Integration of funcX/parsl and ATLAS PanDA

Tadashi Maeno (BNL)

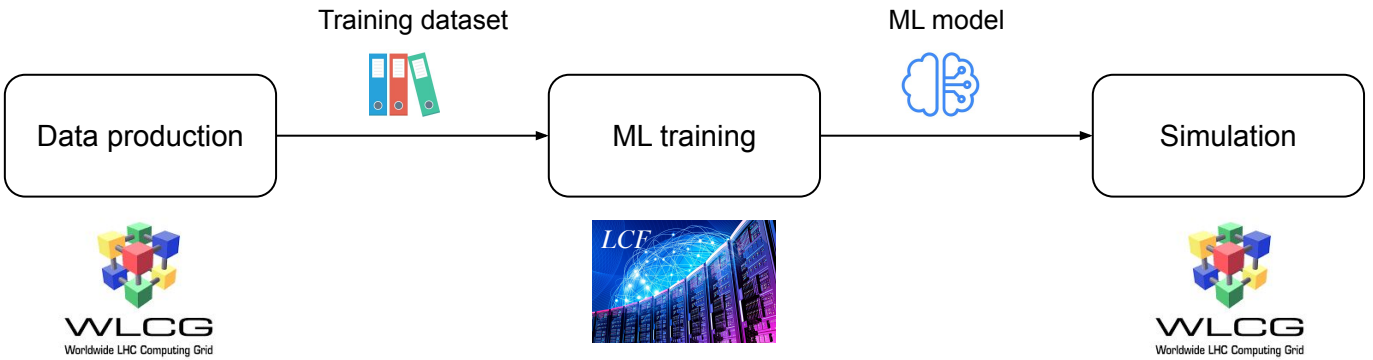
16th Dec 2022
HEP-CCE CW Meeting

Distributed Heterogeneous Computing in ATLAS

- Distributed = Geographically distributed != Multi-node/process
- Most users love local resources, but go to remote providers via PanDA when enough or suitable resources/services are locally unavailable
- A zoo of resource/service providers on the market with various benefits and constraints
  - The WLCG grid, commercial cloud resource/service providers, High-performance computing (HPC) and Leading Computing Facilities (LCFs), volunteer computing, Platform-as-a-Service (PaaS), Function-as-a-Service (FaaS), …
- Complex and emerging workflows
  - Various resources/services even in a single workflow
  - Optimal provider for each part of the workflow

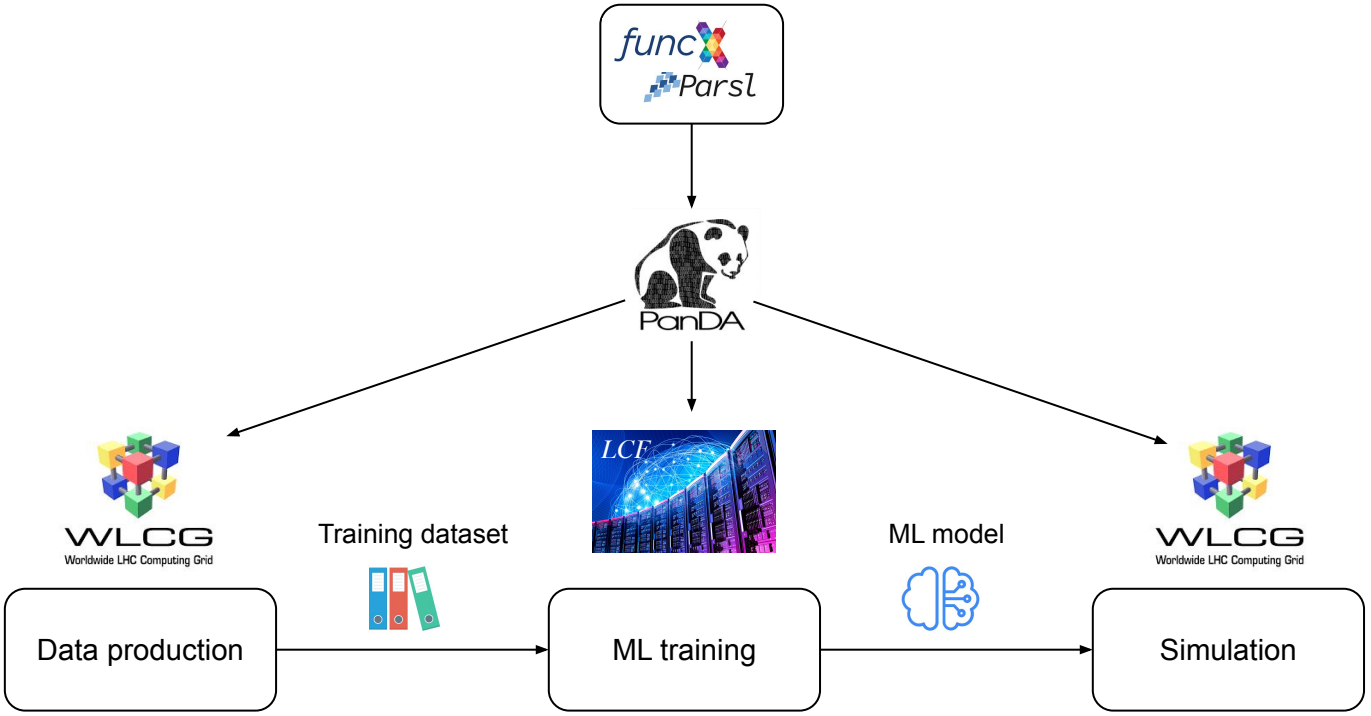
A Simple Usecase with Multiple Remote Providers and Heterogeneous Workflow

