Hyperparameter Optimisation service at ATLAS

Rui Zhang, supported by US-ATLAS
University of Wisconsin-Madison

HL-LHC R&D topics

Mar 03, 2021

Contents

- Introduction to the project: hyperparameter optimisation service at ATLAS
- intelligent Data Delivery Service (iDDS)
- Hyperparameter optimisation using iDDS on the ATLAS Grid
  - The usual HPO workflow
  - Ingredients of the workflow
  - The segmented HPO workflow
- On heterogeneous resources
  - HPC/Summit and its challenges
  - Commercial Cloud for distributed training
- Visualisation support

Introduction: HPO service at ATLAS

- The goal of the project is to provide an HPO service to ATLAS users for ML
  - Minimal user code adaption
  - Support for advanced search algorithms in addition to the traditional grid or random search algorithms
  - Reuse ATLAS production and distributed system (PanDA) - no reinventing the wheel
  - Visualisation of results

- Single-function-call pattern for HPO
  - Computing resources are managed behind the scene
  - Not suitable since ATLAS has its own resource management

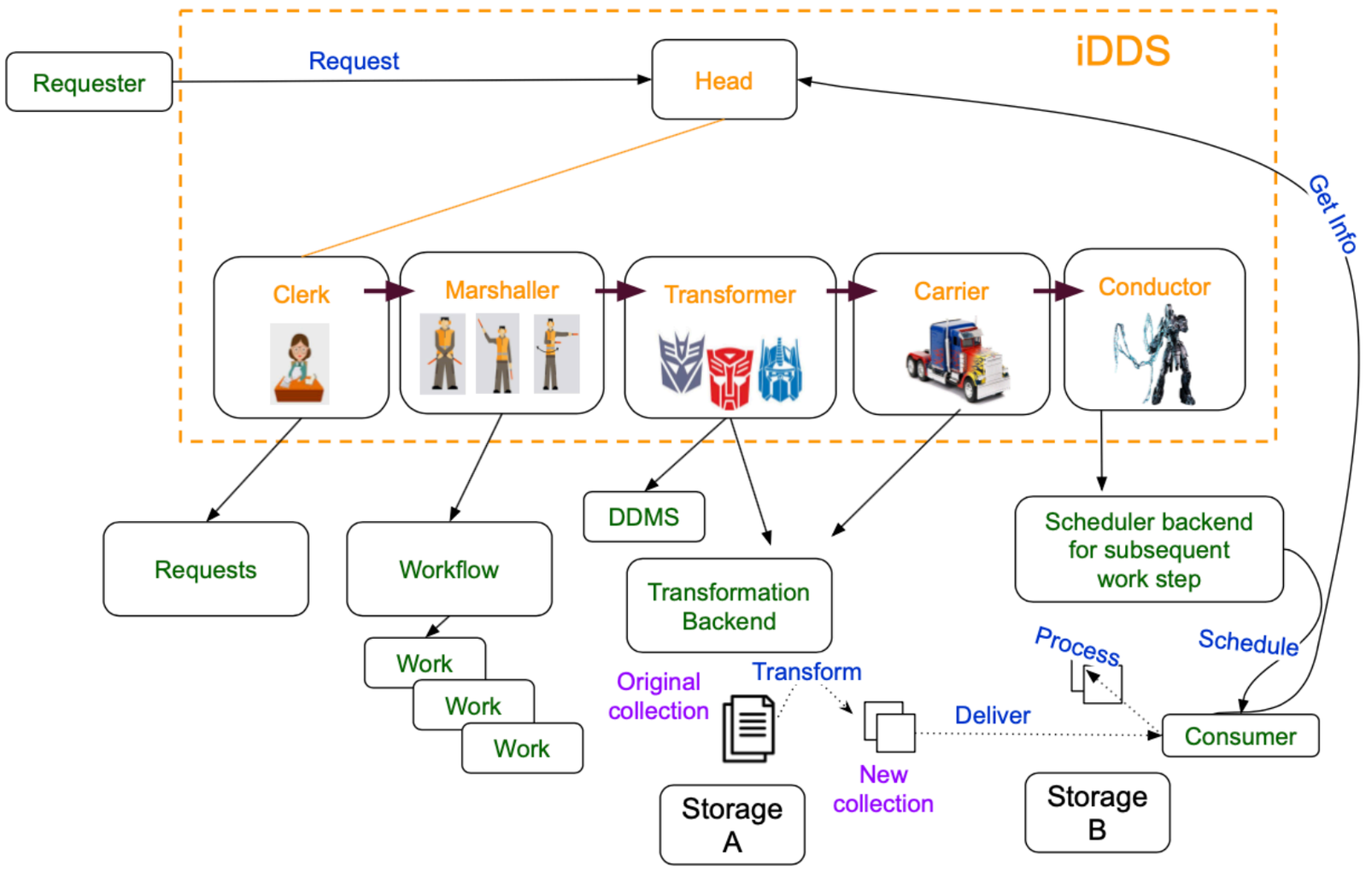
- **Ask-and-tell** pattern for HPO
  - Decoupled optimisation+sampling from training in space-time
  - Purely point searching, no resource management
  - We go in this way

"The ask-and-tell pattern"

```
while ~ opt.stop
  x = ask(opt)
  y = f(x)
  opt = tell(opt, x, y)
end
```

The intelligent Data Delivery Service (iDDS)

