# WORKFLOWS AND DATA CURATION

Thomas Uram
Argonne Leadership Computing Facility

Argonne NATIONAL LABORATORY

# DEFINITIONS

- The title of this talk says that I'm to tell you about workflows and data curation.
  - Workflows essentially refer to running multiple simulation and analysis jobs on one or more systems.
  - Data Curation describes how you manage the data produced by those jobs from the time the data is produced by the simulation, through multi-level analysis, to final data products from which the science has been fully extracted.

# WHY SHOULD YOU CARE ABOUT WORKFLOWS?

- You are spending two weeks at ATPESC learning to use large-scale parallel systems to do science:
  - architectures
  - programming models
  - communication
  - solvers
  - visualization
  - profiling
  - optimizing for node-level performance
  - scaling parallel applications to tens of thousands of nodes
- Once you've written your application and achieved near-peak node-level performance and scalability, what comes next?


WORKFLOWS

# WORKFLOWS

# DATA CURATION

- How should your data be managed during the simulation process and thereafter?

# ALLOCATION EXAMPLES

- Small number of large jobs running on Mira over course of the year followed by some analysis
  - 40 jobs x 16K nodes * 16 cores/node * 12 hours ~= 100M core-hours
  - ~Reasonable to manage manually
- Large number of simulation+analysis jobs running across multiple facilities, requires coordination of jobs submitted to multiple schedulers, large/long data transfers, and interaction with project storage and archival storage
  - Programmatic management would be a clear benefit

Workflows can save you!
…but you might have to code them yourself!

# WHY USE WORKFLOWS?

- Automate job submission
- Simplify computational campaigns
- Increase concurrency by disentangling data dependencies
- Robustness: Improve error handling and recovery (retries)
- Coscheduling of multiple resources
- Systematize data management
- Provenance/Metadata tracking
  - Validation
  - Reuse

See Kerstin Kleese van Dam's talk at ATPESC 2016

# WHAT IS A WORKFLOW?

- It depends who you ask
- Basically a collection of jobs to be run
  - Could be a sequence of individual jobs
  - Could be a sequence of varying numbers of jobs
- Available means of describing and running workflows
  - Script jobs
  - Job dependencies
  - Ensemble jobs
  - In situ
  - Custom workfows
  - Workflow software

# WORKFLOW VIA SCRIPT JOB

- You can submit a job that executes your application
  - qsub -q prod -n 512 -t 10 -A yourproject application.exe
- Alternatively, you can submit a script job that executes multiple applications sequentially
  - qsub -q prod -n 512 -t 10 -A yourproject script.sh


***script.sh***

```
runjob application.exe
runjob application.exe
```

- With this approach, you wait in the queue one time to run your application multiple times
- However, small long jobs tend to stay in the queue longer than small short jobs
  - It may be better to submit individual jobs with dependencies

# WORKFLOW VIA COBALT JOB DEPENDENCIES

- A simple way to achieve a linear workflow is simply to set dependencies between your jobs
  - qsub -q prod -n 512 -t 10 -A yourproject a.out

    Job 12345 submitted
  - qsub -q prod -n 512 -t 10 -A yourproject --dependencies 12345 a.out
  - qsub -q prod -n 512 -t 10 -A yourproject --dependencies 12345:12346 a.out

- Is there an advantage to setting job dependencies?
  - Dependent jobs accumulate score more quickly
- How many jobs can be submitted at once?


multiple independent short jobs

```
qstat -Q
Name      Users     Groups    MinTime   MaxTime   MaxRunning  MaxQueued  MaxUserNodes  MaxNo
default   None      None      00:05:00  01:00:00  5           20         2048          1024
```

  - On Mira, max queued is set to 20

# WORKFLOW VIA ENSEMBLE JOBS: RUN BIGGER JOBS

***http://trac.mcs.anl.gov/projects/cobalt/wiki/BGQUserComputeBlockControl***

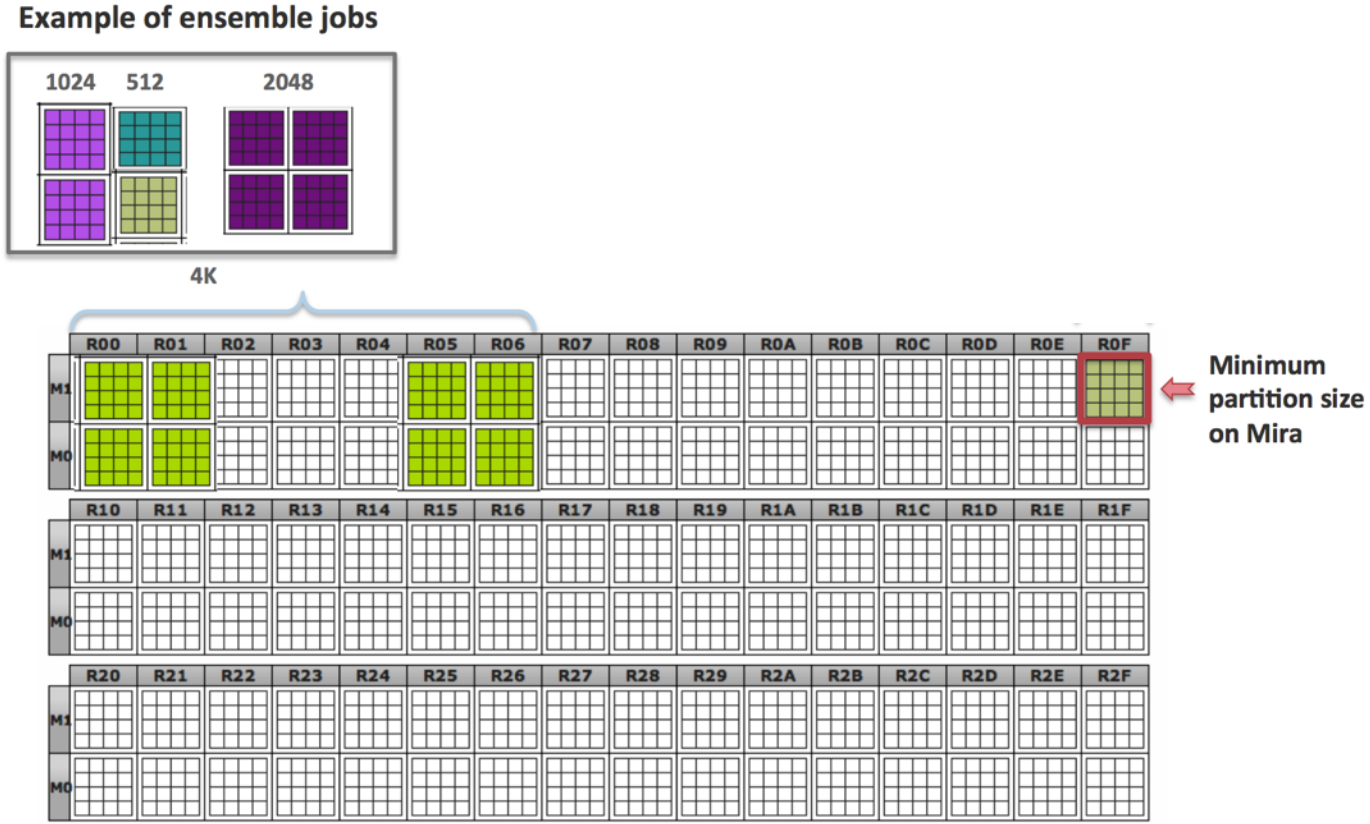
```
#!/bin/bash
BLOCKS=`get-bootable-blocks --size 512 $COBALT_PARTNAME`

for BLOCK in $BLOCKS
do
    boot-block --block $BLOCK &
done
wait

for BLOCK in $BLOCKS
do
    runjob --block $BLOCK : ./my_binary &
done
wait

for BLOCK in $BLOCKS
do
    boot-block --block $BLOCK --free &
done
wait
```

# WORKFLOW VIA ENSEMBLE JOBS: RUN BIGGER JOBS



# WORKFLOW ACROSS ALCF SYSTEMS

- How can I set job dependencies across multiple resources (e.g. Mira and Cooley)?
  - ALCF does not provide a means of doing this currently
  - However, Cetus and Cooley mount the Mira filesystems. Analysis jobs can be run on Cooley without needing to transfer the data.

