PanDA/iDDS HPO workflows

Wen Guan, Lino Gerlach, Tadashi Maeno, Torre Wenaus
on behalf of the PanDA/iDDS team

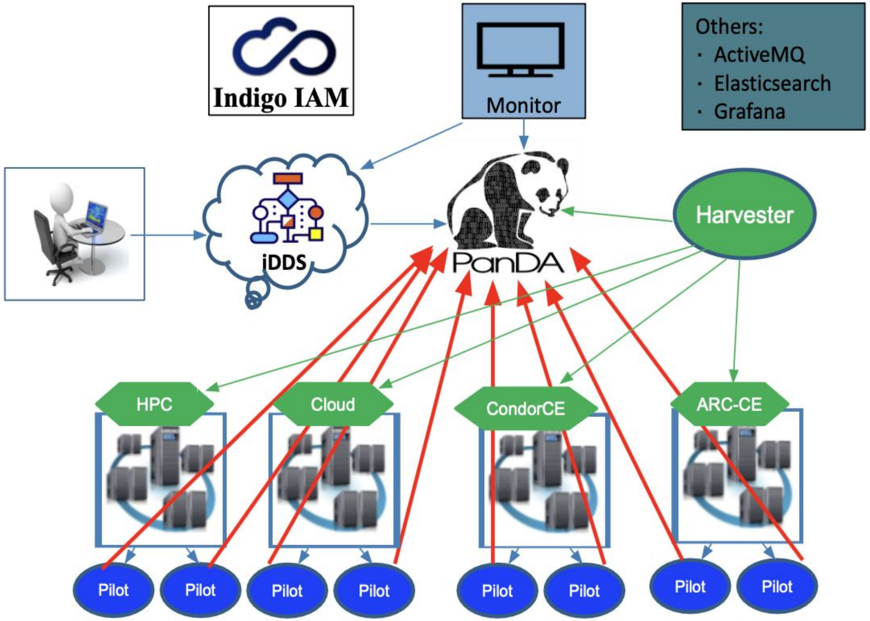
CCE-SML
April 18 2024

Brookhaven National Laboratory

U.S. DEPARTMENT OF ENERGY

@BrookhavenLab

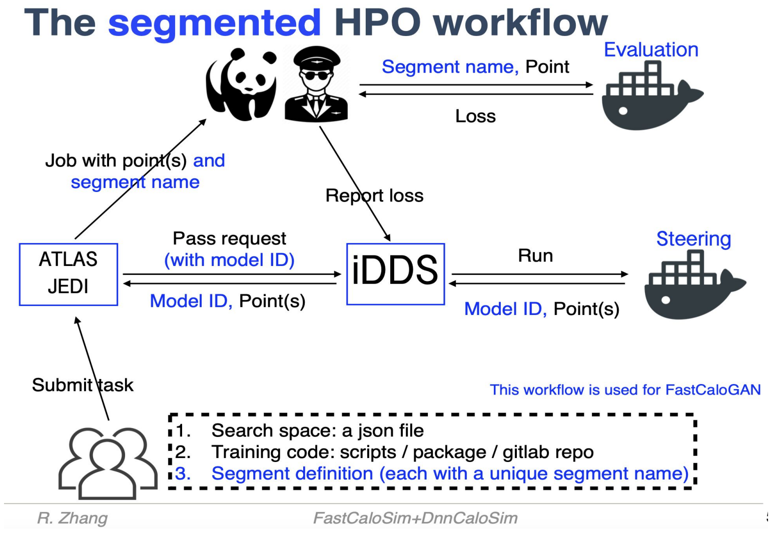
PanDA/iDDS



- **PanDA (Production and Distributed Analysis): Distributed workload Management**
  - Distributed users
  - Distributed computing resources
  - General interface for users, one authentication for all sites
  - Integrated different computing resources (Grid, Cloud, k8s, HPC and so on), hide diversities from users, large scale
- **iDDS (intelligent Data Delivery Service): Workflow management orchestration**
  - DAG (Directed Acyclic Graph)
  - Complex workflow
  - Asynchronous results

