PanDA for Data Processing and Distributed Learning

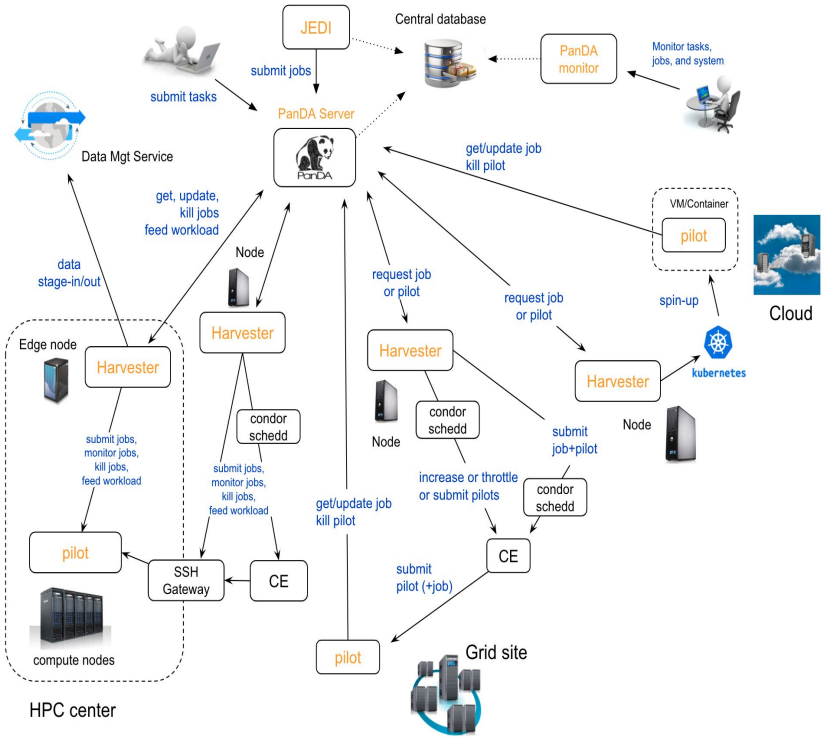
Wen Guan, Lino Oscar Gerlach, Xin Zhao, Tadashi Maeno, Torre Wenaus
on behalf of the PanDA team

ePIC Software & Computing Meeting, CERN
April 22-26, 2024

@BrookhavenLab

Contents

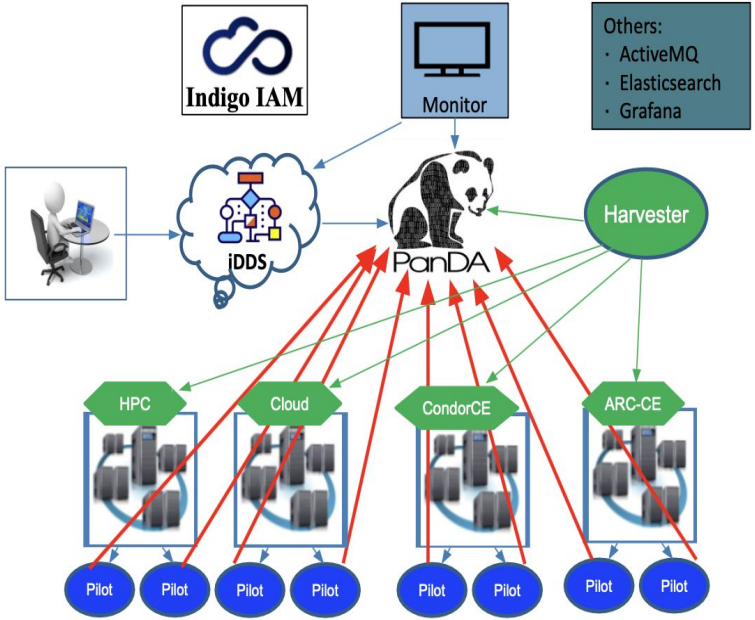
PanDA



- **PanDA (Production and Distributed Analysis): Distributed workload Management**
  - Distributed computing resources
    - Diverse locations
    - Different software
  - Distributed users
    - Different universities/labs
  - General interface for users, one authentication for all sites
  - Integrated different computing resources (Grid, Cloud, kubernetes, HPC and so on), hide diversities from users, large scale
- **Different components of PanDA**
  - PanDA server and JEDI, Harvester, pilot, iDDS and monitor

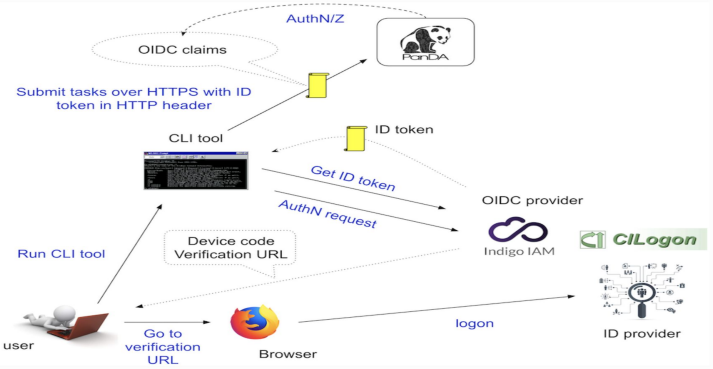
Distributed Computing Resources

- **Worldwide distributed resource providers**
  - Grid
  - Commercial cloud service providers
  - High-performance computing (HPC) and Leading Computing Facilities (LCFs)
  - Volunteer computing
  - Platform-as-a-Service (PaaS) Function-as-a-Service (FaaS)
- **PanDA harvester manages the communication between PanDA and resource providers. It supports a lot of different resource providers**
- **PanDA pilot works as an agent/wrapper to manage the execution environments and monitor the execution of user payloads**



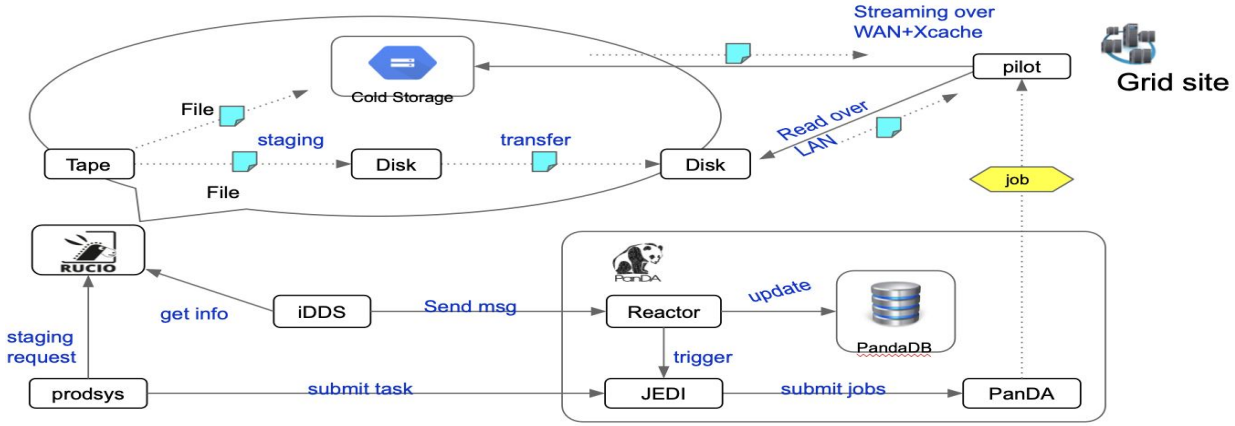
Distributed Users

- **Users from different locations**
  - At different universities, labs, or at home
  - Without requiring users to be registered in a lab before using computing resources in the lab
- **PanDA supports X509 or OIDC for user authentication and authorization**
  - With CILogon, PanDA can authorize users from most of the universities or labs to use PanDA.
  - CILogon is a federated ID broker to delegate authentication to ID providers such as CERN, BNL IT/SDCC, KIT, Google, …
- **PanDA HTTP service**
  - The communication between PanDA client and PanDA server is based on HTTP protocol. It's easy to use PanDA client/API at different locations
- **Containerization**
  - PanDA supports various containers, which makes it possible to run different user payloads
- **PanDA cache**
  - PanDA cache can also be used to transfer user codes from the client side to the executor at a remote site