# SOFTWARE FOR MANAGING WORKFLOWS

This manual effort can be ameliorated by scripting your job submission so that new jobs are automatically submitted as you drop below the max_queued limit. Multiple toolkits are available for this purpose.

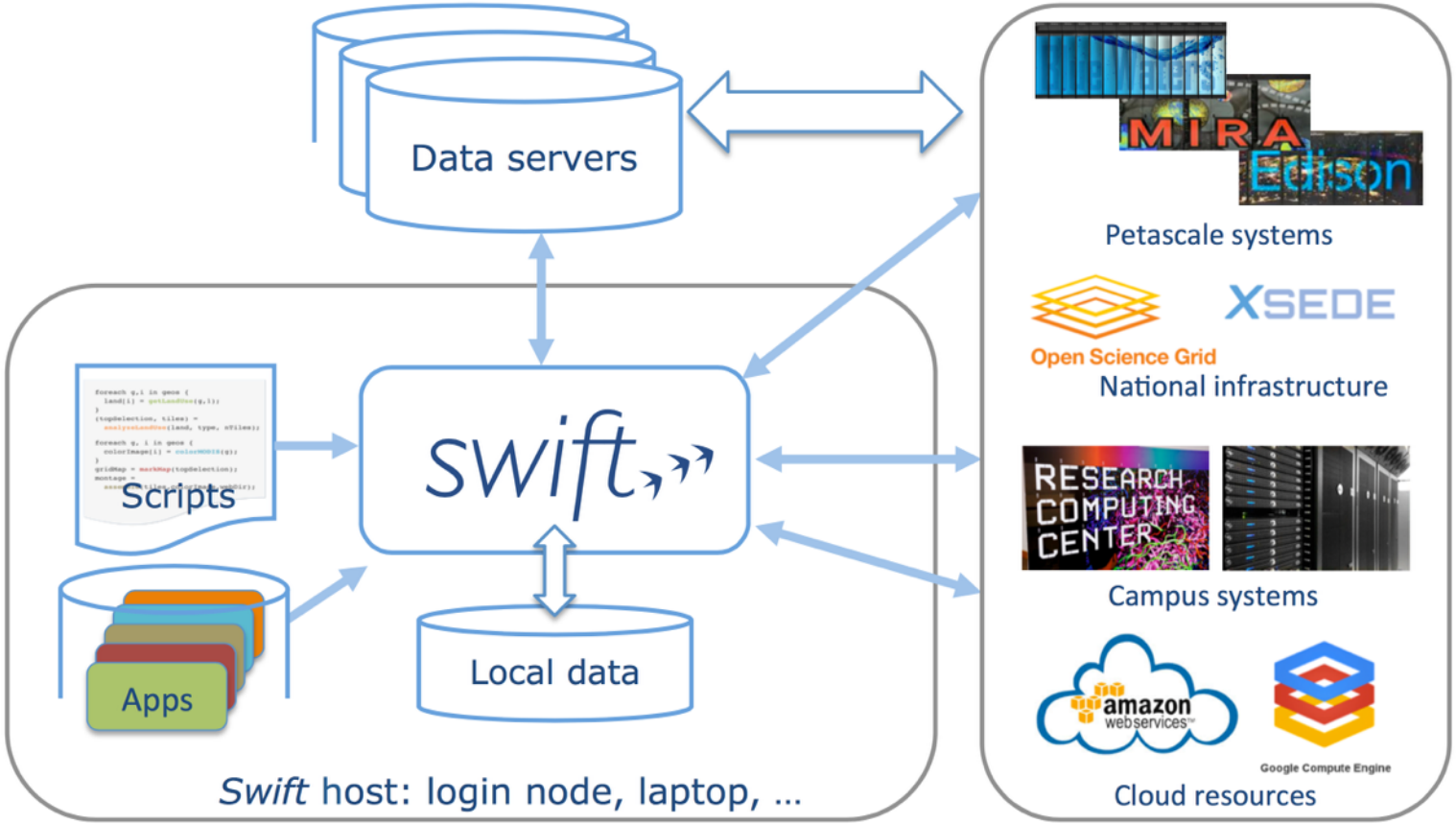
**Swift (http://swift-lang.org/)**
Swift provides a C-like language for expressing workflows that consist of command-line invocations, and an engine for managing their execution.

**Pegasus/Condor (https://pegasus.isi.edu/)**
Pegasus bridges the scientific domain and the execution environment by automatically mapping high-level workflow descriptions onto distributed resources. It automatically locates the necessary input data and computational resources necessary for workflow execution.Pegasus enables scientists to construct workflows in abstract terms without worrying about the details of the underlying execution environment or the particulars of the low-level specifications required by the middleware

**Fireworks (https://pythonhosted.org/FireWorks/)**
FireWorks is a free, open-source code for defining, managing, and executing workflows. Complex workflows can be defined using Python, JSON, or YAML, are stored using MongoDB, and can be monitored through a built-in web interface. Workflow execution can be automated over arbitrary computing resources, including those that have a queueing system. FireWorks has been used to run millions of workflows encompassing tens of millions of CPU-hours across diverse application areas and in long-term production projects over the span of multiple years.

**Many others…**

# Swift enables execution of simulation campaigns across multiple HPC and cloud resources

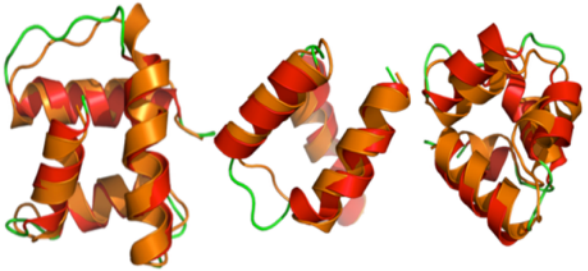



See https://www.alcf.anl.gov/swift

# When do you need HPC workflow?

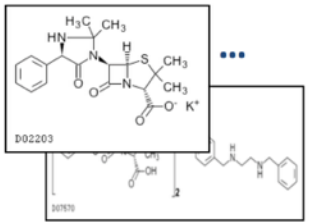
## Example application: protein-ligand docking for drug screening

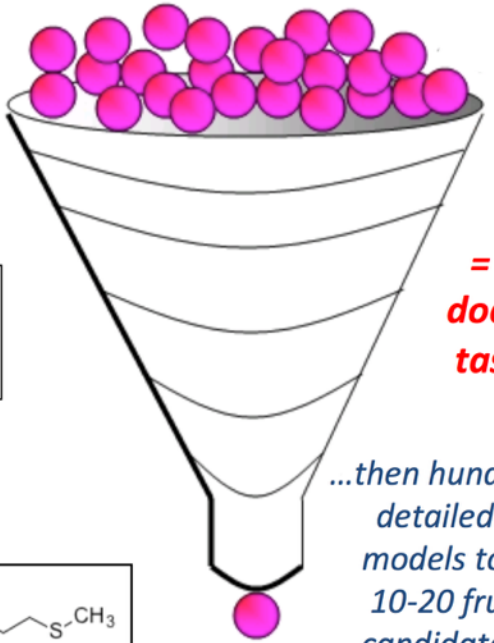
*O(10) proteins implicated in a disease*



