Fast TF streaming with PanDA/iDDS, and 2026 plans

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

Nuclear and Particle Physics Software (NPPS) Group, BNL Physics Dept

Takahashi Tomonori

Osaka University

ePIC Collaboration Meeting, BNL

Jan 21 2026

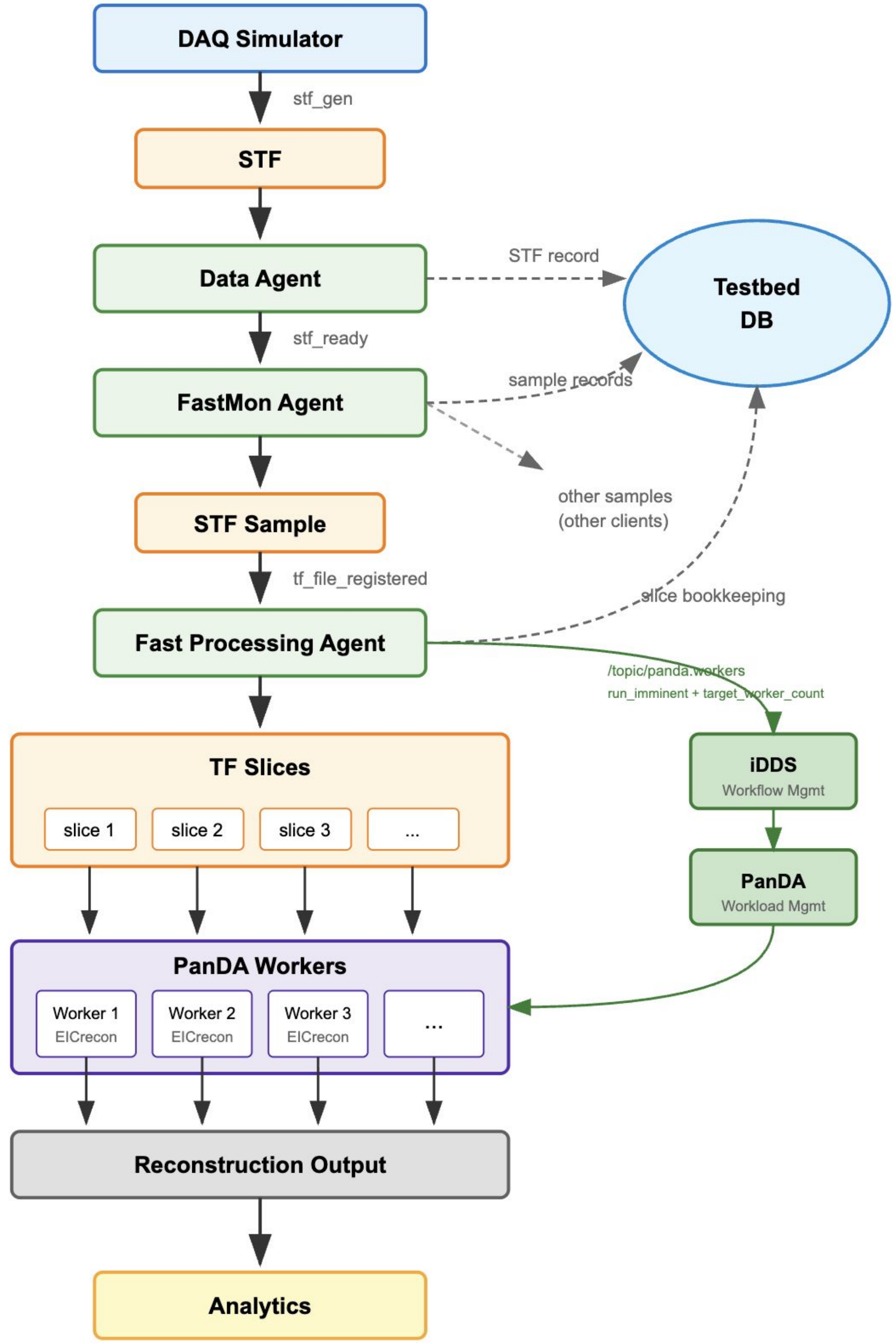
Streaming workflow orchestration testbed components

Bold: new or extensively developed since August 2025

- swf-testbed
    - Testbed umbrella repo: doc, packaging, venv, example agents, service management, test driver, **extensions for complex workflows: orchestration of persistent agents, extensible workflow definitions driven by common runner**
- swf-common-lib
    - Common infrastructure, logging utils, packaging infrastructure
- swf-monitor
    - Django service with full-system browser UI, REST API, ActiveMQ and logging
    - Postgres DB serving the full system via REST
    - Comprehensive **Model Context Protocol (MCP)** service for info and control via LLM
- ActiveMQ based agents
    - All now operate as **persistent agents ready for work**, controlled by CLI or MCP
    - DAQ simulator: **expressing the E0-E1 interface**
    - Data agent: STF file receiving, registering and forwarding to Rucio for transport
    - Fastmon agent: Fast sampling of STFs serving local/remote clients, client-specific filtering
    - STF processing agent: workflow for **end to end PanDA/Rucio processing of STFs**
    - **Fast processing agent:** implementing workflow I will describe

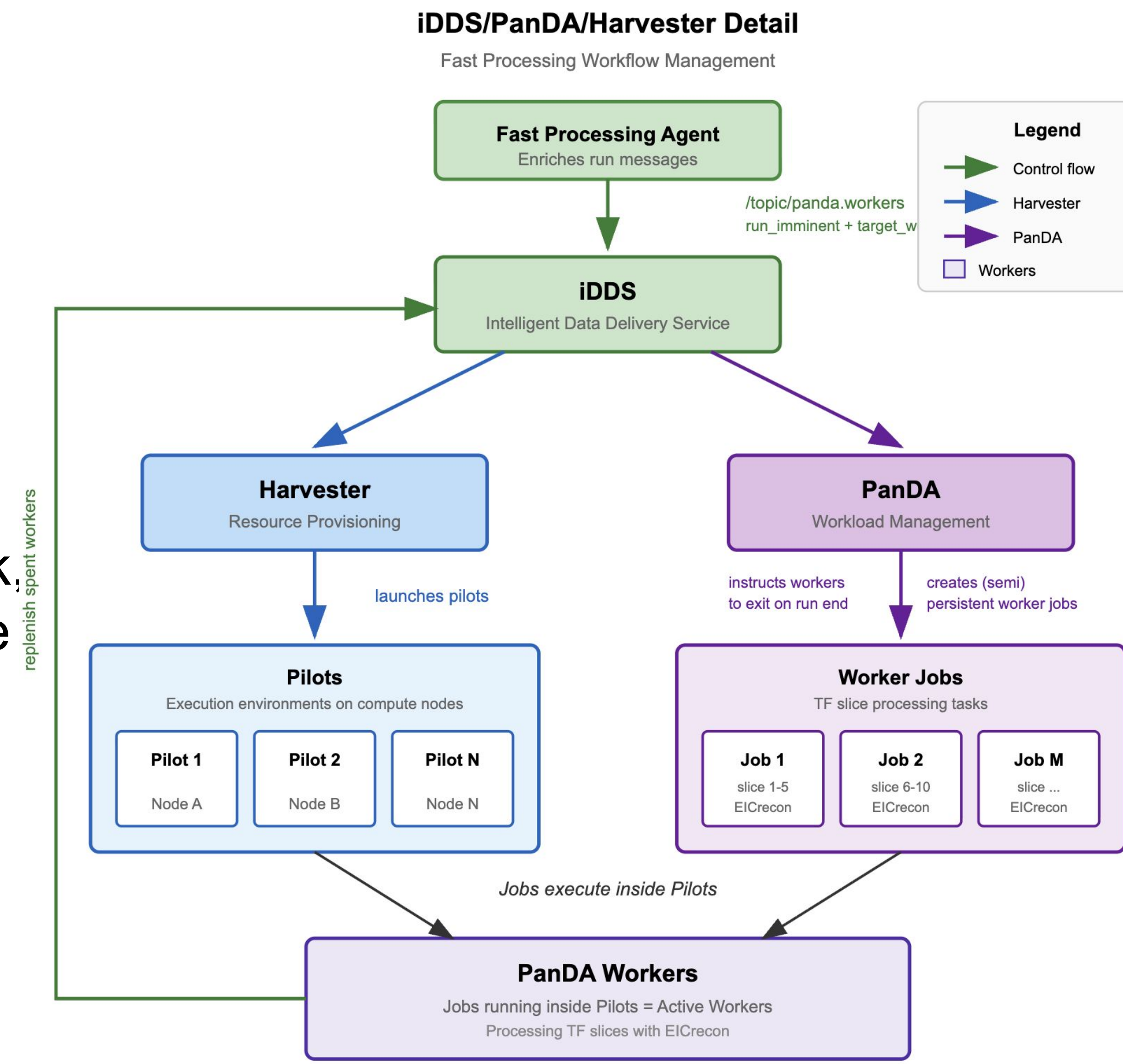
Fast TF streaming with PanDA/iDDS

- Fast Processing Agent
  - Run_imminent
    - Send config to iDDS to set up PanDA workers (e.g. how many workers)
    - iDDS creates the PanDA workflow
  - TF processing
    - Fast proc agent slices STF samples down to smaller TF ranges scaled for quick parallel worker processing: deliver result for control room fast
  - Streaming slices to workers
    - A PanDA worker loop
      - Fetch one slice message
      - Process the payload
      - Send back the result in a message
      - Ack the message to indicate processing complete
  - End_run
    - End the listening loop in PanDA workers
    - Terminate the workers (next run starts clean)