Distributed HyperParameter Optimization (HPO) in PanDA/iDDS

- **Distributed HyperParameter Optimization (HPO)**
  - Provide a full-automated platform for HPO on top of distributed heterogeneous computing resources
  - iDDS orchestrates hyperparameter generation and results collection; automation
  - PanDA evaluates hyperparameters remotely
  - Support for advanced search algorithms in addition to the traditional grid or random search algorithms
  - Integrate geographically distributed GPU resources to provide a single resource pool to end-users

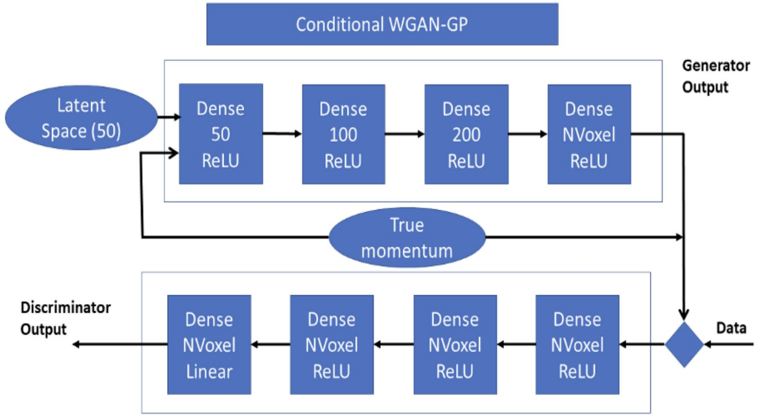


Containerization of the HPO workflow

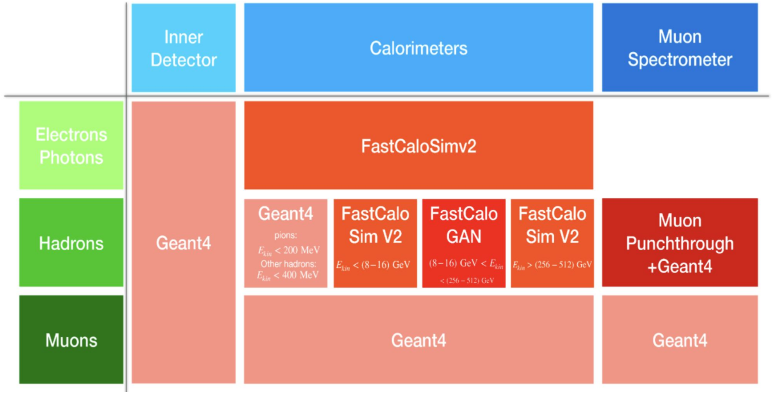
- **SteeringContainer**
  - Optimization executed at iDDS server
  - Generate new HP points with customised method
    - When to trigger; for example, 80% of points of the current iteration finish
    - Number of points to generate per iteration
    - When to finish
  - A wide range of HPO methods are supported
- **EvaluationContainer**
  - ML training at Grid/Cloud/HPC (CPU/GPU) sites
  - Submodule payload contains model definition, training scripts
- **User containers**
  - Both the SteeringContainer and the EvaluationContainer can be replaced with user containers.

HPO for FastCaloGAN

- **FastCaloGAN simulation**
  - One GAN for a particle type and an η range (in total 300 GANs)
  - Hyper parameter optimised for each GAN
    - Bigger networks (due to larger input dimensions)
    - High batchsize
  - 100 GPU-days to train 300 GANs
- **Applied HPO for FastCaloGAN**
  - Applied for ATLAS FastCaloGAN, part of the production ATLAS fast simulation AtlFast3
  - Ref: [FastCaloGAN](), [AML workshop](), [IML](), [ATLAS S&C week]()



GAN



[AtlFast3]()