X

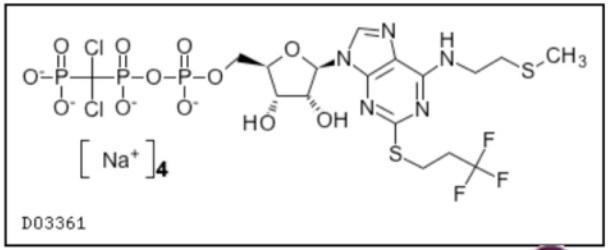
*O(100K) drug candidates*

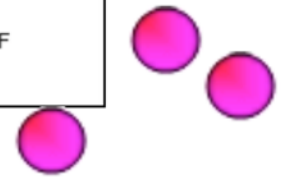


***= 1M docking tasks…***

*…then hundreds of detailed MD models to find 10-20 fruitful candidates for wetlab & APS crystallography*



# Expressing this many task workflow in Swift

*For protein docking workflow:*

```
foreach p, i in proteins {
   foreach c, j in ligands {
      (structure[i,j], log[i,j]) =
       dock(p, c, minRad, maxRad);
}
scatter_plot = analyze(structure)
```

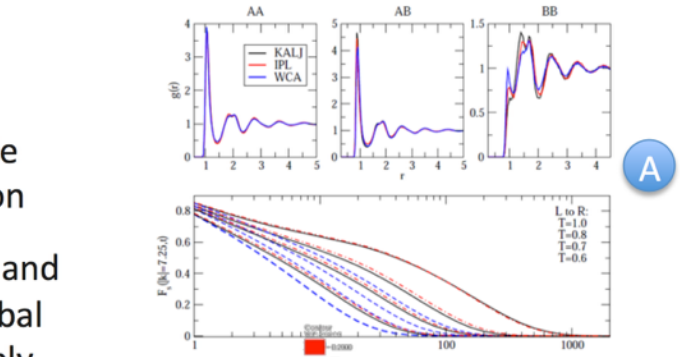
*To run:*

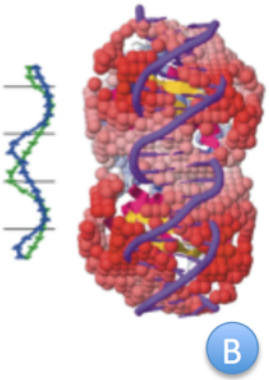
```
swift -site cooley,blues dock.swift
```

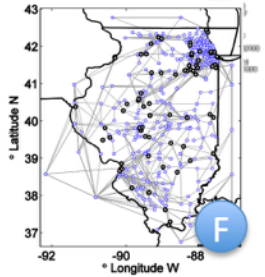
# Large-scale applications using Swift

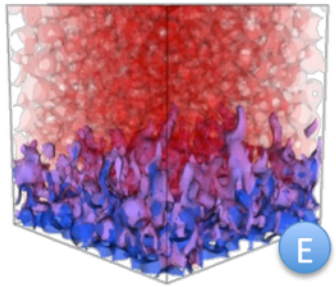
- **A** Simulation of super-cooled glass materials
- **B** Protein and biomolecule structure and interaction
- **C** Climate model analysis and decision making for global food production & supply
- **D** Materials science at the Advanced Photon Source
- **E** Multiscale subsurface flow modeling
- **F** Modeling of power grid for OE applications

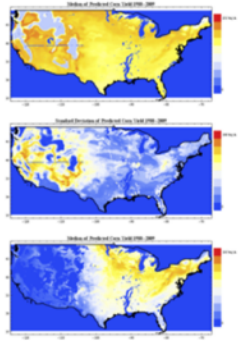
All have published science results obtained using Swift

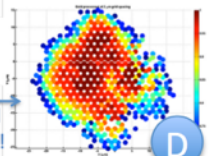


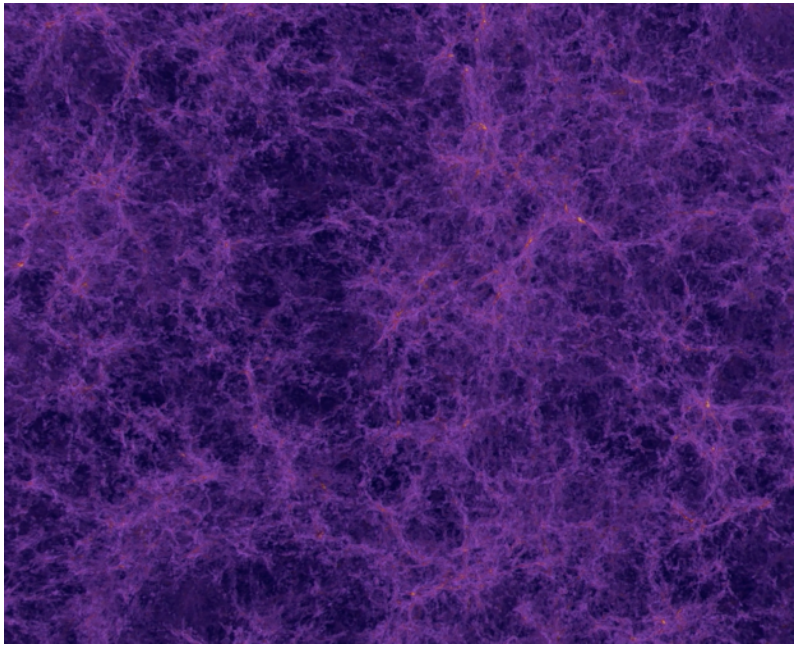





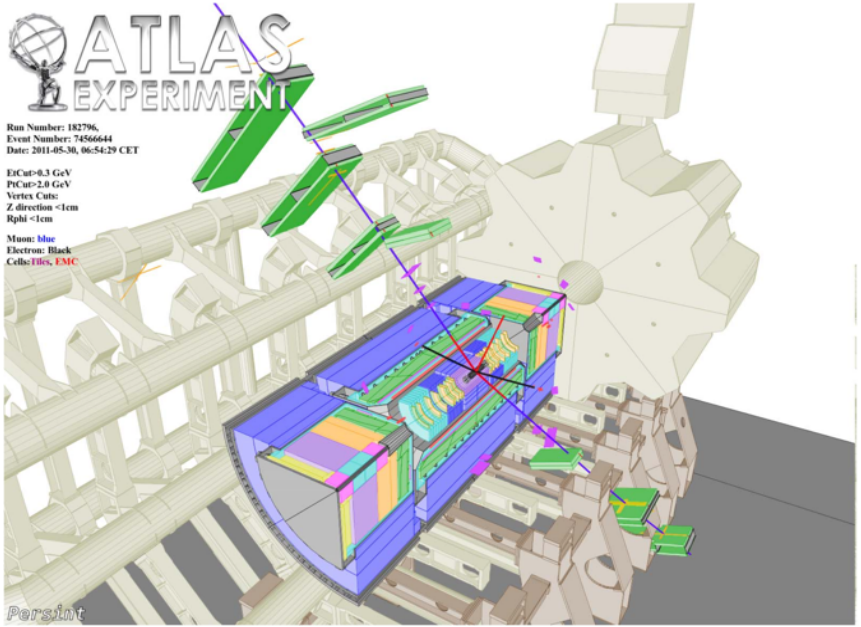
Argonne NATIONAL LABORATORY

# EXAMPLES

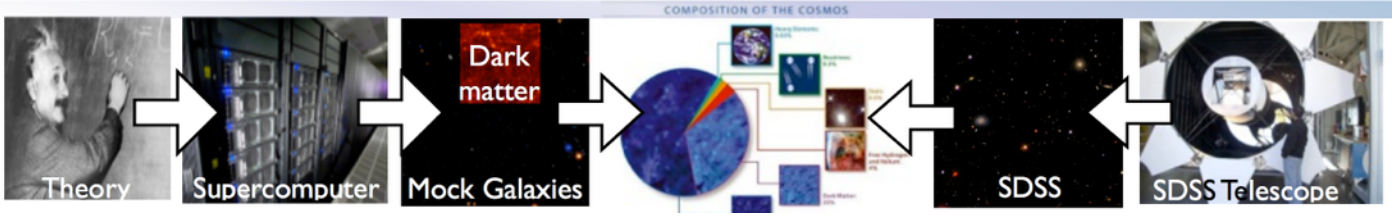
Hardware/Hybrid Accelerated Cosmology Code (HACC)