PanDA/iDDS TF streaming detail

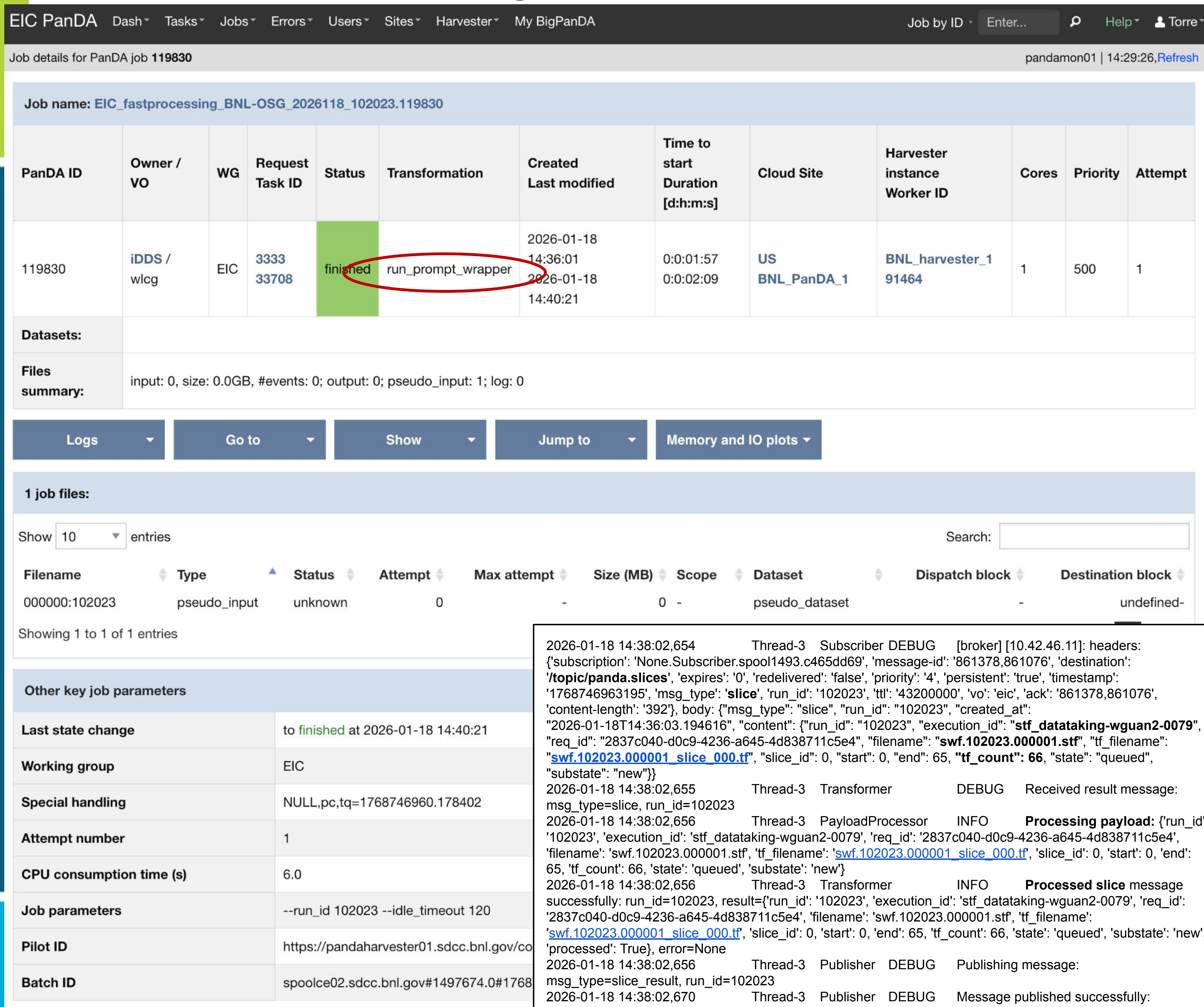
- Run_imminent
  - iDDS creates PanDA task & worker jobs within the tast
  - Message Harvester to create pilots
- PanDA worker
  - Pilot starts in batch slot, asks PanDA for work, is assigned a job: now it's a persistent (for life of the batch slot) slice processor
  - Fetches ActiveMQ configuration from iDDS
  - Listens to ActiveMQ for slice messages
  - Processes slice
  - Sends results to FastProcessingAgent (bookkeeping) and iDDS
    - iDDS calculates the delay from slice creation to getting the results, use this info to adjust the number of workers



PanDA/iDDS workflow working as of a few days ago, thank you Wen! **Example PanDA worker job**

That's the workflow, it's our most complex so far, I'll briefly describe some enabling infrastructure for it

Fast TF streaming PanDA worker job

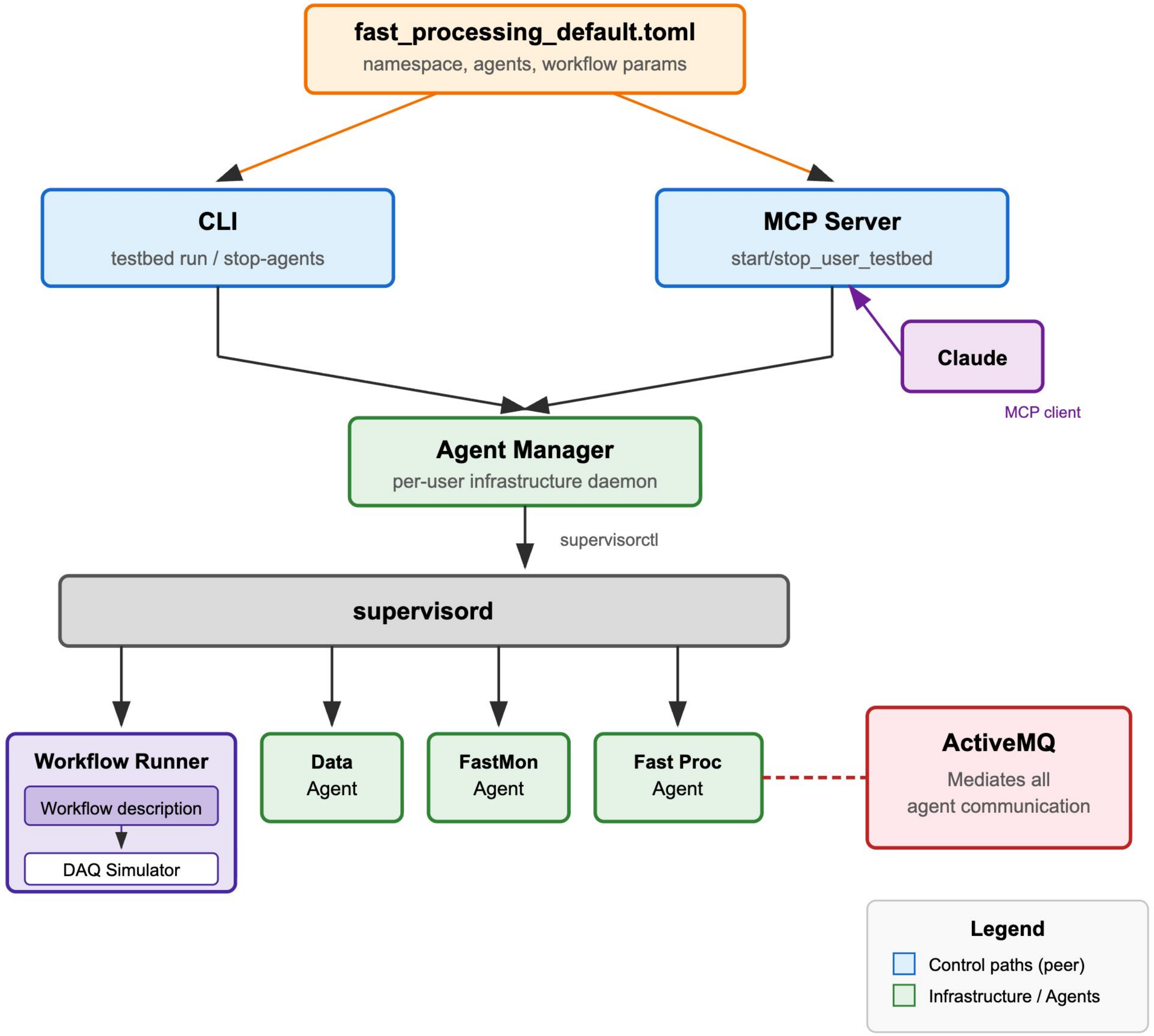



```
2026-01-18 14:38:02,654 Thread-3 Subscriber DEBUG [broker] [10.42.46.11]: headers:
{'subscription': 'None.Subscriber.spool1493.c465dd69', 'message-id': '861378,861076', 'destination':
'/topic/panda.slices', 'expires': '0', 'redelivered': 'false', 'priority': '4', 'persistent': 'true', 'timestamp':
'1768746963195', 'msg_type': 'slice', 'run_id': '102023', 'ttl': '43200000', 'vo': 'eic', 'ack': '861378,861076',
'content-length': '392'}, body: {"msg_type": "slice", "run_id": "102023", "created_at":
"2026-01-18T14:36:03.194616", "content": {"run_id": "102023", "execution_id": "stf_datataking-wguan2-0079",
"req_id": "2837c040-d0c9-4236-a645-4d838711c5e4", "filename": "swf.102023.000001.stf", "tf_filename":
"swf.102023.000001_slice_000.tf", "slice_id": 0, "start": 0, "end": 65, "tf_count": 66, "state": "queued",
"substate": "new"}}
2026-01-18 14:38:02,655 Thread-3 Transformer DEBUG Received result message:
msg_type=slice, run_id=102023
2026-01-18 14:38:02,656 Thread-3 PayloadProcessor INFO Processing payload: {'run_id':
'102023', 'execution_id': 'stf_datataking-wguan2-0079', 'req_id': '2837c040-d0c9-4236-a645-4d838711c5e4',
'filename': 'swf.102023.000001.stf', 'tf_filename': 'swf.102023.000001_slice_000.tf', 'slice_id': 0, 'start': 0, 'end':
65, 'tf_count': 66, 'state': 'queued', 'substate': 'new'}
2026-01-18 14:38:02,656 Thread-3 Transformer INFO Processed slice message
successfully: run_id=102023, result={'run_id': '102023', 'execution_id': 'stf_datataking-wguan2-0079', 'req_id':
'2837c040-d0c9-4236-a645-4d838711c5e4', 'filename': 'swf.102023.000001.stf', 'tf_filename':
'swf.102023.000001_slice_000.tf', 'slice_id': 0, 'start': 0, 'end': 65, 'tf_count': 66, 'state': 'queued', 'substate': 'new',
'processed': True}, error=None
2026-01-18 14:38:02,656 Thread-3 Publisher DEBUG Publishing message:
msg_type=slice_result, run_id=102023
2026-01-18 14:38:02,670 Thread-3 Publisher DEBUG Message published successfully:
msg_type=slice_result, run_id=102023, destination=/topic/panda.results
```

