WORKFLOW MANAGEMENT

Thomas Uram
Argonne Leadership Computing Facility

Argonne NATIONAL LABORATORY

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

WHAT IS A WORKFLOW?

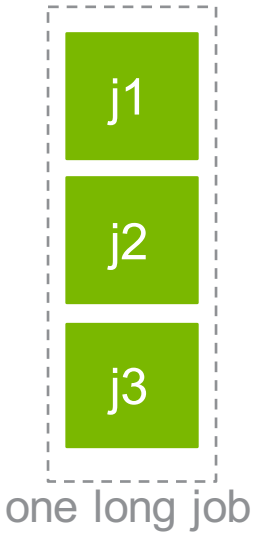
- It depends who you ask
- Basically a collection of jobs to be run
  - Could be a sequence of individual jobs
  - Could be a sequence of varying numbers of jobs
- Available means of describing and running workflows
  - Script jobs
  - Job dependencies
  - Ensemble jobs
  - In situ
  - Custom workfows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

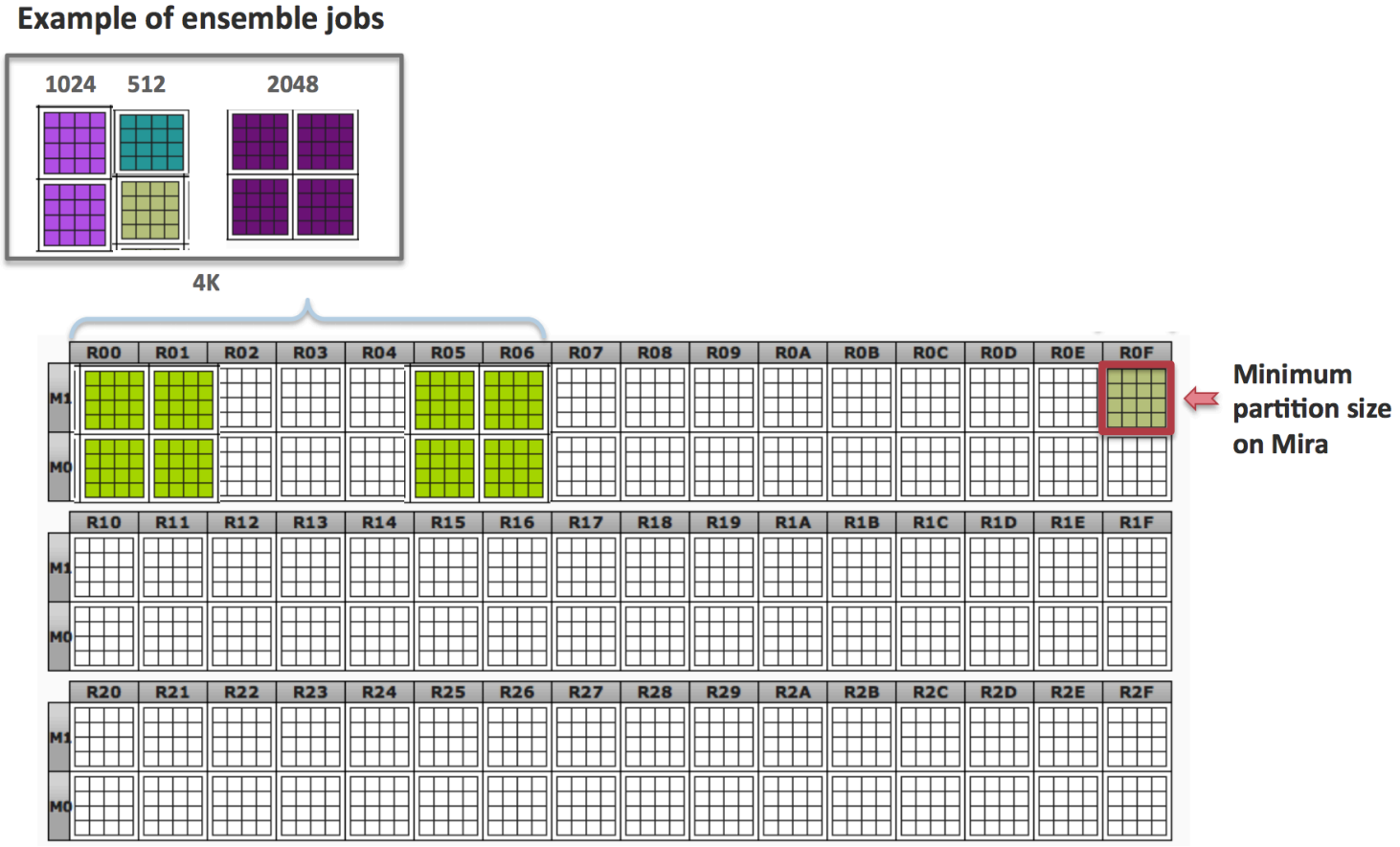
multiple independent short jobs

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

```
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



Argonne NATIONAL LABORATORY

WORKFLOW VIA ENSEMBLE JOBS: RUN BIGGER JOBS

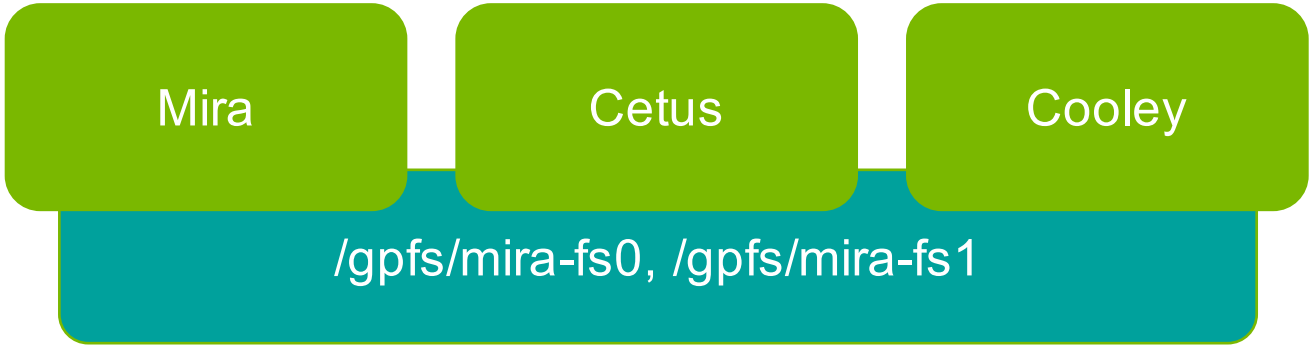
http://trac.mcs.anl.gov/projects/cobalt/wiki/BGQUserComputeBlockControl

```
#!/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
```

See Paul Rich's talk from Monday re: Ensemble 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 run on Cooley without needing to transfer the data.