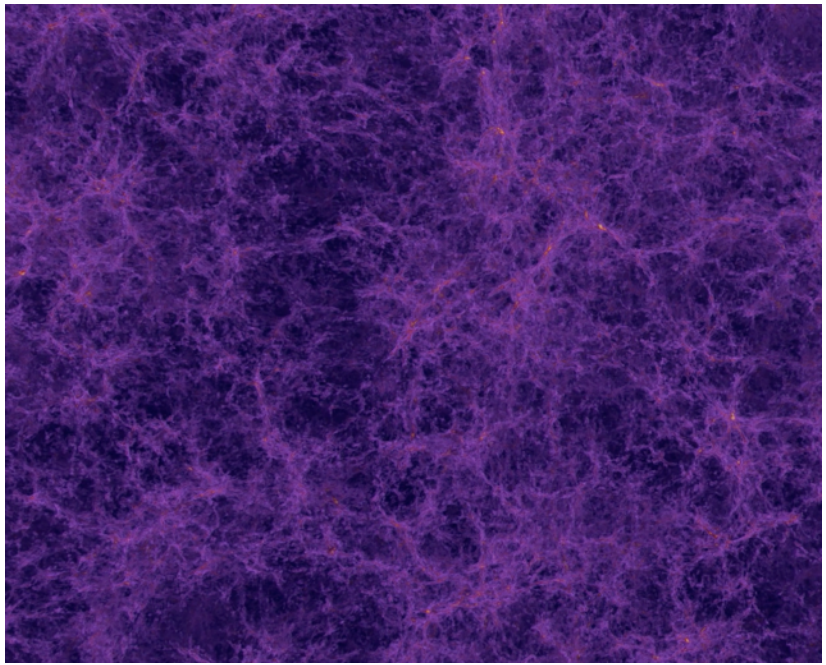
High Energy Physics event generation



# HACC




- N-body code for simulating the evolution of large-scale structure in the universe
- HACC runs on all major computing systems, and leverages heavily optimized kernels to achieve outstanding efficiency and scalability
- Largest HACC simulations have evolved more than 1 trillion particles
- Enormously large, science-rich datasets are managed for analysis within project and also by cosmology community



See Salman Habib's talk from ATPESC 2015
http://extremecomputingtraining.anl.gov/files/2015/08/habib_hacc_atpesc_2015.pdf

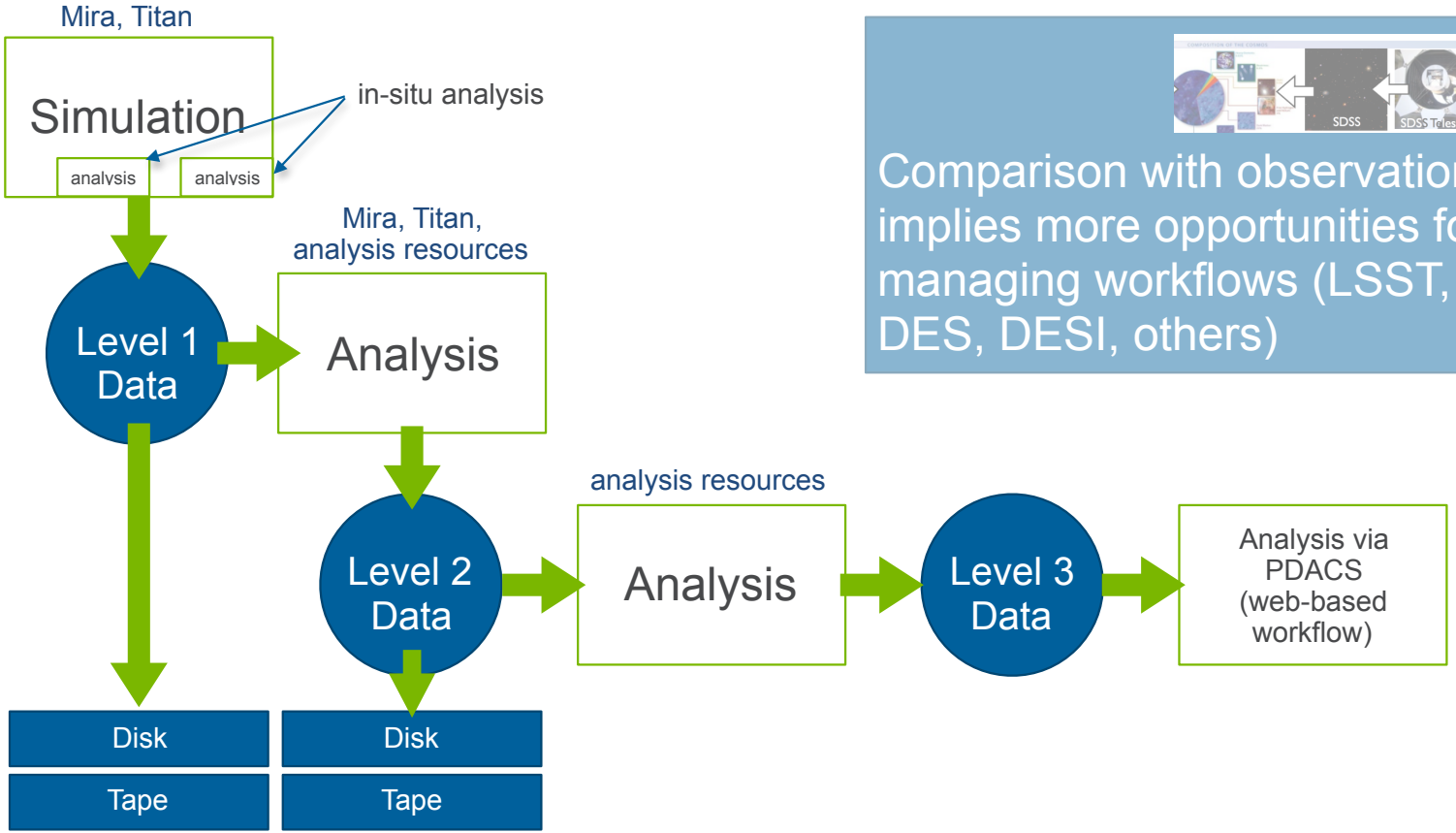
# HACC DATA PRODUCTS

- Level 1 data
  - Raw particle data, 36 bytes per particle (id, position, velocity, mass, etc.)
- Level 2 data
  - Simulation particles (1% sampled from raw particles)
  - Dark matter halo particles
  - Dark matter halo properties (halo membership, halo particle counts, halo profiles)
  - Matter power spectrum
  - Density maps
- Level 3 data
  - Halo power spectrum, merger trees, galaxy catalogs