A New iWorkflow Management (Function-as-a-Task) 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 iWorkflow 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
  - AsyncResult supports with messaging service (ActiveMQ)
  - Simplify the workflow usage for users

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

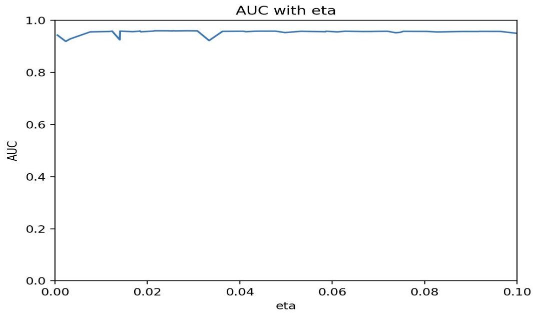
With python decorator **@work** to convert a function to a PanDA task

```
@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))
```

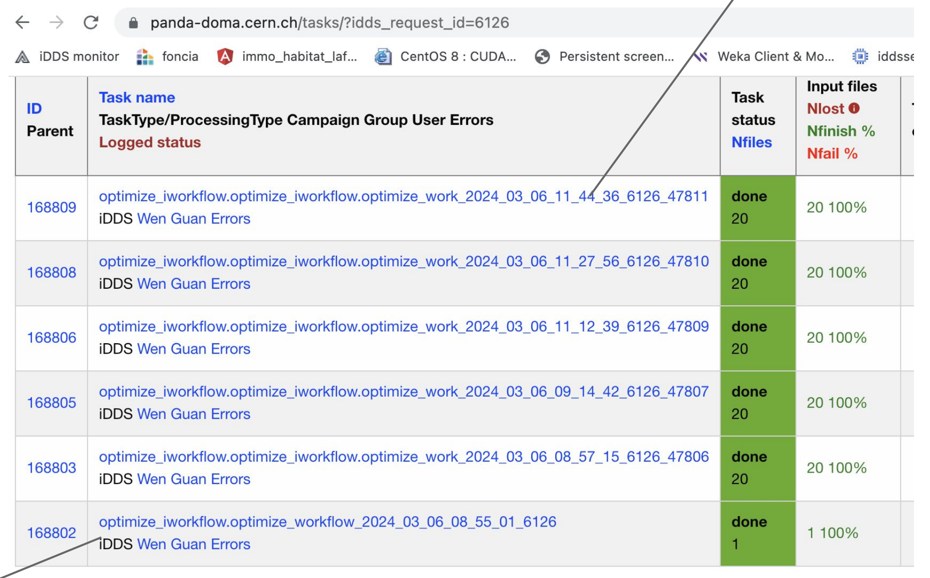
The Workflow calls the task like a local function, transparent to users

Example: new HPO with iWorkflow

- **Apply iWorkflow 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



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%

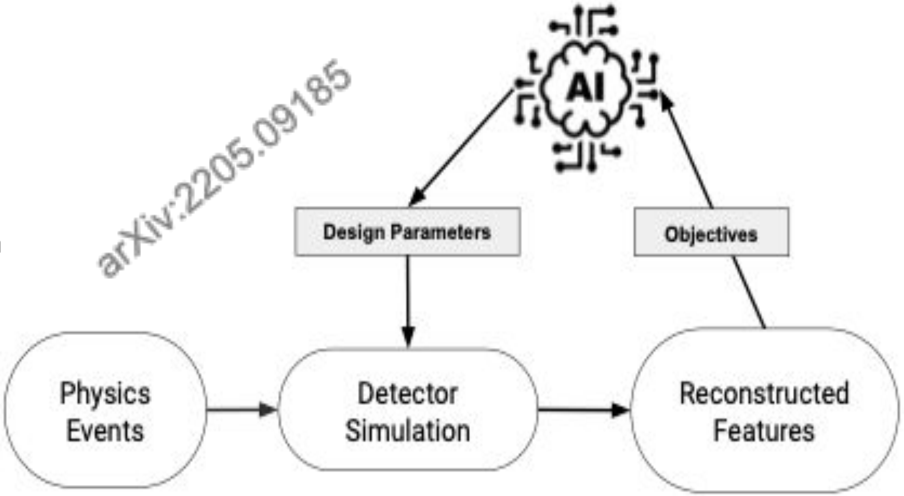
The workflow function

[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), difficult to convert it to workflow description language like CWL (Common Workflow Language) or shakemake
- **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