Automatic Data Placement

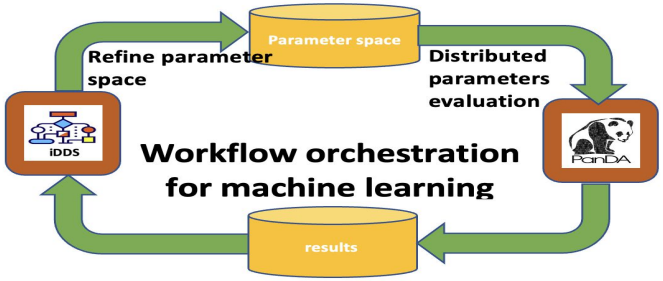
- Integrated with Rucio for automatic data placement
  - Input data caching before releasing jobs
  - Output data replicating after a job finishes
- E.g. Data carousel
  - Fine-grained disk-efficient tape staging
  - On-demand disk replicas and their aggressive deletion
  - Mitigating the disk problem that is the most crucial in ATLAS

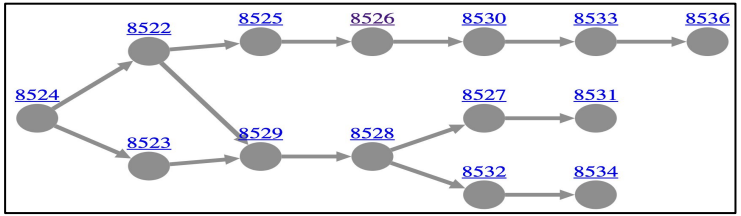


Brookhaven National Laboratory

Complex Workflow Management

- **PanDA iDDS (intelligent Data Delivery Service) works as a workflow engine**
  - Coordinate and orchestrate tasks and data
  - Streamline operations into a workflow, to improve automation and efficiency
- **Workflow management orchestration**
  - DAG (Directed Acyclic Graph)
  - Complex workflow
  - Asynchronous results
- **Supported workflow description languages**
  - YAML + Common Workflow Language (CWL)
  - Python + snakemake
  - Python decorator based Function-as-a-Task workflow





DAG workflow management

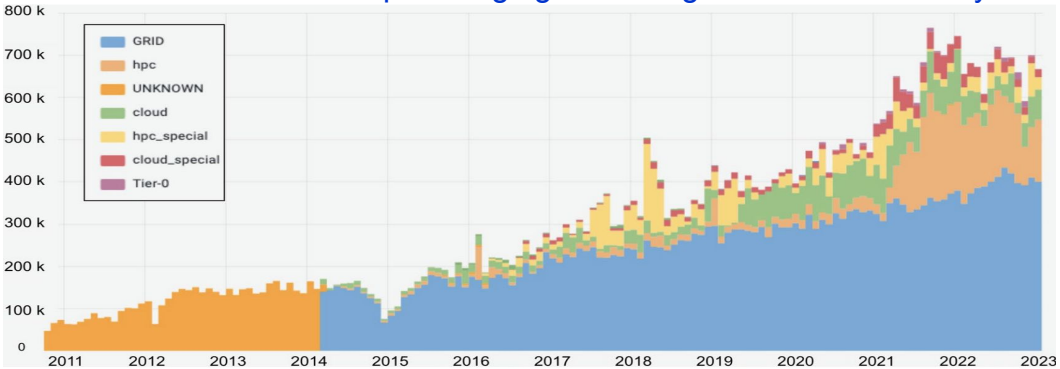
PanDA System

- **Distributed workload and workflow system**
  - Distributed resource providers (Grid, Cloud, HPC, Kubernetes and so on)
  - Distributed users from different universities or labs
- **Powerful for data processing**
  - Automatic data placement (inputs caching and outputs replicating)
  - Workflow supports for constructing user pipelines
  - Smart scheduling which involves user resource requirements, available computing resources, data replicas, workflows and so on

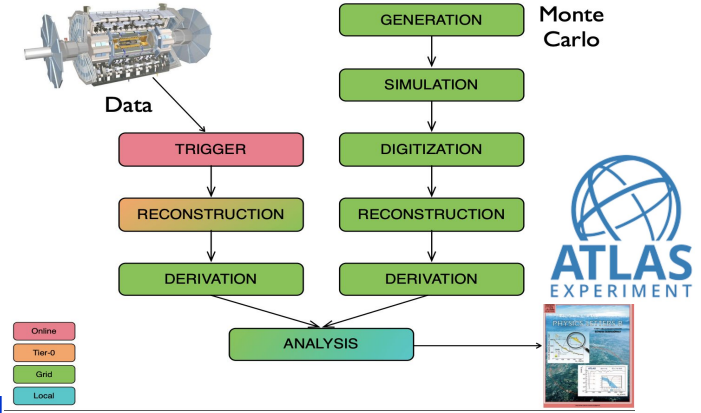
Contents

PanDA @ ATLAS

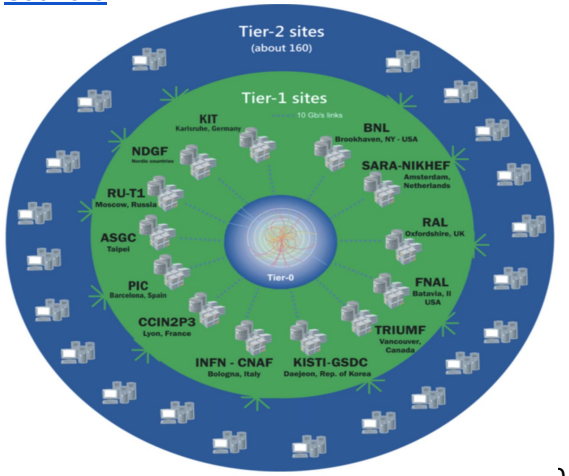
- **PanDA has been developed since 2005 to meet the ATLAS production and analysis requirements**
  - PanDA manages the data processing and analysis on Grid (T1/T2), HPC, Cloud and so on
  - At about 150 sites around the world
  - Has reached 800K concurrent CPU cores
  - Flexiliable to adapt emerging technologies and continuously evolving



Monthly average CPU cores managed by PanDA for ATLAS by resource types between 2011 and 2023. The accounting did not track the resource type until 2014, since the usage of non-Grid resources was negligible (labeled “UNKNOWN”) (Computing and Software for Big Science)



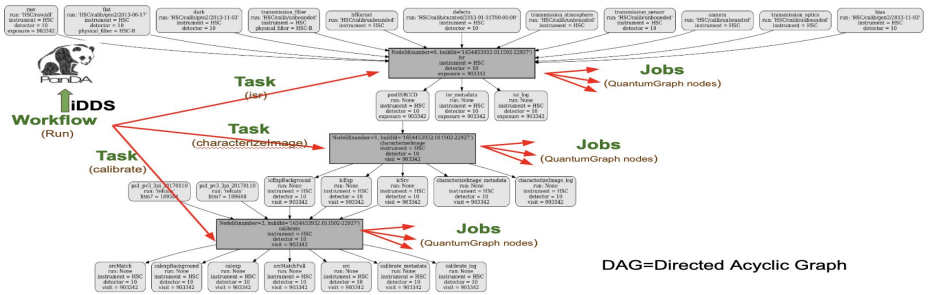
The ATLAS data processing chain, James Catmore



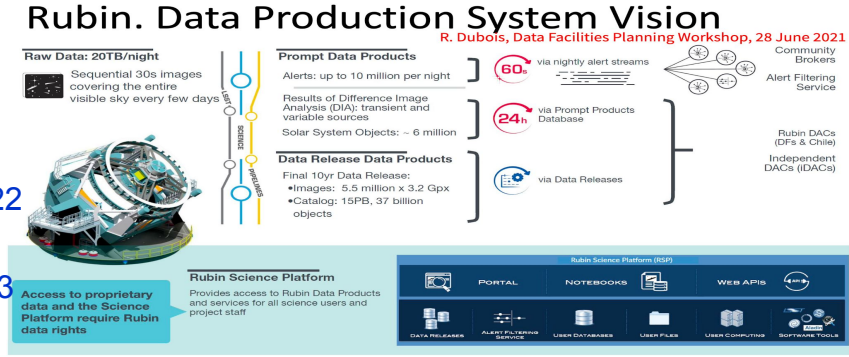
WLCG Grid Computing (T0 -> T1 -> T2)

