https://www.anl.gov/hep-cce

HEP-CCE

# Portable Applications and Workflows

# Early Achievements and Phase 1 Highlights

Somebody **for HEP-CCE**

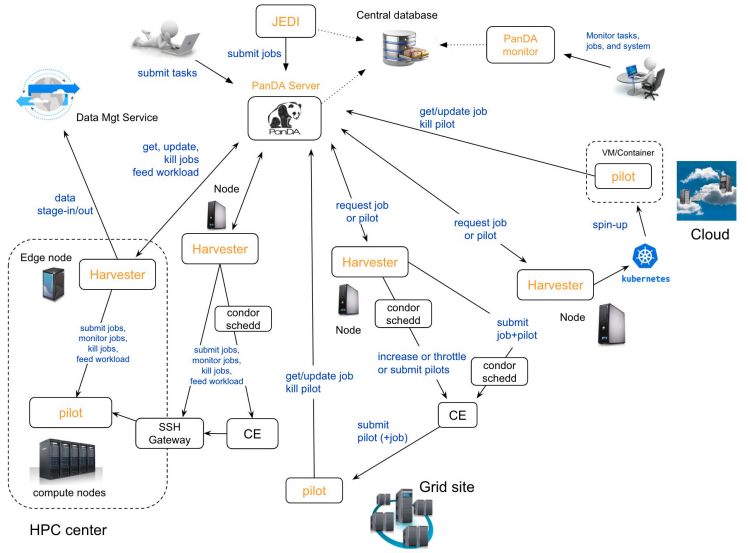
***All-Hands Meeting***
July 22 2024

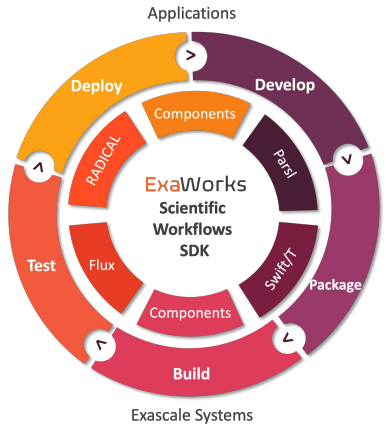
# Recap of CCE-PAW Goals

- **Goal 1** - Broaden the impact of Phase 1 PPS - Portable Parallelization Strategies - findings and software products.
  - Turn the use cases into mini-apps for benchmarking of new systems and software
  - Provide cookbooks and trainings to support refactoring and porting of production HEP software using PPS
  - Monitor technology trends to provide updated guidance on best practices
- **Goal 2** - Investigate requirements and challenges to develop common solutions to support complex HEP workflows on HPC systems.
  - Understand the different requirements and challenges of HEP experiment workflows at different scales.
  - Develop common workflow solutions leveraging existing HEP and ASCR workflow management tools on HPC systems.

| | Kokkos | SYCL | OpenMP | Alpaka | std::par |
|---|---|---|---|---|---|
| Patatrack | Done | Done* | WIP | Done* | Done compiler bugs |
| Wirecell | Done | Done | Done | no | Done |
| FastCaloSim | Done | Done | Done | Done | Done |
| P2R | done | Done | OpenACC | Done | Done |

**Phase I use cases and portability layers studied**




**Examples of HEP and ASCR workflow management tools**

# Early Achievements

HEP-CCE

Selected the **p2r** and **FastCaloSim** testbeds to turn into easily deployable, turn-key mini-apps for the first year investigations

- p2r is investigating spack for automated building
- FCS is using containerization and developing automated CI workflows, consolidating our git repo, and integrating transparent RNG interfaces to hardware generators

Selected the **DUNE Near Detector 2x2 Simulation** and **ATLAS Simulation** workflows to investigate portability solutions for complex workflows

- DUNE ND 2x2 Sim is being ported from Perlmutter to Polaris
- ATLAS workflow is being worked on to run on Perlmutter