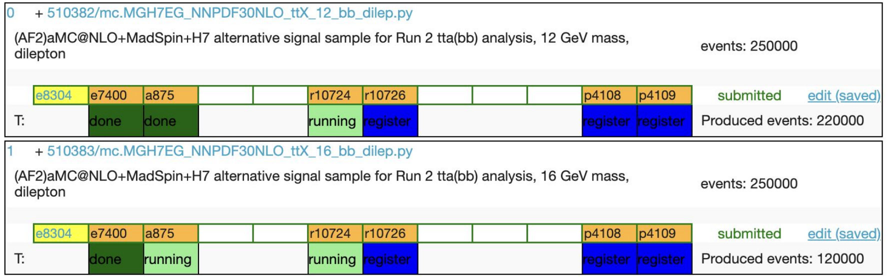
Conclusion and Next

- **PanDA/iDDS has supported complex workflow management for a long time**
  - Various use cases are supported in production
  - Multiple experiments (ATLAS, Rubin, EIC/AID2E, …)
  - Complex workflows are supported, using different workflow descriptions (Snakemake, CWL, iWorkflow)
- **In the future we will improve the structure to support more use cases and platforms**

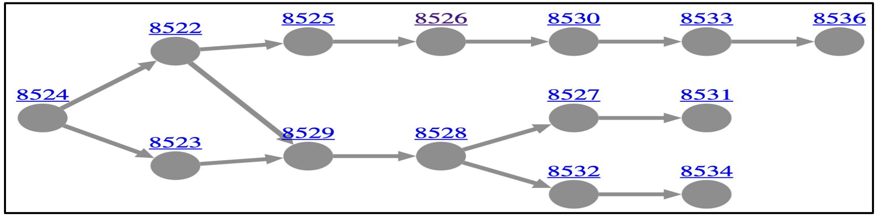
Backups

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



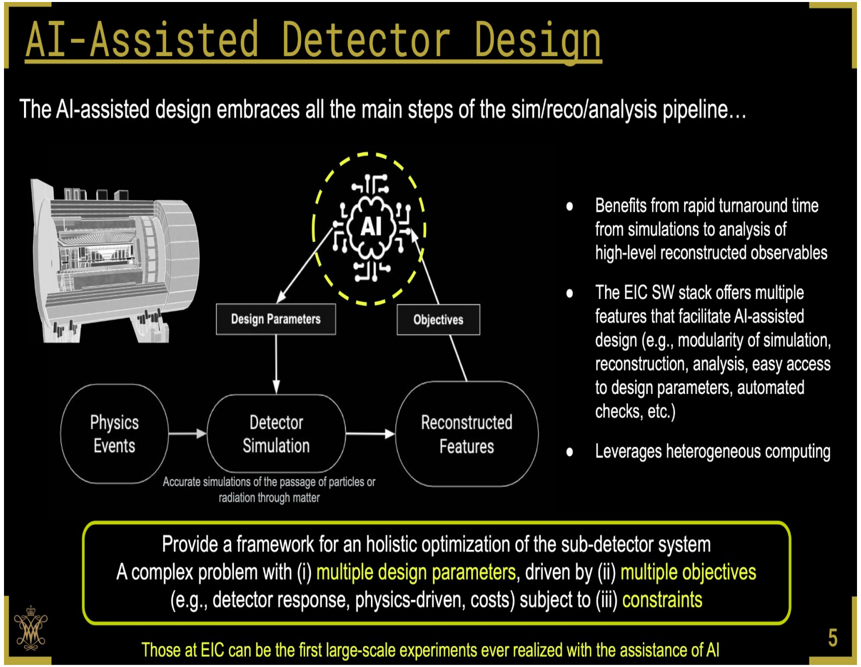
An example of A DAG workflow

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

AI-assisted Detector Design for EIC (AID2E)