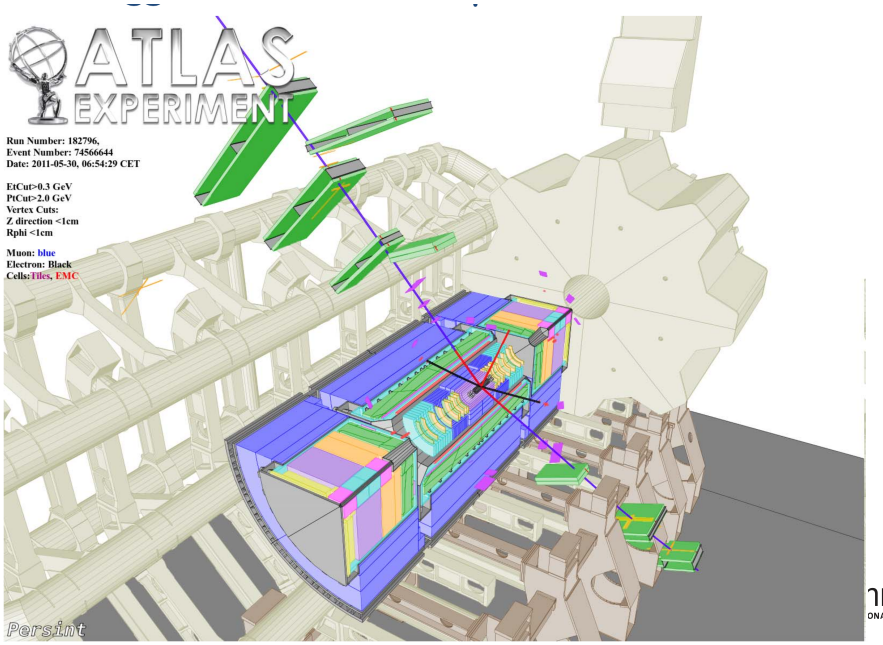


EXAMPLE: HEP EVENT GENERATION

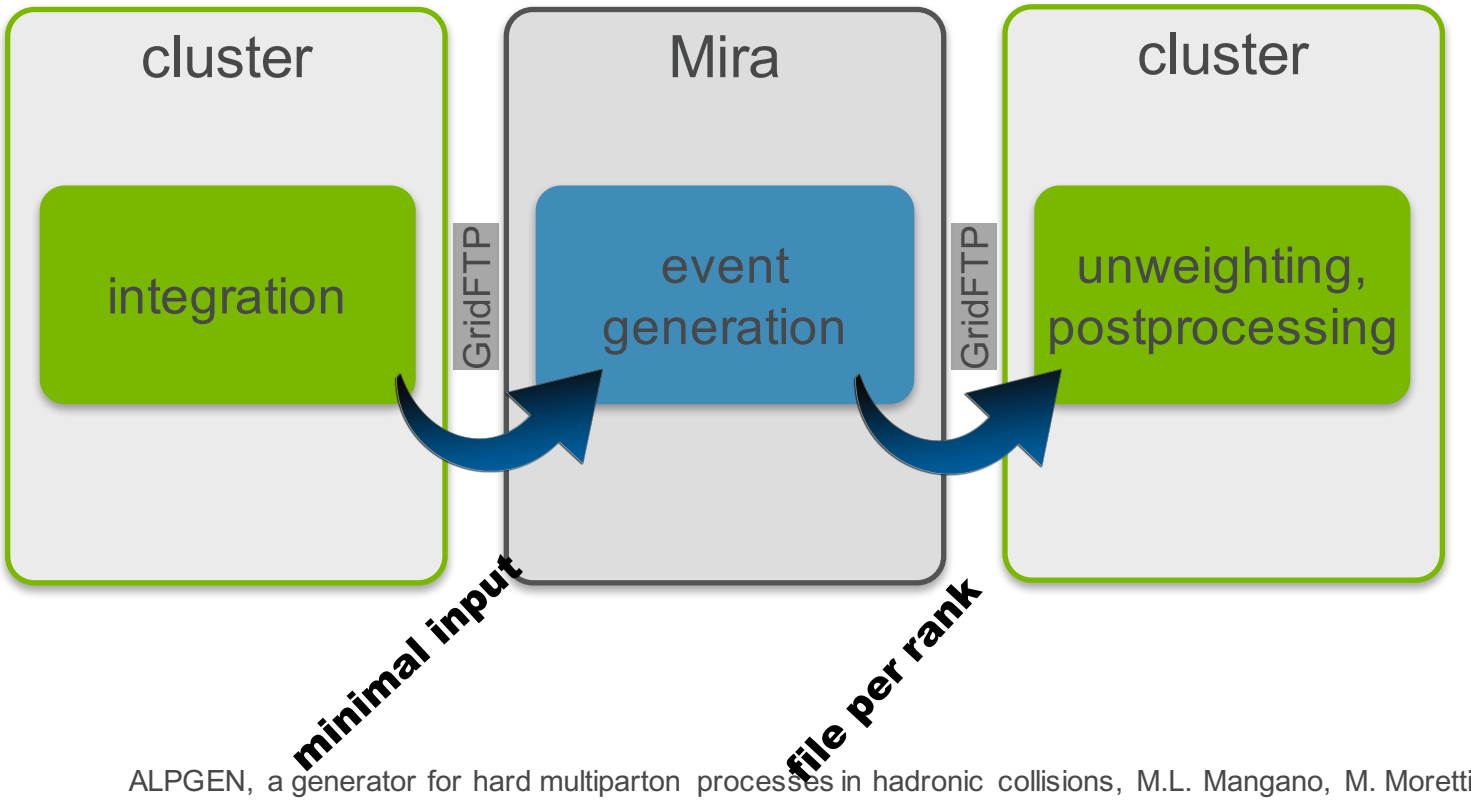
Monte Carlo-based generation of particle collision events such as occur in the ATLAS detector at the Large Hadron Collider, using Alpgen.

