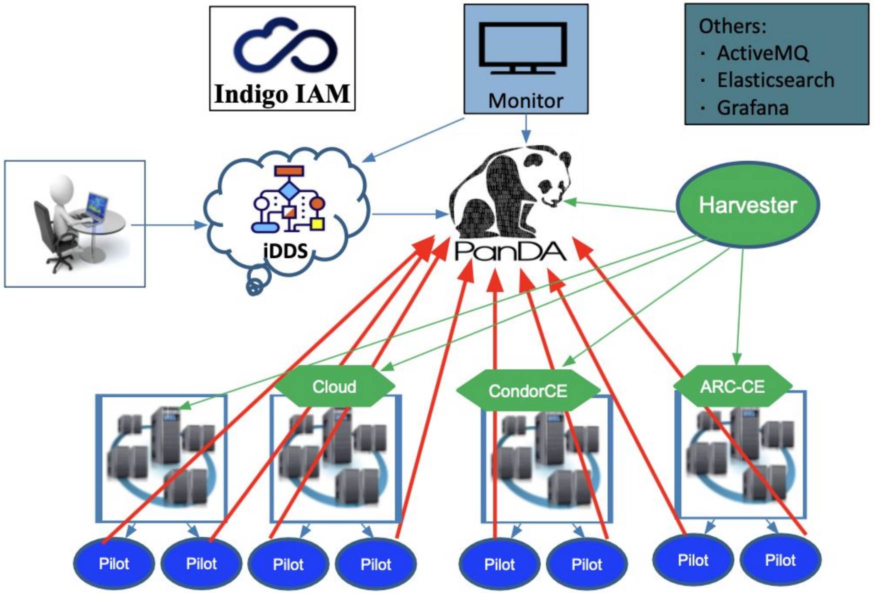
A Function-as-a-Task Workflow Management Approach with PanDA and iDDS

Wen Guan[1], Fernando Harald Barreiro Megino[2], Kaushik De[2], Edward Karavakis[1], Fa-Hui Lin[2], Tadashi Maeno[1], Paul Nilsson[1], Torre Wenaus[1], Rui Zhang[3], Xin Zhao[1], Zhaoyu Yang[1]

1. Brookhaven National Lab
2. University of Texas at Arlington
3. University of Wisconsin Madison

Mar, 2024

Distributed Computing with PanDA/iDDS



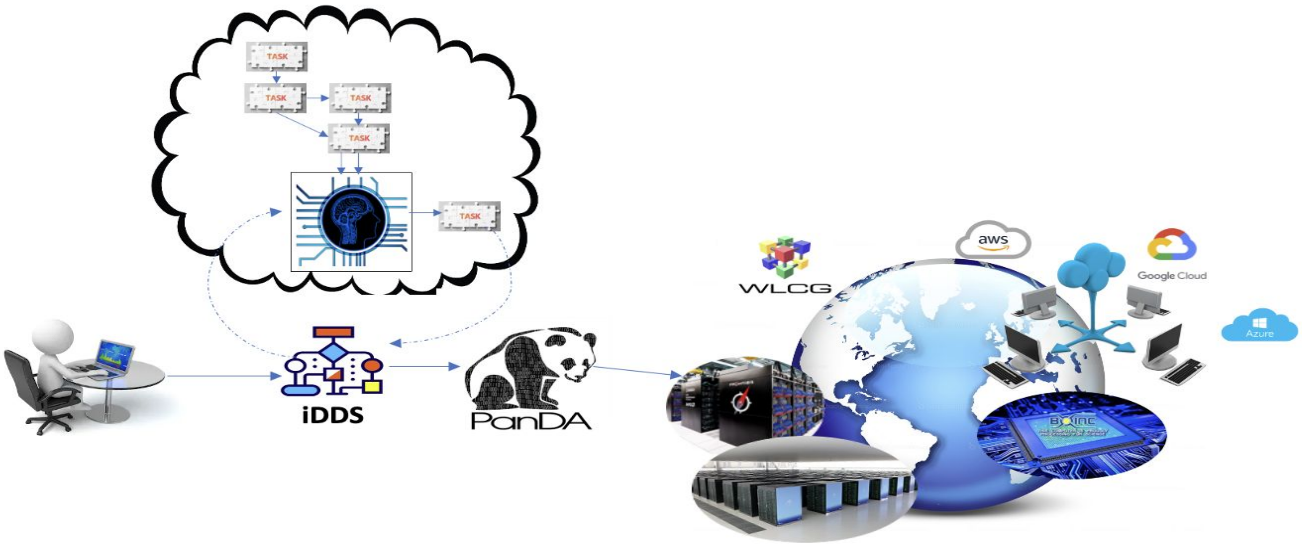
- **Distributed Users**
  - Users from different universities/labs can run jobs on PanDA through a http service
  - X509 or OIDC for authorization
  - LHC ATLAS 170 sites and several thousand users
- **Distributed Heterogeneous computing resources**
  - Diverse locations
  - Different software (slurm,condor,pbs)
  - Site differences will increase user complexity
- **PanDA (Production and Distributed Analysis system): Distributed Workload Management**
  - General interface for users, one authentication for all sites
  - Integrate different resource providers(Grid, Cloud, k8s, HPC and so on), hide the diversities from users, large scale
- **iDDS (intelligent Data Delivery Service): Workflow Management Orchestration**
  - DAG (Directed Acyclic Graph), complex workflow
  - Async result delivery
- **Has been in production in LHC ATLAS, Rubin Observatory and many other experiments.**