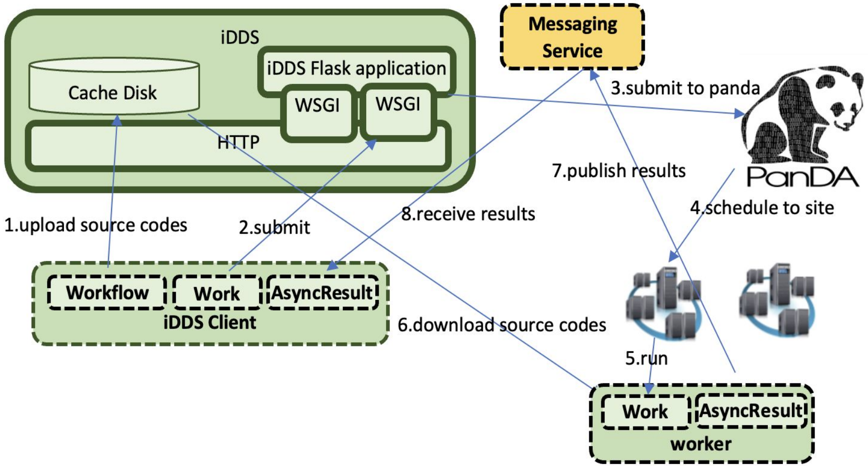
- **Objectives**
  - Employ PanDA/iDDS to manage AI-assisted Detector Design parameter optimization tasks on distributed resources
  - Large scale distributed machine learning
  - Fine-grained automation of multi-step iterative workflows
- **AI-assisted parameter optimization**
  - Many Parameters, multiple detector design objectives
    - Multiple Objective Bayesian Optimization (MOBO)
- **Challenges**
  - AID2E has similarities to existing supported workflows
    - HPO (MOBO)
    - Analysis and combined performance (iterative multiple stages with widely varying characteristics, from full Geant4 simulation to MOBO optimization)
  - AID2E MOBO using AX Adaptive Experiment Platform (pyTorch based)
  - Adapting existing AID2E software stack to PanDA/iDDS
  - To ease the adaptation to PanDA/iDDS, developing a python decorator mechanism to convert functions to iDDS tasks
  - **Function-as-a-Task workflow management in iDDS** (next slides)



Cristiano Fanelli

iWorkflow 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

iWorkflow 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

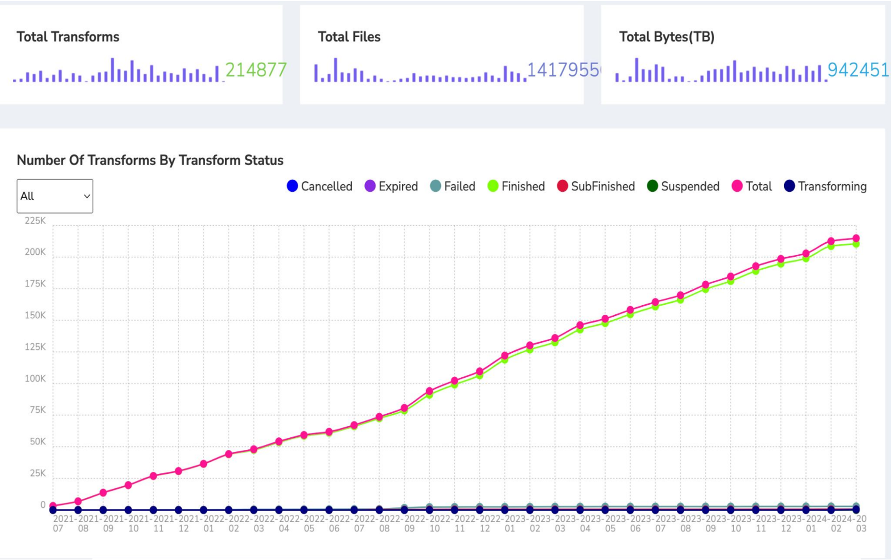
ID Parent	Task name TaskType/ProcessingType Campaign Group User Errors Logged status	Task status Nfiles	Input files Nlost Nfinish % Nfail %
168807	optimize_iworkflow.optimize_iworkflow.optimize_work_2024_03_06_11_12_39_6125_47808 iDDS Wen Guan Errors	done 20	20 100%
168804	optimize_iworkflow.optimize_iworkflow.optimize_work_2024_03_06_08_57_10_6125_47805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