- iDDS worker handler manages PanDA job and Harvester pilot generation based on workflow configuration from fast processing agent
- Transformer running in the pilot establishes it as a worker in the slice processing workflow
- Worker consumes slice messages from /queue/panda.slices.transformer
  - fetch one message, process the payload, send the result back, ack the message as processed
- Processing the payload is a dummy op
  - Move when we can to streaming reco as the payload processor
- Slice processing duration and parallel worker scale-out are scaled to desired latency and stats for presenting fresh data analytics, e.g. ~30sec, will also try ~10sec

Agent management system

- The fast TF streaming workflow uses all the currently existing agents
- Running each in a terminal window, the old approach, is tedious and a poor model of the actual system
- New agent management system uses supervisord to manage agents autonomously, under message-based direction from CLI or MCP
- You just tell it to start the testbed instance for the desired workflow, it sets up the needed agents
- Then tell it to run the workflow
- (Then e.g. change params and iterate)
- The agents are persistent: they wait for work in between workflow runs



Brookhaven National Laboratory

The testbed now has a comprehensive MCP service

- ~30 MCP tools giving LLMs access to ~all testbed system/log knowledge available
  - Uses REST interface to the monitor which is backed by the testbed DB, based on [django-mcp-server]
- Most are passive info providers
- Several are active operators
  - start/stop testbed, based on agents needed for a particular workflow config
  - start/stop workflows operating with those agents
  - various cleanups
- You can start your testbed and run a workflow, monitoring its execution and reporting results, with one Claude prompt (right)
- This is all at the individual user level, autonomous from other testbed users
- It adds an agent to the system: your personal agent
  - Manages your testbed agents, supporting 'start/stop testbed'
  - Probably more to come, it's the agentic era!

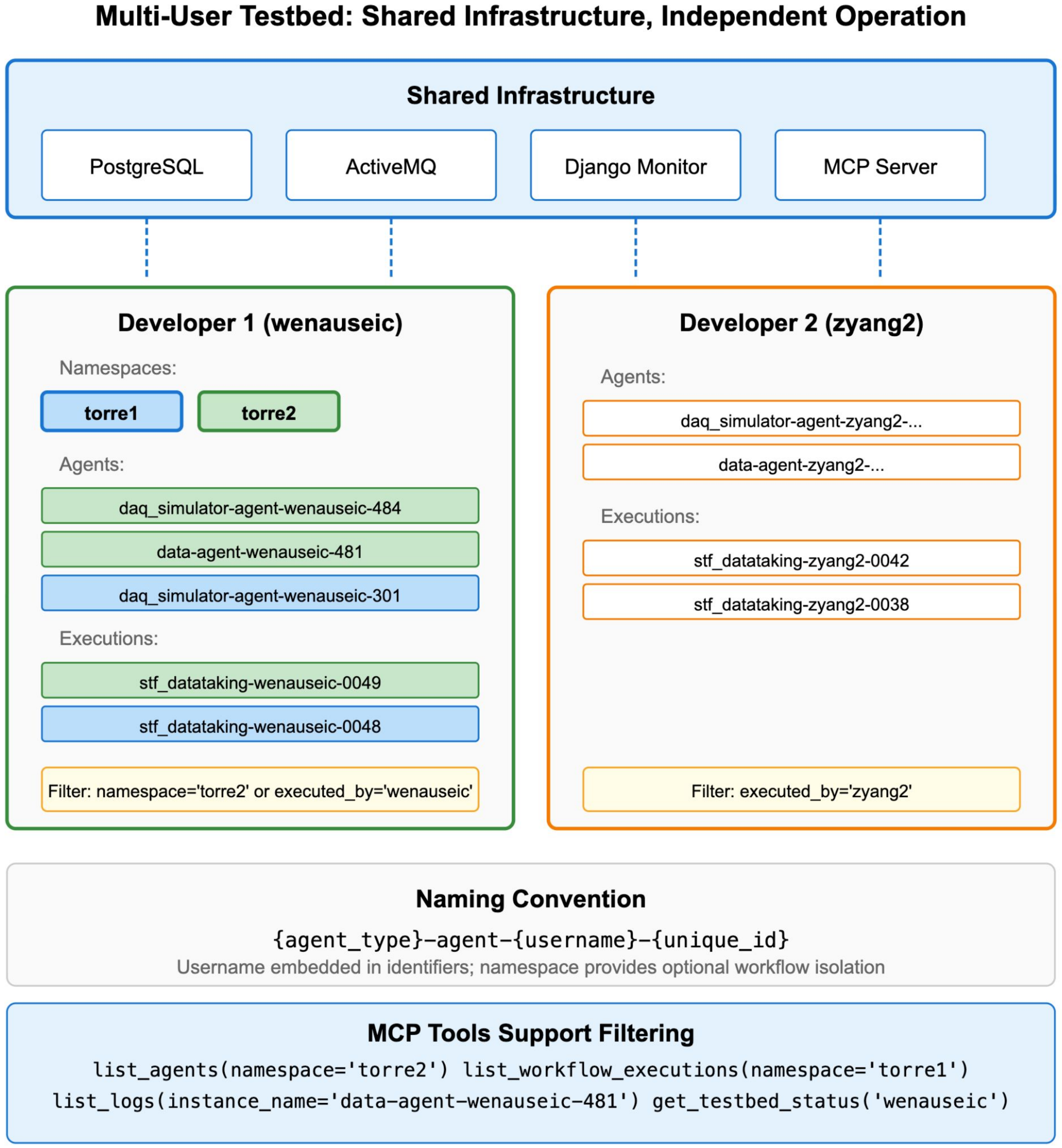
```
Human❯ Let's (re)start the testbed and run the default workflow
• I'll restart the testbed and run the default workflow. Let me first check the current
status, then restart.
• swf-testbed - get_testbed_status (MCP)(username: "wenauseic")
• Agent manager is healthy, all agents stopped. Starting the testbed and then running the
workflow.
• swf-testbed - start_user_testbed (MCP)(username: "wenauseic")
• Testbed starting. Let me wait a moment for agents to come up, then start the workflow.
• Bash(sleep 3)
• swf-testbed - list_agents (MCP)(namespace: "torre1", status: "OK")
• New agents are up and ready:
  - daq_simulator-agent-wenauseic-450 - READY
  - stf_processing-agent-wenauseic-451 - READY
  Now starting the default workflow.
• Workflow started. Let me monitor its progress.
• swf-testbed - list_workflow_executions (MCP)(currently_running: true)
• Workflow stf_datataking-wenauseic-0048 is running. Let me track its progress.
• swf-testbed - get_workflow_monitor (MCP)(execution_id: "stf_datataking-wenauseic-0048")
• Just started - waiting for events. Let me check messages after a few seconds.
• Bash(sleep 8)
• swf-testbed - get_workflow_monitor (MCP)(execution_id: "stf_datataking-wenauseic-0048")
• swf-testbed - list_messages (MCP)(execution_id: "stf_datataking-wenauseic-0048")
• Still in initial DAQ state delays. Let me check logs and wait a bit more.
• swf-testbed - list_logs (MCP)(execution_id: "stf_datataking-wenauseic-0048")
• Workflow completed successfully. Let me get the final status.
• swf-testbed - get_workflow_execution (MCP)(execution_id: "stf_datataking-wenauseic-0048")
• Testbed restarted and workflow completed successfully.
  Summary
```

Item	Value
Execution ID	stf_datataking-wenauseic-0048
Status	completed
Namespace	torre1
Config	fast_processing_default
Run ID	101992
Duration	~40 seconds
STF files generated	10

```
  Agents involved:
  - daq_simulator-agent-wenauseic-450 - broadcasted run_imminent → start_run → 10× stf_gen
→ end_run
  - stf_processing-agent-wenauseic-451 - processed all messages successfully
  Workflow sequence (from logs):
  1. run_imminent broadcast → processing agent prepared resources
  2. start_run broadcast → processing agent ready
  3. 10× stf_gen broadcasts (2s intervals, realtime mode)
  4. end_run broadcast → processing agent completed
  No errors. The testbed is operational and ready for further work
```