Distributed Workflow Management with PanDA/iDDS

- **PanDA as an engine for large scale workload management**
  - PanDA is powerful to schedule jobs to distributed heterogeneous resources
  - Large scale
  - Transparent to users for different computing resources
  - Smart workload routing
  - [CHEP2023 Talk: T. Maeno, et al. Utilizing Distributed Heterogeneous Computing with PanDA in ATLAS]()

- **iDDS orchestrates the workflow for automation**
  - Directed Acyclic Graph (DAG) management.
  - Condition workflow and Loop workflow management
  - Collect results from previous tasks
  - Analyze the results with user predefined jobs
  - Generate new tasks/jobs based on the analyses

  [CHEP2023 Talk: W.Christian, et al. Distributed Machine Learning with PanDA and iDDS in ATLAS]()

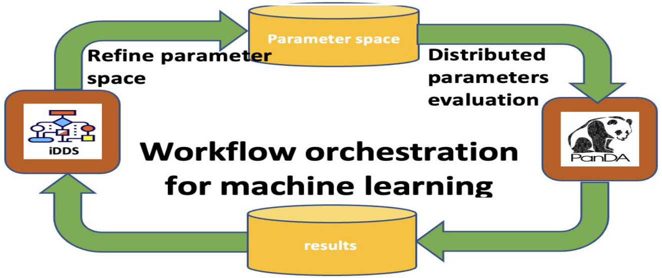


Brookhaven National Laboratory

Distributed Workflow Management Use Cases

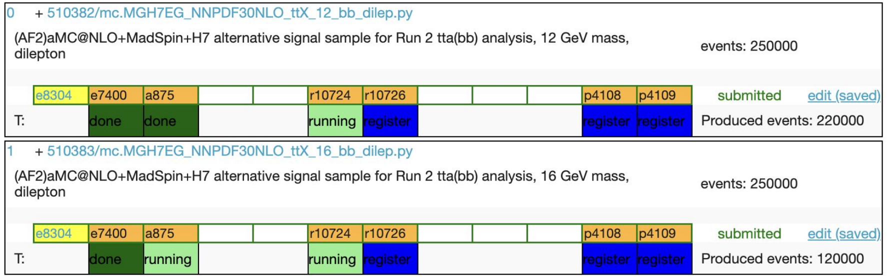
- **Started workflow integration in PanDA and iDDS for a long time, different use cases in production**
  - Fine-grained Data Carousel for **LHC ATLAS**
  - DAG management for **Rubin Observatory** to sequence data processing
  - Distributed HyperParameter Optimization (HPO)
  - Monte Carlo Toy based Confidence Limits
  - Active Learning assisted technique to boost the parameter search in New Physics search space
  - AI-assisted Detector Design for **EIC** (currently working)
- **Has involved processing a lot of data and different physics analysis**
- **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

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

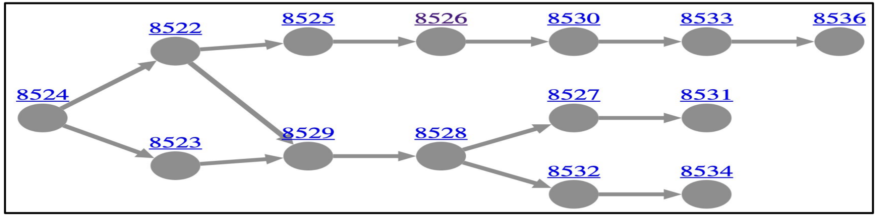


Different Types of Workflow Management

- **Workflow Management**
  - Coordinate and orchestrate tasks and data
  - Streamline operations into a workflow, to improve automation and efficiency
- **Data flow based workflow**
  - Output of current task -> input of next task
  - Easy to generalize and manage
- **Logics based workflow**
  - No data dependencies
  - Different parameters or different functions are triggered based on the previous return value
    - Eg: Hyperparameter Optimization
  - Different use cases can be very different
  - Supported workflow description tools:
    - Common Workflow Language (CWL), snakemake, …
  - Easy to make mistakes between user requirements and system behaviors when a complicated logic is defined