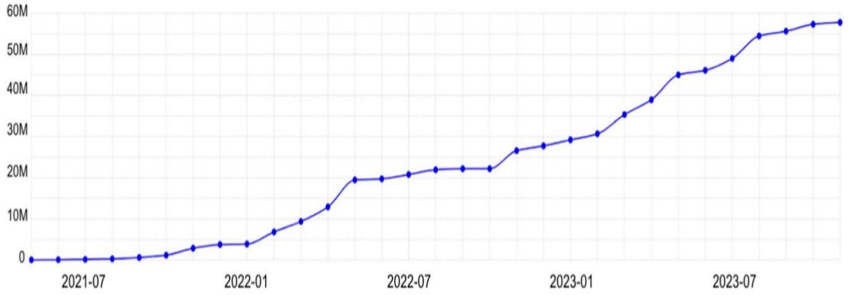
PanDA @ Rubin

- **DAG management for Rubin Observatory to sequence data processing based on dependencies since 2020**
  - Largely stable since Oct 2021
  - DP0.2 (Phase 2 of Data Preview 0) campaign successfully, 2022
  - HSC (Hyper-Suprime Cam) processing, 2022
  - Dedicated PanDA deployed at SLAC for Rubin production, 2023
    - Multiple Data Facilities (DF)
      - USDF (SLAC), FrDF (IN2P3), UKDF (RAL&LANCS)
    - Kubernetes based deployment
    - Postgresql database



DAG=Directed Acyclic Graph

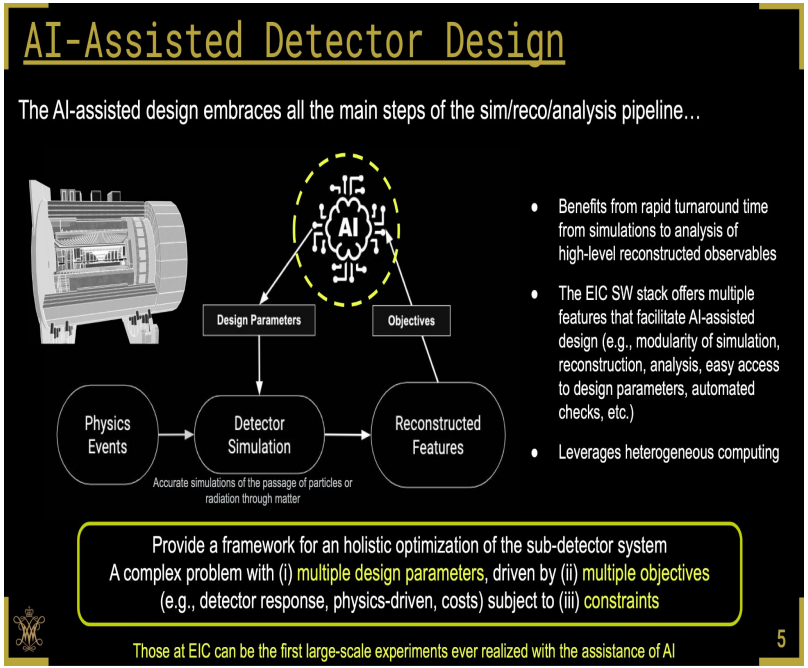




Accumulated number of Vera C. Rubin jobs managed by PanDA from 2021 to 2023 (In 2023 winter, a new dedicated PanDA system was deployed for Rubin) (Computing and Software for Big Science)

PanDA @ EIC

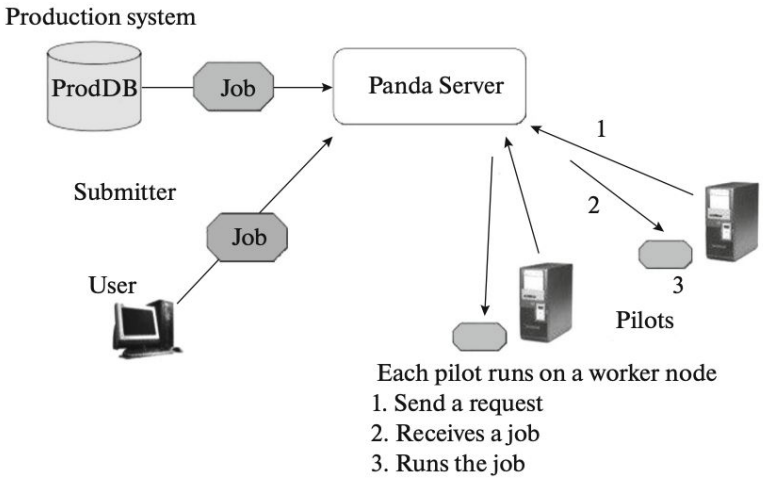
- **AI-assisted Detector Design for EIC (AID2E)**
  - Employ PanDA to manage parameter optimization tasks for AID2E on distributed resources
  - Large scale distributed machine learning
  - Fine-grained automation of multi-step iterative workflows
  - Multiple Objective Bayesian Optimization (MOBO) with AX Adaptive Experiment Platform (pyTorch based)
  - Successfully integrated PanDA with AX
  - Containerization
- **DOMA PanDA @ CERN**
  - DOMA PanDA @ CERN is deployed for pre-production tests for different experiments
  - AID2E is using this one currently
- **Integration works**
  - Continuously improving it for parallelization and scaling



Cristiano Fanelli

PanDA @ others

- **COMPASS**
  - COMPASS is using PanDA for data processing for a long time
  - Currently it's upgrading PanDA with kubernetes deployments
- **Some other experiments are also interested in PanDA**
  - sPHENIX
  - SKA
  - SPD at NICA