Consists of three stages:

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



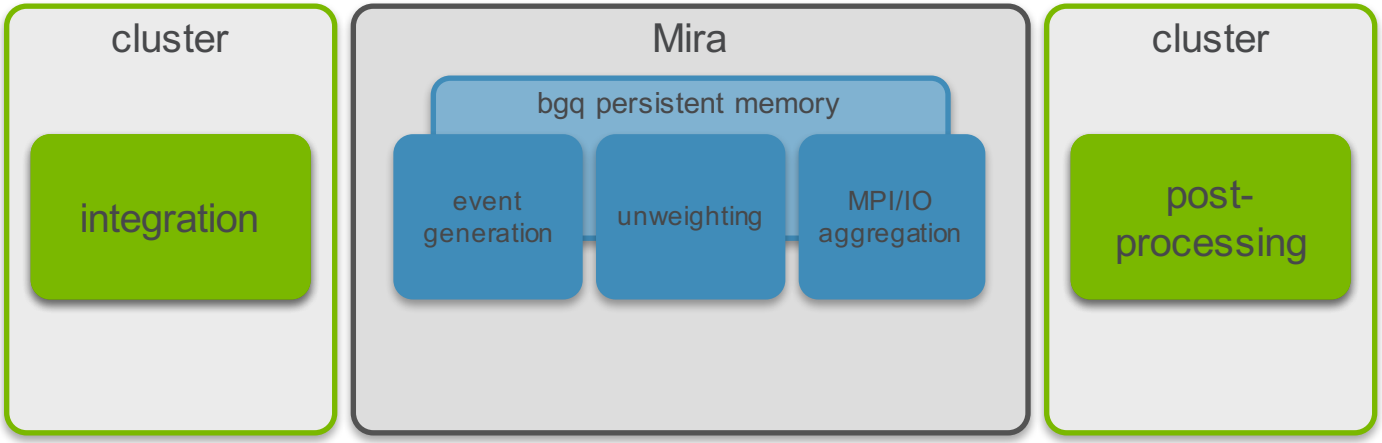
EXAMPLE: HEP EVENT GENERATION



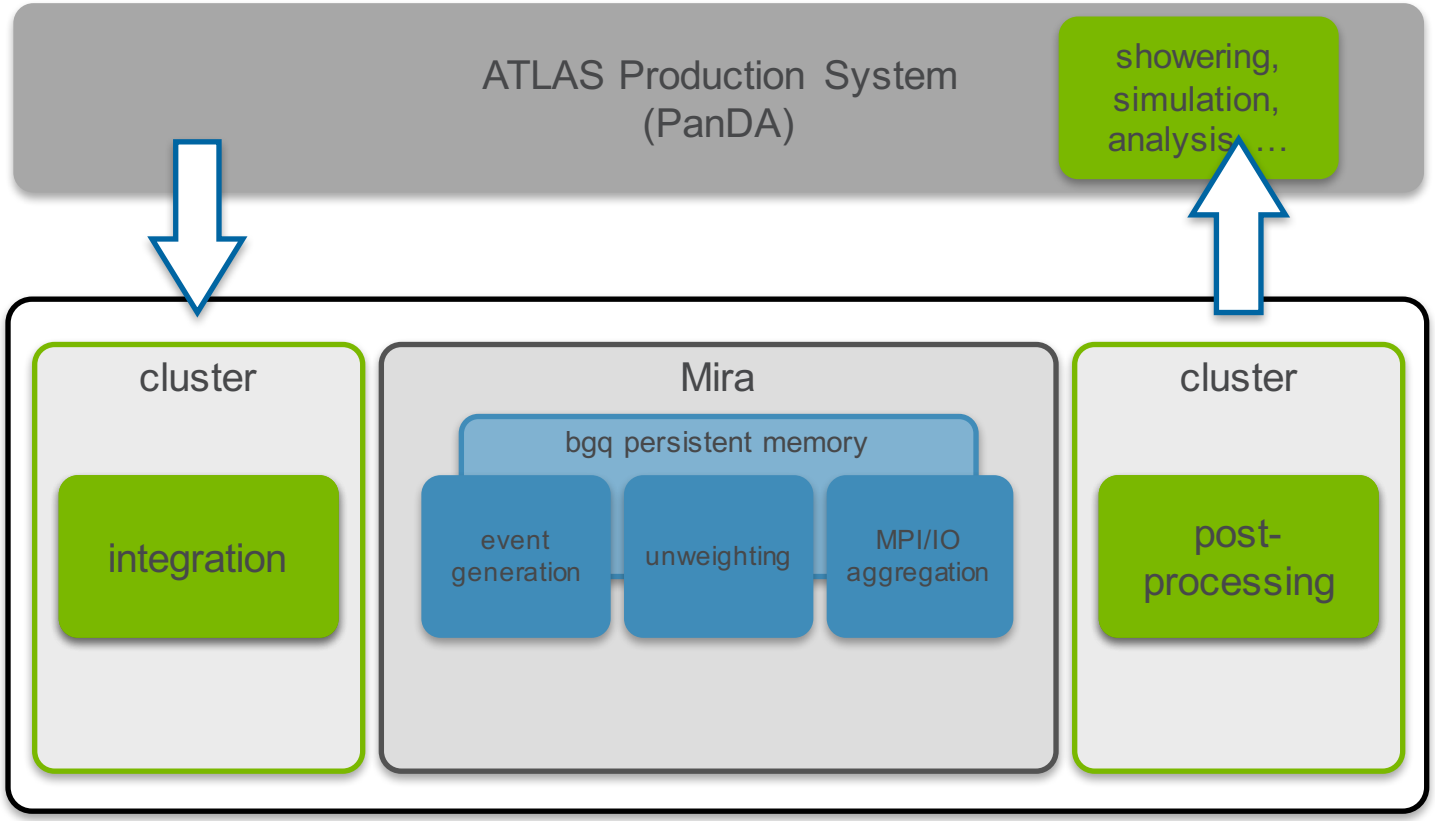
ALPGEN, a generator for hard multiparton processes in hadronic collisions, M.L. Mangano, M. Moretti, F. Piccinini, R. Pittau, A. Polosa, JHEP 0307:001,2003

EXAMPLE: HEP EVENT GENERATION

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



EXAMPLE: HEP EVENT GENERATION



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 info on persistent memory, see
http://www.**redbooks**.**ibm**.com/**redbooks**/pdfs/sg247948.pdf