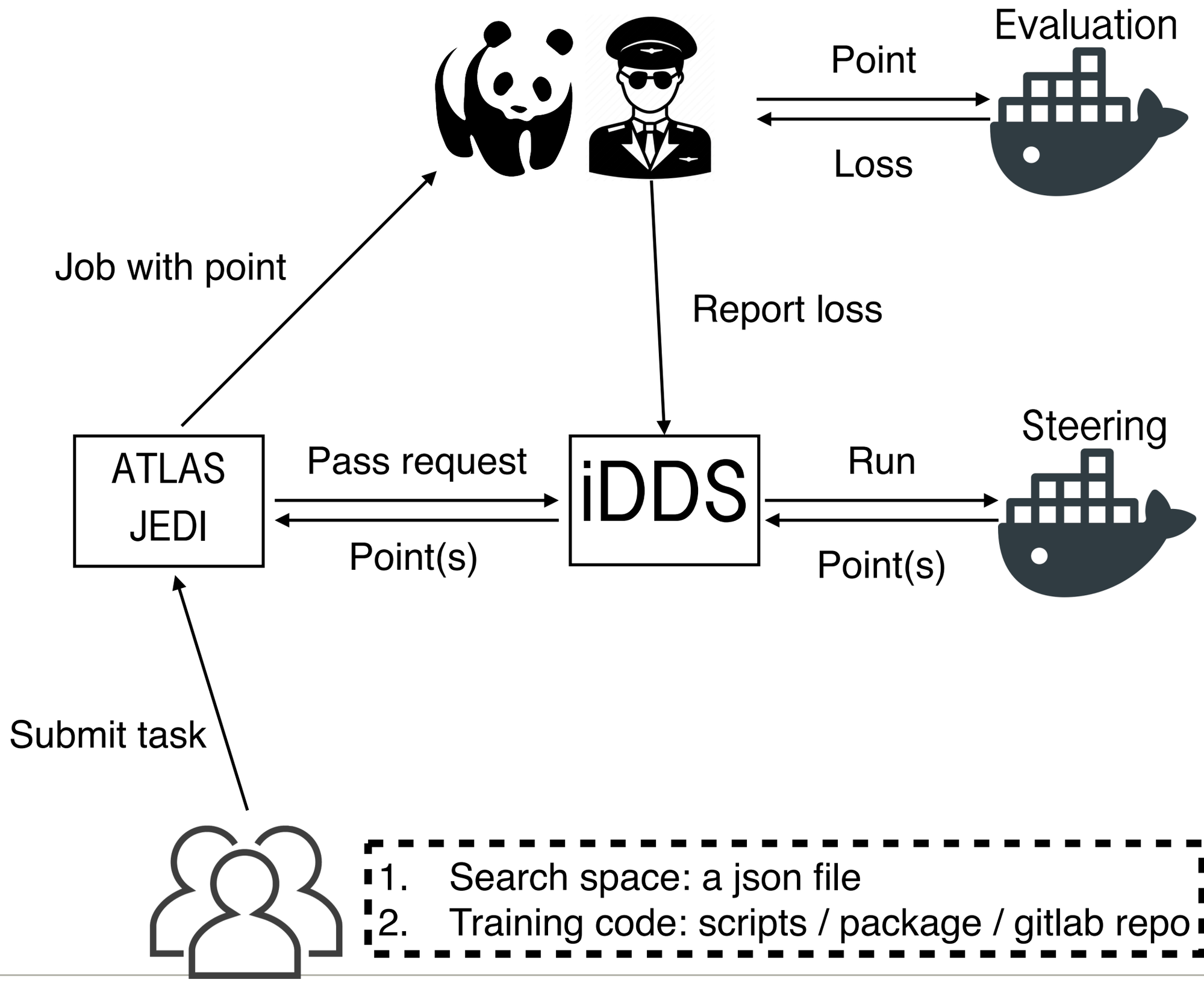
- iDDS (a joint project with IRIS-HEP) is designed to intelligently transform and deliver needed data to workflows in a fine-grained way.
  - Takeaway: jobs of successive tasks start as soon as possible, no need waiting for precedent tasks to finish, optionally making decisions in between



The intelligent Data Delivery Service (iDDS)

- Many applications share this paradigm, e.g.:
  - Data Carousel: job starts when its input is ready, no waiting for the full dataset to be transferred
  - A chain of tasks (DOMA): successive jobs start when enough inputs are produced by the precedent tasks
  - A chain of tasks (Active Learning): successive jobs are created and submitted by iDDS based on results of precedent tasks => extendable to a generic function-as-a-service type of workflow
- HPO is a series of tasks with decision-making in between —- a suitable use case