# HACC SIMULATIONS

- OuterRim (ALCF)
  - 1 trillion particle simulation
  - Level 1 data: 4PB
  - Level 2 data: 600TB
- QContinuum (OLCF)
  - 0.5 trillion particle simulation
  - Level 1 data: 2PB
  - Level 2 data: 400TB
- Mira-Titan Universe (ALCF, OLCF)
  - 32 billion particles per simulation, 50 simulations with varying cosmological parameters
  - Level 1 data: 1.1PB (OLCF), 400TB (ALCF)
  - Level 2 data: 78TB (ALCF)

# HACC WORKFLOW







Comparison with observation implies more opportunities for managing workflows (LSST, DES, DESI, others)

# DOE COMPUTING RESOURCES

| | Argonne | Oak Ridge | NERSC |
|---|---|---|---|
| |  |  |  |
| | ALCF<br>30PB disk | OLCF<br>32PB disk | NERSC<br>30PB disk |
| HACC data | 6PB on disk<br>6PB on tape | 3.5PB on disk<br>~3.5PB on tape | 300TB on disk<br>->6PB on tape |

# IN SITU ANALYSIS WORKFLOW

- In situ == running analysis inline with the simulation
- In situ analysis has several advantages
  - Operates on data already in memory; during postprocessing, reading large data from the filesystem can be expensive
  - Is able to perform analysis at the same scale as simulation
  - Operates on full simulation data sets before reduction as opposed to offline analysis on reduced data
  - How to reproduce analysis results based on simulation-time datasets that were not written to disk? Rerun the simulation.

# HACC - COSMOTOOLS (IN-SITU ANALYSIS)

- Cosmotools is a configurable in-situ analysis framework
  - The simulation configuration specifies analyses that should be performed between time steps during the simulation, and their parameters
  - On execution, particle data is passed from the simulation to analysis routines for processing
  - When finished, the simulation proceeds to the next timestep
- Analysis runs prior to data reduction; potentially the only opportunity to analyze this data
- Efficient: avoids the need to subsequently load data from disk
- Delivers intermediate analysis results during simulation runs; especially helpful during long runs

# HACC - COSMOTOOLS (IN-SITU ANALYSIS)

In-situ analysis is appropriate for some tools and not others, according to a few criteria

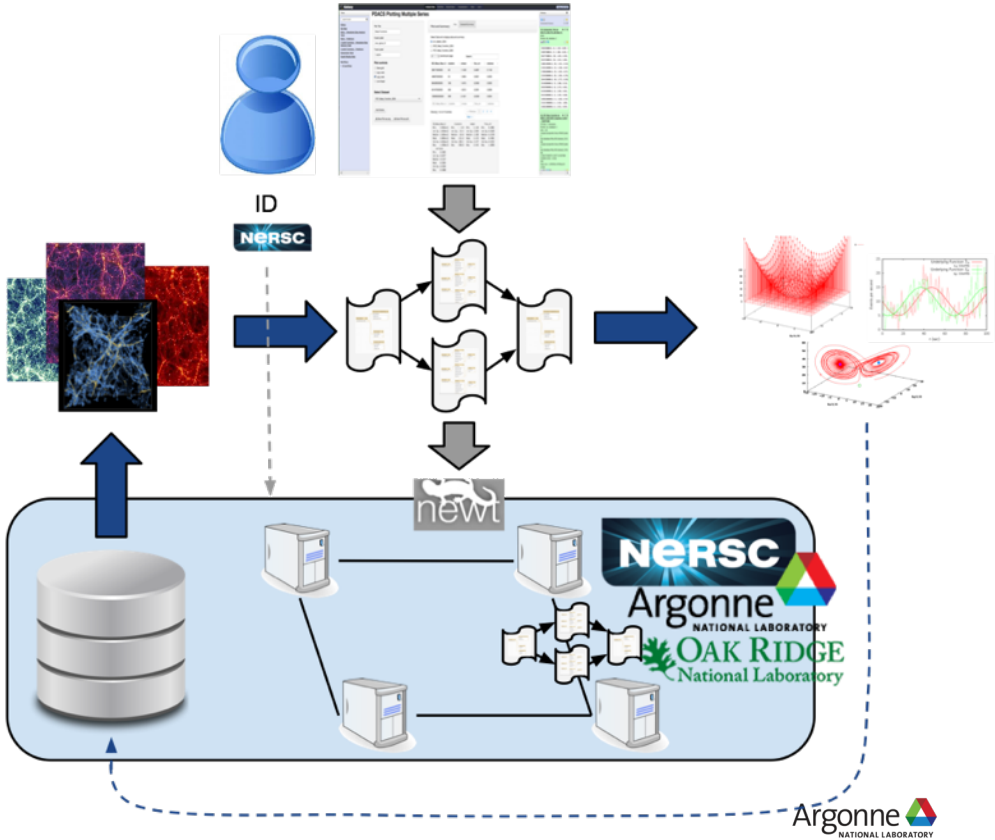
- The analysis runtime must not significantly hinder simulation progress
  - long runtime
  - high memory usage

  } implies optimized analysis codes
- The analysis must run at a scale comparable to the simulation
- The analysis must run on the same architecture as the simulation (analyses that are more appropriate for the GPU, for example, are run in post-processing instead of in-situ on the Blue Gene)
- The analysis operates on the same data distribution as the simulation (not strictly required; an analysis could modify the data distribution, but must Leave No Trace of itself in the simulation data)
- Analysis and the arrow of time: Can the data be analyzed in the order it is produced?