IN SITU ANALYSIS WORKFLOW

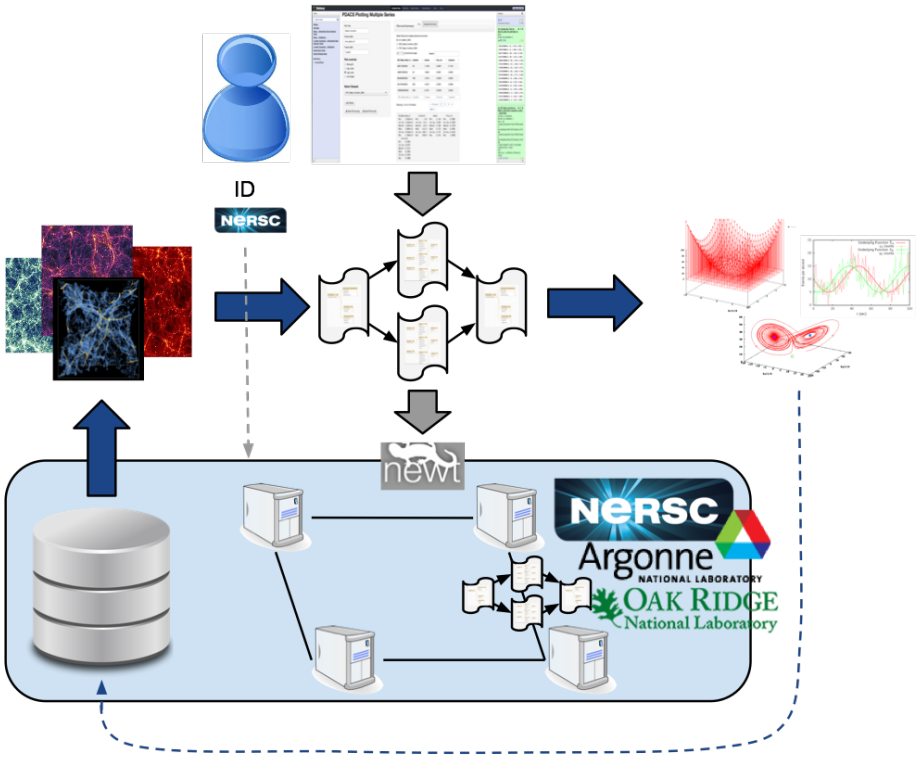
- In situ == running analysis inline with the simulation
- In situ analysis has several advantages
  - Operates on data already in memory; during postprocessing, reading large data from the filesystem can be expensive
  - Is able to perform analysis at the same scale as simulation
  - Operates on full 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

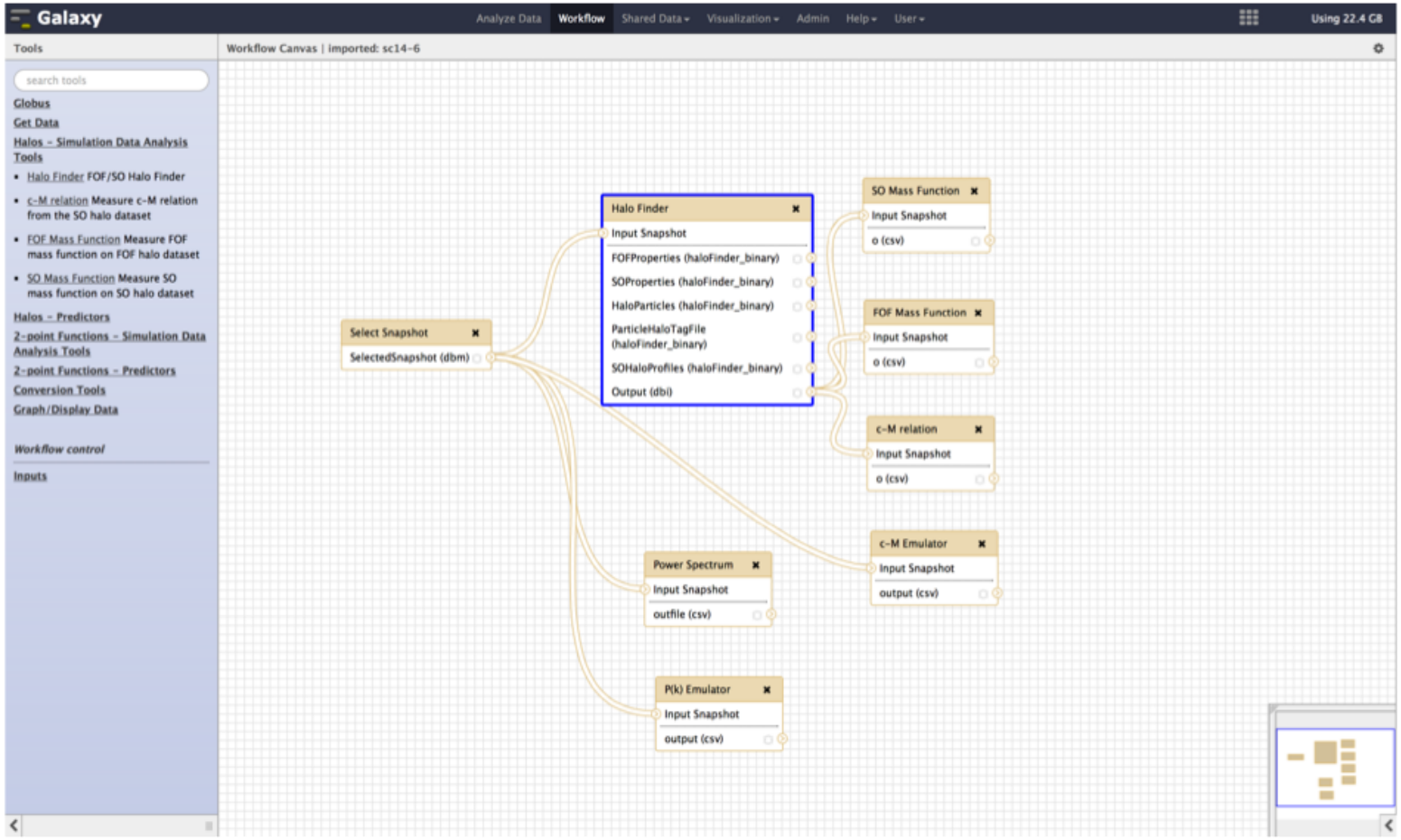
- Cosmotools is a configurable in-situ analysis framework
  - Simulation configuration specifies analyses that should be performed between time steps during the simulation, and their parameters