- To produce Monte-Carlo (MC) samples with a Machine Learning-based (ML-based) detector simulation
  - Three tasks in the workflow
    - Training data production, ML training, MC simulation
- Different resource/service requirements for each task
  - Training data production and MC simulation
    - Traditional High Energy Physics (HEP) workloads
    - CPU/IO intensive → the WLCG grid distributed over hundreds of computing centers
  - ML training
    - GPU intensive → LCF
- Data transfers among many computing centers/facilities are not negligible due to the considerable data size



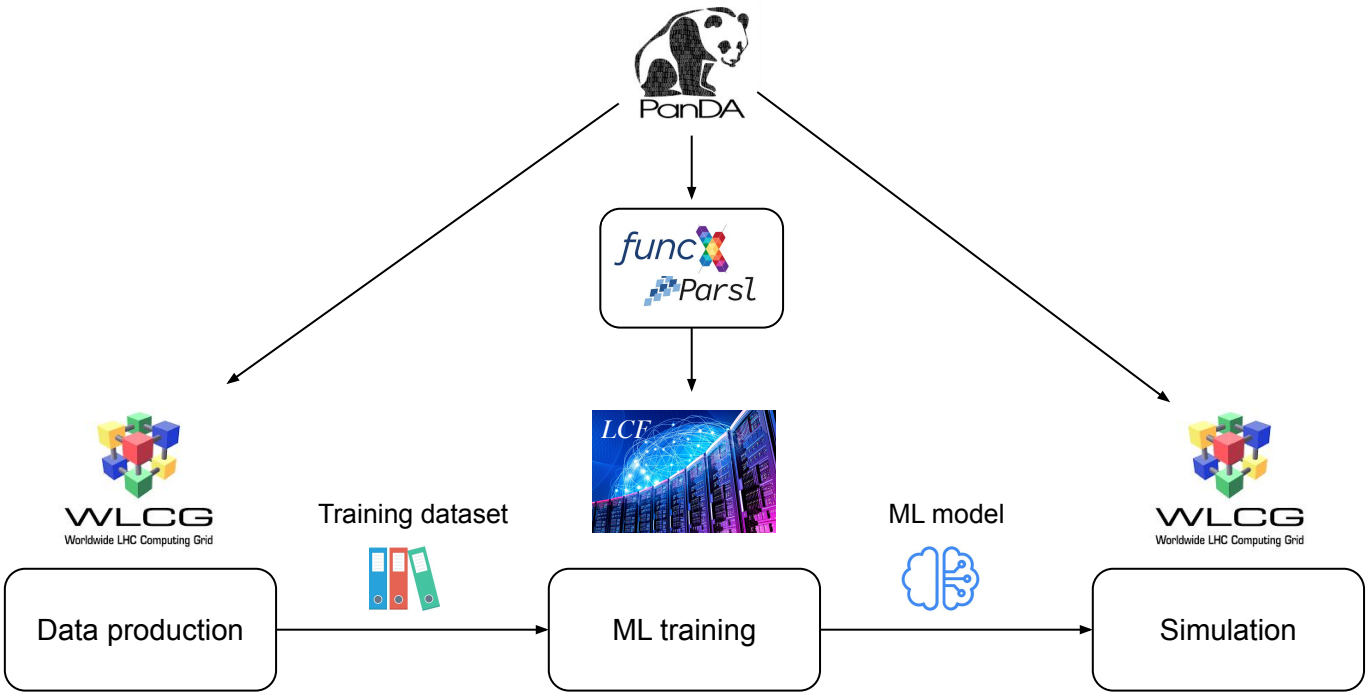
Two Scenarios for funcX/Parsl Integration