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



Examples of A DAG workflow

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

- **A new Function-as-a-Task Workflow Management framework is developed**
- **Python function as a task/job**
  - More granular for scientific workloads
  - Complicated logics can be defined with python language
  - Easy for users with python to define workflows
- **Workflow**
  - Manages tasks (functions) in groups
  - Python source codes preparation for distributed processing
  - General environments for all tasks (functions)
- **AsyncResult**
  - Employ messaging service to publish/receive results between function executor and submitter

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

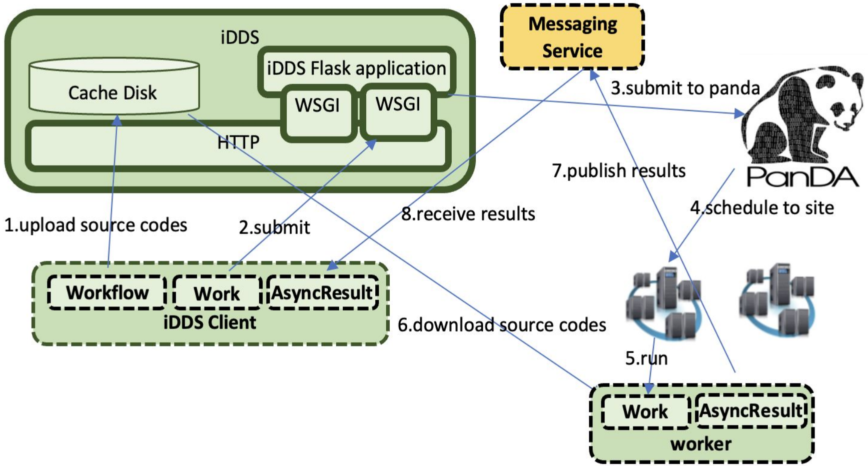
With python decorator @work to convert a function to a PanDA task

```
@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

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

Function-as-a-Task Workflow Management Advantages

- Source codes are managed transparently, no additional steps
- Support different ways to run user functions at distributed resources
  - With/without container
  - With base container + source codes caching, users don't need to build the container for a code update, the workflow will automatically update the source codes in the cache
  - For some experiments, different base containers are already provided and deployed on cvmfs. Users don't need to build personal containers
- Make use of the current PanDA structure and related middlewares, no additional requirements for sites
- Distributed resources, possible to large scale
- AsyncResults based on messaging service improves the efficiency

Example: Hyperparameter Optimization

- **Apply Function-as-a-Task for hyperparameter optimization**
- **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
  - List of parameters is provided to generate multiple jobs in a task
  - Singularity container is used as the base container

```python
@work(map_results=True)
def optimize_work(opt_params, retMethod=None, hist=True, saveModel=False, input_weight=None, **kwargs):
    from optimize import evaluate_bdt, load_data

    data, label = load_data()
    train, val = data
    y_train_cat, y_val_cat = label
    input_x = [train, val]
    input_y = [y_train_cat, y_val_cat]

    ret = evaluate_bdt(input_x=input_x, input_y=input_y, opt_params=opt_params, retMethod=retMethod, hist=hist,
                       saveModel=saveModel, input_weight=input_weight, **kwargs)
    return ret
```

```python
    bayesopt, util = get_bayesian_optimizer_and_util(optFunc, opt_params)

    n_iterations, n_points_per_iteration = 10, 20
    for i in range(n_iterations):
        print("Iteration %s" % i)
        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, opt_method=opt_method, hist=True, saveModel=False, input_weight=None,
                                retMethod=opt_method, group_kwargs=group_kwargs)
        print("points: %s" % str(points))

        for u_id in points:
            points[u_id]['ret'] = results.get_result(name=None, args=points[u_id]['kwargs'])
            print('ret :%s, kwargs: %s' % (points[u_id]['ret'], points[u_id]['kwargs']))
            bayesopt.register(points[u_id]['kwargs'], points[u_id]['ret'])