  - On execution, particle data is passed from the simulation to analysis routines for processing
  - When 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 (long) simulation runs

HACC – PDACS WORKFLOWS

- Analysis tools 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



HACC – PDACS WORKFLOWS

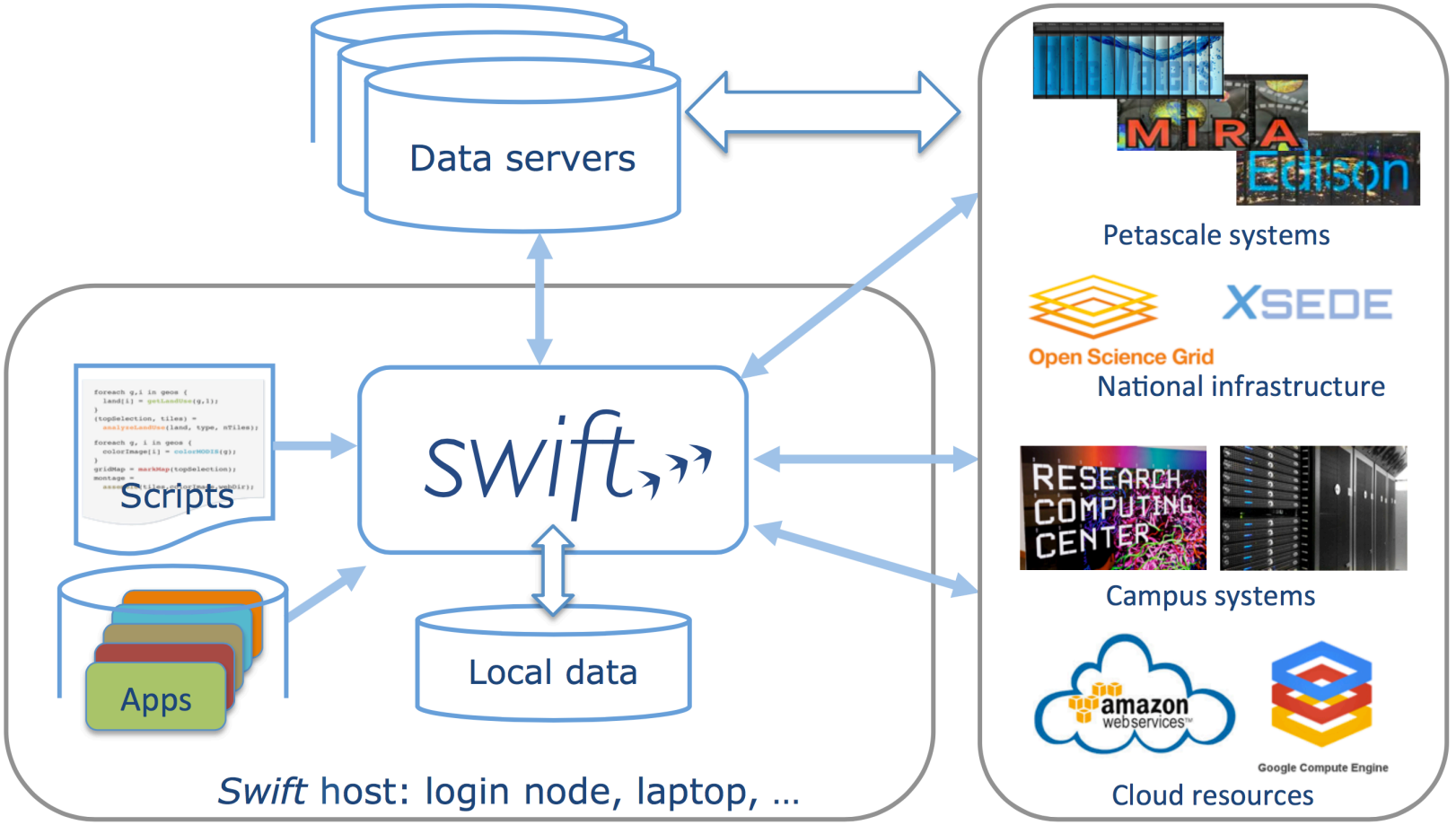


Argonne NATIONAL LABORATORY

WORKFLOW SOFTWARE TO CONSIDER

- Swift (http://swift-lang.org)
  - **C-like language** for expressing workflows by wrapping command-line applications or function calls
  - **Execution engine** for translating workflows into job submissions on compute resources, maximizing concurrency by exploiting data (in)dependencies