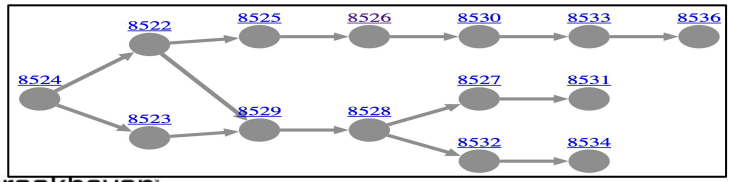


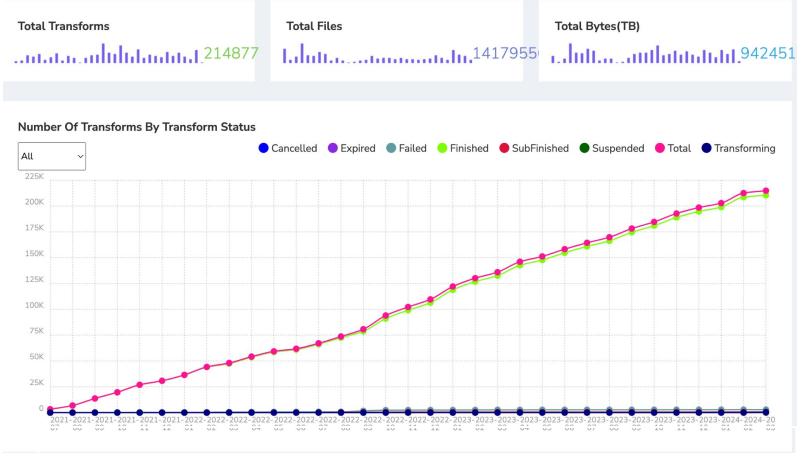
PanDA for COMPASS at JINR1

Contents

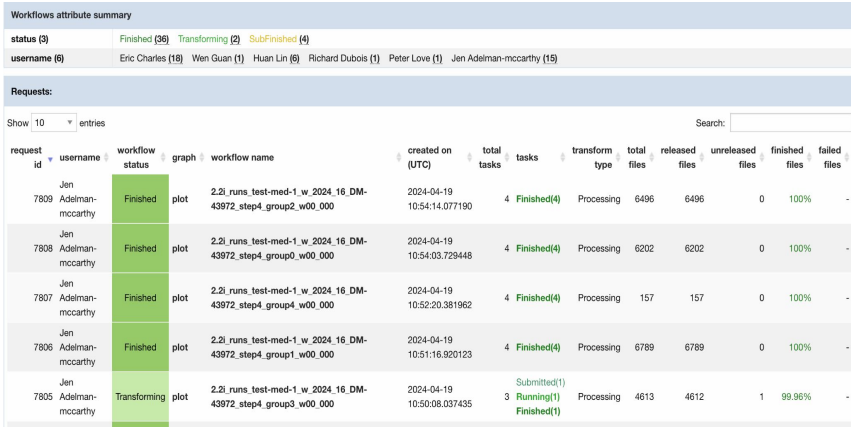
PanDA DAG workflow

- **ATLAS Data Carousel**
  - Employs messages to manage data staging and data processing dependencies in a proper granularity
  - In production since 2020
  - From 2021, has processed 942 PB data
- **Rubin Pipeline**
  - PanDA manages tasks DAG and jobs DAG dependencies
  - Stably integrated since 2021
  - A dedicated PanDA was deployed in Rubin at 2023





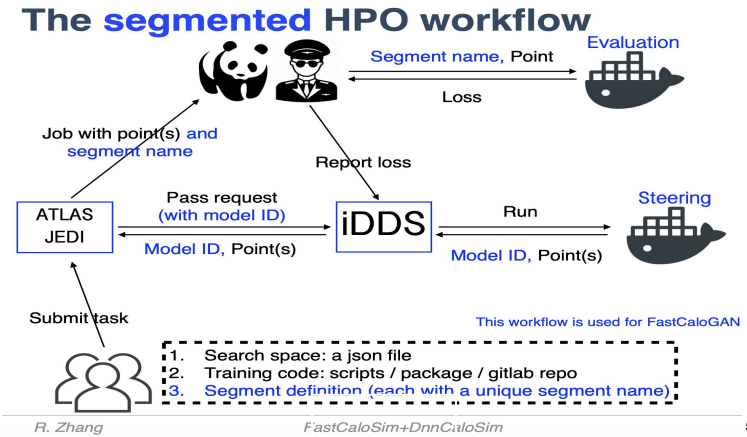
Since late 2021, ATLAS Data Carousel has processed 942 PB data (old monitor information are archived)



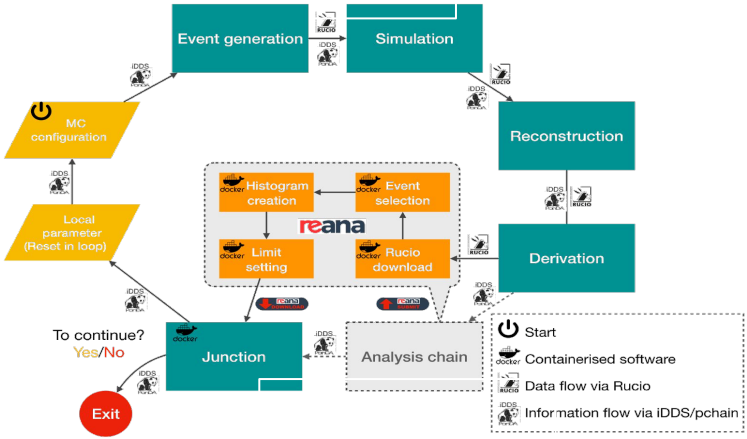
Latest workflows

PanDA ML workflow

- **HyperParameter Optimization (HPO), a full-automated platform for HPO on top of distributed computing resources**
  - Schedules ML training jobs to distributed heterogeneous GPUs to evaluate the performance of the hyperparameter
  - Orchestrates to collect the results and search new hyperparameters based on the previous results
  - The HPO service is in production for FastCaloGAN, part of the production ATLAS fast simulation AtlFast3
- **Active Learning, an iterative ML assisted technique to boost the parameter search in New Physics search space**
  - Automate the multi-steps parameter redefining and evaluation chain
  - Integrated REANA (Reusable Analyses) for learning processing
  - Applied the Active Learning service in the H → ZZ_d → 4ℓ dark sector analysis in ATLAS
- Ref: CHEP2023



HPO



Active Learning

PanDA Function-as-a-Task Workflow Management

- **Challenges for complex workflow management**
  - Complicated to support different logical requirements in different use cases
  - Complicated for users to define different dependency logics
    - Eg. easy to make mistakes between user requirements and system behaviors when a complicated logic is defined
  - Complicated for user experience
    - Difficult to convert some user software stack to other workflow management tools
    - User preference is important
- **A new Function-as-a-Task Workflow Management framework is developed**
  - With python functions to define the workflow steps
  - With python decorators to convert functions to distributed parallized tasks
  - Workflow executes python tasks like local functions, transparent to users
  - With messages (ActiveMQ) to transfer the outputs back
- **Apply Function-as-a-Task workflow for AI-assisted Detector Design for EIC (AID2E)**
- Ref: ACAT2024

```
@work(map_results=True)
def optimize_work(opt_params):
```

With python decorator **@work** to convert a function to a PanDA task

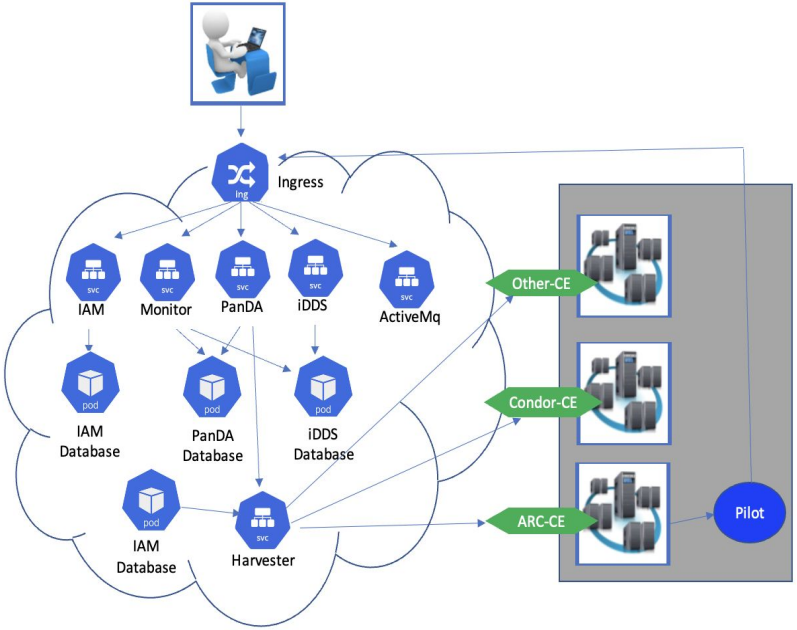
```
@workflow
def optimize_workflow():
    from optimize import evaluate_bdt, get_bayesian_optimizer_and_util

    ...
    n_iterations, n_points_per_iteration = 10, 20
    for i in range(n_iterations):
        points = {}
        group_kwargs = []
        for j in range(n_points_per_iteration):
            x_probe = bayesopt.suggest(util)
            u_id = get_unique_id_for_dict(x_probe)
            print('x_probe (%s): %s' % (u_id, x_probe))
            points[u_id] = {'kwargs': x_probe}
            group_kwargs.append(x_probe)

        results = optimize_work(opt_params=params, group_kwargs=group_kwargs)
        print("points: %s" % str(points))
```

The Workflow calls the task like a local function, transparent to users

Contents

PanDA system deployment

- **Containerization**
  - All PanDA services are containerized
  - Including postgresql and mariadb database
- **Helm deployment to kubernetes**
  - Prepared helm deployment templates
  - PanDA can be deployed to kubernetes easily with helm
  - Users only need to update the configuration files
- **Site service**
  - PanDA supports a wide range of resource providers which may already deployed at different sites
  - It's easy to integrate sites to PanDA system