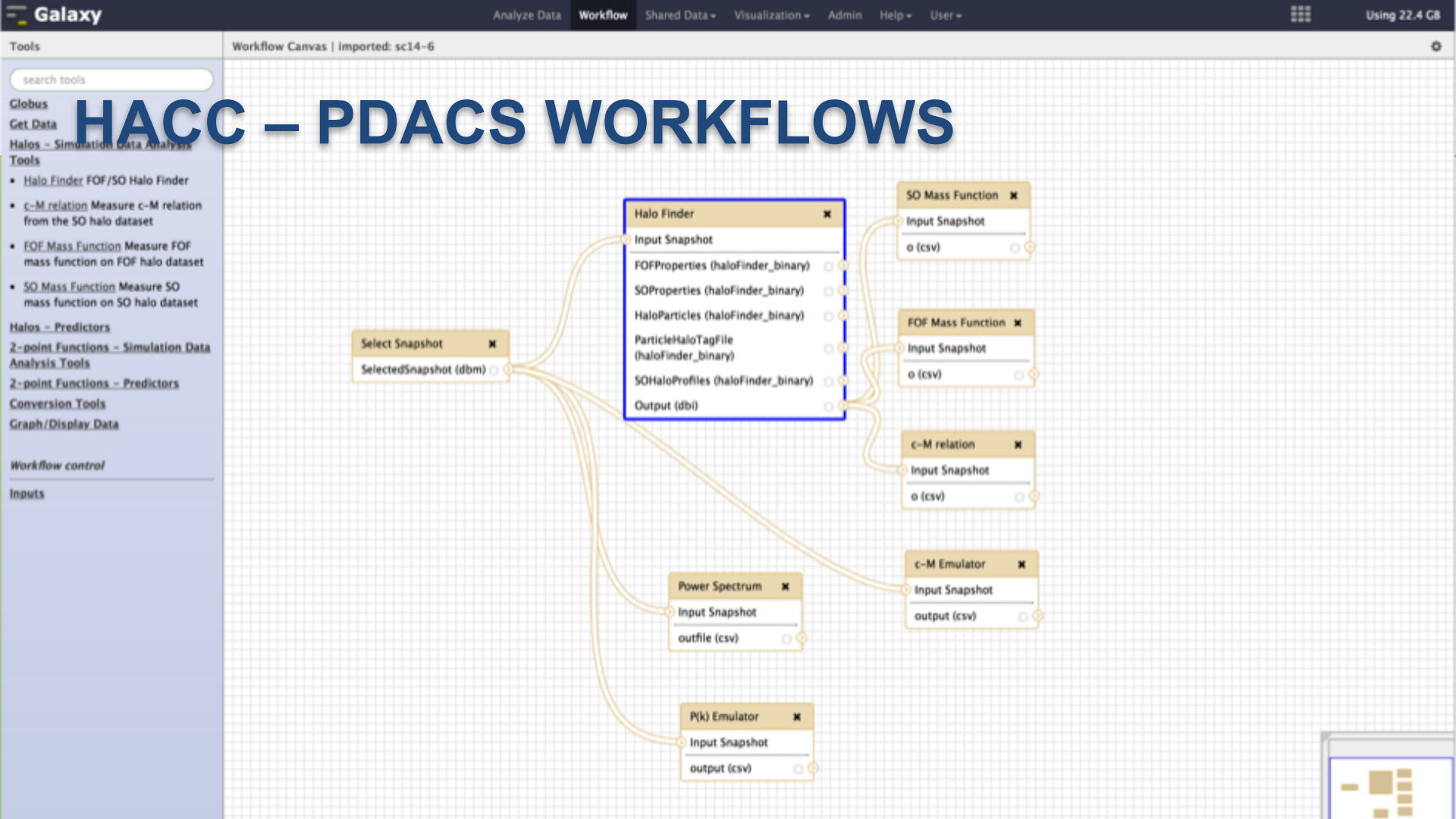
For simulation data that is subject to data reduction, analyzing data in-situ represents the only opportunity to analyze the data before it is reduced and written to storage.

# HACC – PDACS WORKFLOWS

- Analysis tools are wrapped as strongly-typed modules
- Users construct workflows from modules using a graphical tool
- Workflows are executed on resources according to underlying configuration
- PDACS instances running against resources at Argonne, Oak Ridge, and NERSC
- Benefit: Empowers community to analyze large-scale simulation data, without requiring them to own the large storage and compute resources required to do so
  - They also don't have to write the analysis tools



# HACC – PDACS WORKFLOWS



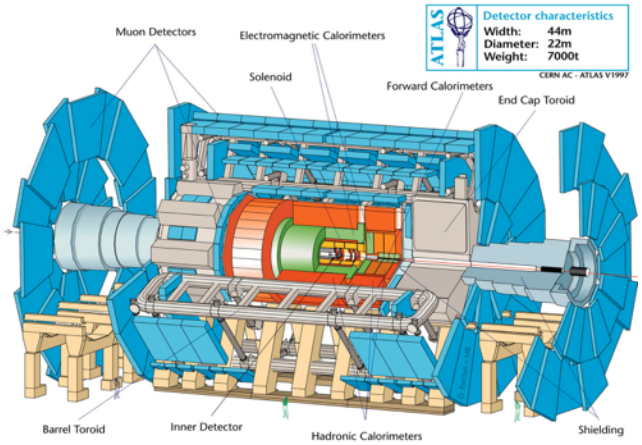
# HACC SUMMARY

Workflow applicable in separate areas

- Simulations running at multiple sites
- Analysis in-situ with simulation
- Data transfer between sites (Globus Online very helpful)
- Analysis in post-processing at simulation site and beyond
- Level 3 data products available through web-based analysis platform

# HIGH ENERGY PHYSICS EXAMPLE

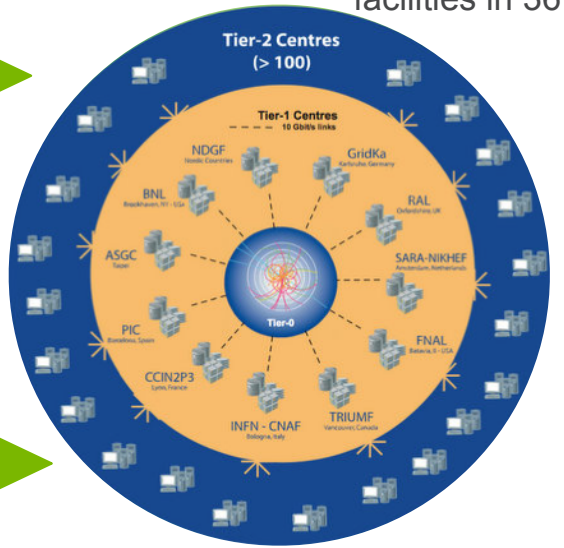
ATLAS detector at LHC




*Experimental data (>25PB/year)*

LHC Computing Grid (analysis)

>170 computing facilities in 36 countries




ALCF


*Simulation data*



Argonne NATIONAL LABORATORY

# HIGH ENERGY PHYSICS EXAMPLE

- The ATLAS experiment uses more than 1 billion compute hours per year
- This consists of integration, event generation, showering, simulation, reconstruction, and analysis
- This (very loosely coupled) workflow propagates step-by-step through the ATLAS production system based on requests from users.
- Simulation with Geant4 accounts for 60% of ATLAS's computing. We wanted to leverage Mira to offload some of this computing, and chose to target event generation (not simulation) as a first step. This task required coordination of multiple workflow-like steps:
  - fetching job descriptions from ATLAS (manual process)
  - serial phase-space grid integration on local cluster
  - transfer of data from local cluster to ALCF
  - large-scale event generation on Mira, based on incoming integration grids
  - transfer of data from ALCF to local cluster
  - post-processing of output data to exportable format

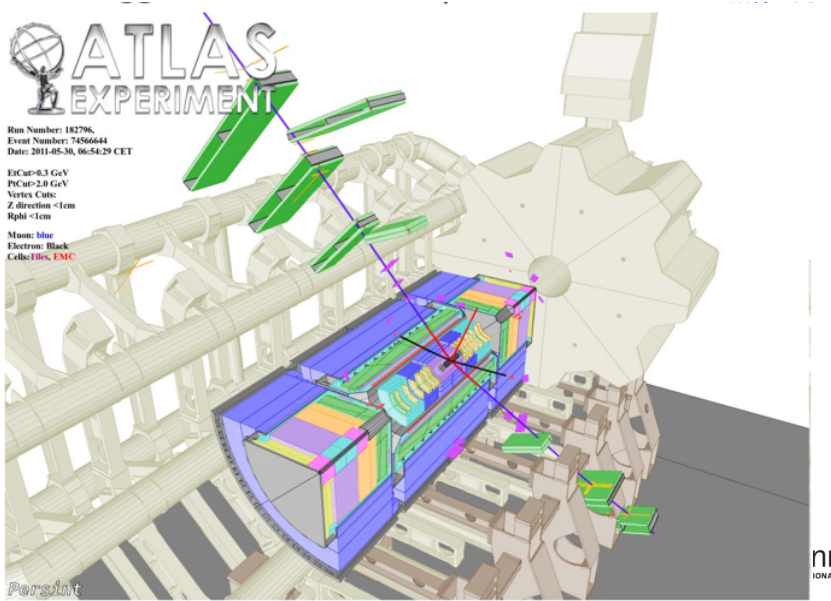
See Tom LeCompte's talk from ATPESC 2015
http://extremecomputingtraining.anl.gov/files/2015/08/LeCompte_HEP-Code-and-Lessons.pdf

# EXAMPLE: HEP EVENT GENERATION

Monte Carlo-based generation of particle collision events such as occur in the ATLAS detector at the Large Hadron Collider, using Alpgen.