  - **(Not the Apple product)**
- Fireworks (https://github.com/materialsproject/fireworks)
  - Infrastructure for defining and running workflows

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

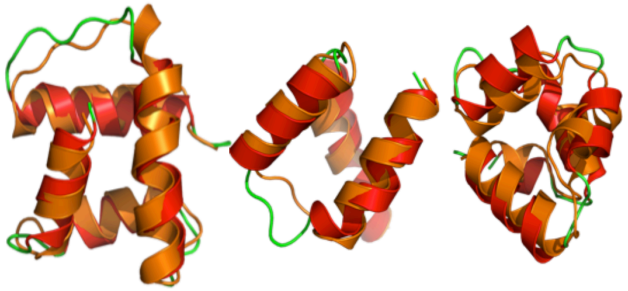


See https://www.alcf.anl.gov/swift

When do you need HPC workflow?

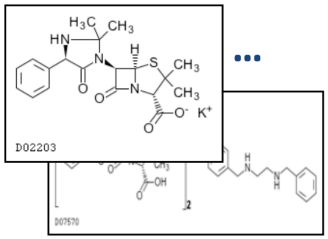
Example application: protein-ligand docking for drug screening

O(10) proteins implicated in a disease

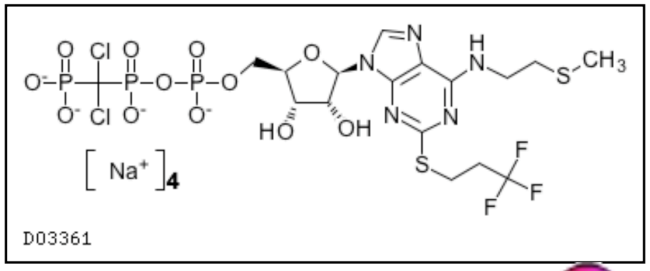


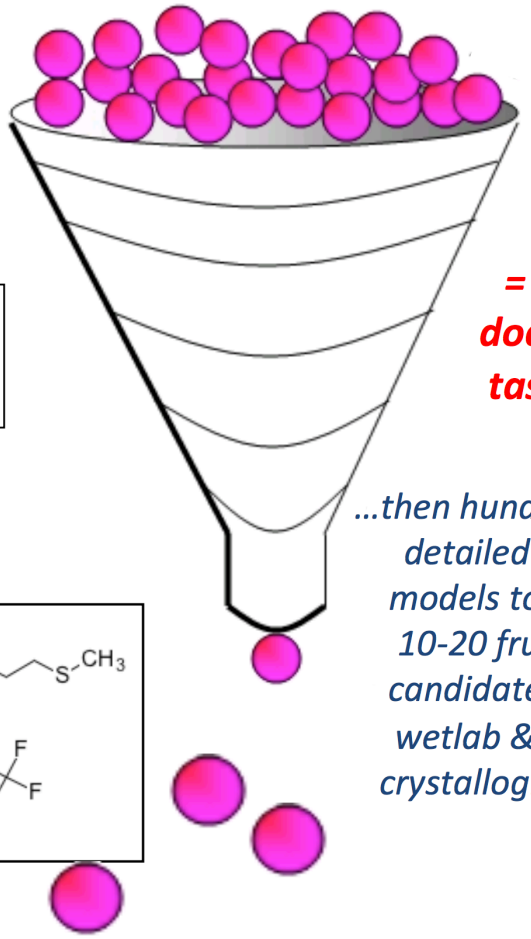
X

O(100K) drug candidates



...





