PanDA and Rucio

Paul Nilsson (BNL), Tadashi Maeno (BNL), Alexei Klimentov (BNL), Kaushik De (UTA)

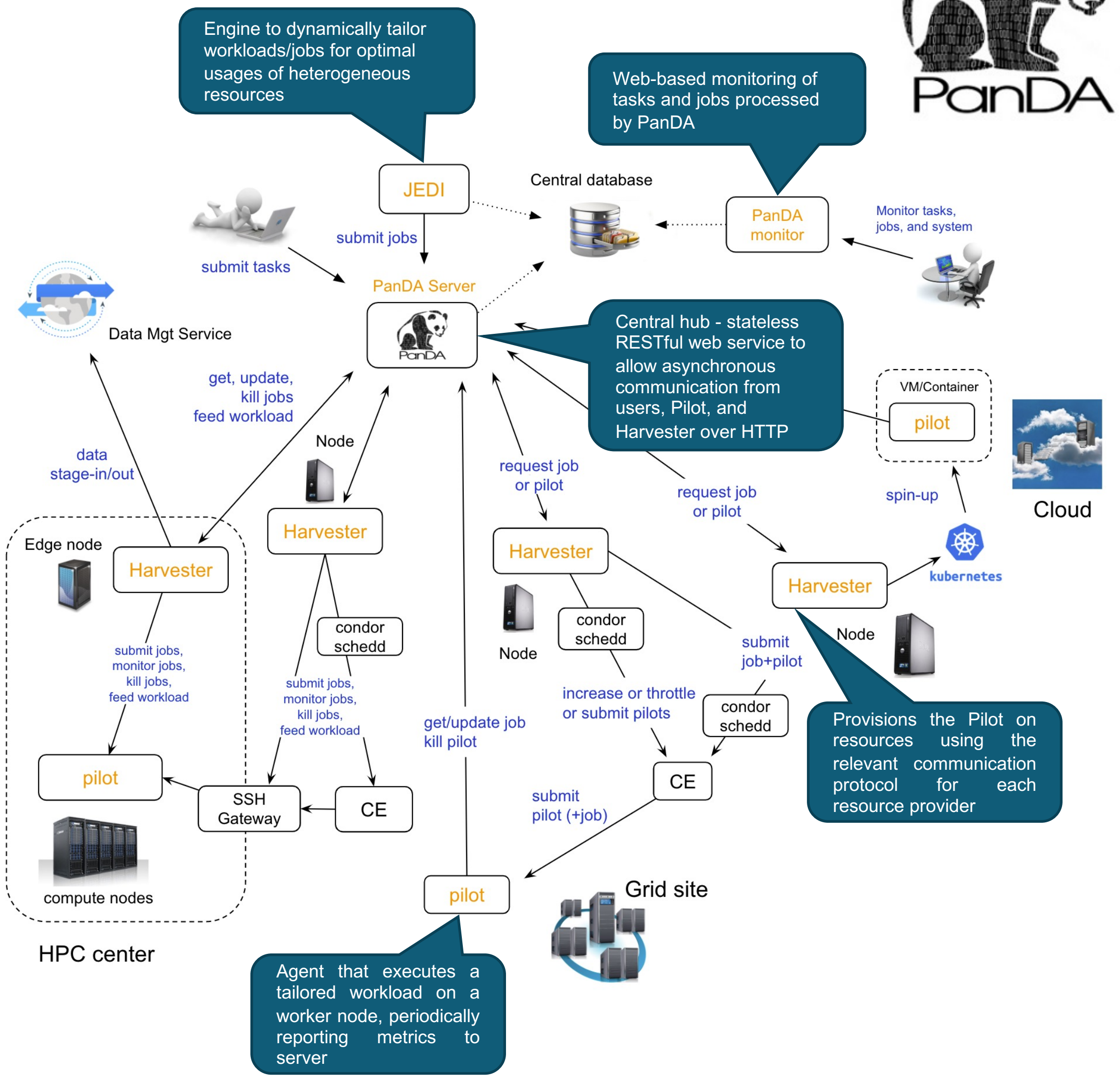
September 27, 2021

ATLAS TCB

@BrookhavenLab

PanDA Overview

- PanDA = Production and Distributed Analysis
- Developed to meet ATLAS requirements for a data-driven workload management system capable of operating at the LHC data processing scale
- Used by ATLAS since 2006, integrating a wide range of computing resources
  - WLCG grid, commercial clouds, volunteer computing and HPCs
- Scalability has been demonstrated through the rapid increase in usage over the last decade
- In ATLAS alone, some 600,000 jobs are typically running in the system at any time – more than 5 millions jobs processed per week
- Currently, PanDA is also used by Rubin(LSST), COMPASS, sPHENIX
- The proven scalability and flexibility make PanDA well suited for adoption by various exabyte scale sciences