PanDA kubernetes deployment

Available PanDA systems

- **ATLAS @ CERN**
  - Oracle database
  - Virtual machines based deployment
- **Rubin @ SLAC**
  - Kubernetes based deployment
  - Postgresql database
- **DOMA @ CERN**
  - Pre-production tests for multiple experiments
  - AID2E
- **PanDA @ BNL**
  - Coming soon
  - To support new activities for multiple communities
  - Full production PanDA system with OIDC supports for users from different universities and labs
  - Rucio integration

Contents

- PanDA System
  - Distributed Computing Resources
  - Distributed Users
  - Automatic Data Placement
  - Complex Workflow Management
- PanDA at Different Experiments
  - ATLAS
  - Rubin Observatory
  - EIC
  - Others
- PanDA Workflow and ML
  - DAG (Directed Acyclic Graph)
  - Distributed ML
  - Function-as-a-Task workflow
- PanDA Deployment
  - Kubernetes deployment
  - Available PanDA systems
- **Conclusion**

PanDA Benefits

- **Distributed users**
  - General user interface
  - Without requiring user accounts at different sites
- **Distributed sites**
  - Hide differences and difficulties of various sites
  - Supported resource providers: HTCondor, Slurm, ARC CE, CondorCE, saga, Cloud GCE, kubernetes, PBS, LSF and so on
- **Pipeline**
  - Automatic data placement
  - Workflow supports (CWL, Snakemake, Function-as-a-Task)
  - Smart scheduling which involves user resource requirements, available computing resources, data replicas and workflows
- **Monitoring**
  - Global PanDA monitors to monitor jobs at different site
  - Grafana/ElasticSearch integration
- **Grid/Cloud log access**
  - Distributed log access for logs at different sites
  - Realtime log access

Conclusions

- PanDA is a powerful large scale distributed production data processing and analysis system
- Automatic data placement and workflow supports will simply user pipelines
- Different DAG workflows and ML workflows supports
- Pre-production PanDA demonstration system available for tests and getting experience
- With kubernetes, it’s easy to deploy a PanDA system
- Nearly 20 years of production experience, many adaptations to handle different situations to get good performance
- Flexiliable to adapt emerging technologies and continuously evolving

Backups

PanDA

The flagship: PanDA workload manager

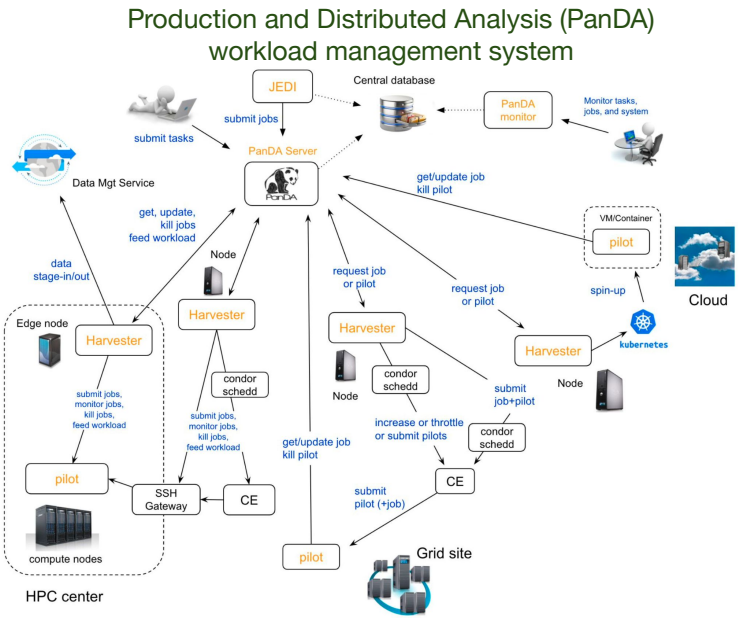
- Developed by BNL and UT Arlington (UTA) 2005-present
- 24x365 processing on ~800k cores globally for ATLAS
- All workloads, production and analysis, at ~150 facilities, ~1500 users, ~1M jobs/day, Exabyte throughput/year
- All resource types: WLCG, clouds (GCP, AKS, k8s-based clusters), HPCs, BOINC, …

Built to scale smoothly to HL-LHC (2030+)

- Efficient optimization of workflows to economize storage and processing at the HL-LHC
- Tightly integrated with data management (Rucio) delivering e.g. Data Carousel functionality (cf. RHIC since the 2000s)

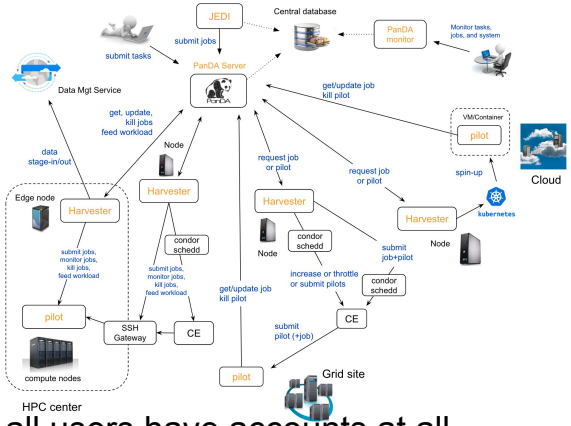
Developed in recent years into an engine for complex workflows

- Deliver the same capability of scaling up to large problems that it provides to production computing
- e.g. Hyperparameter optimization capability: basis for training the first ML application in the ATLAS production simulation
  - FastCaloGAN: 600 networks across different particle types, energies, solid angles, 100 GPU training days



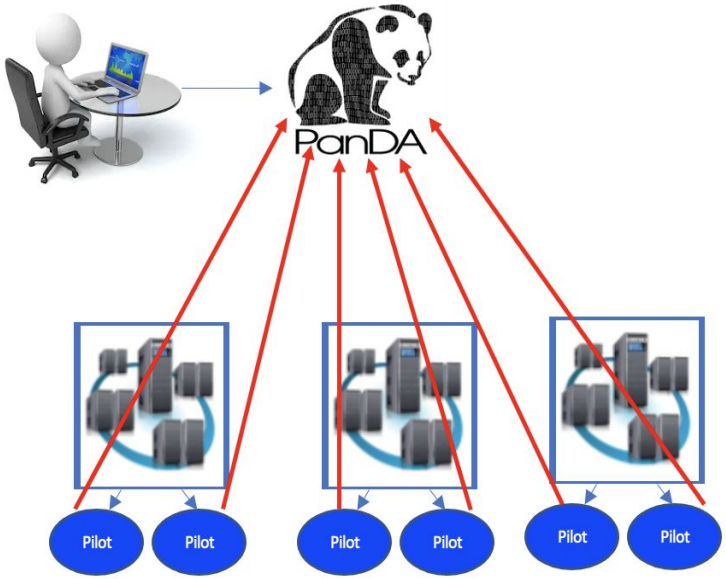
Why PanDA - Distributed Workload Management

- Distributed Heterogeneous computing resources
  - Diverse locations
  - Different software (slurm,condor,pbs)
  - Different work directory and so on
  - Site differences will increase user complexity. One user normally can only work on 1~2 sites, will ignore other sites even they have free CPUs.



- Distributed Users
  - Users from different universities/labs, difficult/impossible to make sure all users have accounts at all sites.
    - LHC 170 sites and several thousand users, Rubin: 3 DFs (more and more sites and users in the future?)
    - Have an account at a site is complicated: request form, security courses and so on
    - It may take a half year or even longer for a user to get accounts in 170 sites.
- PanDA - Distributed workload Management system
  - General interface for users, one authentication for all sites
  - Integrate different resource providers(Grid, Cloud, k8s, HPC), hide the diversities from users
  - Manage complex workflow, data scheduling and so on.