Overview of the HPO workflow



Ingredients of the workflow

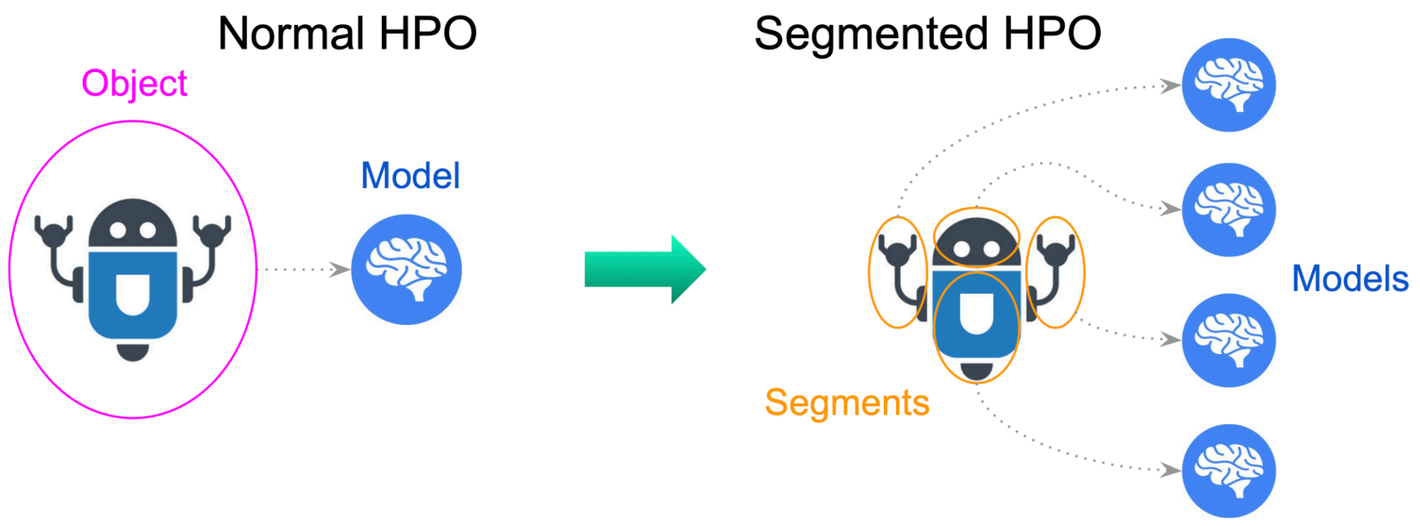
- Two containers to fulfil the loop:
  - `SteeringContainer` - optimisation at iDDS servers
    - Generate next HP points with customised method
    - A wide range of HPO methods are supported

  - `EvaluationContainer` - training at Grid (GPU) sites
    - Submodule `payload` contains model definition, training scripts (user specific)

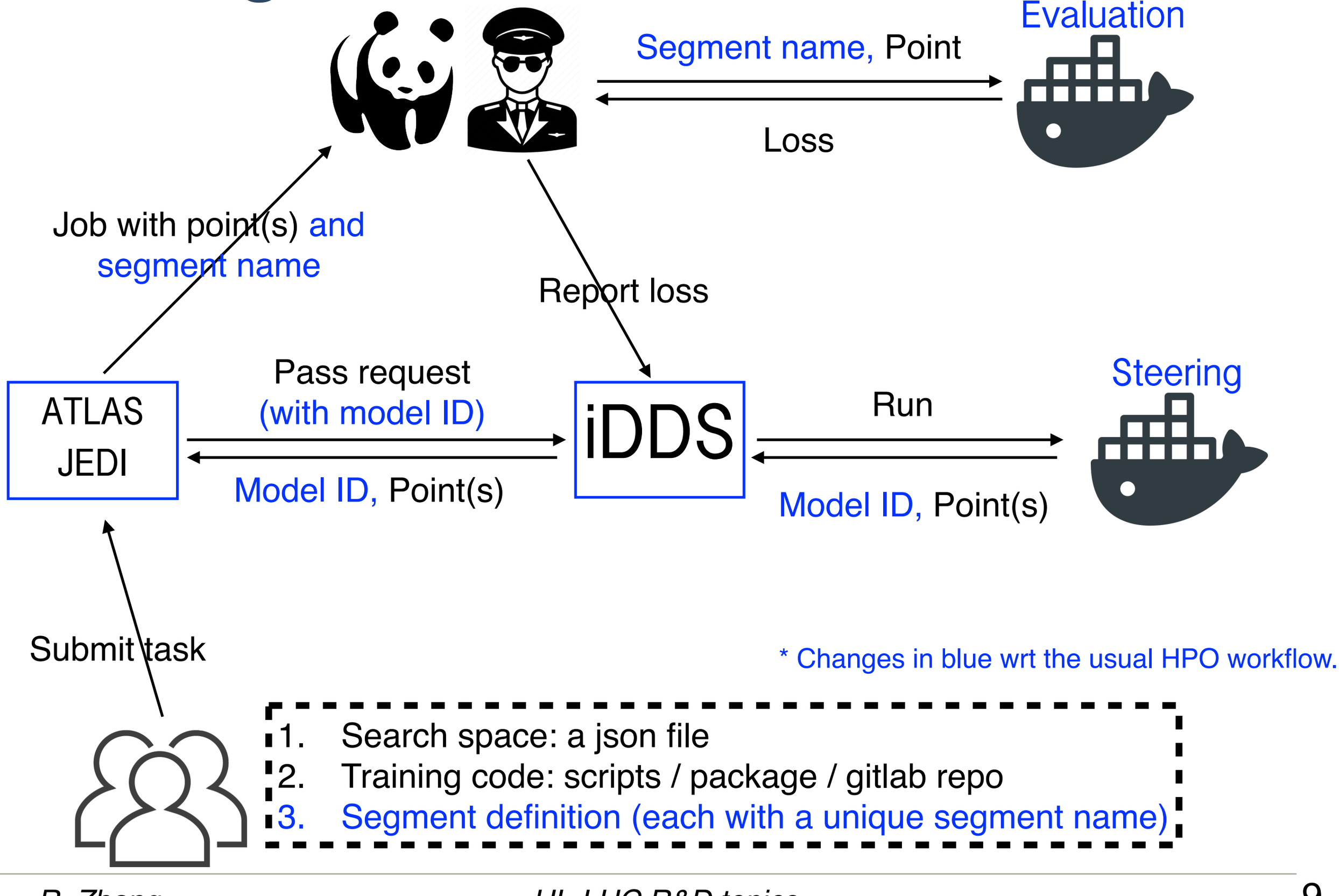
- Checkpointing:
  - Periodically upload checkpoints to Grid
  - Download the checkpoint when the same job is retrying
  - Resume training if checkpoint is found