    print(bayesopt.res)
    p = bayesopt.max
    print("Best params: %s" % p)
```

```python
init_env = 'singularity exec /afs/cern.ch/user/w/wguan/workdisk/iDDS/test/eic/idds_ml_al9.simg '
wf = Workflow(func=optimize_workflow, service='idds', init_env=init_env)
```

With python decorator to transparently convert function to distributed tasks and collect the results transparently.
[sources]()

Example: Hyperparameter Optimization Test Examples

- **Test workflows with different iterations**
  - Every iteration is mapped to one panda task
  - In every iteration, multiple hyperparameters are generated. As a result, multiple jobs are generated in a task

Workflow: Group tasks together

request id	username	workflow status	graph	workflow name	created on (UTC)	total tasks	tasks	transform type	total files	released files	unreleased files	finished files
6128	Wen Guan	Finished	plot	optimize_iworkflow.optimize_workflow_2024_03_06_13_18_47	2024-03-06 13:18:47	11	Finished(10)	N/A	0	0	0	100%
6127	Wen Guan	Finished	plot	optimize_iworkflow.optimize_workflow_2024_03_06_13_18_41	2024-03-06 13:18:45	11	Finished(10)	N/A	0	0	0	100%
6126	Wen Guan	Finished	plot	optimize_iworkflow.optimize_workflow_2024_03_06_08_55_01	2024-03-06 08:55:04	6	Finished(5)	N/A	0	0	0	100%