Fine-grained Data Carousel for LHC ATLAS

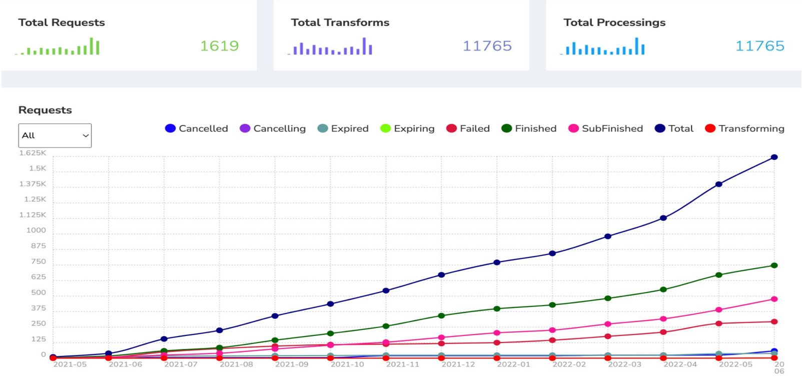
- Fine-grained **Data Carousel** for **LHC ATLAS** enables processing in proper granularities and grouping to efficiently use disk storage
  - iDDS employs messages to trigger PanDA processing data in proper granularity, instead of per dataset
  - In production since 2020
  - From 2021, has processed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



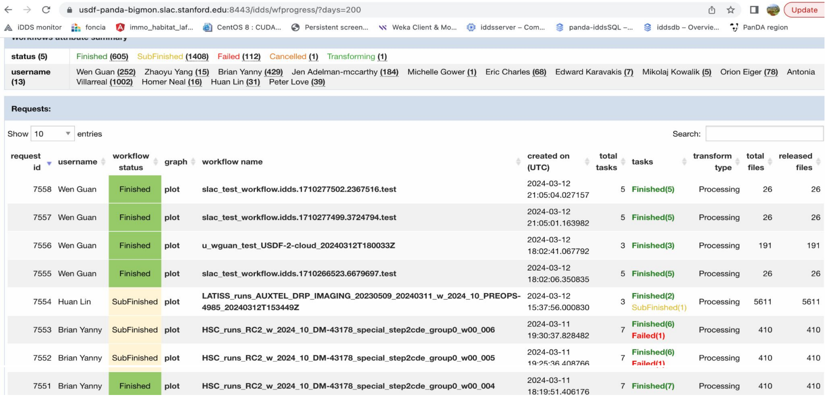
[Since late 2021, ATLAS Data Carousel has processed 942 PB data] (old monitor information are archived)

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.