Multi-user support

- Testbed is based on shared infrastructure + independent operation via mechanisms to isolate/filter one user from another
- Agent names have assured isolation for some time: username and unique ID are built in
- Additional mechanism introduced: namespaces
- Namespaces support isolation/filtering at the higher workflow and testbed instance levels
- Namespaces also support collaboration: multiple people can use the same namespace



Looking back at May 2025 planned scope

Components and functionalities of the system to define and implement in a first prototype:

- **stream definition, structure and associated metadata**
  - **bulk data stream**
    - **stream delivering 100% of the data to both E1s**
  - **monitoring stream**
    - **finer grained, lower latency stream of a fraction of the data for fast (O(1min)) monitoring and validation at E1s**
    - **complements the very fast monitoring and validation happening in DAQ**
- **E0 - E1 data flows**
  - **how bulk data moves from E0 to E1**
  - **how fast monitoring data moves from E0 to E1**
- **E1 bulk data orchestration**
  - **How and where bulk data lands at E1s**
  - **Triggering actions on the data (archiving, prompt processing) as it arrives**
- **E1 fast monitoring infrastructure**
  - **How fast monitoring data moves from E0 to E1**
  - **How and where it is received and processed by monitoring agents/workers**
- **E1 data processing orchestration**
  - **Workflows for processing physics data at the E1s**
- **E1 calibration orchestration**
  - **Workflows, from simple to complex, for performing prompt calibration/alignment at E1s**
  - **Supporting many such activities concurrently across the detector systems**
  - **Tools to define and automatically execute complex workflows with dependencies down the processing chain**
- **Detector/data state machine**
  - **Infrastructure to manage and interface to detector/data state in an E1 resident service**
  - **cataloging of detector/data states, clients served and their needs**
  - **Needs further development as part of defining the E0-E1 interface, and to begin to incorporate calibration**
- **E2 extension**
  - **An extension of the testbed will prototype the inclusion of Echelon 2 (E2) computing facilities**

Green: implemented in the testbed, at V1 level

Blue: planned for 2026

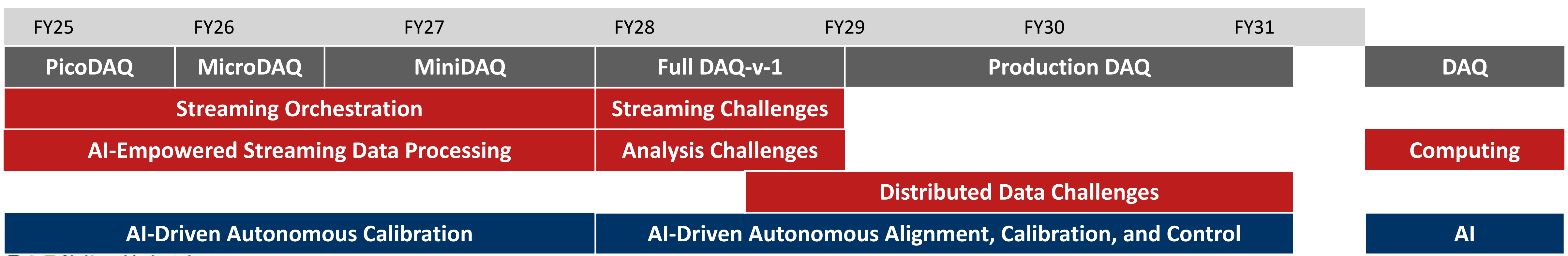
What did we do that's not on here?
Some May jokes became later reality

Let's let LLMs write the code, lol

Let's use MCP to control the whole thing
and as a full system info source, lol

2026 Objectives

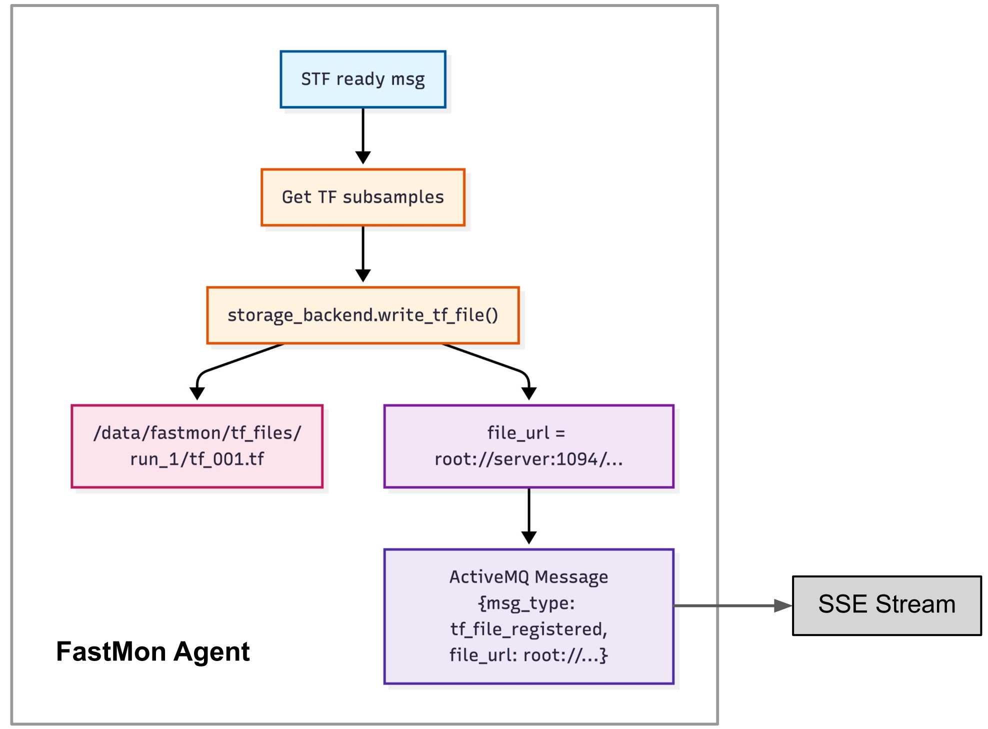
- Defining and developing the **E0-E1 interface** **Cross-testbed collaboration**
- We now have the **STF processing and fast STF streaming workflows**, let's start using them!
  - Begin to scale and operate under quasi realistic conditions, as permitted by 2026 infrastructure
  - **Exercise and test the system - are we using the right system components, is it a system that meets the requirements**
- Flesh out the workflows as real data processors using simulation data as TF 'DAQ input' and **TF-based ElCrecon streaming reconstruction** **Cross-testbed collaboration**
- Operational **Osaka Japan extension** of the testbed
- **Calibration workflows** integrated in testbed to drive PanDA based processing, with state machine integration **Cross-testbed collaboration**
- Building out **MCP, helping answer what is AI-empowered streaming data processing**
  - Adept/extend the testbed's MCP service based on experience
  - Add to our MCP suite the PanDA and Rucio MCP tools
    - AskPanDA, MCP analytics service, Rucio MCP
- **The ePIC streaming workflow orchestration testbed is a system simulator from E0 egress through E1 and soon to proto E2. Explore what we can do and address with such a simulator!**



2026 Objectives: Fast monitoring

Streaming Fast Monitoring Files (TFs)

- Extend swf-fastmon-agent to allocate TF files in a storage instance
- Enable monitoring clients to retrieve them from anywhere via root:// protocol
- XRootD Authentication
  - X.509 grid certificate (current)
  - Aiming to switch to WLCG/EGI tokens
- Aiming to O(seconds) latency for TF retrieval



Fast Monitoring Applications

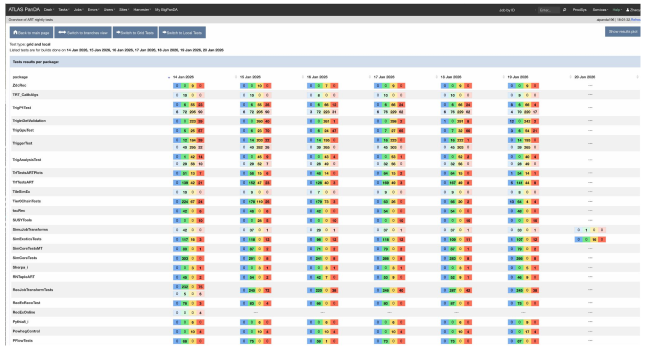
- Further develop testbed interfaces applications to real data (instead of mock records)
  - Introduce realistic payloads using current ePIC reconstruction framework
  - Demonstrate integrated running of monitoring and fast calibration workflows (Snakemake workflows) within the testbed (a fastmon client)

2026 Objectives: PanDA in production and CI

In the PanDA for ePIC production effort proceeding in parallel with the testbed…

- PanDA operational as ePIC simulation production back end
  - Including monitoring, analytics, and Perlmutter as well as OSG operation
- AskPanDA operational for ePIC production jobs

- Develop prototype distributed CI system with PanDA as backend – enables processing-intensive CI/testing workflows for ePIC
  - Evaluate applicability of ATLAS's processing-intensive distributed test/CI system ART
  - Snakemake-defined, PanDA-executed validation workflows demonstrated