6125	Wen Guan	Finished	plot	optimize_iworkflow.optimize_workflow_2024_03_06_08_54_32	2024-03-06 08:54:36	3	Finished(2)	N/A	0	0	0	100%

tasks, sorted by jeditaskid-desc

ID Parent	Task name TaskType/ProcessingType Campaign Group User Errors Logged status	Task status Nfiles	Input files Nlost Nfinish % Nfail %
168807	optimize_iworkflow.optimize_iworkflow.optimize_work_2024_03_06_11_12_39_6125_47808 iDDS Wen Guan Errors	done 20	20 100%
168804	optimize_iworkflow.optimize_iworkflow.optimize_work_2024_03_06_08_57_10_6125_47805 iDDS Wen Guan Errors	done 20	20 100%
168801	optimize_iworkflow.optimize_workflow_2024_03_06_08_54_32_6125 iDDS Wen Guan Errors	done 1	1 100%

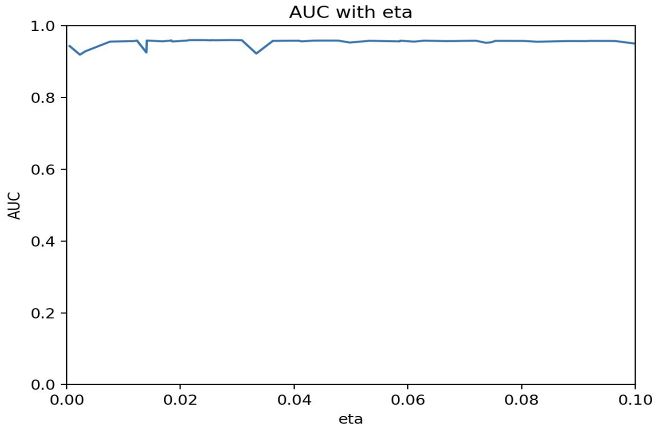
Iterations: this workflow has 10 iterations

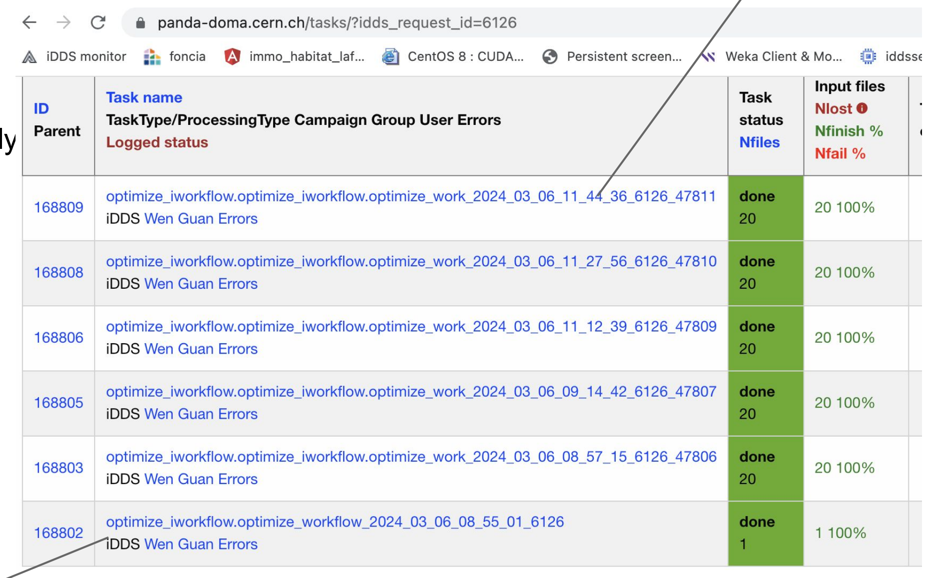
Jobs per iteration: this iteration has 20 jobs

Example: Hyperparameter Optimization Test Results

The work function

- **Function-as-a-Task**
  - In the example, the workflow is also converted to a remote task. However, the workflow can also run locally
  - Not many events, improvements based on ‘eta’ (learning rate) is small




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

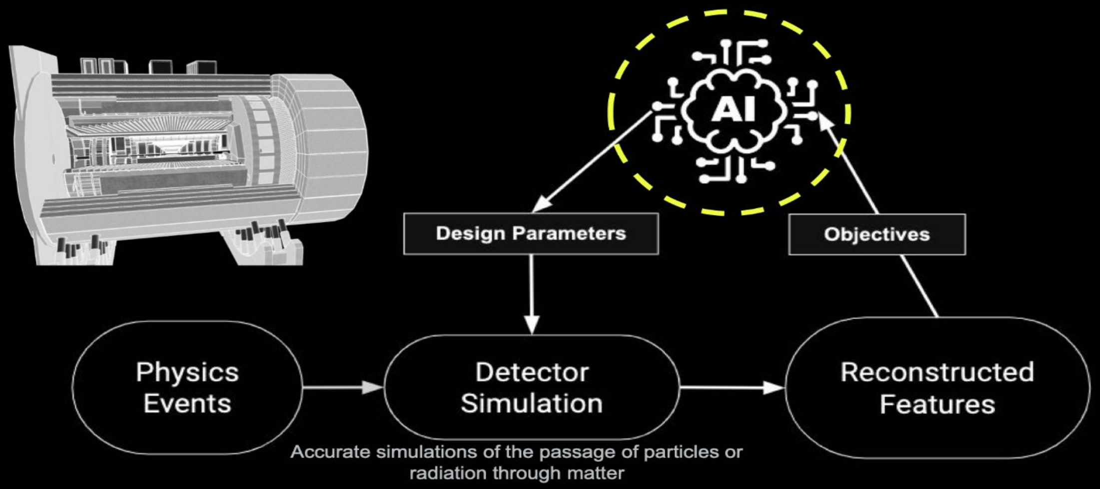
[One example workflow: (1) workflow function; (2) 5 iterations and 20 parallel jobs per iteration]