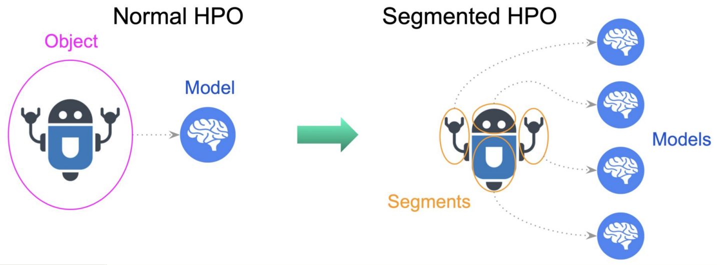
request id	username	workflow status	graph	workflow name	created on (UTC)	total tasks	tasks	transform type	total files	released files
7558	Wen Guan	Finished	plot	slac_test_workflow.idds.1710277502.2367516.test	2024-03-12 21:05:04.027157	5	Finished(5)	Processing	26	26
7557	Wen Guan	Finished	plot	slac_test_workflow.idds.1710277499.3724794.test	2024-03-12 21:05:01.163982	5	Finished(5)	Processing	26	26
7556	Wen Guan	Finished	plot	u_wguan_test_USDF-2-cloud_20240312T180033Z	2024-03-12 18:02:41.067792	3	Finished(3)	Processing	191	191
7555	Wen Guan	Finished	plot	slac_test_workflow.idds.1710266523.6679697.test	2024-03-12 18:02:06.350835	5	Finished(5)	Processing	26	26
7554	Huan Lin	SubFinished	plot	LATISS_runs_AUXTEL_DRP_IMAGING_20230509_20240311_w_2024_10_PREOPS-4985_20240312T153449Z	2024-03-12 15:37:56.000830	3	Finished(2) SubFinished(1)	Processing	5611	5611
7553	Brian Yanny	SubFinished	plot	HSC_runs_RC2_w_2024_10_DM-43178_special_step2cde_group0_w00_006	2024-03-11 19:30:37.828482	7	Finished(6) Failed(1)	Processing	410	410
7552	Brian Yanny	SubFinished	plot	HSC_runs_RC2_w_2024_10_DM-43178_special_step2cde_group0_w00_005	2024-03-11 19:25:36.408766	7	Finished(6) Failed(1)	Processing	410	410
7551	Brian Yanny	Finished	plot	HSC_runs_RC2_w_2024_10_DM-43178_special_step2cde_group0_w00_004	2024-03-11 18:19:51.406176	7	Finished(7)	Processing	410	410

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

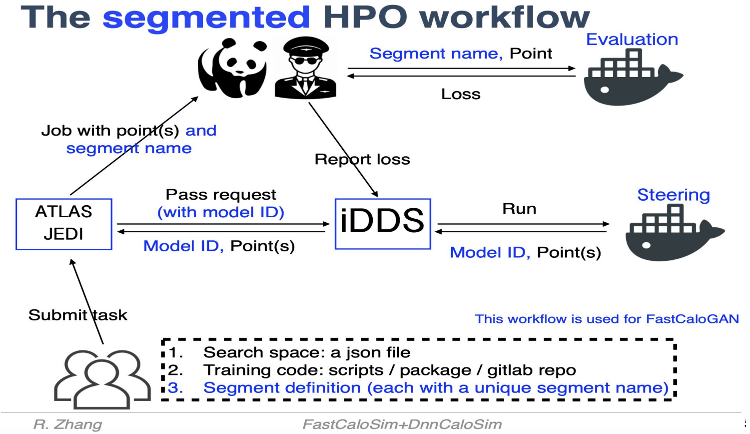
Distributed HyperParameter Optimization (HPO)

- 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]()

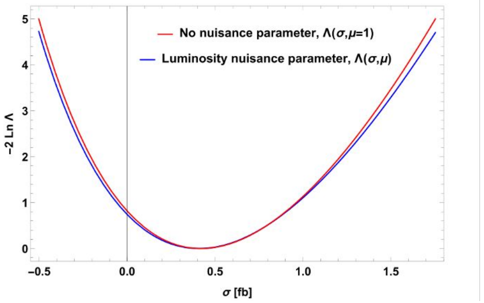


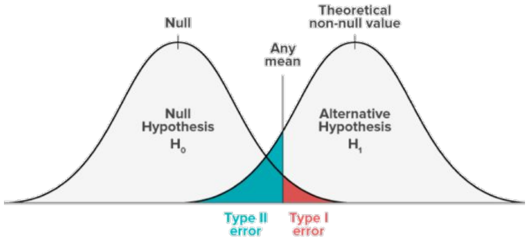
R. Zhang 5th ATLAS Machine Learning Workshop