Segmented HPO - Why do we need it

- Some machine learning payloads have similar architecture of models targeting different physics regions/objects
  - Essentially multiple models formed with different training datasets
  - Once the amount is large, bookkeeping is challenging
- A real ATLAS example: FastCaloGAN, a calorimeter image generation model
  - 300 GANs = 3 PIDs x 100 η slices
  - 300 individual tasks in the usual workflow
- Now this can be done with **Segmented HPO**



The segmented HPO workflow

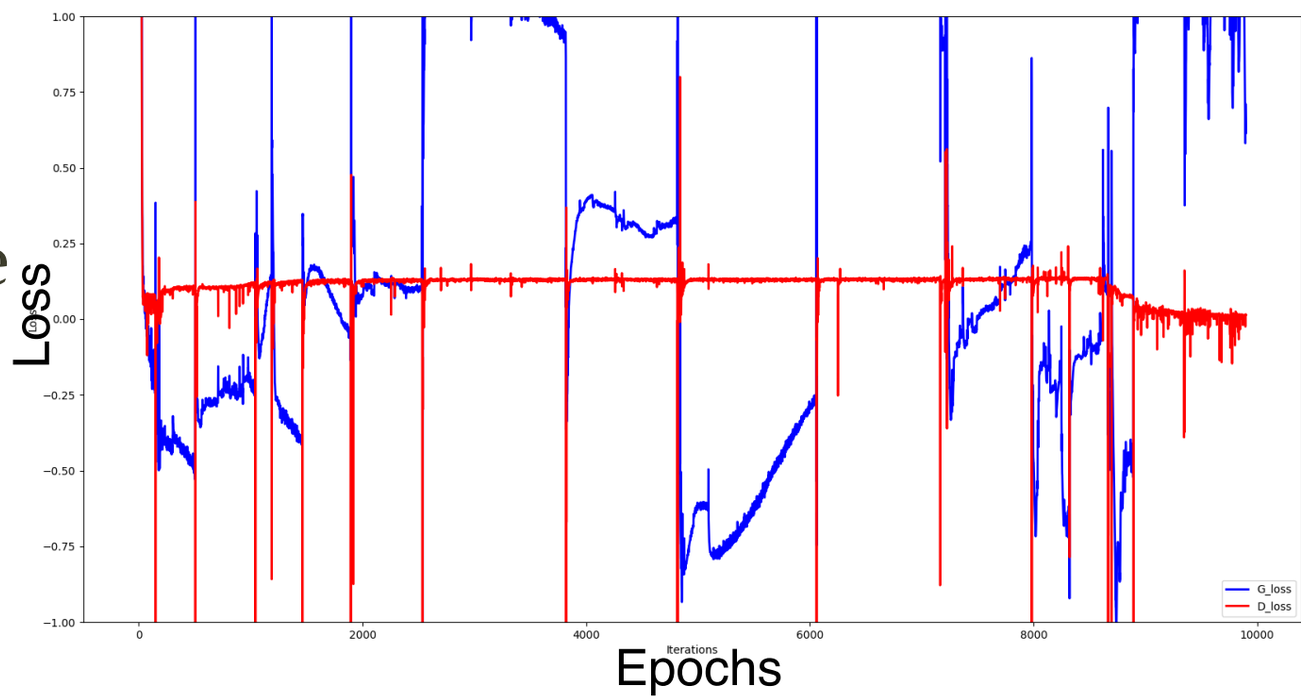


Features and test results

- Features:
  - Support both usual (not segmented) and segmented HPO tasks
    - An argument to fill when submitting
  - Support separating input for each segment
    - To reduce the load of the sites
- Tested with 15 GANs
  - 3 particle types × 5 η slices
- Plot on the right is from a **10K epochs** job from the BNL GPU site
  - The training ideally needs **1 million epochs** and a dedicated offline analysis is required

Input dataset (PID 22, 0 < $|\eta|$ < 0.05):
pid22_eta_0_5.v02.tar

pid22_eta_0_5.v02.tar	input	ready
binning.xml	input	ready
user.zhangr.seg_gpuBNL.bd6dda1e-8db4-483b-b4f9-3d04dee8b340.log.24393491.000010.log.tgz	log	ready
photons_0_5.24393491.metrics.000010.tgz_1	output	ready
photons_0_5.24393491.metrics.000010.tgz_0	output	ready