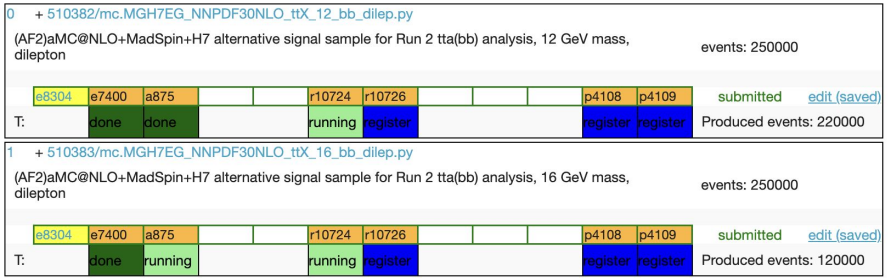
PanDA



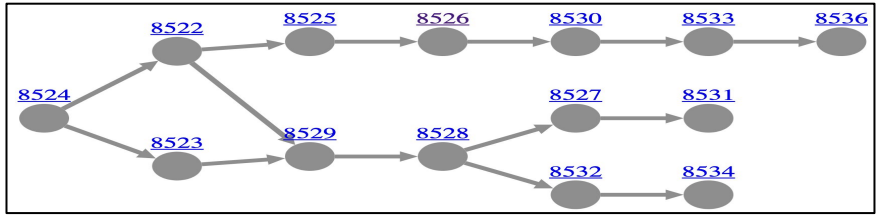
- PanDA provides general REST interface for all users.
- PanDA and Pilots (red lines) work together as a workload management system to integrate distributed computing resources
  - Pilot works as an agent to pre-occupy the CPUs, validate the environments and pull jobs from PanDA.
  - Pilot starts user jobs, monitors user jobs and heartbeats to PanDA server
- PanDA-pilots works like a distributed virtual cluster.
  - Pilot works as an agent, which can avoid a lot of user job failures.
  - Distributed pilots to hide differences of heterogeneous computing resources.
- PanDA is more than a virtual cluster.
  - Data scheduling
  - Complex workflow management

Workflow Management in PanDA/iDDS

- **Workflow Management**
  - Coordinate and orchestrate tasks and data
  - Streamline operations into a workflow, to improve automation and efficiency
- **Workflows**
  - N x M dependencies between upstream and downstream tasks
  - Conditional branching for downstream task execution depending on the results of upstream tasks
  - Loop of task chain based on the results of previous iterations
    - Eg: HPO
- **Supported workflow description languages**
  - Common Workflow Language (CWL), snakemake, …



Examples of data flow based workflow: Even Gen -> Simul -> Reco -> Deriv



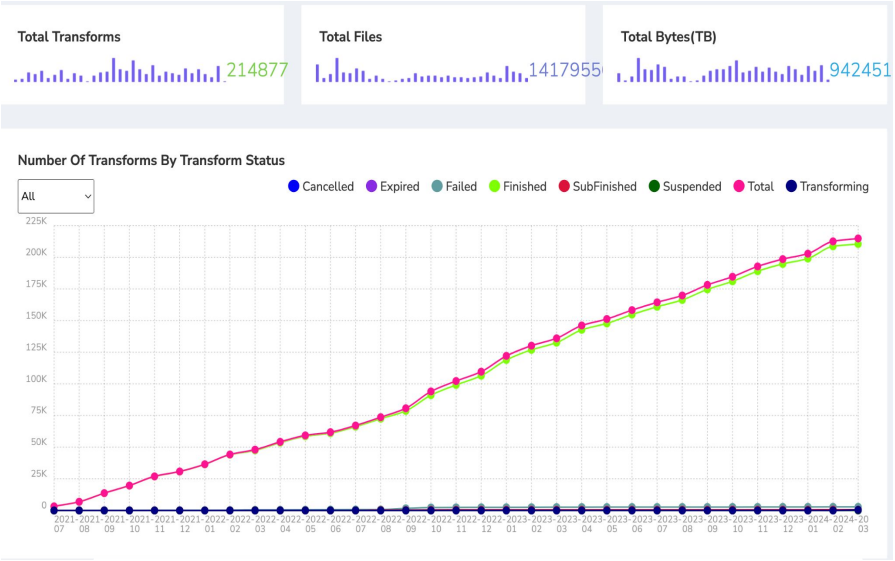
An example of A DAG workflow

Distributed Workflow Management Use Cases

- **Started workflow integration in PanDA and iDDS for a long time, various use cases in production, processing a lot of data and different physics analysis**
  - Fine-grained Data Carousel for **LHC ATLAS**
  - DAG management for **Rubin Observatory** to sequence data processing
  - Distributed HyperParameter Optimization (HPO)
  - Monte Carlo Toy based Confidence Limits
  - Active Learning assisted technique to boost the parameter search in New Physics search space
- **Currently developing**
  - AI-assisted Detector Design for **EIC (AID2E)**
    - A new Function-as-a-Task workflow implementation
    - An enhancement of the HPO mechanism

Fine-grained Data Carousel for LHC ATLAS

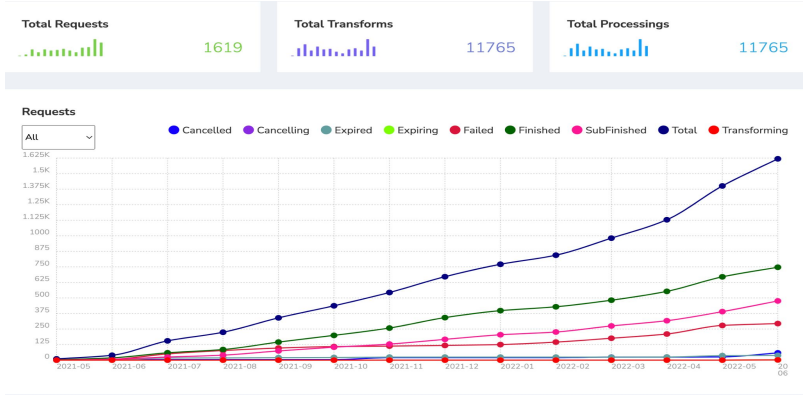
- Fine-grained **Data Carousel** for **LHC ATLAS** enables processing in proper granularities and grouping to efficiently use disk storage
  - iDDS employs messages to trigger PanDA processing data in proper granularity, instead of per dataset
  - In production since 2020
  - From 2021, has processed 942 PB data (old processing information has been archived)



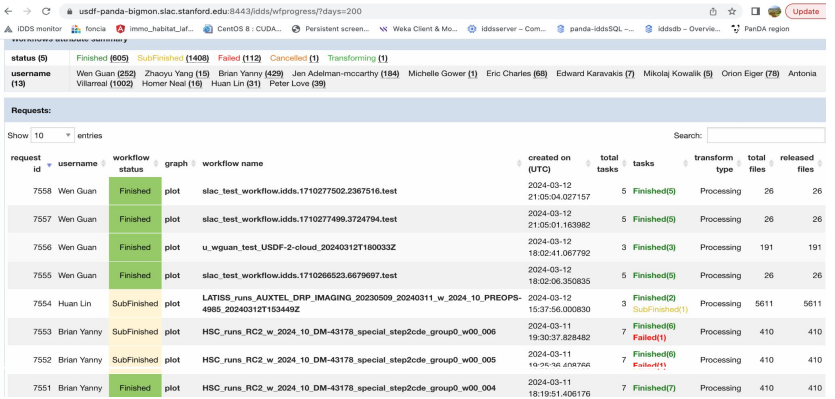
Since late 2021, ATLAS Data Carousel has processed 942 PB data (old monitor information are archived)

DAG management for Rubin Observatory

- **DAG** management for **Rubin Observatory** to sequence data processing based on dependencies since 2020
  - Largely stable since Oct 2021
  - DP0.2 (Phase 2 of Data Preview 0) campaign successfully, 2022
  - HSC (Hyper-Suprime Cam) processing, 2022
  - Dedicated PanDA/iDDS deployed at SLAC for Rubin production, 2023
    - Multiple Data Facilities (DF)
      - USDF (SLAC), FrDF (IN2P3), UKDF (RAL&LANCS)
    - Kubernetes based deployment
    - Postgres database



From May 2021 to May 2022 in Rubin Observatory, iDDS-PanDA within the LSST framework has processed more than 11000 tasks.