= 1M docking tasks…

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

Argonne NATIONAL LABORATORY

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:

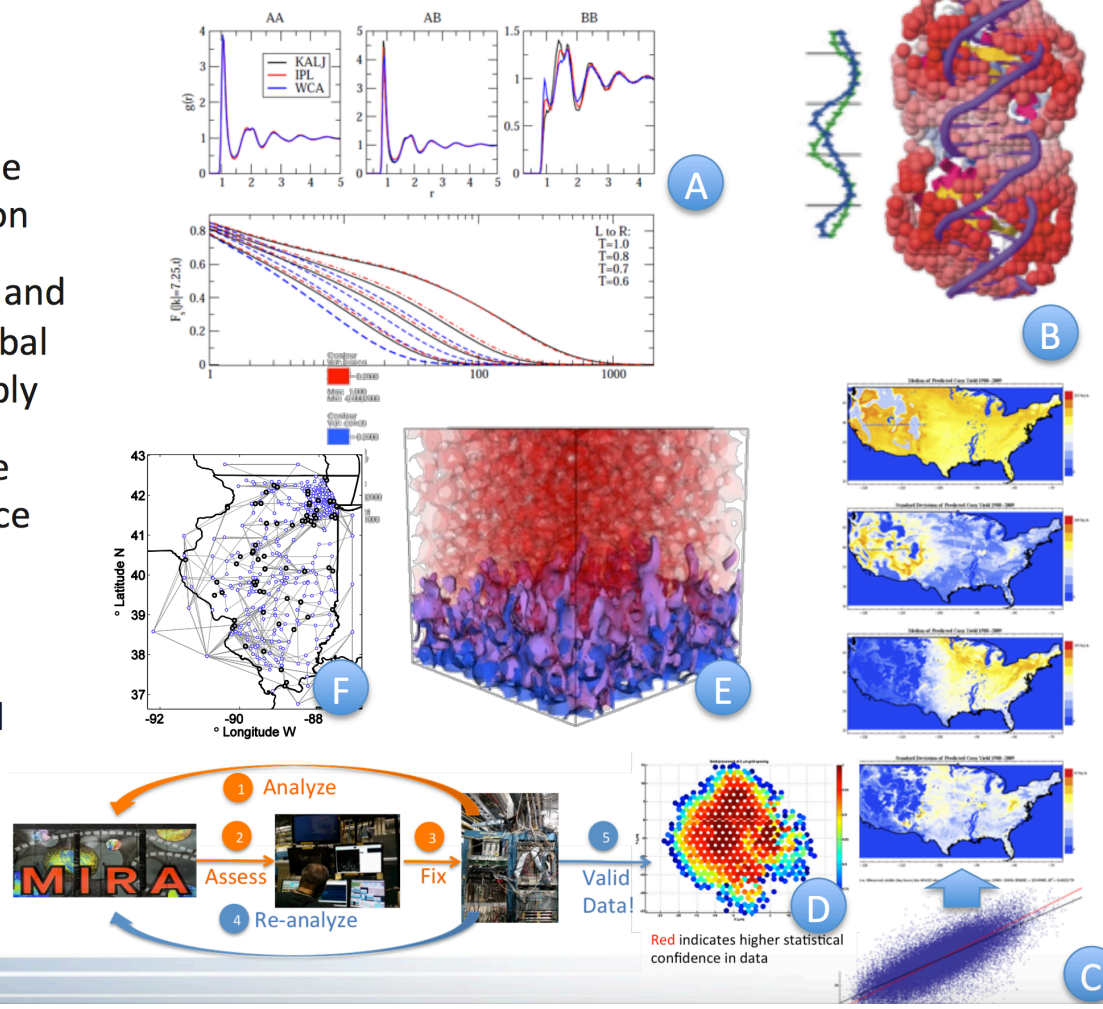
```
swift –site tukey,blues dock.swift
```

Argonne NATIONAL LABORATORY

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

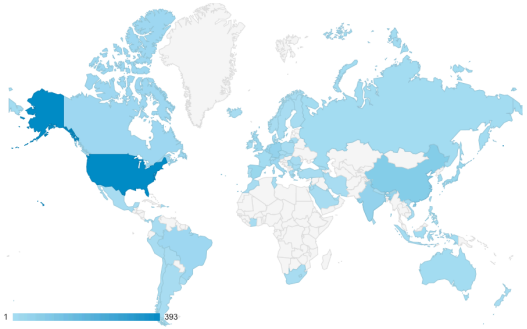
All have published science results obtained using Swift



Argonne NATIONAL LABORATORY

FireWorks v1.3.1

FireWorks is an application-agnostic workflow software for defining and executing large numbers of calculations



Usage and outreach:

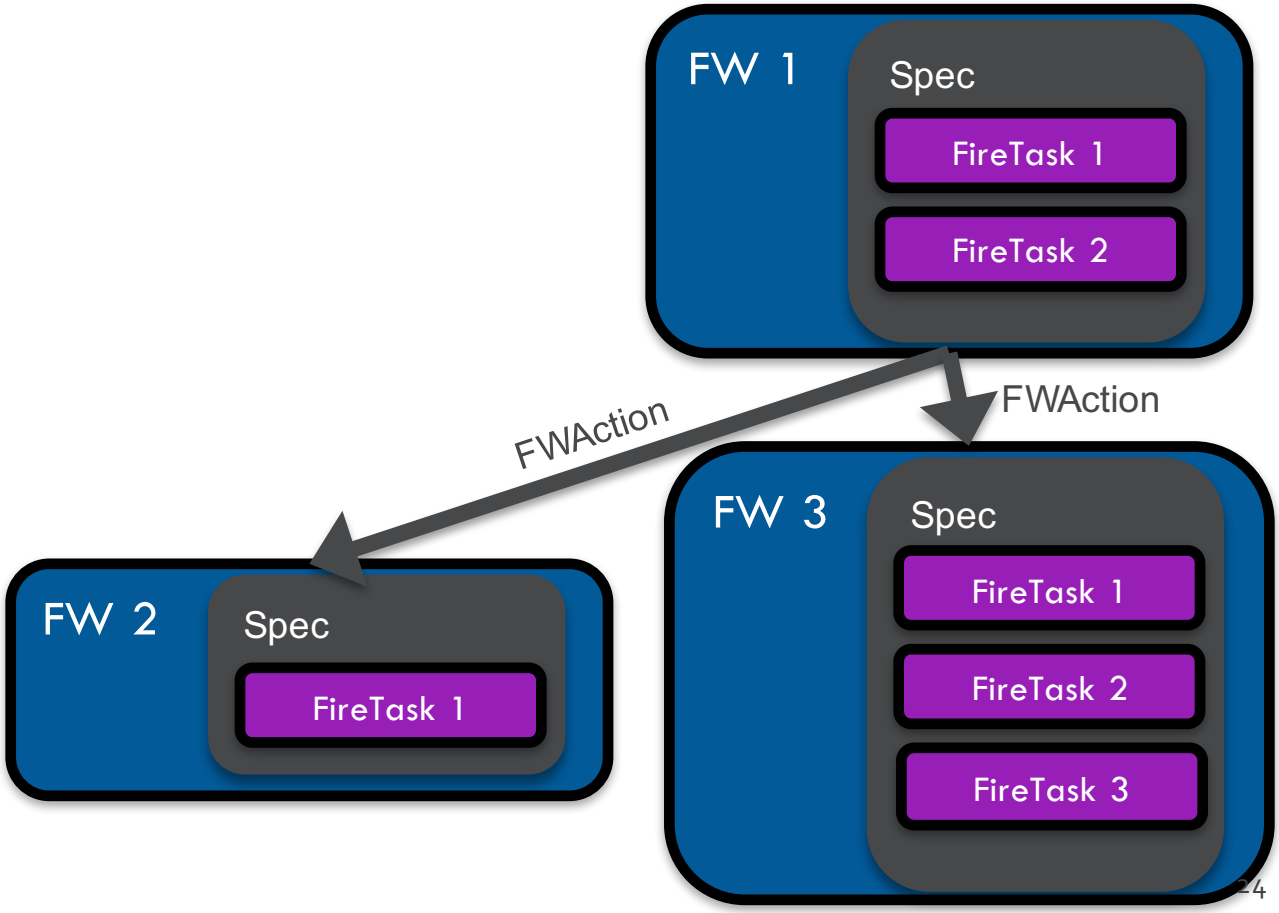
- Built w/Python+MongoDB. Open-source, pip-installable: http://pythonhosted.org/FireWorks/
- >7500 downloads per month
  - #1 Google hit for "Python workflow software"
  - #4 software hit for "scientific workflow software"
- 1 of 2 workflow softwares officially supported by NERSC
- Worldwide usage
- 97th percentile, scientific Python software impact (Depsy) paper published in C.S. journal

Major new features:

- Completely redesigned frontend,
- Auto-reporting compiled using Mongo aggregation framework
- Compatibility with Argonne Leadership Computing Facility scheduler (Cobalt)
- Major features such as task-level restart contributed by community

Argonne NATIONAL LABORATORY

HOW WORKFLOWS ARE STRUCTURED



CODE EXAMPLE

```
from fireworks import Firework, Workflow, LaunchPad, ScriptTask