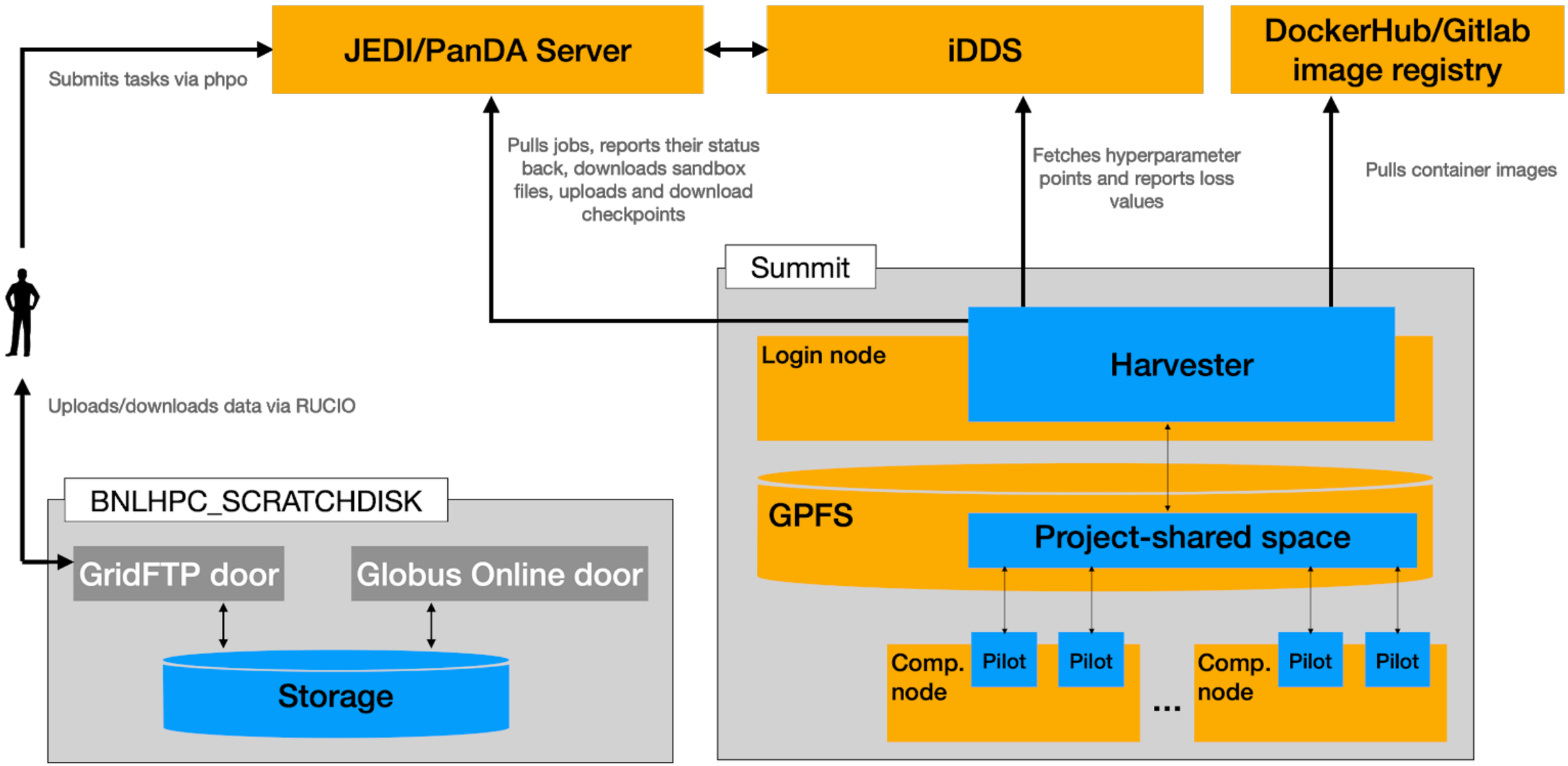


HPCs as GPU resources

- Summit as an example
  - 4608 computer nodes
    - 2 Processors x 22 cores / node
    - 6 V100 GPUs / node
  - Wonderful workstation for ML/HPO
- Challenges
  - Short wall time
  - Standard Grid services and workflows unavailable or suboptimal

Integration map on Summit

- Harvester is a key component in the HPC environment
  - Need to connect it with JEDI, iDDS and image repository



Challenges on Summit / HPC

- Various issues need to be addressed for HPCs, which are different among HPCs. For Summit, there are following striking factors:

  1) PowerPC architecture - a non-x86 architecture
     - Encounter different compilers for PowerPC 😐
     - Docker image is architecture-dependent; not straightforward to provide for a production service ☹️

  2) Short walltime - 2h for each job*
     - Checkpointing is almost always needed 😐

  3) No network inbound/outbound connectivity of worker nodes
     - Requires all demands to be downloaded in advance
     - This hurts a lot for many payloads that are designed without this limitation ☹️

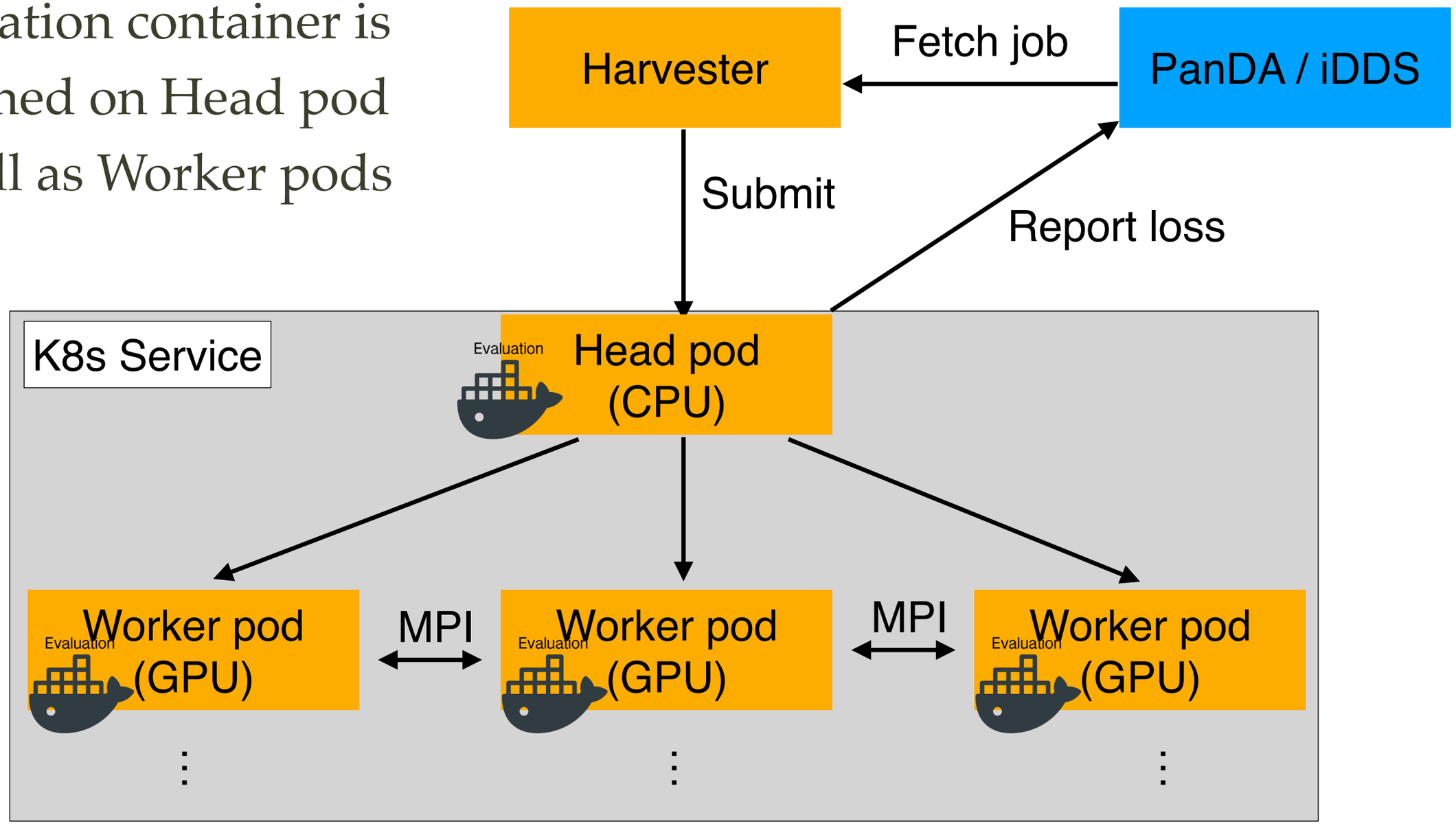
***1-45 nodes: 2h, 46-91 nodes: 6h, 92-921 nodes: 12h, 922-4608 nodes: 24h**