```
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}}
```

Next Step: Apply Function-as-a-Task workflow management for AI-assisted Detector Design for EIC (AID2E)

AI-Assisted Detector Design

The AI-assisted design embraces all the main steps of the sim/reco/analysis pipeline…



- Benefits from rapid turnaround time from simulations to analysis of high-level reconstructed observables
- The EIC SW stack offers multiple features that facilitate AI-assisted design (e.g., modularity of simulation, reconstruction, analysis, easy access to design parameters, automated checks, etc.)
- Leverages heterogeneous computing

[Cristiano Fanelli]

Provide a framework for an holistic optimization of the sub-detector system
A complex problem with (i) multiple design parameters, driven by (ii) multiple objectives (e.g., detector response, physics-driven, costs) subject to (iii) constraints

Those at EIC can be the first large-scale experiments ever realized with the assistance of AI

Next step: What to do in AID2E

- **Objectives**
  - Employ PanDA/iDDS to manage AI-assisted Detector Design parameter optimization tasks to distributed resources
  - Large scale distributed machine learning
  - Fine-grained automatization of multi-step iterative workflows
- **AI-assisted parameter optimization**
  - Many Parameters
  - Multiple Objectives
    - Multiple Objective Optimization
    - Multiple Objective Bayesian Optimization (MOBO)
- **Function-as-a-Task workflow structure**
  - It will simplify the integration between PanDA/iDDS and AID2E
  - PanDA will be able to provide a platform for large scale distributed computing
  - iDDS will convert AID2E functions to tasks, with AsyncResult to achieve good efficiency

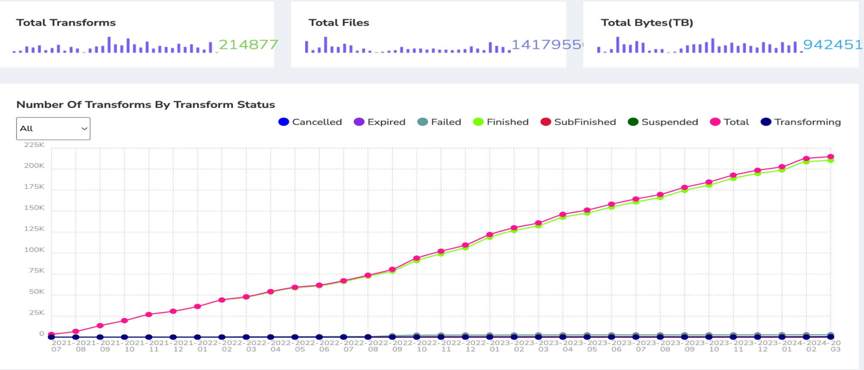
Conclusion

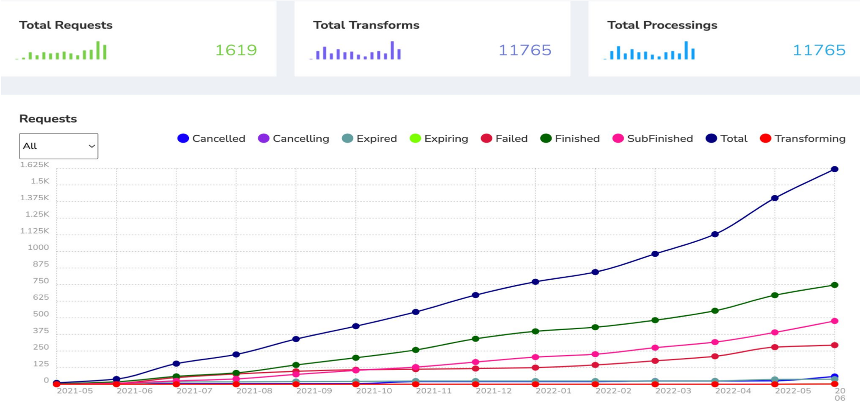
- PanDA/iDDS has supported complex workflow management for a long time
  - Different use cases are supported in production in different experiments
  - Complex workflow logics require an easy way to define the workflows