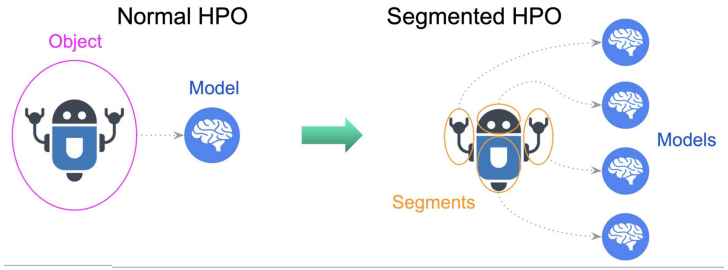


Since middle of last year, the USDF PanDA starts production. It has processed about 2000 workflows (every workflow can have 5~10 tasks

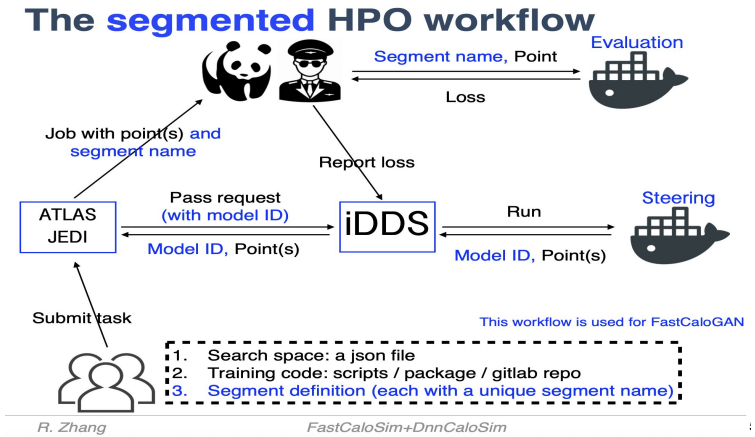
Distributed HyperParameter Optimization (HPO)

- Provide a full-automated platform for HPO on top of distributed heterogeneous computing resources
  - Hyperparameters are generated centrally in iDDS
  - PanDA schedules ML training jobs to distributed heterogeneous GPUs to evaluate the performance of the hyperparameter
  - iDDS orchestrates to collects the results and search new hyperparameters based on the previous results
- Applied for ATLAS FastCaloGAN
  - The HPO service is in production for FastCaloGAN, part of the production ATLAS fast simulation AtlFast3
  - With hyperparameters to tune various models targeting different particles and slices
  - Distributed GPUs, HPCs, commercial cloud
  - Ref: FastCaloGAN, AML workshop, IML, ATLAS S&C week
- Used in ATLAS, however not specific to ATLAS

❖ Ref: CHEP2023



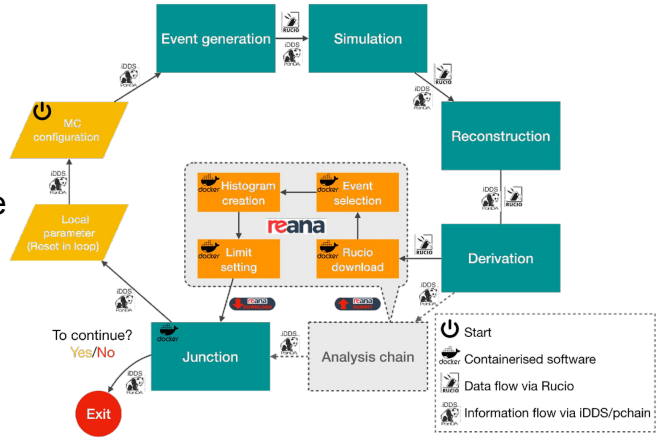
R. Zhang 5th ATLAS Machine Learning Workshop

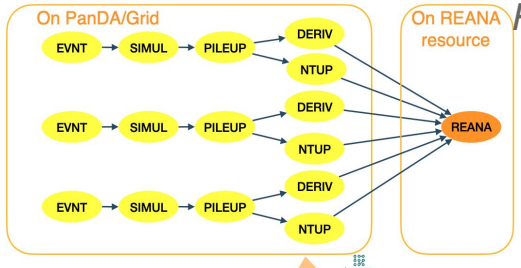


R. Zhang FastCaloSim+DnnCaloSim

Active Learning for ATLAS

- An iterative ML assisted technique to boost the parameter search in New Physics search space
  - The Active Learning technique we are applying was developed by Kyle Cranmer et al, “Active Learning for Excursion Set Estimation”, ACAT 2019
  - Automate the multi-steps parameter redefining and evaluation chain
  - Integrated REANA (Reusable Analyses) with PanDA/iDDS for learning processing
- Applied the Active Learning service in the $H \rightarrow ZZ_d \rightarrow 4\ell$ dark sector analysis
  - Apply Bayesian Optimization to refine the parameter space
  - Greater efficiency, scalability, automation enables a wider parameter search (instead of 1D, 2D or even 4D on large scale resources) and improved physics result
  - Has demonstrated active learning driven re-analysis for dark sector analysis
  - ATLAS PUB NOTE in progress

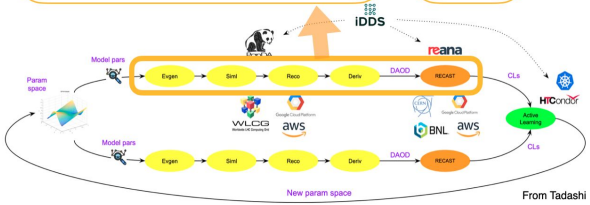




R. Zhang

pchain workflow:

- Three independent chains with different random seeds run in parallel
- Six input files are fed to the final REANA task



From Tadashi

[CHEP2023 Talk: C. Waber, et al. An Active Learning application in a dark matter search with ATLAS PanDA and iDDS]

A New Function-as-a-Task Workflow Management in iDDS

- **Challenges for complex workflow management**
  - Complicated to support different logical requirements in different use cases
  - Complicated for users to define different dependency logics
    - Eg. easy to make mistakes between user requirements and system behaviors when a complicated logic is defined
  - Complicated for user experience
    - Difficult to convert some user software stack to other workflow management tools
    - User preference is important
- **A new Function-as-a-Task Workflow Management framework is developed**
  - With python functions to define the workflow steps
  - With python decorators to convert functions to distributed tasks
  - Workflow executes python tasks like local functions, transparent to users

```
@work(map_results=True)
def optimize_work(opt_params):
```

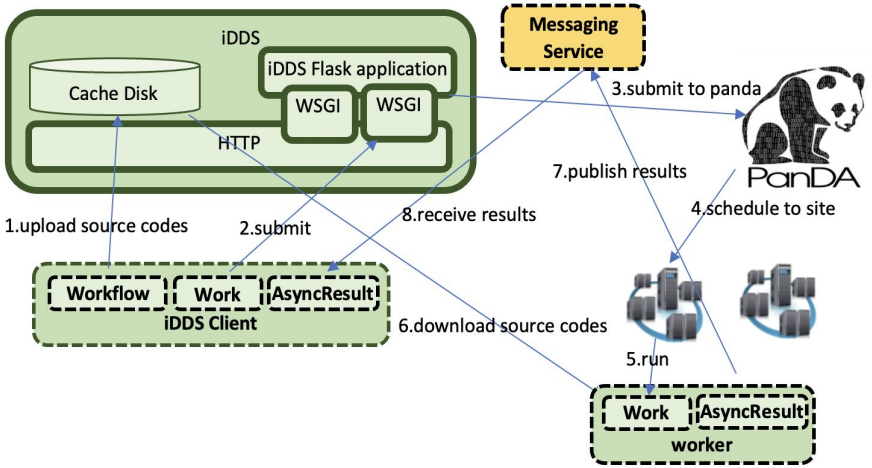
With python decorator **@work** to convert a function to a PanDA task

```
@workflow
def optimize_workflow():
    from optimize import evaluate_bdt, get_bayesian_optimizer_and_util

    ...
    n_iterations, n_points_per_iteration = 10, 20
    for i in range(n_iterations):
        points = {}
        group_kwargs = []
        for j in range(n_points_per_iteration):
            x_probe = bayesopt.suggest(util)
            u_id = get_unique_id_for_dict(x_probe)
            print('x_probe (%s): %s' % (u_id, x_probe))
            points[u_id] = {'kwargs': x_probe}
            group_kwargs.append(x_probe)

    results = optimize_work(opt_params=params, group_kwargs=group_kwargs)
    print("points: %s" % str(points))
```