Frontend for ART in PanDA monitoring systen

Brookhaven National Laboratory

The participants

- Wen Guan
  - PanDA, iDDS and testbed-interface components of fast STF sample processing
- Dmitry Kalinkin
  - Calibration, validation and CI workflows; EICrecon payload integration; realization of ‘real’ STF data structure as it becomes defined
- Maxim Potekhin
  - PanDA and Rucio based STF processing workflows; DAQ interfaces and simulation
- Tomonori Takahashi (Osaka)
  - Extending testbed to Osaka, fast STF sampling client, proto Echelon 2, production integration
- Michel Villanueva
  - Fast STF sample preparation, distribution, processing
- Torre Wenaus
  - Testbed infrastructure, fast STF sampling workflows, DAQ interfaces and simulation
- Zhaoyu Yang
  - Monitoring & analytics (PanDA, Opensearch, Grafana), AI MCP (AskPanDA, analytics)
- Xin Zhao
  - Computing facilities, SCDF liaison, foundation services including PanDA, Rucio, ActiveMQ

Conclusion

- Two TF processing workflows implemented thus far
  - Prompt processing of STF files at E1s
    - Transfer to E1s with Rucio and process entire STF files
    - Jobs processed via PanDA as conventional batch jobs with STF file input
  - Fast streaming of finer-grained TFs for quick results in control room
    - STF subsamples messaged to parallel workers for high throughput on fresh data
    - iDDS mediates between testbed processing agent and PanDA's parallel persistent workers
    - Workers have lifetime of batch slot, with small assignments constantly streamed in during a run
- Time to start using them towards testbed objectives!
  - Evaluate against WFMS requirements, scaling, robustness, defining E0-E1 interface, …
- Other threads of activity are growing
  - Thanks to Japan participation, progress to proto E2 at Osaka integrated with the testbed
  - PanDA for production operational, distributed CI planned
  - Application of fast monitoring workflow to real reco workloads, calibration
- Managing workflow complexity motivated very helpful infrastructure: comprehensive MCP and an agent management system -- models the real system better, and demonstrates natural language interaction/control via fully system-aware LLM
- Next big objective to add in 2026: the first calibration workflows

More information

- ePIC workflow management system requirements draft
- Streaming workflow testbed planning document
- Streaming workflow testbed progress document
- Testbed monitor
  - InCommon federated login access, ask Torre for a django account on the monitor itself
- BNL PanDA (and Rucio) startup guide (testbed uses BNL R&D instances of each)
- BNL PanDA monitor

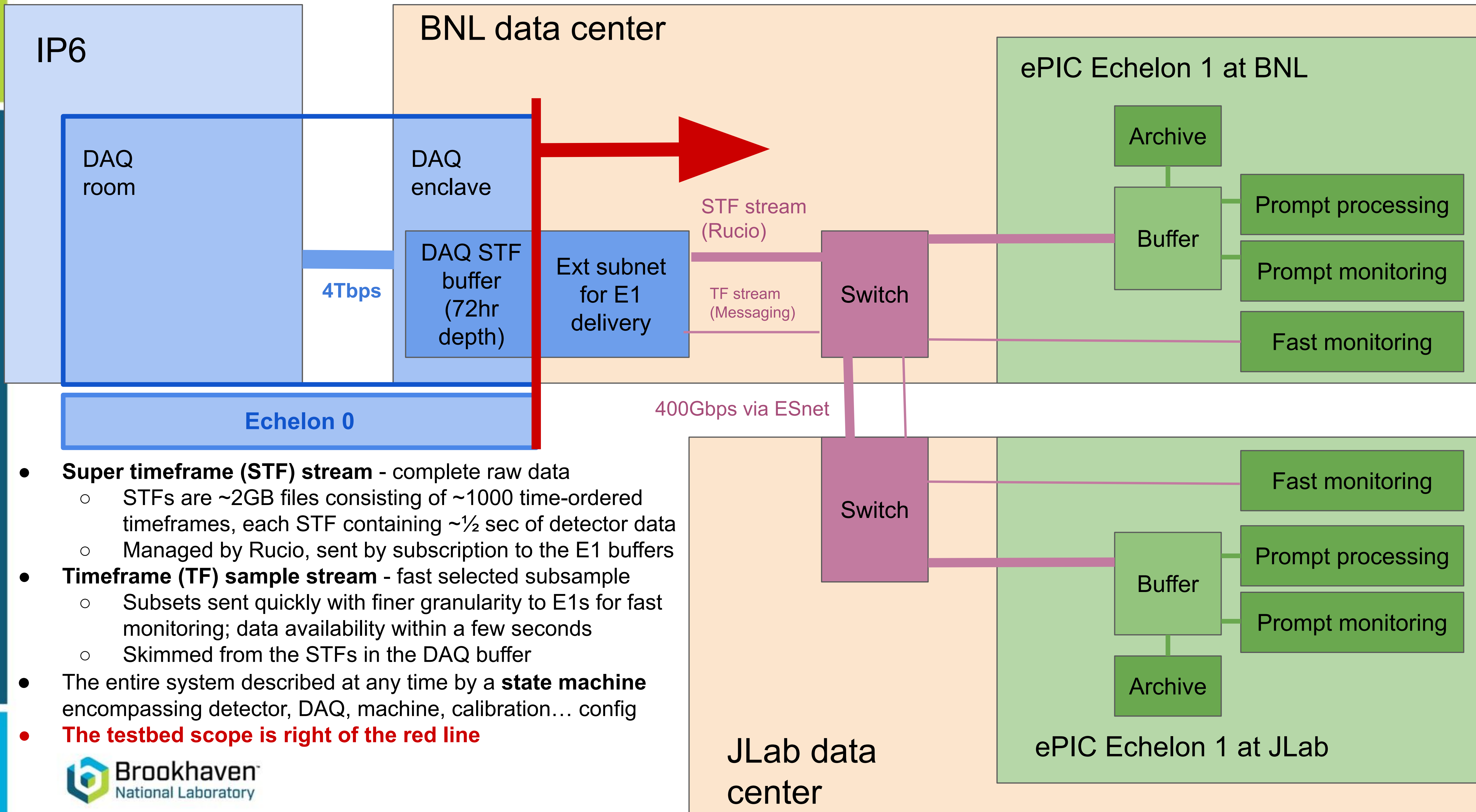
- ePIC Streaming Computing Model Report (currently V2 Fall 2024, V3 in dev)
- ePIC streaming computing model WG meeting notes

- New iDDS paper in arXiv

Thank you to everyone in NPPS, SCDF and the ePIC streaming computing model working group community who have contributed to making this testbed a reality!

Backup

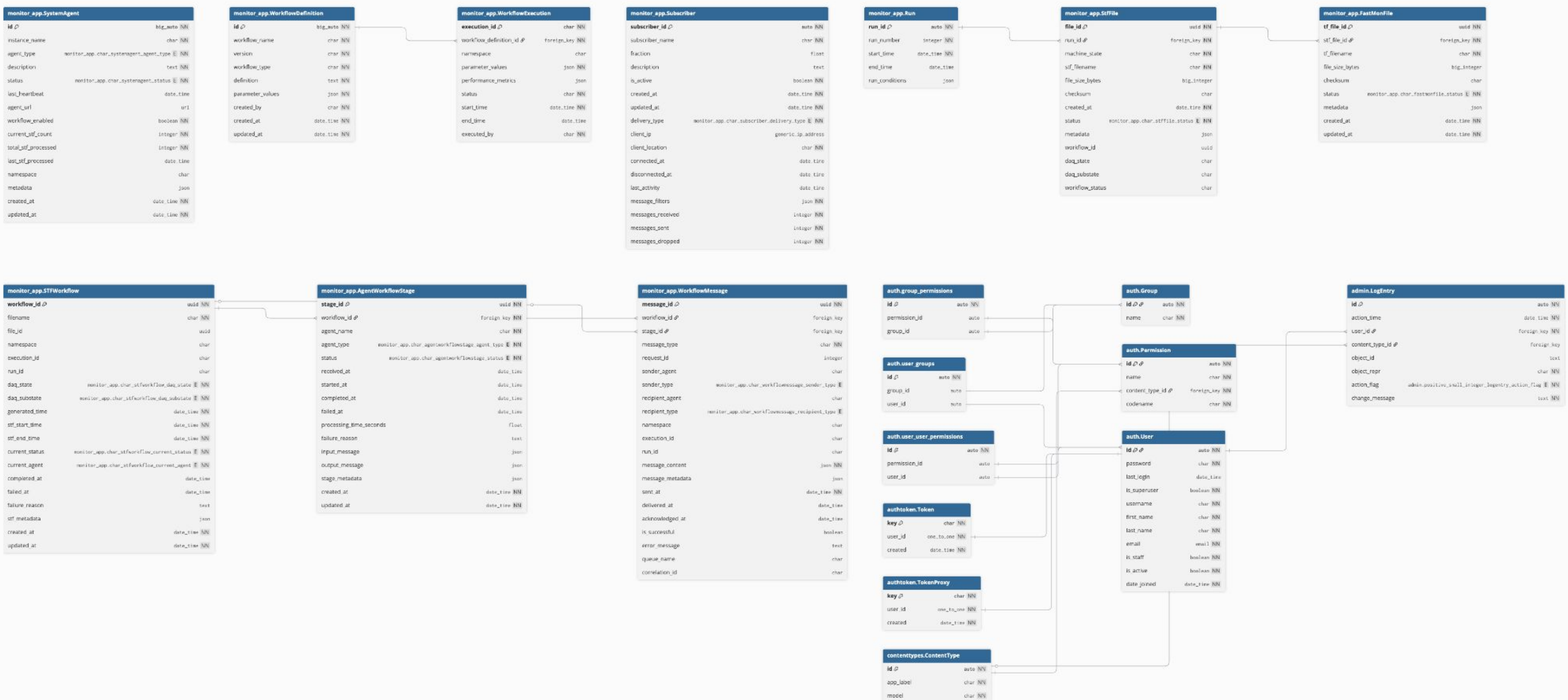
ePIC Echelon 0 - Echelon 1 workflows



- **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**

Brookhaven National Laboratory

DB model Jan 2026



State machine implementation

Our baseline workflows exercise most of it (no detector/machine at this point)