- A python Function-as-a-Task workflow structure is developed
  - With python functions to define the workflow steps
  - With python decorators to convert functions to distributed tasks
- It can easily convert a Bayesian hyperparameter optimization program to a distributed workflow
  - Examples has achieved good automation and efficiency
- Next step we will apply it to AI-assisted Detector Design for EIC, to achieve Multiple Objectives Bayesian Optimization
- In the future we will improve the structure to support more use cases and platforms

Backups

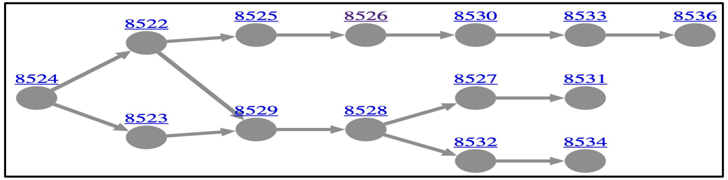
Workflow Management with PanDA and iDDS

- Fine-grained **Data Carousel** for LHC ATLAS enables processing in proper granularities and grouping to efficiently use disk storage
  - In production since 2020
  - From 2021, has processed 942 PB data (old processing information has been archived)
- DAG management for Rubin Observatory sequences data processing based on dependencies since 2020
  - Largely stable since Oct 2021
  - **DP0.2 (Phase 2 of Data Preview 0) campaign successfully** from the beginning of 2022 to June 2022.
    - From Jan-Jun 2022. Workflows: 351, tasks: 2732, jobs: 16483753
  - **HSC (Hyper-Suprime Cam) processing ongoing.**
    - Started from Jun 2022



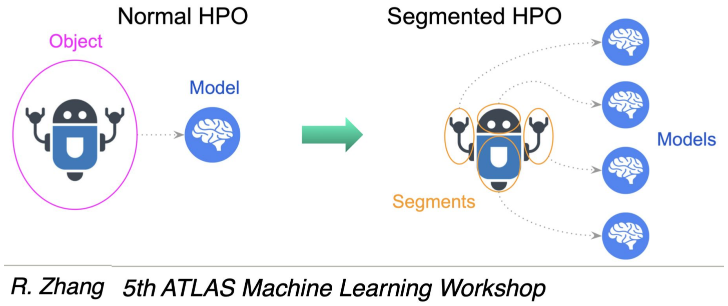


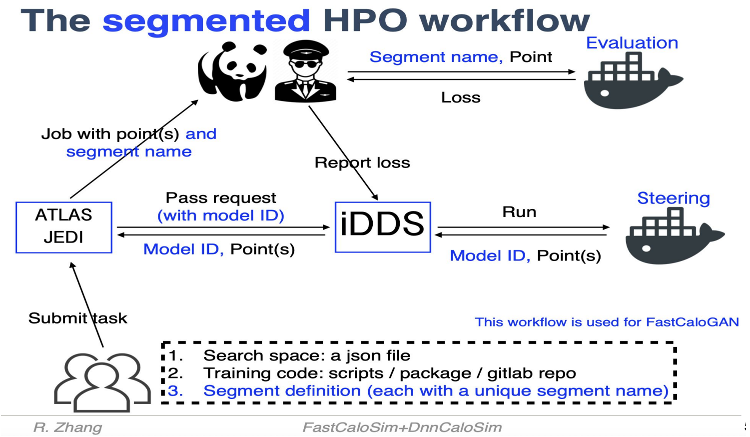
Since late 2020 in Rubin Observatory, iDDS-PanDA within the LSST framework has processed more than 11000 tasks.



Distributed HyperParameter Optimization (HPO)

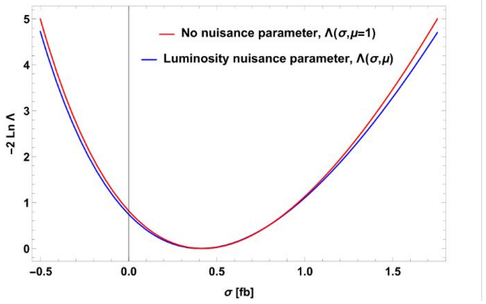
- Provide a full-automated platform for HPO on top of distributed heterogeneous computing resources
  - Hyperparameters are generated centrally in iDDS
  - PanDA schedules ML training jobs to distributed heterogeneous GPUs to evaluate the performance of the hyperparameter
  - iDDS orchestrates to collects 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]()

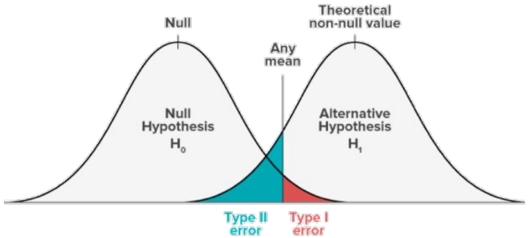


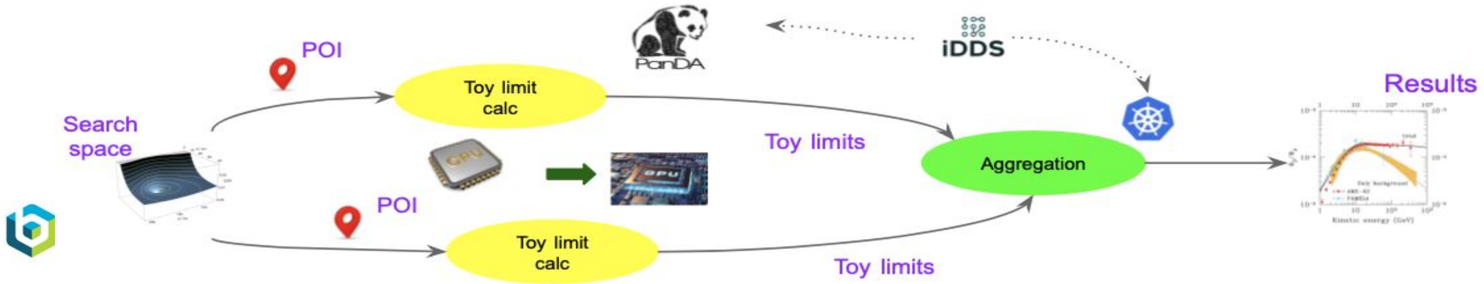