# Mini-app: p2r

Martin Kwok

HEP-CCE

- Decided to use spack for the distribution of p2r
  - HPC friendly: no installation of spack needed; can re-use system module etc)
  - What we get from spack: automated build + install via easy python config.
- Suitable for p2r since:
  - Limited external dependencies
  - Lightweight source code (easy clone-and-go)

Green = results available
Red = no results/ unsupported
Orange = could be made available with effort

**Goal: One (spack) package to rule them all**

| | Implementations | | | | | | |
|---|---|---|---|---|---|---|---|
| | TBB | CUDA | HIP | Kokkos | Alpaka | Std::par | SYCL |
| NVDIA | | | | | | *nvc++* | |
| AMD | | | | | | | |
| Intel | | | | | | | *dpl* |
| CPU | | | | *OMP* | *TBB* | | |

# Mini-app: p2r

- Pros: Spack recognizes different build systems
  - Make, Ninja, Autotools, CMake
- Cons: p2r does NOT have a build system
  (Been using loose shell scripts for different implementation)
  (Side benefit: collect all the versions/run parameters for var. implementations)
- Decided to use CMAKE as the build system
  - pros: already used in Kokkos/Alpaka
- Step (1) progress:
  - Reorganized p2r repo for CMAKE(here)
  - Able to build simple versions(CUDA, TBB)
- Step (2) progress:
  - Set up the skeleton of the `package.py` (main spack config) similar to Kokkos
    (p2r have similar multi-backend structure)

# Mini-app: FastCaloSim

HEP-CCE

## Containerization of app for multiple platforms

- CUDA
- Kokkos/NVIDIA
- HIP/AMD

**Pengfei Ding**
**Dhruva Kulkarni**
**Charles Leggett**

## Development of git CI pipelines for automatic build tests

- can be triggered to run on perlmutter (for NVIDIA GPUs) or exalearn (AMD)
- moved FCS repo to hep-cce github org for better integration with CI workflow management