States

- no_beam
  - Collider not operating
- beam
  - Collider operating
- run
  - Physics running
- calib
  - Dedicated calibration period
- test
  - Testing, debugging
  - Any substates can be present during test

Substates

- not_ready
  - detector not ready for physics datataking
  - occurs during states: no_beam, beam, calib
- ready
  - collider and detector ready for physics, but not declared as good for physics
  - when declared good for physics, transitions from beam/ready to run/physics
  - occurs during states: beam
- physics
  - collider and detector declared good for physics
  - if collider or detector drop out of good for physics, state transitions out of ‘run’ to ‘beam’ or ‘off’
  - occurs during states: run
- standby
  - collider and detector still good for physics, but standing by, not physics datataking (dead time!)
  - occurs during states: run
- lumi
  - detector, machine data that is input to luminosity calculations
  - occurs during states: beam, run
- eic
  - machine data, machine configuration
  - occurs during states: all
- epic
  - detector configuration, data
  - occurs during states: all
- daq
  - info, config transmitted from DAQ
  - occurs during states: all
- calib
  - a catch-all for a great many calib data types, we can start small
  - occurs during states: all (assuming there are cases where calib data is taken during beam on)

Fast STF sample processing for fast control room monitoring

- Have started to build on our ePIC computing resource requirements estimates with a **new prompt processing workflow orchestration spreadsheet**
- It addresses the question, what will shifters have on their screens from E1-based monitoring immediately after run start, and ongoing, presenting near-live detector state?
- **Rucio STF delivery takes minutes, it doesn't play a role in this**
- STFs arrive from DAQ every ~half second with ~45k events each, **getting high-stats reco up on the screens quickly will be a priority, and will require quickly sampling them**
  - Target plucked out of the air: **process an STF equivalent (~45k events) every 30sec**
- **This is what the STF sample based fast monitoring stream from fastmon agent is for**
- **Fan out STF samples** of new data in near real time across E1 workers
  - A given **worker processes (fractions of) hundreds of STFs** over its batch slot lifetime
- The **PanDA fine grained processing services** -- the event service and iDDS (intelligent Distributed Dispatch and Scheduling system) -- were designed for this
  - Worker acts as a (semi) persistent agent accepting fine grained tasks
  - iDDS mediates the fast assignment to workers of STF samples produced by fastmon
  - PanDA does the fine grained bookkeeping and automated retry
- **This is now implemented as the fast processing workflow**

Testbed monitor

Home Workflows STF Files TF Files Agents Subscribers Messages Logs Database State PanDA/Rucio

Log Summary

Server-side aggregated log counts by application and instance, with level breakdowns and drill-down access.

View All Logs

Applications: All base_agent daqsim daqsim-agent data django_integration_test example_agent fastmon monitor processing test test-app

Instances: All daq-simulator-wenauseic-5 daqsim-agent daqsim-agent-1 daqsim-simulator-1 data-agent data-agent-1 data-agent-2 data-agent-example data-agent-wenauseic-
9998 fastmon-agent fastmon-agent-e1 fastmon-agent-e2 fastmon-agent-wenauseic-10 fastmon-agent-wenauseic-11 fastmon-agent-wenauseic-12 fastmon-agent-wenauseic-13 f
fastmon-agent-wenauseic-8 fastmon-agent-wenauseic-9 monitor-system processing-agent processing-agent-1 processing-agent-example processing-agent-wenauseic-4 processi
processing-agent-wenauseic-9266 processing-agent-wenauseic-9481 simulation-main sse_sender-agent temp-instance test test-instance test_django_dual_server workflow_sim
wenauseic-17 workflow_simulator-agent-wenauseic-18 workflow_simulator-agent-wenauseic-19 workflow_simulator-agent-wenauseic-20 workflow_simulator-agent-wenauseic-21 wo
agent-wenauseic-23 workflow_simulator-agent-wenauseic-24 workflow_simulator-agent-wenauseic-25 workflow_simulator-agent-wenauseic-26 workflow_simulator-agent-wenauseic-

Levels: All DEBUG INFO WARNING ERROR CRITICAL

Show 100 entries per page. 50 total entries passing filter.

Application Name	Instance Name	Latest Timestamp	INFO	WARNING	ERROR	CRITI			
base_agent	workflow_simulator-agent-wenauseic-28	20250926 15:46:10	2	0	0	0			
base_agent	workflow_simulator-agent-wenauseic-27	20250926 15:45:52	2	0	0	0			
base_agent	workflow_simulator-agent-wenauseic-26	20250926 15:37:31	2	0	0	0			
base_agent	workflow_simulator-agent-wenauseic-25	20250926 14:35:04	2	0	0	0			
base_agent	workflow_simulator-agent-wenauseic-24	20250926 14:34:26	2	0	0	0			
base_agent	workflow_simulator-agent-wenauseic-23	20250926 14:32:20	2	0	0	0			
base_agent	workflow_simulator-agent-wenauseic-22	20250926 14:30:17	2	0	0	0			
base_agent	workflow_simulator-agent-wenauseic-21	20250926 14:28:26	2	0	0	0			
base_agent	workflow_simulator-agent-wenauseic-20	20250926 14:27:42	1	0	1	0			
base_agent	workflow_simulator-agent-wenauseic-19	20250926 14:27:15	1	0	1	0			
base_agent	workflow_simulator-agent-wenauseic-18	20250926 14:17:28	2	0	0	0			
base_agent	workflow_simulator-agent-wenauseic-17	20250926 13:41:56	2	0	0	0			
base_agent	workflow_simulator-agent-wenauseic-16	20250926 13:40:25	2	0	0	0			
base_agent	fastmon-agent-wenauseic-15	20250919 17:03:10	2	14	0	0			
base_agent	processing-agent-wenauseic-7	20250919 17:03:10	6	0	0	0	0	6	View Logs
base_agent	fastmon-agent-wenauseic-14	20250918 17:00:16	2	0	0	0	0	2	View Logs
base_agent	fastmon-agent-wenauseic-13	20250918 16:54:46	2	0	0	0	0	2	View Logs
base_agent	fastmon-agent-wenauseic-12	20250918 16:53:48	2	0	0	0	0	2	View Logs
base_agent	fastmon-agent-wenauseic-11	20250918 16:50:00	2	0	0	0	0		
base_agent	fastmon-agent-wenauseic-10	20250918 16:46:52	2	0	0	0	0		
base_agent	fastmon-agent-wenauseic-9	20250918 16:41:49	2	0	0	0	0		
base_agent	fastmon-agent-wenauseic-8	20250918 16:31:52	2	0	0	0	0		
daqsim-agent	daq-simulator-wenauseic-5	20250918 16:29:51	43	0	0	0	15		
base_agent	processing-agent-wenauseic-4	20250918 16:29:50	6	0	0	0	0		

Home Workflows STF Files TF Files Agents Subscribers Messages Logs Database State PanDA/Rucio

ePIC Streaming Workflow Testbed Monitor

Real-time monitoring and management dashboard for the ePIC streaming workflow testbed

Runs

View and manage DAQ runs with their status, timing, and file counts. Monitor run lifecycle from imminent to completion.

STF Files

Browse Super TimeFrame files generated during runs. Track file sizes, checksums, and processing status.

TF Files

Monitor Time Frame files sampled from STFs for fast monitoring. Track TF generation and quality assessment metrics.

Agents

Monitor workflow agents status, types, and heartbeats. Track DAQ simulator, data, processing, and monitoring agents.

Logs

Access centralized logging from all workflow components. Filter by application, level, and timeframe for troubleshooting.

Detailed log list

Subscribers

Manage message queue subscribers and their connection status. Monitor ActiveMQ topic subscriptions.

Messages

View workflow messages. Track runs, STF generation, and agent communications.

Database

Inspect database tables, record counts, and schema. Monitor data growth and last insertion times.

State

View and manage persistent workflow state including run numbers, configuration, and system parameters.

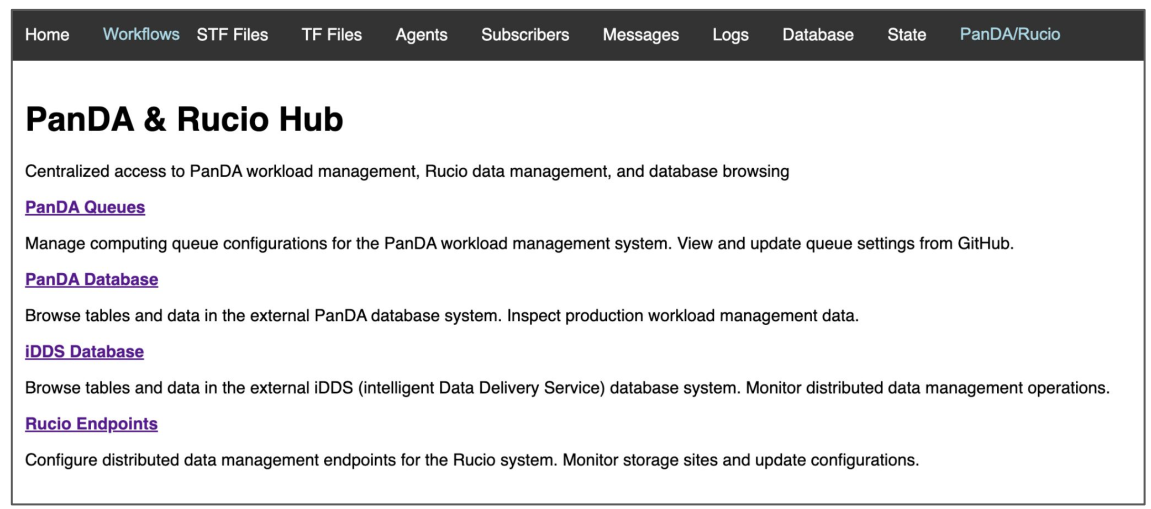
PanDA/Rucio

Centralized access to PanDA workload management, Rucio data management, and database browsing for external systems.