Engine to dynamically tailor workloads/jobs for optimal usages of heterogeneous resources
Web-based monitoring of tasks and jobs processed by PanDA
JEDI
Central database
PanDA monitor
Monitor tasks, jobs, and system
submit jobs
submit tasks
PanDA Server
Data Mgt Service
Central hub - stateless RESTful web service to allow asynchronous communication from users, Pilot, and Harvester over HTTP
get, update, kill jobs feed workload
data stage-in/out
Node
VM/Container
pilot
Cloud
request job or pilot
request job or pilot
spin-up
Edge node
Harvester
Harvester
Harvester
Harvester
kubernetes
condor schedd
Node
condor schedd
Node
submit jobs, monitor jobs, kill jobs, feed workload
submit jobs, monitor jobs, kill jobs, feed workload
submit job+pilot
increase or throttle or submit pilots
get/update job kill pilot
condor schedd
Provisions the Pilot on resources using the relevant communication protocol for each resource provider
pilot
SSH Gateway
CE
CE
submit pilot (+job)
compute nodes
pilot
Grid site
HPC center
Agent that executes a tailored workload on a worker node, periodically reporting metrics to server

PanDA and Rucio

- Bidirectional communication between PanDA and Rucio
  - PanDA → Rucio: HTTP requests using python API of rucio-client
  - Rucio → PanDA: Messages through ActiveMQ
- No data management component in PanDA
  - Fully rely on Rucio
  - PanDA system has its own dataset/file catalog
    - Used internally e.g. for data caching
- Most workloads read input files and upload output files only from/to local storages via LAN
  - To avoid chaotic data traffic over WAN

T. Maeno

PanDA and Rucio: Usages

- Dataset and file location discovery
  - To schedule workloads to computing resources that have input data locally
- File distribution
  - To send input files to computing resources that are idle but don’t have the input files locally
  - Possible to distribute workloads more widely
- Preplacement of input files before running workloads on actual computing resources
- File aggregation
  - To aggregate output files produced by workloads
  - Asynchronous stage-out of output files to the final destination
- Additional hints for workload brokerage
  - Status of data transfer channels
  - Storage and free space sizes
  - Blocklist of storages
- User information lookup such as email address
- Lifetime-based data cleanup

Pilot

From slide 2

- Pilot: Agent that executes a tailored workload on a worker node, periodically reporting metrics to server
  - Responsible for any related file transfer to/from the worker node
- Interacting with Rucio via rucio.api for stage-in/out
- Pilot also sends traces to Rucio server via https for each file transfer
  - Errors on the Rucio end are reported back to Pilot, which in turn reports to PanDA server
  - Bulk transfer capability would be nice instead of having to send individual traces in sometimes massive loop

A. Klimentov

Data Carousel

- **Data Carousel**: on-demand reading from tape without pre-staging
  - Ultimate goal : use tape more efficiently and actively
- Uses a rolling disk buffer whose size can be tuned to suit available resources and production requirements
- Key to success: rate at which data can be staged to disk at the Tier-0 and Tier-1 sites
- Technique can eventually be used for any experiment
- **The Data Carousel R&D Project** was initiated at BNL for use with RHIC experiments
  - In full production for STAR and PHENIX for more than 10 years
  - Files were fetched at tape speed for weeks in a row
  - Today it is one of the vital R&D projects to address High Luminosity LHC data processing challenge (scope and scale are different)

ATLAS Data Carousel Project Phases

A. Klimentov

completed

Phase I : Tape Sites Evaluation (Y2018)