**Marco Lorenz**
**Pengfei Ding**
**Dhruva Kulkarni**

## Extension of HIP to exercise NVIDIA backend

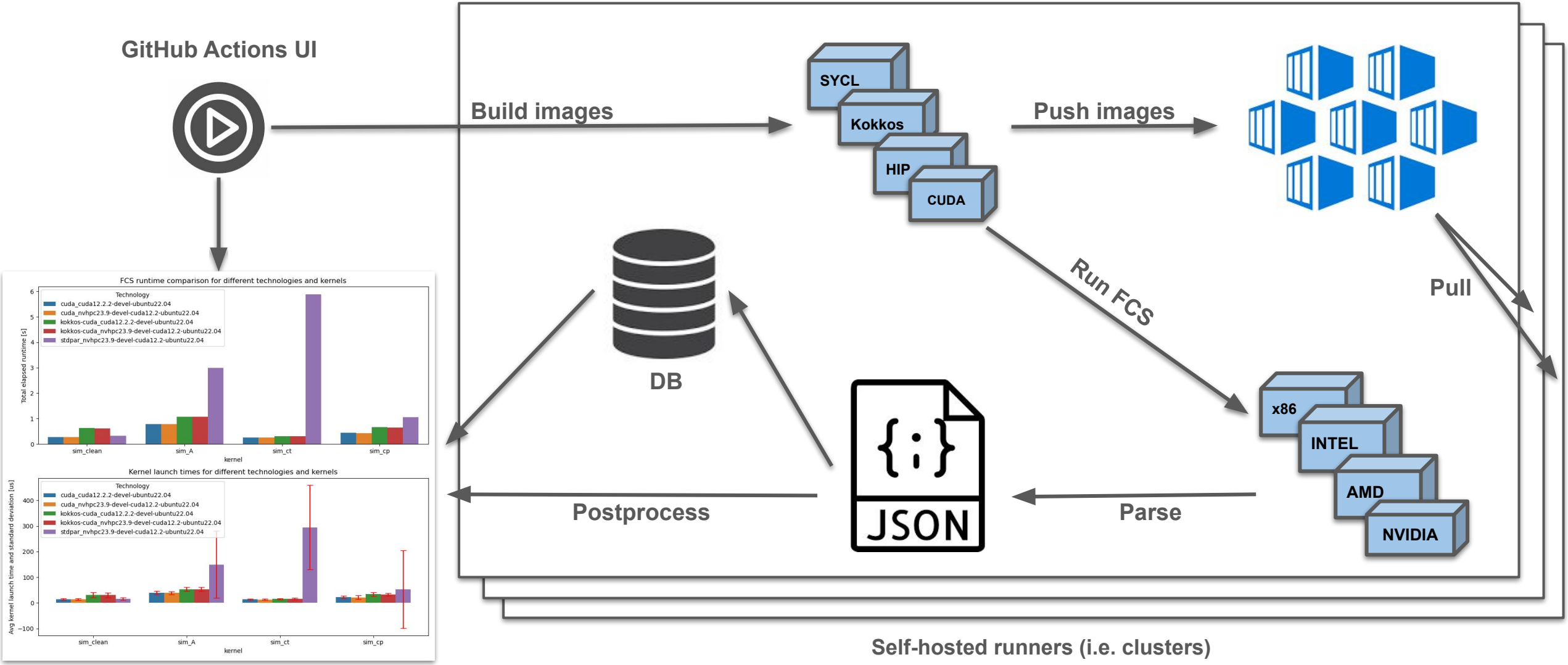
**Charles Leggett**
**Mohammad Atif**

## Integration of architecture-independent RNG wrapper for full device transparency

- using header only interfaces developed in Phase 1

**Vedit Venkatesh**

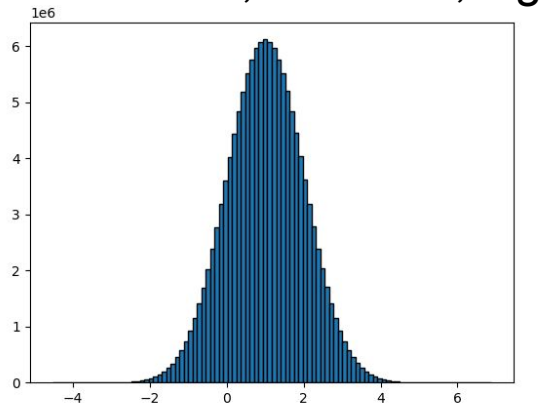
# Pipeline - Automating Performance Comparison



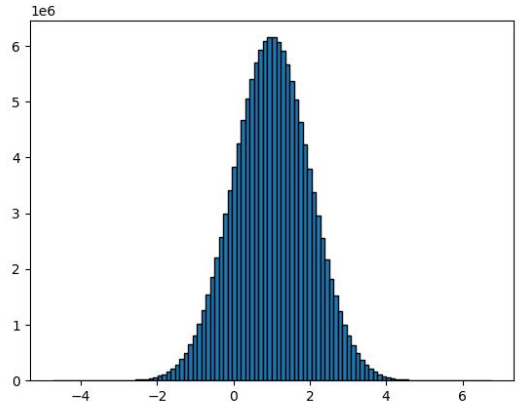
# Mini-app: FastCaloSim Summer Student Project

- SULI student **Vedit Venkatesh**, Lafayette College (Computer Science and Physics)
- Tasked to help with testing and benchmarking the portable RNG wrapper integration in FastCaloSim using https://github.com/GKNB/test-benchmark-OpenMP-RNG (**Tianle Wang**)
- First step: verify the generated RNGs follow the expected distributions
- Next step: verity the FCS results and performance comparison with the new RNG implementation (WIP with **Mohammad Atif**)