Distributed training on commercial cloud

- Commercial cloud is one of the best places for distributed training
  - GPUs on the grid are mostly for single-GPU training
  - HPCs have many development and operational challenges
- So far in the R&D phase
  - Big investment for pledged GPU would be a bit risky since there are not many real use cases now
  - Should be prepared since distributed training is quite popular outside of HEP
- Horovod is currently being experimenting
  - A useable Evaluation container is created, to be tested with multi-GPU resources.
  - Open to support other distributed solutions, e.g. DASK, Ray

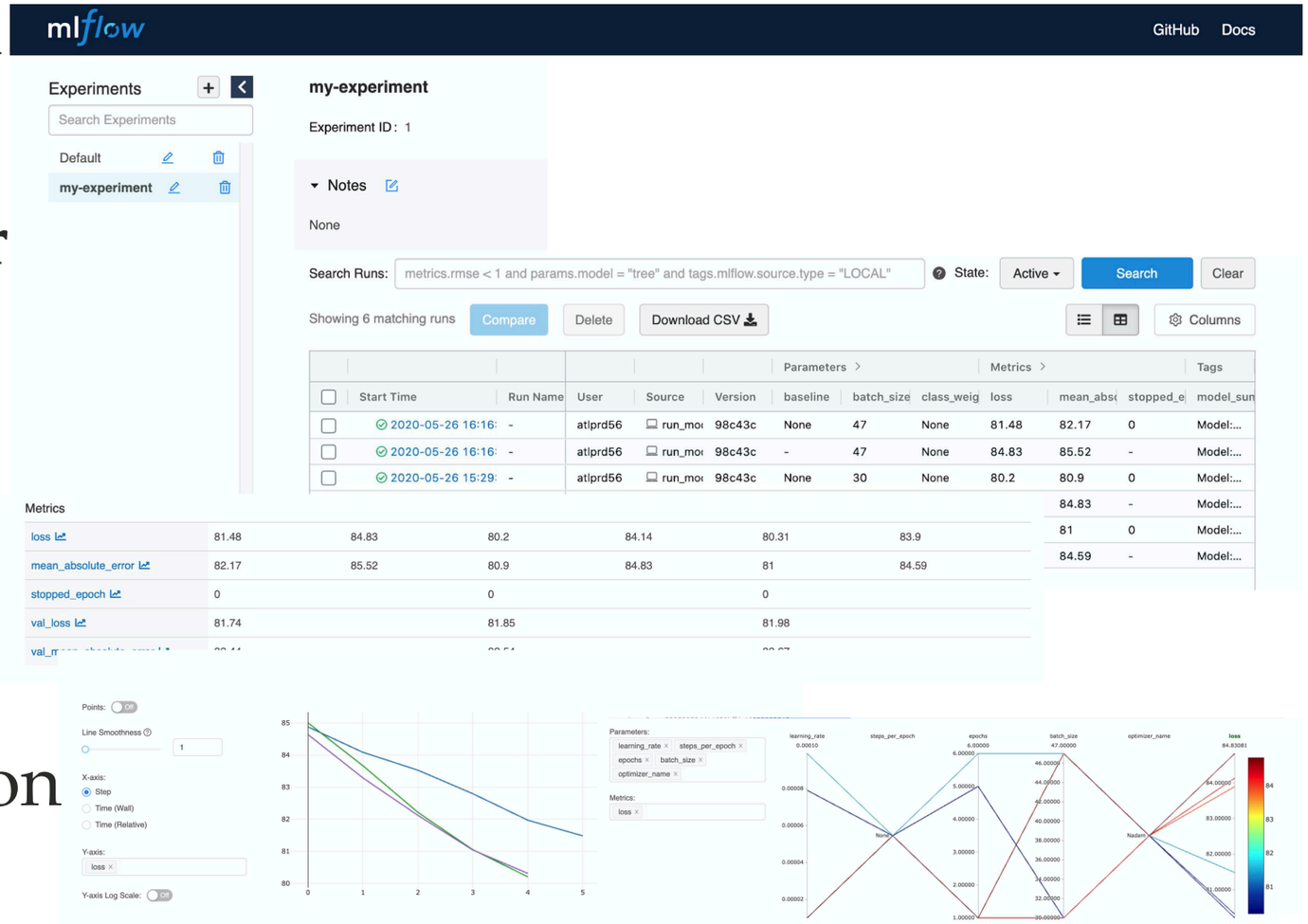
(Simplified) Integration map on Amazon K8s

