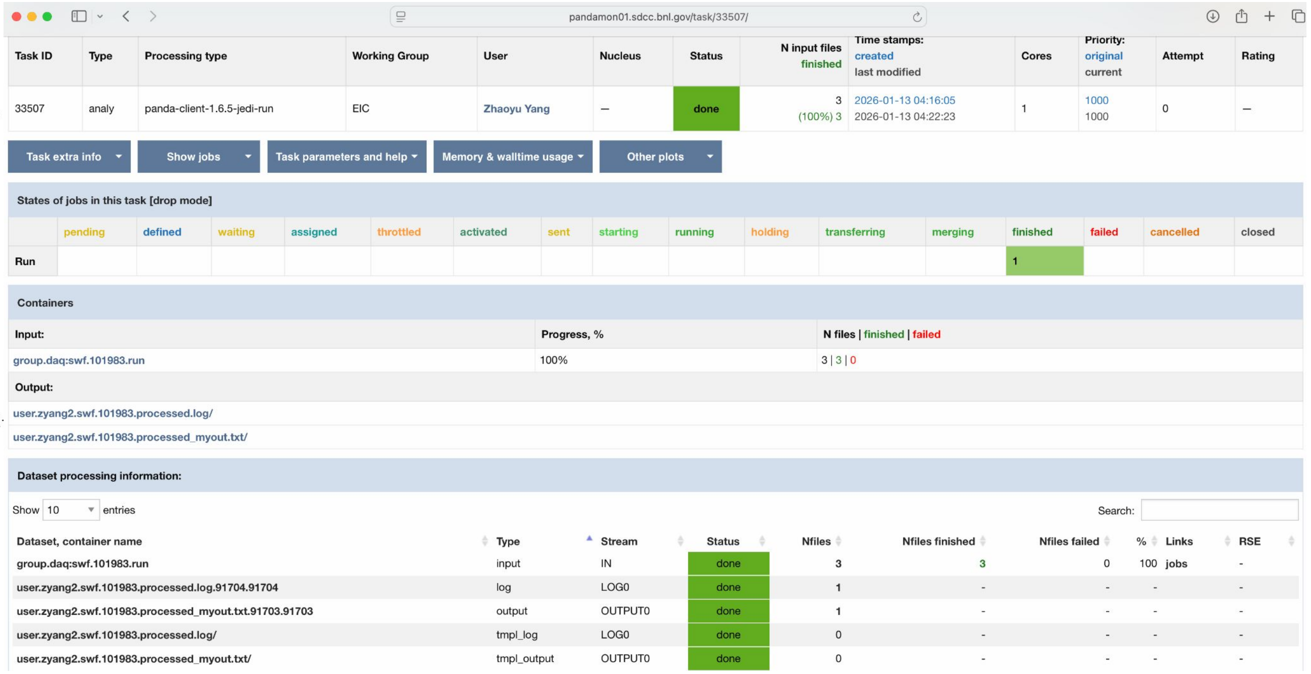
MQ receive:
- data_ready

Submit job once the first STF arrives

STF processing: full chain as reflected in the monitor (cont'd)

- The processing agent received the "data_ready" message and submits the job to PanDA, task 33507
- The payload at this point is simply echo to the output file, not real ePIC processing. It read the input dataset: group.daq:swf.101983.run, produced a rucio dataset: group.daq:swf.101983.processed

```
*** MQ: data ready for run 101983 ***
*** Named datasets for run 101983 ***
*** inDS: swf.101983.run ***
*** outDS: swf.101983.processed ***
INFO: Subscribed to queue: 'epictopic'
2026-01-12 23:14:53,824 - INFO - base_agent - Subscribed to queue:
*** Initialized the PROCESSING class, test mode is False ***
*** PROCESSING class run method called ***
*** Processing agent is running. Press Ctrl+C to stop. ***
*** PANDA PARAMS ***
taskName             : user.zyang2.swf.101983.processed/
uniqueTaskName       : True
vo                   : wlcg
architecture         : @centos8#x86_64
transUses            :
transHome            : None
processingType       : panda-client-1.6.5-jedi-run
prodSourceLabel      : test
site                 : E1_BNL
excludedSite         : []
includedSite         : None
cliParams            : prun -v
skipScout            : True
osInfo               : Linux-4.18.0-513.24.1.el8_9.x86_64-x86_64-wi
workingGroup         : EIC
nMaxFilesPerJob      : 200
respectSplitRule     : True
sourceURL            : https://pandaserver01.sdcc.bnl.gov:25443
dsForIN              : group.daq:swf.101983.run
runUntilClosed       : True
********************
*** Getting PanDA API client... ***
(0, 'succeeded. new jediTaskID=33507')
```



pandamon01.sdcc.bnl.gov/task/33507/

Task ID	Type	Processing type	Working Group	User	Nucleus	Status	N input files finished	Time stamps: created last modified	Cores	Priority: original current	Attempt	Rating
33507	analy	panda-client-1.6.5-jedi-run	EIC	Zhaoyu Yang	–	done	3 (100%) 3	2026-01-13 04:16:05 2026-01-13 04:22:23	1	1000 1000	0	–

Task extra info | Show jobs | Task parameters and help | Memory & walltime usage | Other plots

States of jobs in this task [drop mode]

	pending	defined	waiting	assigned	throttled	activated	sent	starting	running	holding	transferring	merging	finished	failed	cancelled	closed
Run													1			

Containers

Input:	Progress, %	N files | finished | failed
group.daq:swf.101983.run	100%	3 | 3 | 0

Output:

user.zyang2.swf.101983.processed.log/

user.zyang2.swf.101983.processed_myout.txt/

Dataset processing information:

Show 10 entries Search:

Dataset, container name	Type	Stream	Status	Nfiles	Nfiles finished	Nfiles failed	%	Links	RSE
group.daq:swf.101983.run	input	IN	done	3	3	0	100	jobs	-
user.zyang2.swf.101983.processed.log.91704.91704	log	LOG0	done	1	-	-	-		-
user.zyang2.swf.101983.processed_myout.txt.91703.91703	output	OUTPUT0	done	1	-	-	-		-
user.zyang2.swf.101983.processed.log/	tmpl_log	LOG0	done	0	-	-	-		-
user.zyang2.swf.101983.processed_myout.txt/	tmpl_output	OUTPUT0	done	0	-	-	-		-

Testing the STF processing chain: summary

The full STF processing chain has been tested, from the emulated DAQ data source to the processing in PanDA. The testing process included all of the following:

1. Operation of the DAQ emulator, driven by an easy-to-configure schedule, presented in a declarative format. Generation of STF mockup data and MQ communication to other agents.
2. Operation of the data-agent, directed my MQ messages received from the DAQ emulator, resulting in creation of datasets, upload of the data to the emulated storage element, registration of each parcel of the data (STF) in Rucio, and closing of the dataset at the run completion.
3. Operation of the processing-agent, directed by MQ messages received from the data-agent, resulting in definition of the PanDA processing tasks, designed to run continuously until the input dataset is marked “closed” in Rucio.
4. Communication with the monitor web application, providing the monitoring interface to the operators.

Backup

Processing orchestration: recent developments and plans

1. Recently, the ActiveMQ interface has been enhanced with namespaces and other devices necessary for parallel testing by many users (e.g. to prevent message loss) and other optimization.
2. The testbed currently emulates the JLab data endpoint with storage allocated within BNL SCDF, i.e. at present the data does not leave the BNL perimeter. A full implementation of data transmission to JLab and development of the data processing strategies would be the next steps in this work area.