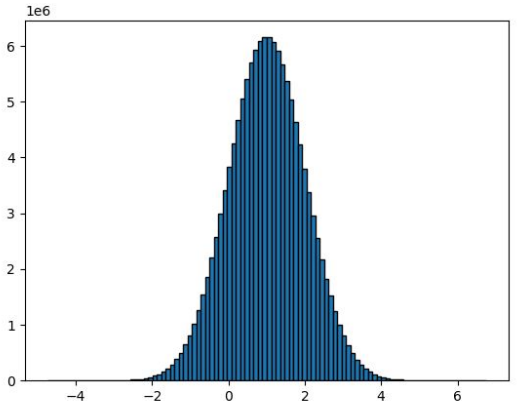
float normal distribution,mean 1.0, sigma 1.0




CPU backend



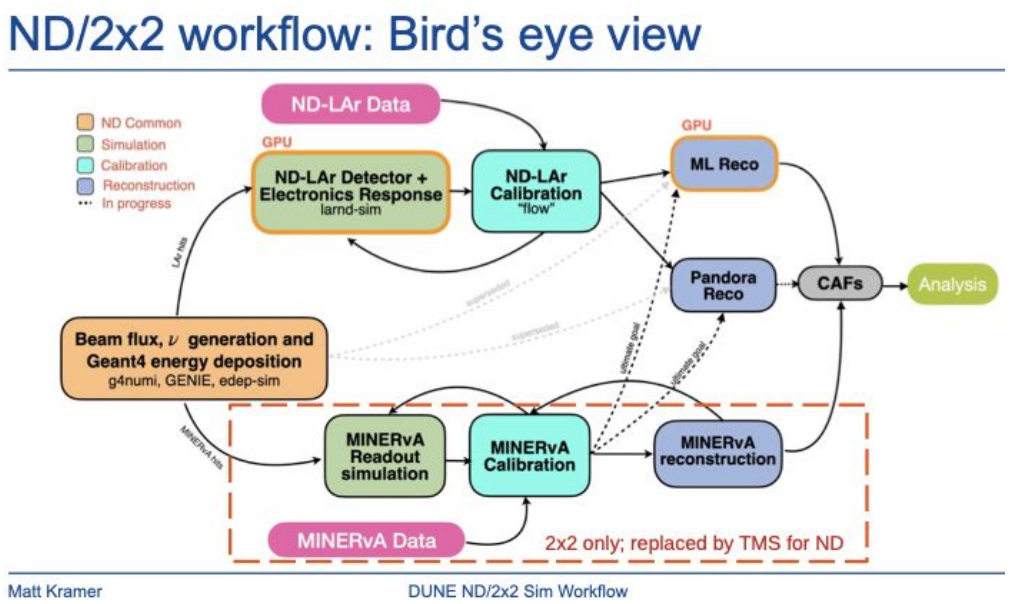

AMD backend




NVIDIA backend

# Workflows: DUNE

- Following discussions with several people from DUNE, and in particular presentation by Matt Kramer, we decided to use DUNE Near-Detector 2x2 LArTPC prototype module analysis workflow as a case study (ND/2x2 Workflow)
  - Lots of pileup, opportunities for parallelization/GPU utilization
  - Sufficiently complex workflow, with potential ML components
- First, try to replicate the workflow on Perlmutter
- Then, to test portability, move the workflow to Polaris
  - we will try Frontier next