- Horovodrun (MPI launcher) runs on the Head pod
- Number of Worker pods is scalable by K8s
- Evaluation container is launched on Head pod as well as Worker pods



Visualisation support

- A visualisation tool `MLflow` is turned on in `EvaluationContainer`
  - Useful for offline visualisation as it is part of the output
- An α-version of the tool also integrated into PanDA Monitoring system
  - Fetch output from Evaluation container (training job) and spin-up an MLFlow container to display results
  - Extendable to other nice visualisation tools (Neptune, WandB, Tensorboard, etc.) if outputs match the visualisation backend or if additional conversion step is implemented



Summary (1)

- Goal of the project is to provide an ML/HPO service at ATLAS
  - To fulfil the demands that are expected from ML-topical physics, in particular in HL-LHC when larger dataset comes
  - An ATLAS directed work, but could be used by others as well
- The HPO workflow is developed and tested
  - "Ask" and "tell" are separated such that they are incorporated into ATLAS PanDA system
  - Docker container is used to preserve rapidly changing ML libraries
  - Tested with several use cases in ATLAS
    - Documented what need to change for a user with a mature training code

Summary (2)

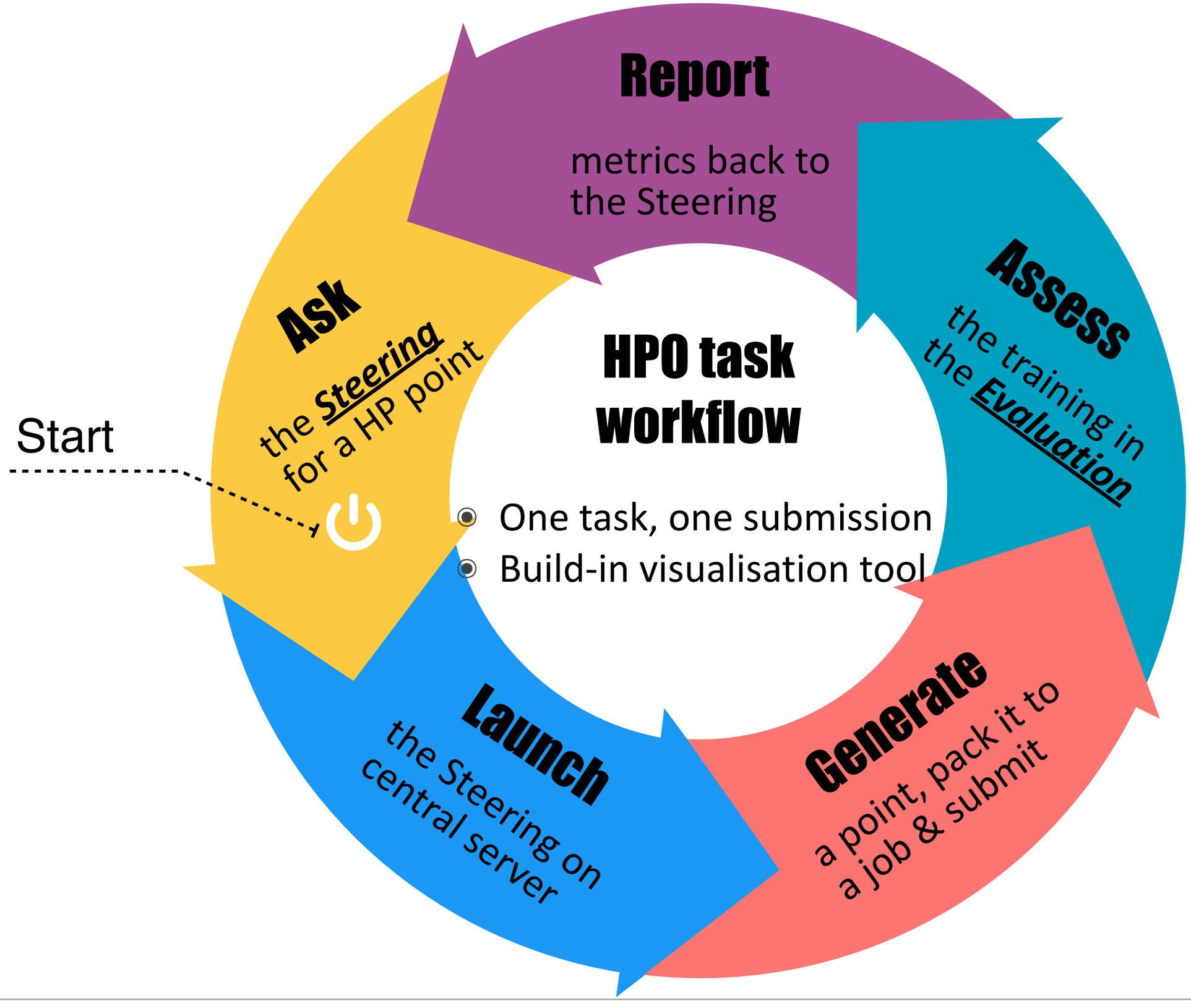
- Special scenarios
  - Distributed training
    - Some challenges were faced on Summit
    - Commercial Cloud (AWS) is currently being R&D
  - Segmented HPO
    - Implemented and tested with succeed
      - Convenient framework to train hundreds of models in one go; quite useful if ATLAS migrates to ML-based simulation
- Visualisation being supported centrally via the PanDA Monitoring system
  - Many new ideas can be implemented