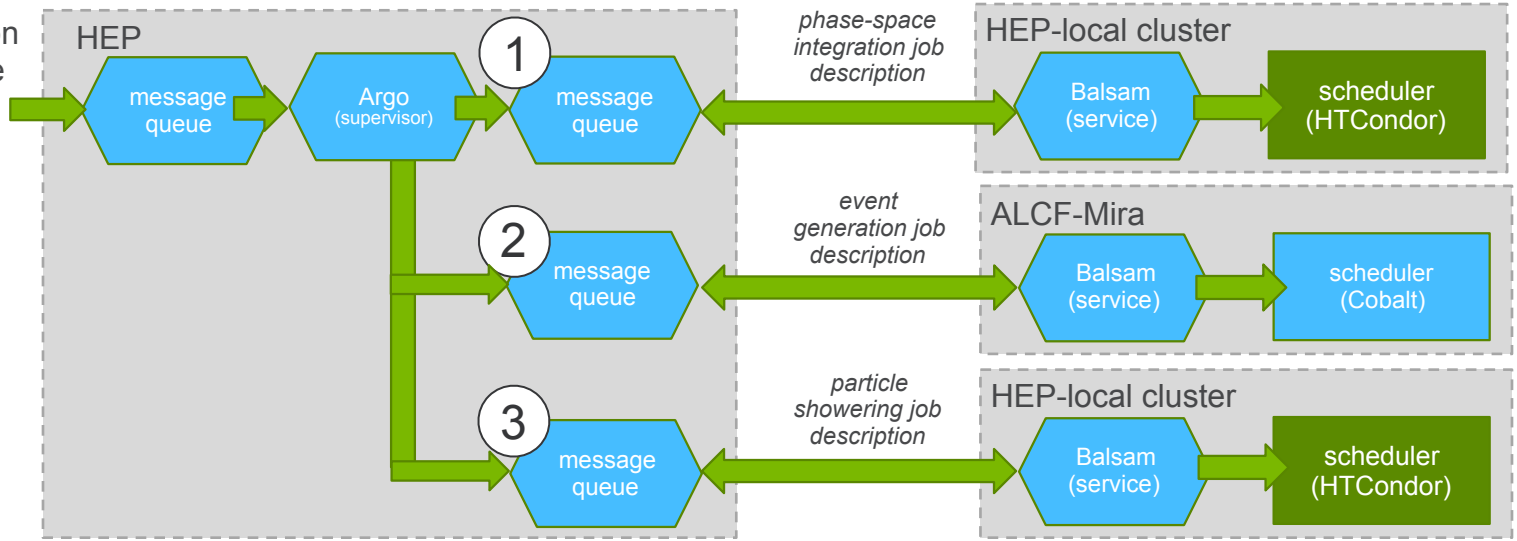
Consists of three stages:

1. Generation of phase-space integration grid
2. Generation of weighted events
3. Unweighting of events



# ARGO ENABLES MULTI-RESOURCE EVENT GENERATION WORKFLOW

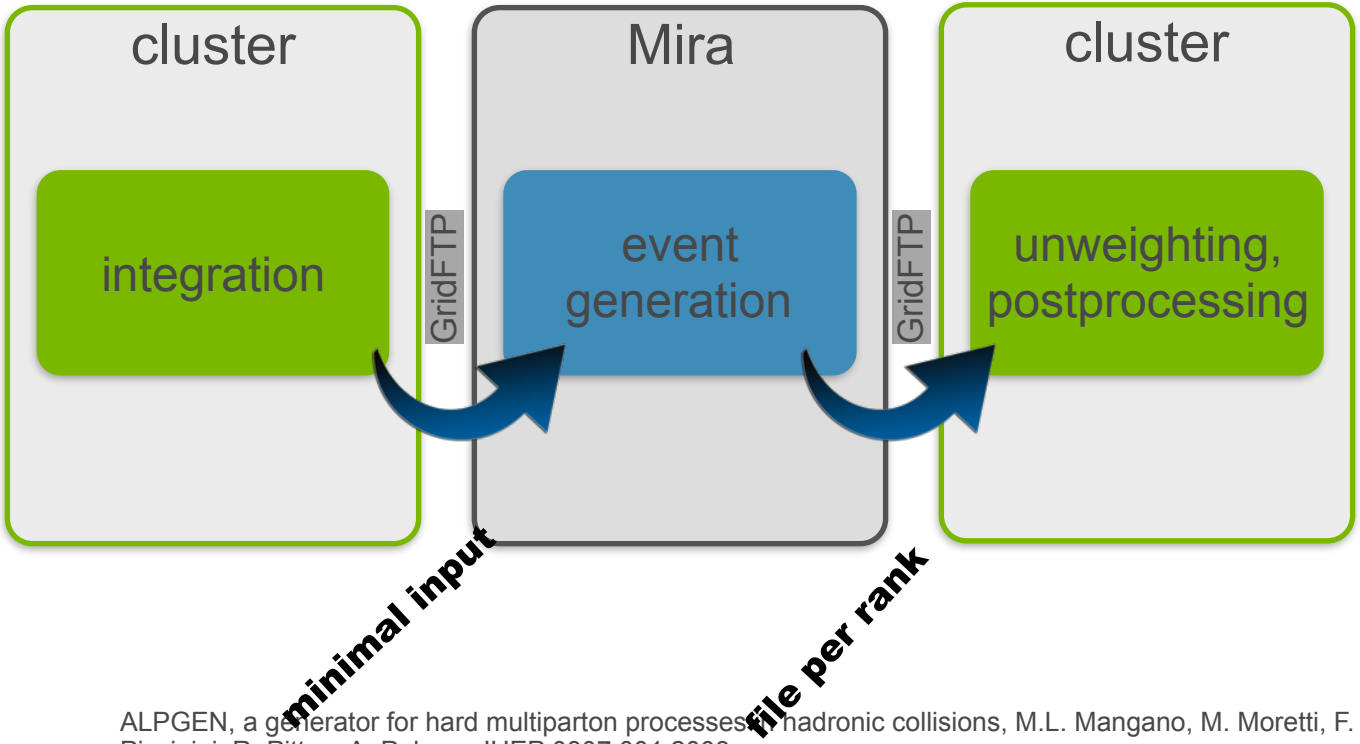
Job description messages are placed on the incoming message queue*




Final events are transferred manually to ATLAS; this could be automated in future

* Note: To date, all jobs have been injected manually; integration with external services (e.g. PanDA) is incomplete