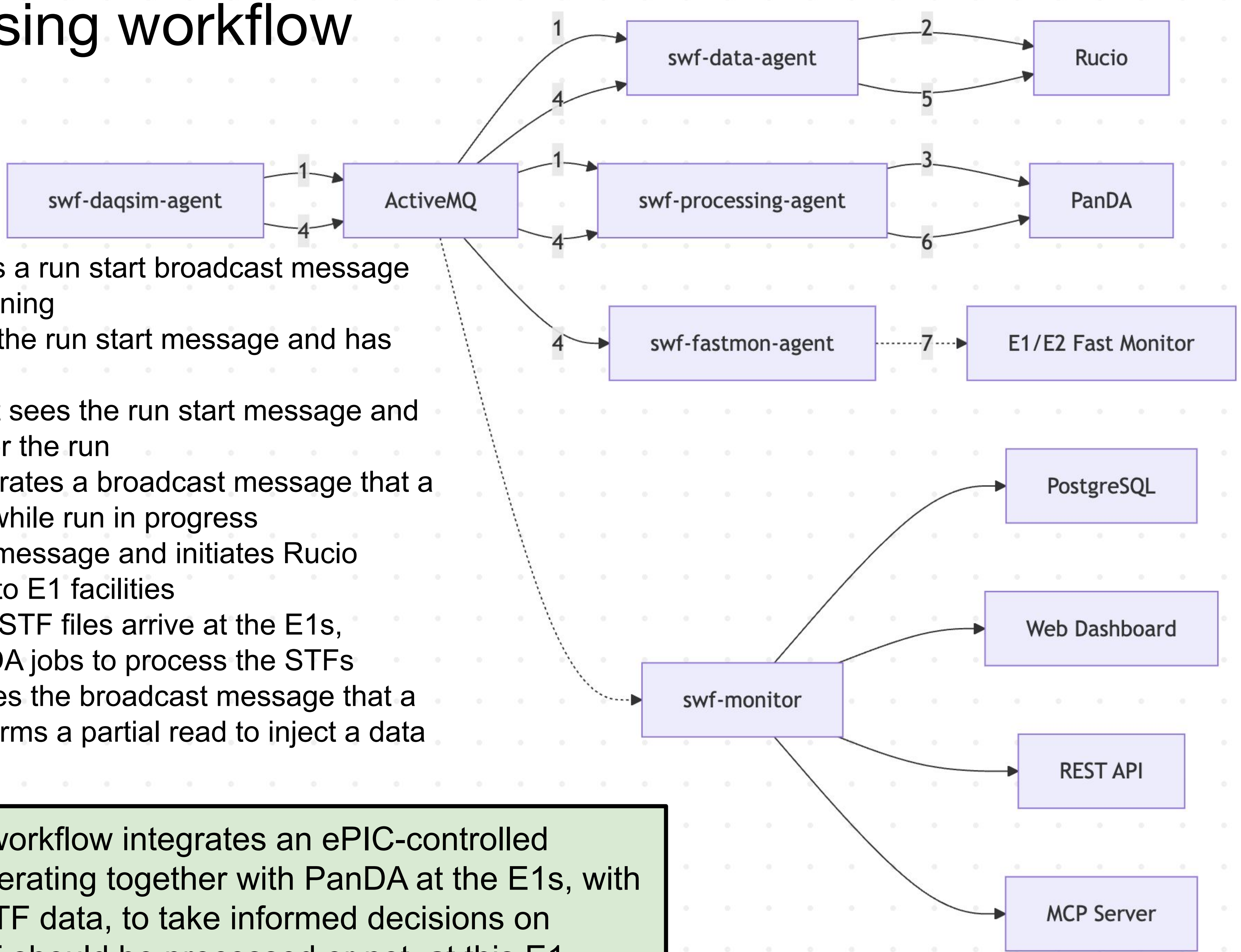
https://pandaserver02.sdcc.bnl.gov/swf-monitor
Two auth layers:
InCommon (BNL or other login)
Django account (ask Torre)

PanDA, Rucio integration in the testbed monitor/system

- PanDA and Rucio testbed configurations are browsable in the testbed monitor
  - PanDA queues
  - Rucio endpoints
- New PanDA, Rucio configurations can be activated via the monitor by superusers (ie Xin)
- This is an extremely simple (yet fully adequate for the testbed) replacement for the complex CRIC information management system developed by ATLAS PanDA and used by WLCG
  - Based on the same json schema as CRIC
- The full BNL-instance PanDA and Rucio databases are browsable in the testbed monitor
  - Not in itself greatly useful except to experts, but **opens the door to integrating PanDA and Rucio information directly in testbed monitor views**
  - Will use complementary integration of PanDA monitor via links, avoid duplication
- Next: integrate more PanDA services
  - Opensearch for analytics (BNL instance in place)
    - Also Grafana
  - 'Ask PanDA' MCP service
    - Also Rucio MCP?
  - Pilot/batch monitoring service
    - Add links to a service in Lancaster, they kindly extended their ATLAS/Rubin service to support our PanDA instance



STF prompt processing workflow in the testbed



1. Run Start - DAQ simulator generates a run start broadcast message indicating a new datataking run is beginning

2. Dataset Creation - data-agent sees the run start message and has Rucio create a dataset for the run

3. Processing Task - processing-agent sees the run start message and establishes a PanDA processing task for the run

4. STF Available - DAQ simulator generates a broadcast message that a new STF data file is available, iterates while run in progress

5. STF Transfer - data-agent sees the message and initiates Rucio registration and transfer of the STF file to E1 facilities

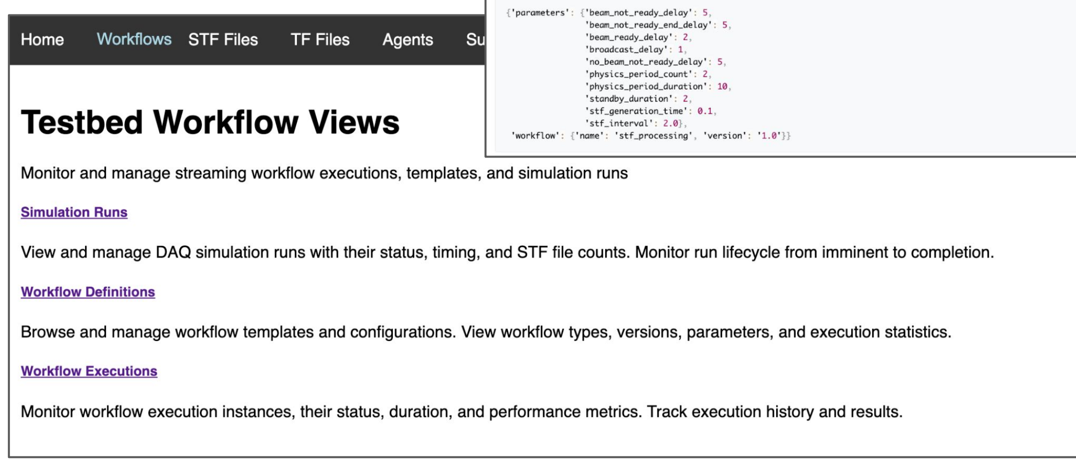
6. STF Processing - PanDA sees new STF files arrive at the E1s, transferred by Rucio, and initiates PanDA jobs to process the STFs

7. Fast Monitoring - fastmon-agent sees the broadcast message that a new STF data file is available and performs a partial read to inject a data sample into E1/E2 fast monitoring

The automated workflow integrates an ePIC-controlled 'decision box' operating together with PanDA at the E1s, with visibility to the STF data, to take informed decisions on whether this STF should be processed or not, at this E1

Workflow orchestration framework

- The fast STF sample processing is an example of complex workflows we want to **flexibly define, configure, parameterise, adjust**
- A system allowing us to do this is now part of the testbed, the workflow orchestration framework
- **Provides for defining, parameterising, executing and monitoring a catalog of workflows**
- Implements concise **python workflow definitions**, full parameterisation in config files, a generic execution framework to run these, and a cataloging system for workflow definitions and their executions, with presentation in the monitor
  - 557 line python script for the STF processing workflow became **56 lines**
- WorkflowRunner orchestration layer with SimPy discrete event simulation to turn definitions into running workflows
- TOML-based config for workflow parameters
- WorkflowDefinition, WorkflowExecution models
- Unique execution tracking `workflow-username-N’
- Detailed workflow definition and execution inspection in monitor



Coming in 2026: Calibration workflows in the testbed

- Taku Gunji gave a recent talk on calibration in the computing model, the topic is gaining attention
- As this activity develops it will guide integrating calibration into the testbed
- Objective: demonstrate capability for autonomous (automated) calibration of a detector system
  - In an example subdetector system, extensible to others

Toward (AI-Driven) Autonomous Calibration

9

- **Our primary objective is to build an *autonomous calibration system*, capable of detecting when calibrations are needed, executing them reliably, and integrating results into the reconstruction.**
- **AI/ML methods serve as powerful tools that can enhance selected components.**

Three Core Tasks for autonomy

1. Calibration Logic (AI-assisted decision engine)
- Software must detect when new calibration is required (e.g., change detection, drift detection) and update the state machine accordingly. **(AI can assist)**

2. Calibration Integration Into the State Machine
- The calibration workflow must connect to a calibration/conditions DB to track calibration status and link to calibration data, define who reads/writes constants, and manage workflow transitions.