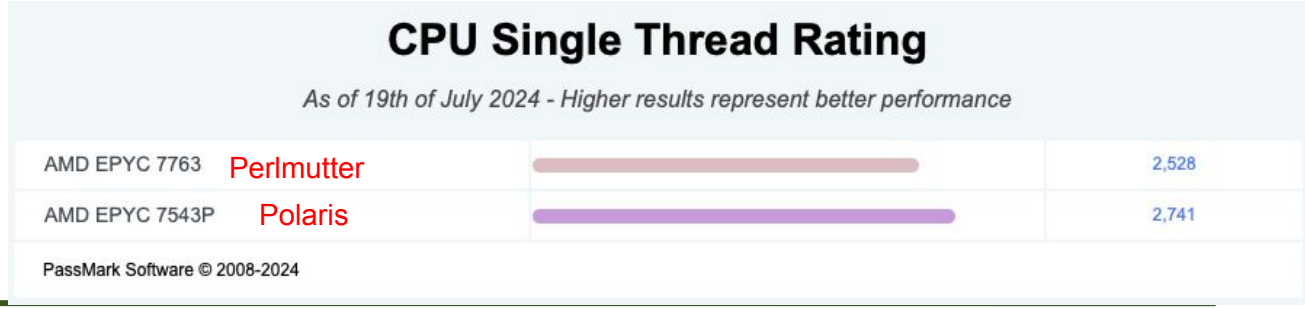
# DUNE ND/2x2 Portability and Performance Analysis

**Bruno Coimbra**
**Ozgur Ozan Kilic**

| Task | Perlmutter | Polaris |
|---|---|---|
| Genie (Step1/Step2) | 0m46s/28m32s | 0m36s/26m20s |
| edep_sim (Step1/Step2) | 0m37s/25m42s | 0m25s/24m1s |
| Hadd (Step1/Step2) | 0m18s/0m18s | 0m4s/0m4s |
| spill_build | 0m34s | 0m6s |
| edep2flat | 3m00s | TBD: Python venv |
| minerva | 2m59s | —- |
| convert2h5 | 0m33s | —- |
| larnd_sim | 0m17s | —- |
| ndlar_flow | 2m12s | —- |
| flow2spera | 19m56s | —- |
| MLreco (inf/analysis) | 2m01s/1m15s | —- |
| cafmaker | 0m43s | —- |
| validation | 2m43s | —- |

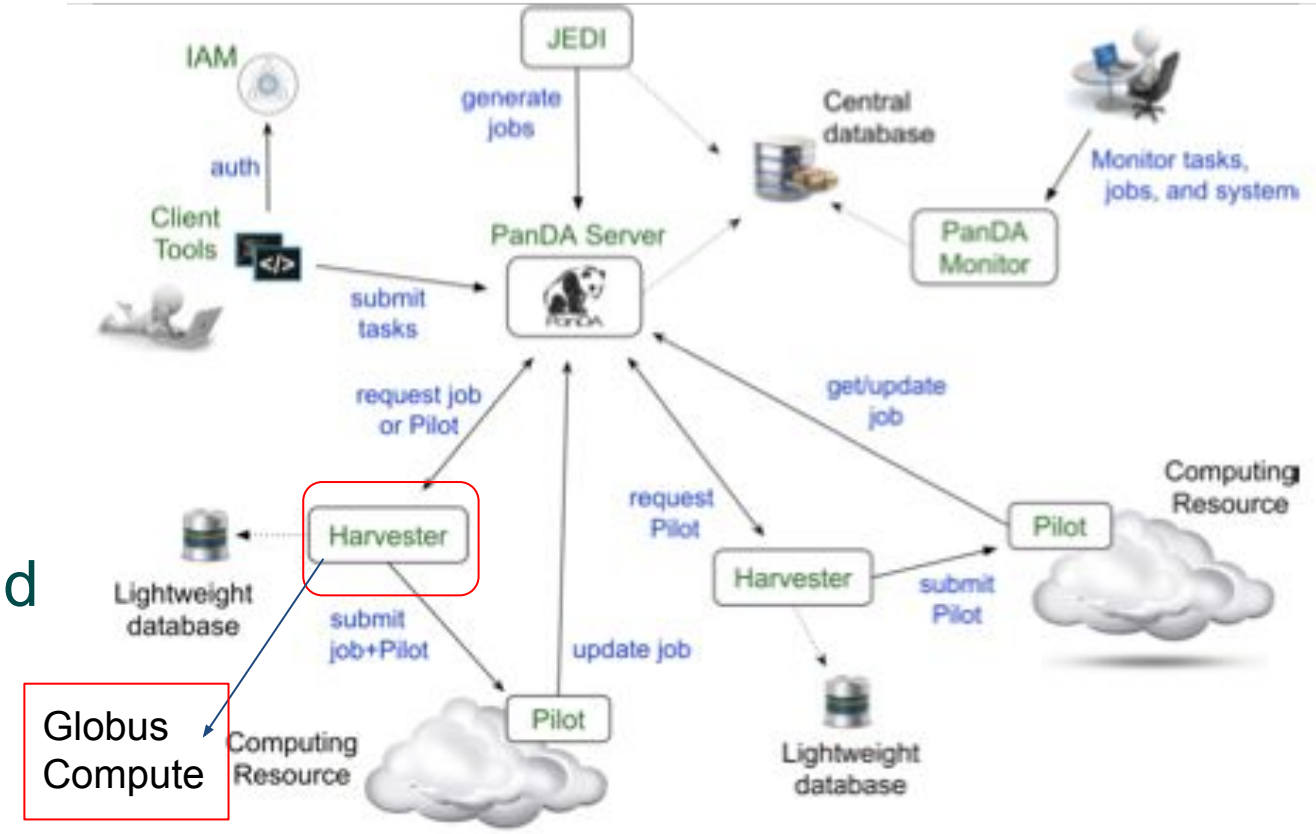
Portability Issues:

- ./install_everything.sh needs heavy modification and modules are not the same between two systems
- 2x2EventGeneration is not available on Polaris
  - Copied over from Perlmutter
- Container images are not available in Polaris
  - Pulled from Docker Hub with Singularity
- The workflow expects Perlmutter directory tree
  - Used a lightweight container to replicate it



# Workflows: ATLAS

- ATLAS already runs Simulation workflows on Perlmutter through PanDA+Harvester.
- Goal is to see if we can integrate standard workflow tools as Harvester plugins to manage the access to HPC resources.
- The idea is that the Harvester plugins will handle the system-specific configurations and access to alleviate the burden on the experiment workflows.
- Candidate under investigation: **Globus Compute** (formerly FuncX).



# ATLAS Workflow Status

- Set up test queue and Harvester test instance on Perlmutter
- Test PanDA pilot jobs running on Perlmutter by HEP-CCE
- Learned to set up Globus Compute on Perlmutter to support both direct Python script submission and submission through pilots
- **Issues**
  - Problem accessing ATLAS directories without ATLAS accounts (resolved)
  - Have to use the collaboration account of “usatlas” to submit job due to a harvester-policy permission issue, which is incompatible with globus compute right now (might be fixed relatively easy)
  - PanDA pilot could not retrieve a job (under investigation)

3: 2024-07-17 04:13:32,284 | WARNING | pilot.control.job | retrieve | did not get a job -- max number of job request failures reached: 5

**Doug Benjamin**
**Mikhail Titov**
**Tianle Wang**

# Summary and Plans

HEP-CCE

- In Year 1 of Phase 2, PAW has made concrete plans towards both goals (Portable Applications and Portable Workflows)
- Initial progress has been made:
  - Development of ready-to-deploy miniapps for FastCaloSim and p2r is in progress.
  - Identified DUNE and ATLAS workflows on HPC as initial targets for portability studies.
  - Tests on replicating these workflows have varied degrees of successes and failures.
- We also identified several issues that need to be addressed to enable portable workflows:
  - HEP workflows have complicated dependencies and existing workflow management tools. Integrating them with new workflow managers or porting them to a different platform may not be easy.
  - HEP relies heavily on cvmfs, which is not currently available on all HPC systems.
- We need to continue to build, test and benchmark the example workflows before we can implement a portable solution.
  - Based on our Year 1 findings, we will expand the number of apps/workflows in Year 2 and beyond