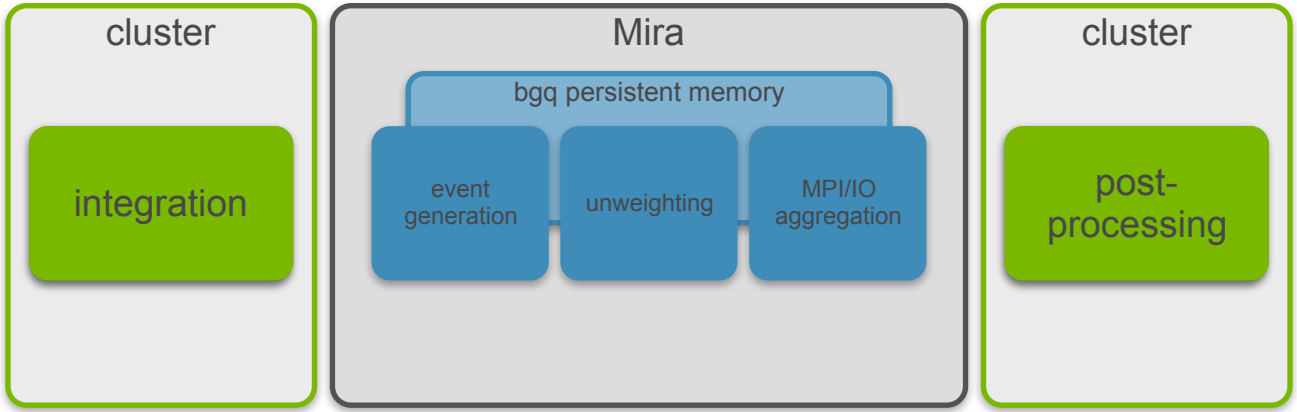
# EVENT GENERATION ON MIRA



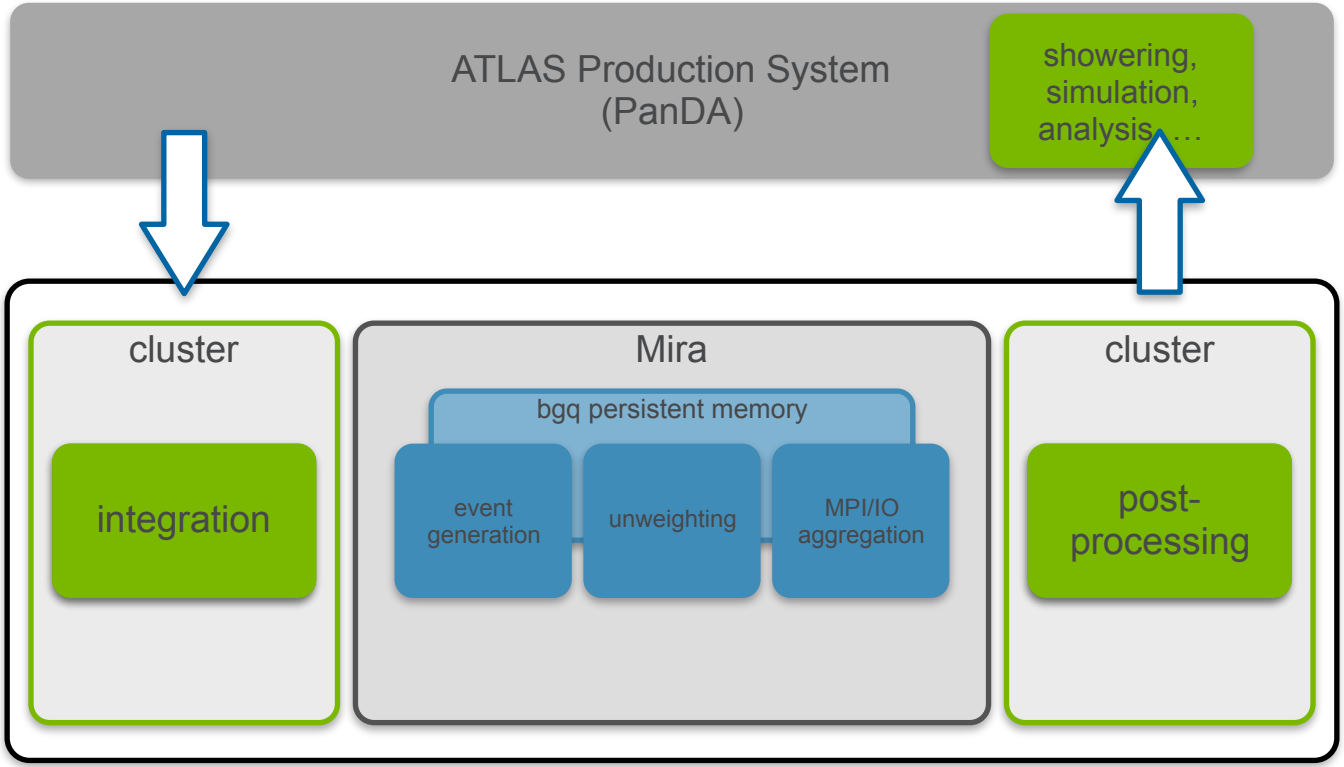

ALPGEN, a generator for hard multiparton processes in hadronic collisions, M.L. Mangano, M. Moretti, F. Piccinini, R. Pittau, A. Polosa, JHEP 0307:001,2003

# EVENT GENERATION ON MIRA

- Combine multiple application invocations in single script job
- Use persistent memory for exchanging data between invocations
- Aggregate data from persistent memory to filesystem



# EVENT GENERATION ON MIRA



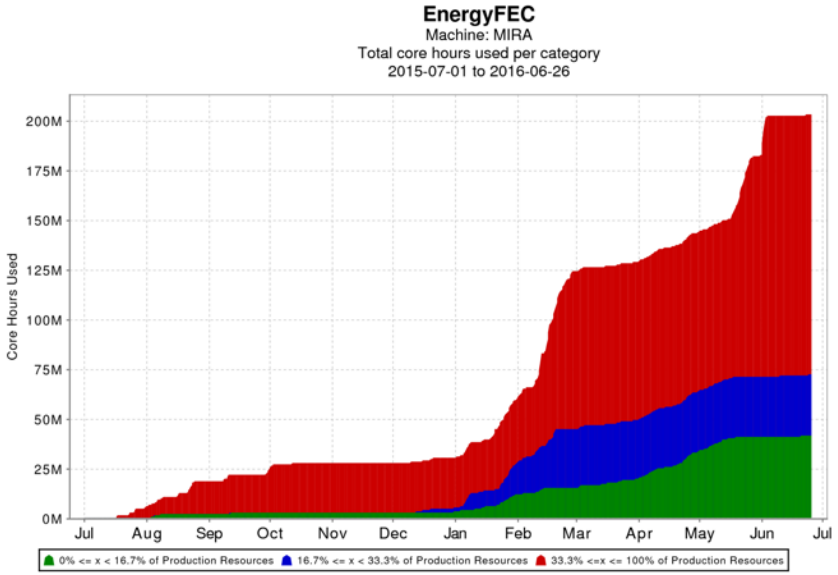
# COMPUTE-NODE PERSISTENT MEMORY

- Environment variable BG_PERSISTMEMSIZE sets MB of memory to use as ramdisk, mounted at /dev/persist
- Persistent memory persists for lifetime of (qsub) job
  - Specify BG_PERSISTMEMSIZE to runjob command
  - If persistent memory size changes between runjob executions, memory will be available but it will be cleared
- Caveat: Persistent memory clearly reduces the amount of memory available for the simulation; if you are operating near the limit of node memory, this solution is not for you

For more details of persistent memory, see
http://www.redbooks.ibm.com/redbooks/pdfs/sg247948.pdf

# RESULTS

- **70M hours used for ATLAS event generation through Argo**
- Unprecedented rates of event generation (largest event generation jobs on Mira requested >1T events)
- More complex/rare events than could be produced on the Grid
- Mira has become the primary site for ATLAS event generation
- 70 million compute hours have been offloaded from the WLCG to Mira, freeing this time to be used for other purposes
- Events that would have taken years to produce on the Grid were generated within a couple months, accelerating the simulation and analysis pipeline and therefore publications



# SUMMARY

- Workflows are inevitable in computational science
- Small-scale workflow management should be easy and is normally managed using scripting
- Large-scale workflow management can benefit from a workflow system
  - In the transition from small to large workflows, you might build your own workflow system
- Data curation has a similar growth pattern
  - How will you manage your data today so you can find it when you need it in N years?