- funcX/Parsl orchestrates the entire workflow while using PanDA as an executor
  - Wouldn't make much sense for now since
    - PanDA has its own workflow management engine, iDDS
    - Not sure if traditional HEP workloads benefit unless they are rewrote to work with funcX/parsl better
    - ATLAS wouldn't appreciate any big-bang evolutions during the data taking period (~2025)



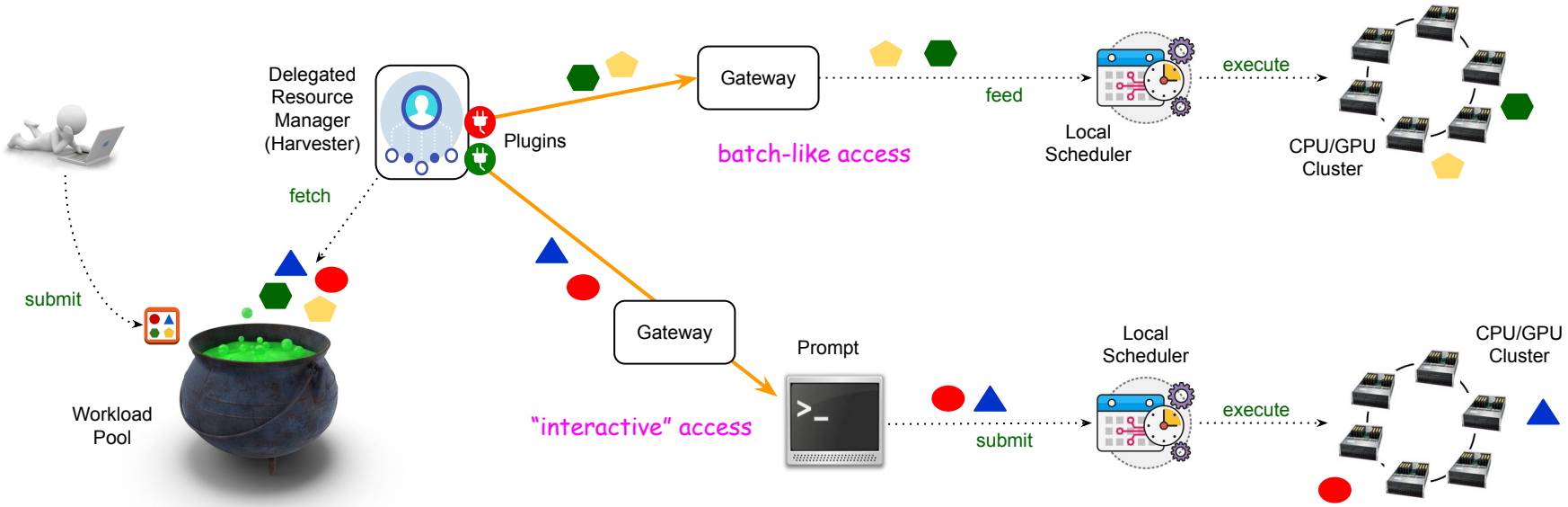
Two Scenarios for funcX/Parsl Integration (contd)

- funcX/Parsl works as a service provider in front of LCF and PanDA interacts with it on behalf of user
  - No changes to the traditional workload execution scheme
  - To leverage strength of funcX/parsl at LCF in terms of multi-CPU/GPU payload execution and stable operation
  - Will focus on newly emerging AI/ML workloads suitable for LCF



Integration of Remote Provider

- In ATLAS, central workload pool + delegated resource manager
  - The user submits workflows to the central workload pool where workflows are decomposed to smaller workload entities (tasks and jobs)
  - The delegated resource manager (Harvester) accesses to remote providers on behalf of users using common or user's credentials through plugins applicable for those providers
    - ~ One plugin for each provider
- Advantages
  - Isolation between the user and remote providers
  - Centrally managed fair-share and priorities among multiple users
  - Workload routing based on fine-grained requirements and central knowledge of provider's characteristics and availabilities