The Workflow calls the task like a local function, transparent to users

Function-as-a-Task Workflow Management Schema

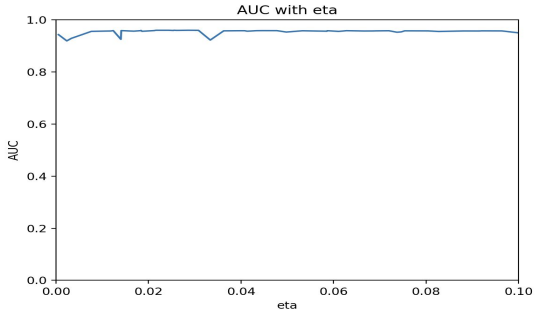
- **Workflow**
  - Source codes caching
    - workflow as the basic unit to manage source codes
    - Source codes in the workflow directory will be uploaded into the iDDS or PanDA http cache
    - During running time, the source codes will be downloaded to the current running directory
  - Running environment
    - Base environment (eg: cvmfs) + source codes caching
    - Base container + source codes caching
      - Container for user, ShuWei, Ye, 2024 ATLAS S&C
- **Work**
  - Submit function as tasks/jobs to workload management system PanDA
  - Load and run a function as a job at distributed sites
  - List of parameters can be used to call a function, which will create a task with multiple jobs and every job uses item of the list of parameters
- **AsyncResults**
  - When a function finishes, the 'Work' executor will publish the result in a message
  - The 'Work' at submission side will receive the result
- **iDDS also monitors the tasks/jobs submitted. It will publish messages to AsyncResult, to avoid AsyncResult waiting for failed remote workers**



Schema of how a workflow executes a function at remote distributed resources

Example: HPO with Function-as-a-Task

- **Apply Function-as-a-Task for a HPO example analysis**
  - ttH analysis (simulated events with Delphes)
  - Boosted Decision Tree (BDT): xgboost
  - Bayesian based hyperparameter optimization: bayes_opt
- **Base container**
  - Alma9 Singularity container with installed xgboost, bayes_opt
- **Distributed tasks**
  - With one line python decorator '@work(map_results=True)' to convert local functions to distributed tasks
  - Transparent for users to run function as remote tasks and collect results



The work function

panda-doma.cern.ch/tasks/?idds_request_id=6126

iDDS monitor | foncia | immo_habitat_laf... | CentOS 8 : CUDA... | Persistent screen... | Weka Client & Mo... | iddsse

ID / Parent	Task name / TaskType/ProcessingType Campaign Group User Errors / Logged status	Task status / Nfiles	Input files / Nlost / Nfinish % / Nfail %
168809	optimize_iworkflow.optimize_iworkflow.optimize_work_2024_03_06_11_44_36_6126_47811 iDDS Wen Guan Errors	done 20	20 100%
168808	optimize_iworkflow.optimize_iworkflow.optimize_work_2024_03_06_11_27_56_6126_47810 iDDS Wen Guan Errors	done 20	20 100%
168806	optimize_iworkflow.optimize_iworkflow.optimize_work_2024_03_06_11_12_39_6126_47809 iDDS Wen Guan Errors	done 20	20 100%
168805	optimize_iworkflow.optimize_iworkflow.optimize_work_2024_03_06_09_14_42_6126_47807 iDDS Wen Guan Errors	done 20	20 100%
168803	optimize_iworkflow.optimize_iworkflow.optimize_work_2024_03_06_08_57_15_6126_47806 iDDS Wen Guan Errors	done 20	20 100%
168802	optimize_iworkflow.optimize_workflow_2024_03_06_08_55_01_6126 iDDS Wen Guan Errors	done 1	1 100%

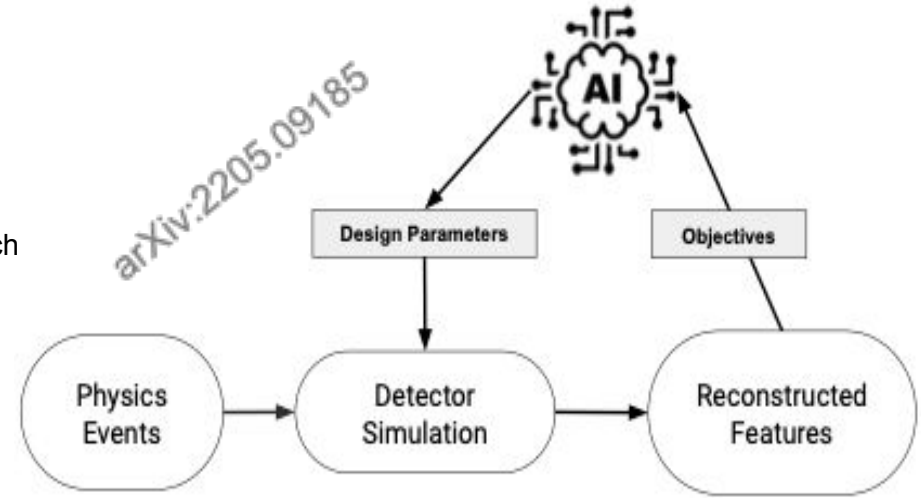
The workflow function

One example workflow: (1) workflow function; (2) 5 iterations and 20 parallel jobs per iteration

```
Best params: {'target': 0.960055699094351, 'params': {'alpha': 0.23406035151804216, 'colsample_bytree': 0.579042534809806, 'eta': 0.028600368999834338, 'gamma': 0
.23818222147295043, 'max_delta_step': 0.8490976659073024, 'max_depth': 19.211553258551916, 'min_child_weight': 70.89679426305557, 'scale_pos_weight': 0.4108082725
8102803, 'seed': 47.50122115129428, 'subsample': 0.9416281815903255}}
```

Apply iWorkflown for AI-assisted Detector Design for EIC (AID2E)

- **Objectives**
  - Employ PanDA/iDDS to manage AI-assisted Detector Design parameter optimization tasks on distributed resources
  - Large scale distributed machine learning
  - Fine-grained automation of multi-step iterative workflows
- **AI-assisted parameter optimization**
  - Many Parameters, multiple detector design objectives
    - Multiple Objective Bayesian Optimization (MOBO)
- **Challenges**
  - AID2E has similarities to existing supported workflows
    - HPO (MOBO)
  - AID2E MOBO using [AX Adaptive Experiment Platform] (pyTorch based)
- **Integration**
  - Successful integrated with AID2E Closure-Test-2.
- **Containerization**
  - The container includes packages such as AX, pytorch, botorch, which is needed by this test.
  - Todo: to apply the EIC simulation container



PanDA Benefits

- User:
  - General user interface
  - Without requiring user accounts at different sites.
- Sites:
  - Hide differences and difficulties of distributed heterogeneous sites
  - Support sites: htcondor,slurm,ARC CE, CondorCE,saga, cloud GCE, k8s, pbs, lsf and etc.
- Distributed virtual cluster
  - Share resources between distributed computing systems.
  - Trigger data movements to resource locations.
  - Complex workflow managements
- Grid/Cloud log access
  - Distributed system requires international log access, to avoid requiring users to login to remote sites to access logs.
  - (PanDA can write logs to local file systems.).
- Monitoring
  - Global PanDA monitor to monitor jobs at different site.
- Grafana/ElasticSearch integration (will deploy)
  - Can automatically publish job information with detail performance statistics such as user jobs' setup time, running time, CPU efficiency, memory usage, error classification and so on.
  - To generate long term monitor for computing resource usage (budget justification for example), error analysis and so on.

Thanks