- Conduct tape staging tests, understand tape system performance at sites and define primary metrics

completed

Phase II : ProdSys2/Rucio/Facilities integration (Y2019-2020) - *CHEP2019* *talk*

- Address issues found in Phase I
- Deeper integration between workflow, workload and data management systems (ProdSys2/PanDA/Rucio), plus facilities
- Identify missing software components

completed

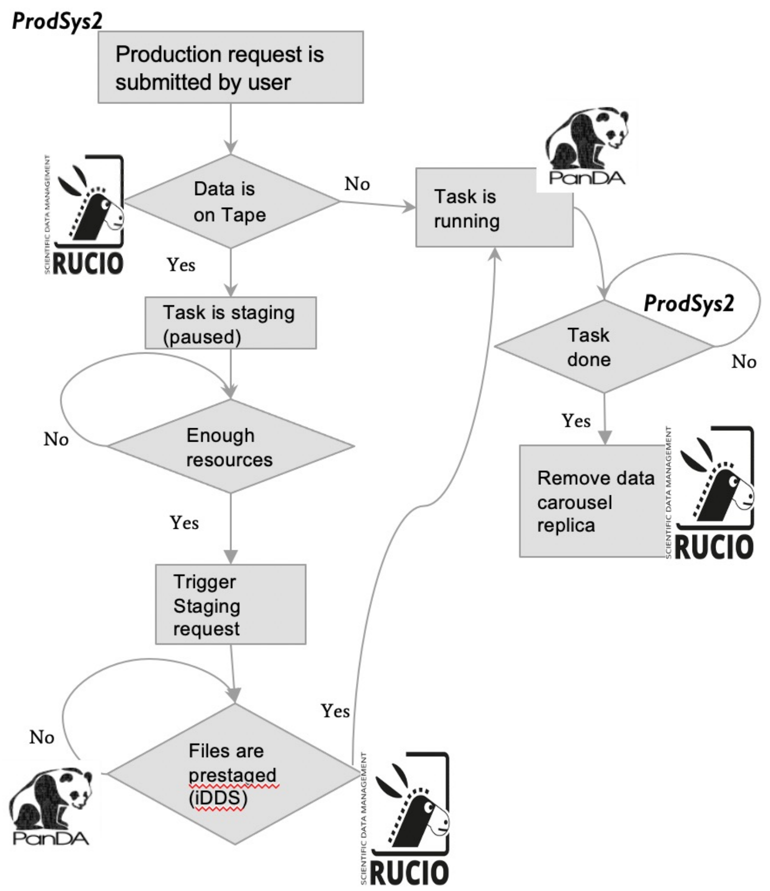
Phase III : Run production, at scale, for selected workflows (Y2020)

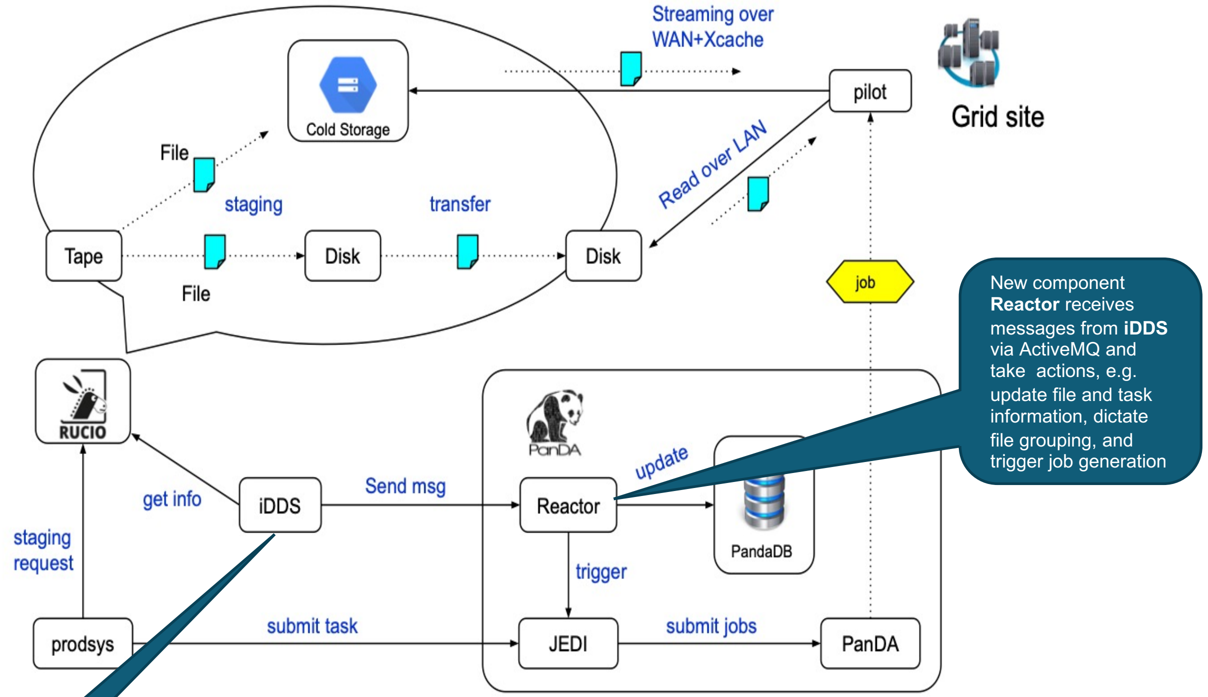
Phase IV : Use data carousel for many workflows in parallel, respect computing share per workflow. Run Data Carousel jointly for more than one experiment (Y2021)