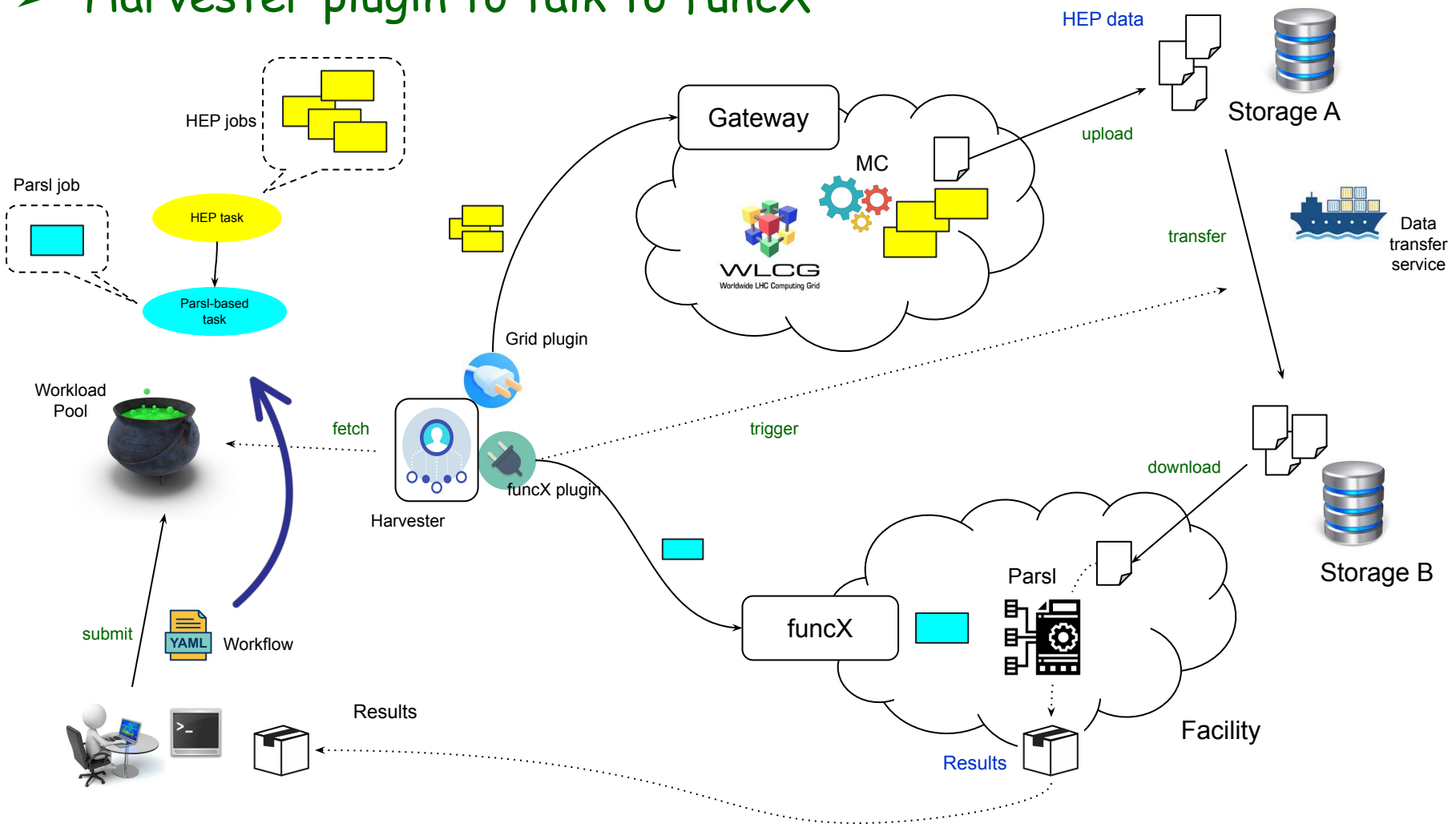


Resource/Service Providers Available via Harvester

- Traditional batch systems
  - HTCondor, Slurm, Pbs, Torque, …
  - Many academic institutes including LCFs
  - In production
- Kubernetes-based resource providers
  - Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Service (EKS), Azure Kubernetes Service (AKS), …
  - De-facto standard available in many cloud services
  - In production
- Multi-node software
  - Dask, Horovod, Ray, …
  - In production
- PaaS and FaaS
  - Google AI, Amazon ML service, REANA, funcX, ServiceX, …
  - REANA available for data preservation and analysis, while others including funcX/parsl to be integrated based on their needs

Possible Integration of funcX/Parsl in ATLAS

- Parsl-based workload in a workflow
  - E.g. HEP data production → Parsl-based AI/ML
- funcX as a gateway to allow users to run parsl-based workloads at the facility
- Harvester plugin to talk to funcX



Functions of funcX Plugin

- Fetches parsl-based payloads from the workload pool
- Pre-places input data and stages out output data to/from LCF using the data transfer service such as Globus Online
- Talks to funcX using its SDK + personal or common OIDC tokens
- Periodically reports the payload execution status back to PanDA

Harvester can be remote outside of LCF network, or be local inside of LCF network if outbound network connection is available