Backup

Documentations

- Walk-through the Calo Image-based DNN example
  - SteeringContainer: https://gitlab.cern.ch/zhangruihpc/SteeringContainer
  - EvaluationContainer: https://gitlab.cern.ch/zhangruihpc/EvaluationContainer
- How to submit HPO task
  - https://twiki.cern.ch/twiki/bin/view/PanDA/PandaHPO
- iDDS Readme about the interfaces of ask-and-tell pattern
  - https://idds.readthedocs.io/en/latest/usecases/hyperparemeter_optimization.html

The HPO workflow

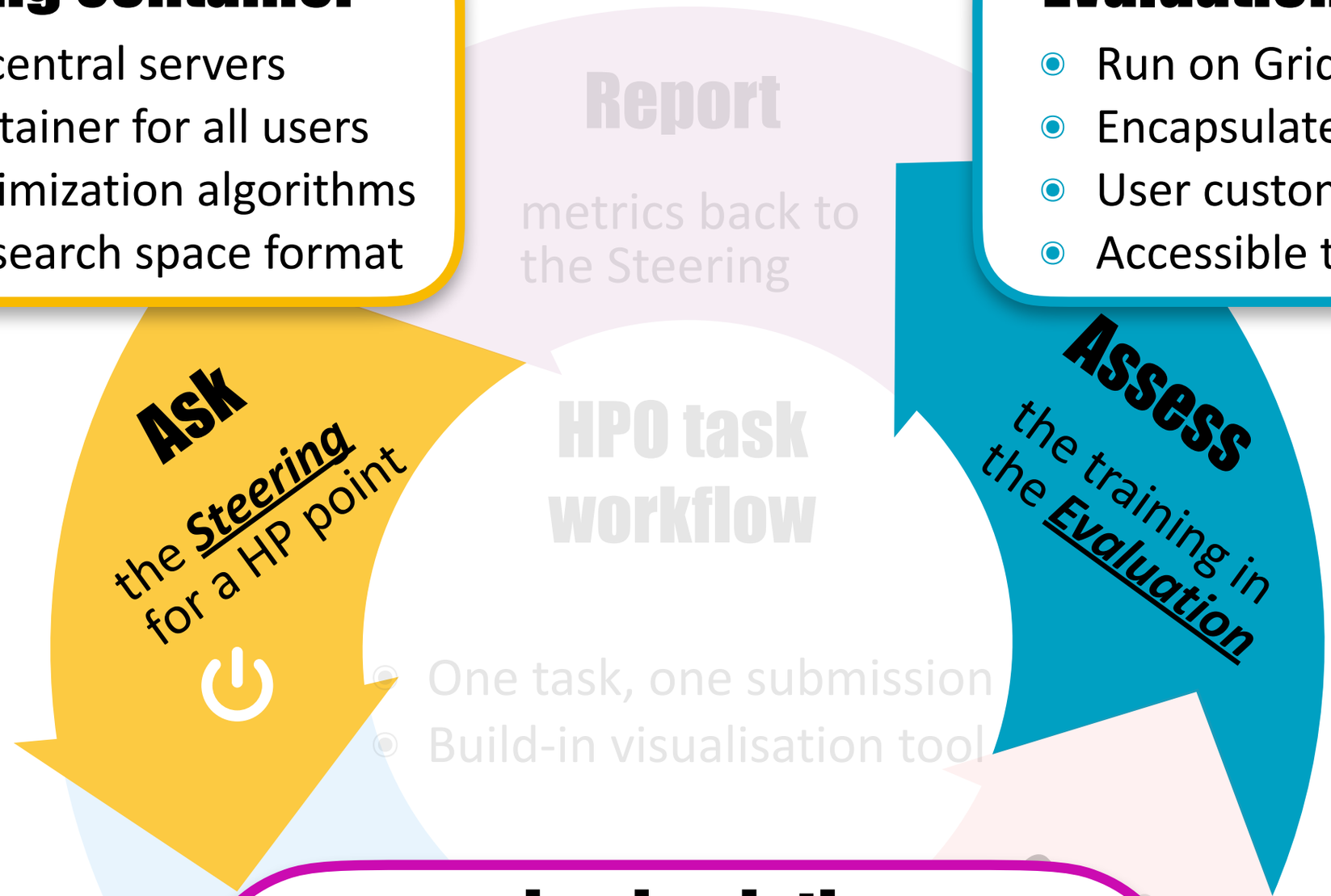


Steering container

- Run on central servers
- One container for all users
- Rich optimization algorithms
- Unified search space format

Evaluation container

- Run on Grid sites
- Encapsulate training job
- User customizable
- Accessible to data on Grid



checkpointing

- In case of short walltime on sites
- Periodically upload checkpoints to Grid
- Download the checkpoint when retrying
- Resume training if checkpoint is found

More references

- Summit/HPC distributed training
- New workflows for HPC
- Summit/HPC challenges
- New workflows
- 4th Inter-experiment Machine Learning Workshop
- Future analysis facility
- AI for Big data