Monte Carlo Toy based Confidence Limits

- Confidence Limits in Analyses
  - Exclude some ranges of relevant phase space for future processing
  - Show that obtained results are meaningfully different from what could have obtained by chance
- An Monte Carlo (MC) Toy based confidence limits workflow requires multiple steps of grid scans, where the current step depends on the previous steps
- Automate the workflow of Toy limits calculation and aggregation
  - Point of Interest (POI) generation based on the search space and results aggregation to generate new POIs in iDDS
  - Distributed Toy limits calculation to distributed resources with PanDA
- Ref: [CHEP2023]

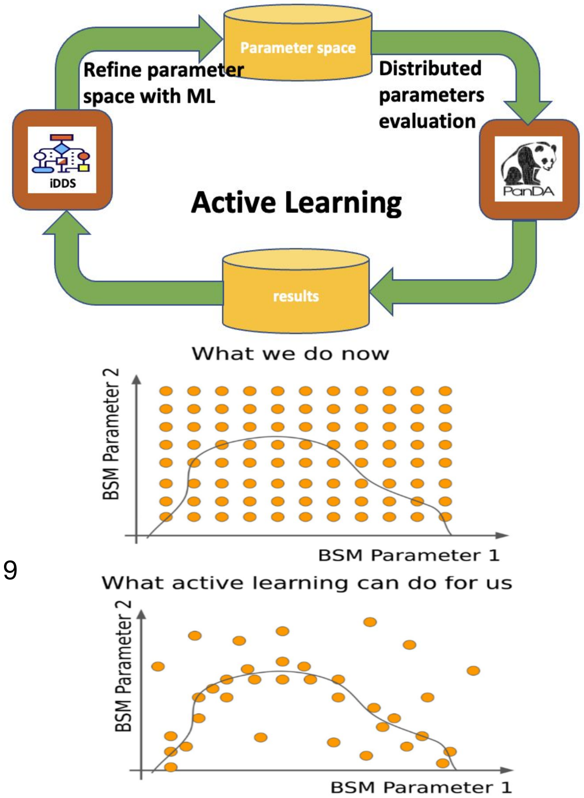






Active Learning

- ❖ 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
  - ➢ Redefine the parameter space for the next iteration based on the previous results with ML, more efficient than a single-step processing
  - ➢ Optimize the parameter space points for evaluation to maximise the information gain from each evaluation
  - ➢ Distributed computing resources for parameter evaluation
- ❖ Automate the multi-steps processing chain with PanDA and iDDS for ATLAS
  - ➢ Integrated REANA (Reusable Analyses) with PanDA for learning processing
  - ➢ iDDS orchestrates the workflow to trigger new tasks/jobs based on the previous results

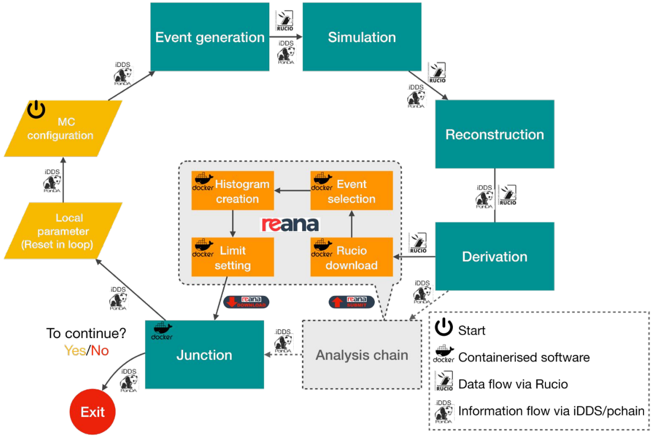


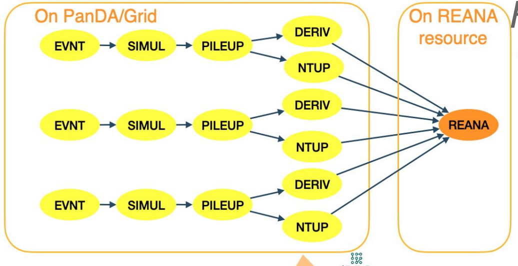
Active Learning via iterative regression on a limit surface

Active Learning for ATLAS

- ❖ Applied the Active Learning service in the $H \rightarrow ZZ_d \rightarrow 4\ell$ dark sector analysis
  - ➢ Avoids a complex interpolation scheme, costly in development and validation
  - ➢ 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

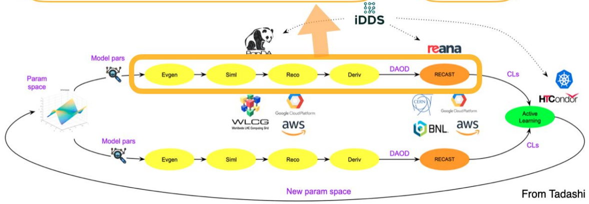
[CHEP2023 Talk: C. Waber, et al. An Active Learning application in a dark matter search with ATLAS PanDA and iDDS]





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

Thanks