We are now in the middle of Phase IV and have reached the initial goal of the data carousel in full production for LHC Run 3

A. Klimentov

Data Carousel Workflow and New Distributed Software Component : intelligent Data Delivery Service (iDDS)





New component **Reactor** receives messages from **iDDS** via ActiveMQ and take actions, e.g. update file and task information, dictate file grouping, and trigger job generation

See *T.Maeno et al.* iDDS talk

iDDS: designed to intelligently transform and deliver the needed data in a fine-grained approach for High Energy Physics workloads

- **Data Carousel is a use-case of iDDS** - The purpose of Data Carousel with iDDS is to release tasks quickly enough, avoid redundant job attempts, and improve error handling

PanDA-ML

- What is PanDA-ML?
  - Project to modernize PanDA over next 3 years
  - Necessary for scalability, and to lower operations effort during HL-LHC
    - Currently in R&D stage, to be ready for HL-LHC – deploy and test during Run 3
- Goals
  - Add internal ML engine to PanDA, across all modules
  - Improve usability of PanDA for ML workloads
  - Replace hand-tuned algorithms with ML/AI tools
- PanDA-ML engine
  - Training and inference modules, tools and DB bindings
- PanDA-ML framework
  - Enable ML workloads for scalability on heterogeneous resources

K. De

PanDA-ML Use Cases

- ML for brokerage, dispatch, global share, job matching, dynamic job sizing, iDDS, harvester auto-scaling, dynamic data placement, data carousel, anamoly detection, job/task retry, pilot retries, data locating, data caching, fault tolerance, task completion, ..
- Brokerage example
- Anomaly detection – task/site blacklisting
- Task completion – reduce long tails, especially user tasks
- Popularity
- Dynamic data placement – first copy, extra copies, deletions

Goals and Keys to Dynamic Data Placement

- It is clear from studies presented in Data Popularity WG that ATLAS stores a lot of rarely used data on disk
- **Goals** of dynamic placement
  - Rely more on data carousel to retrieve data from tape as needed
  - Reduce primary data volume on disk
  - Achieve better model of a global data cache
  - Improve physicist experience – faster access to important data
- **Key 1**: New data placements (first copy)
  - Optimal placement of newly produced data
- **Key 2**: Data replications
  - Extra copies of popular data
- **Key 3**: Data deletions
  - Deletion of less used data

- First copy always to nucleus
- Second copy tape only
- Create clear mapping of Rucio-PanDA-ProdSys interactions

- Reduce long tails for jobs and data
- Reduce wait time for job/task starts
- ML in PanDA and Rucio?

- Goal: reduce primary fraction of disk usage to 50-60%
- Mainly Rucio actions – but help from PanDA/ProdSys
- All PanDA generated data gets short lifetime (2-4w, immediate copy to tape)
- Rucio to automatically secondarize after lifetime ends
- Rucio to delete as watermark reached