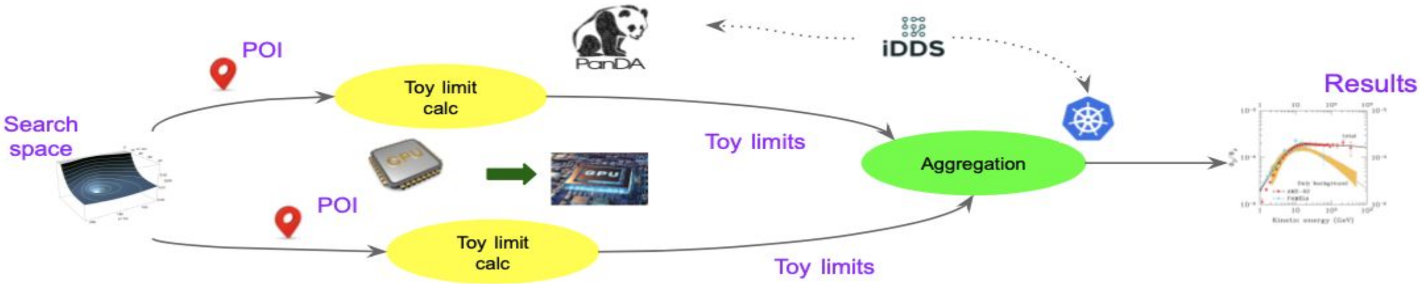


Monte Carlo Toy based Confidence Limits

- 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]



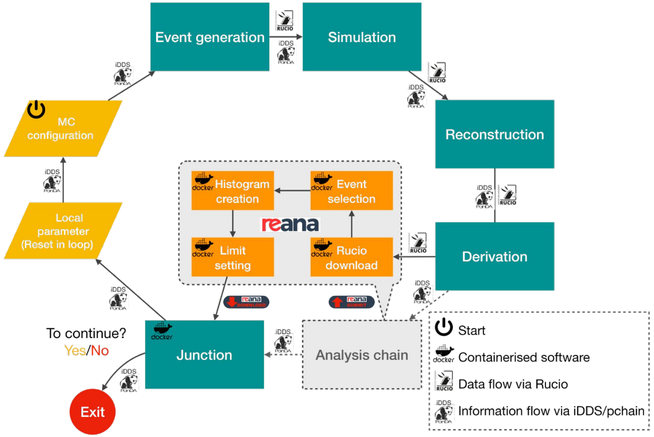


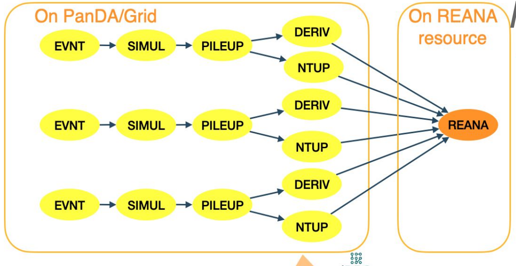


Active Learning for ATLAS

- 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

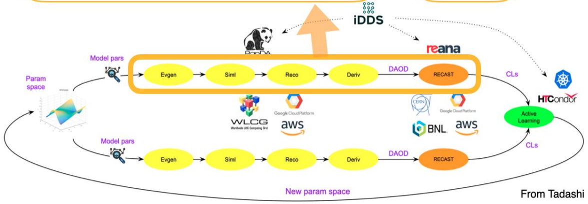
[CHEP2023 Talk: C. Waber, et al. An Active Learning application in a dark matter search with ATLAS PanDA and iDDS]





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