3. Calibration Execution & Validation (AI-supported QC)
- Calibration scripts must compute new constants, validate them, and register them in the database. **(AI can assist in automated quality checks or validation scoring.)**

Operational Boundary Conditions

1. Online Condition
- Must function during live data-taking
- File-based workflows as an initial stage but target is fully streaming, low-latency calibration loops

2. Human Condition
- Define necessary manual checkpoints
- Specify where human approval or override is required
- Aim for automation first → integrate human-in-the-loop later

3. Cybersecurity Condition
- Access control, signing of calibration constants
- Addressed in later implementation phase

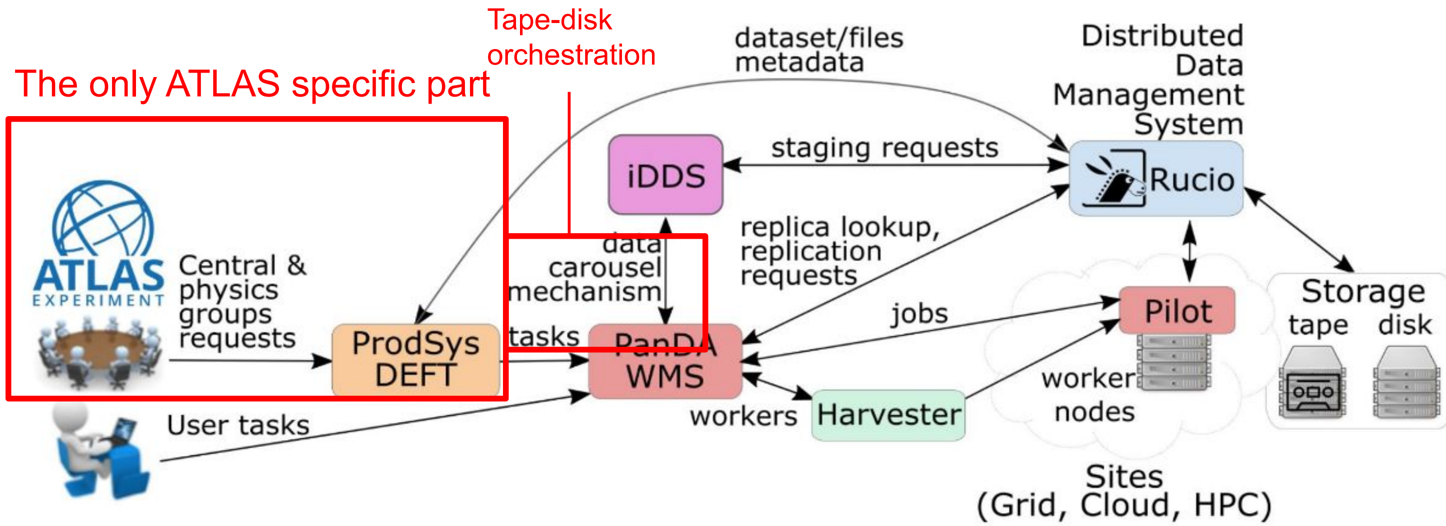
Testbed relevant

Level of Calibration

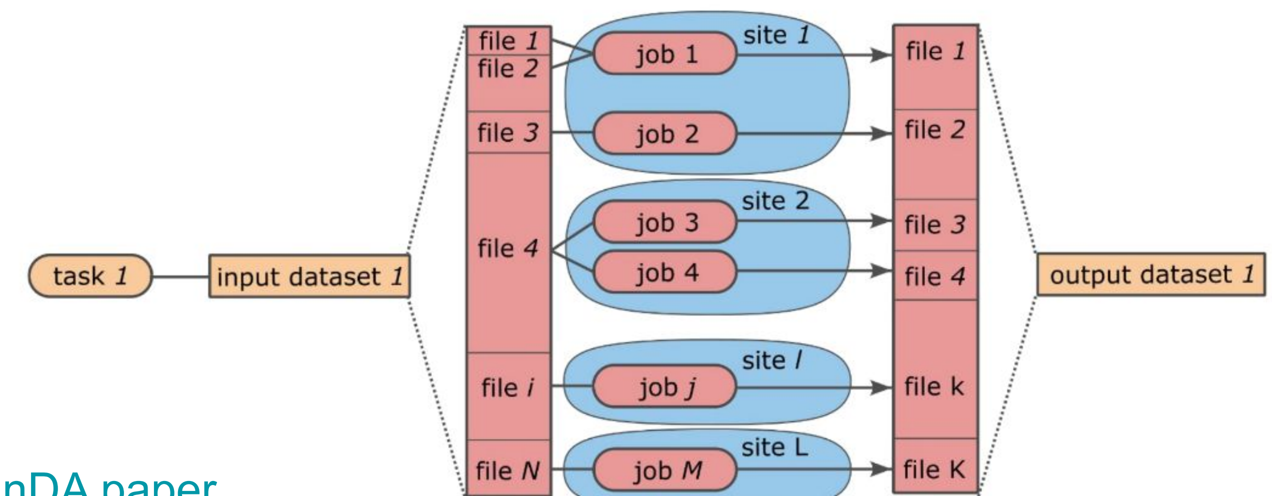
7

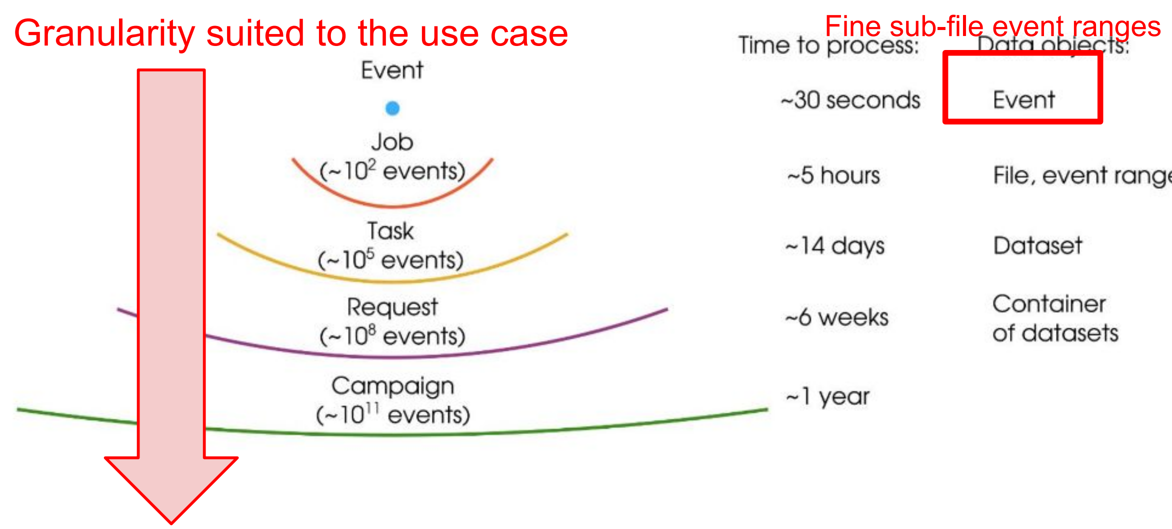
- **What is calibrated? (Calibration Content)**
  - **Detector physical parameters**
    - Bias voltages, gain settings, Temperature / radiation damage corrections, etc
      → Calibration affecting the *physical operation point* of detectors.
  - **Electronics and readout calibration**
    - Pedestals / offsets, Channel-to-channel timing, Amplifier gains, ADC linearity
      → Calibration of *electronics configuration parameters*.
  - **Reconstruction-level calibration**
    - T0 offsets, Energy calibration, Detector alignment
      → *Reconstruction-critical* calibration.
  - **Time-dependent corrections**
    - Clock drift, Temperature-induced slow drifts, Event-by-event T0 corrections via vertex
      → *Time-evolving* calibration parameters.
- **How calibration data is obtained (Calibration Methods)**
  - **Special runs**
    - Pedestal/noise runs, special bunch patterns, Low luminosity runs, Vernier scans
  - **Dedicated on-detector calibration systems**
    - Lasers, LEDs, Pulsers, Diodes
  - **Continuous monitoring during normal beam operations**
    - Built-in calibration features, Streaming pedestal monitoring, Online gain tracking
  - **Physics-based calibration using high-statistics events**
    - Calorimeter energy scale (π^0, MIP, electrons), Tracking alignment (residual-based)
  - **Time-dependent parameter estimation**
    - Clock correction, Slow thermal drift monitoring, Event-by-event T0 estimation

The PanDA workload manager (developed by BNL and UT Arlington)



– **DEFT**: Database Engine For Tasks
– **PanDA**: Production ANd Distributed Analysis System
– **Harvester**: resource-facing service between the PanDA and collection of pilots
– **Pilot**: the execution environment on a worker node Fine grained workflow orchestration
– **iDDS**: Intelligent Data Delivery System
– **Rucio**: Distributed Data Management System





[PanDA paper CSBS 2024]

Brookhaven National Laboratory