from fireworks.core.rocket_launcher import rapidfire

# set up the LaunchPad and reset it (first time only)
launchpad = LaunchPad()
launchpad.reset('', require_password=False)

# define the individual FireWorks and Workflow
fw1 = Firework(ScriptTask.from_str('echo "To be, or not to be,"'))
fw2 = Firework(ScriptTask.from_str('echo "that is the question:"'))
wf = Workflow([fw1, fw2], {fw1:fw2}) # set of FWs and dependencies

# store workflow in LaunchPad
launchpad.add_wf(wf)
# pull all jobs and run them locally
rapidfire(launchpad)
```

WORKFLOWS ARE SIMPLE JSON/YAML DOCUMENTS THAT HAVE VERY LITTLE "FLUFF"

(this is YAML, a bit prettier for humans
but less pretty for computers)

```
fws:
- fw_id: 1
  spec:
    _tasks:
    - _fw_name: ScriptTask:
      script: echo 'To be, or not to be,'
- fw_id: 2
  spec:
    _tasks:
    - _fw_name: ScriptTask
      script: echo 'that is the
question:'
links:
  1:
  - 2
metadata: {}
```

The same JSON document will produce the same result on any computer (with the same Python functions).

JSON + MONGODB MEANS YOU CAN STORE WORKFLOWS DIRECTLY AND MAKE RICH QUERIES

(this is YAML, a bit prettier for humans but less pretty for computers)

```
fws:
- fw_id: 1
  spec:
    _tasks:
    - _fw_name: ScriptTask:
      script: echo 'To be, or not to be,'
- fw_id: 2
  spec:
    _tasks:
    - _fw_name: ScriptTask
      script: echo 'that is the
question:'
links:
  1:
  - 2
metadata: {}
```

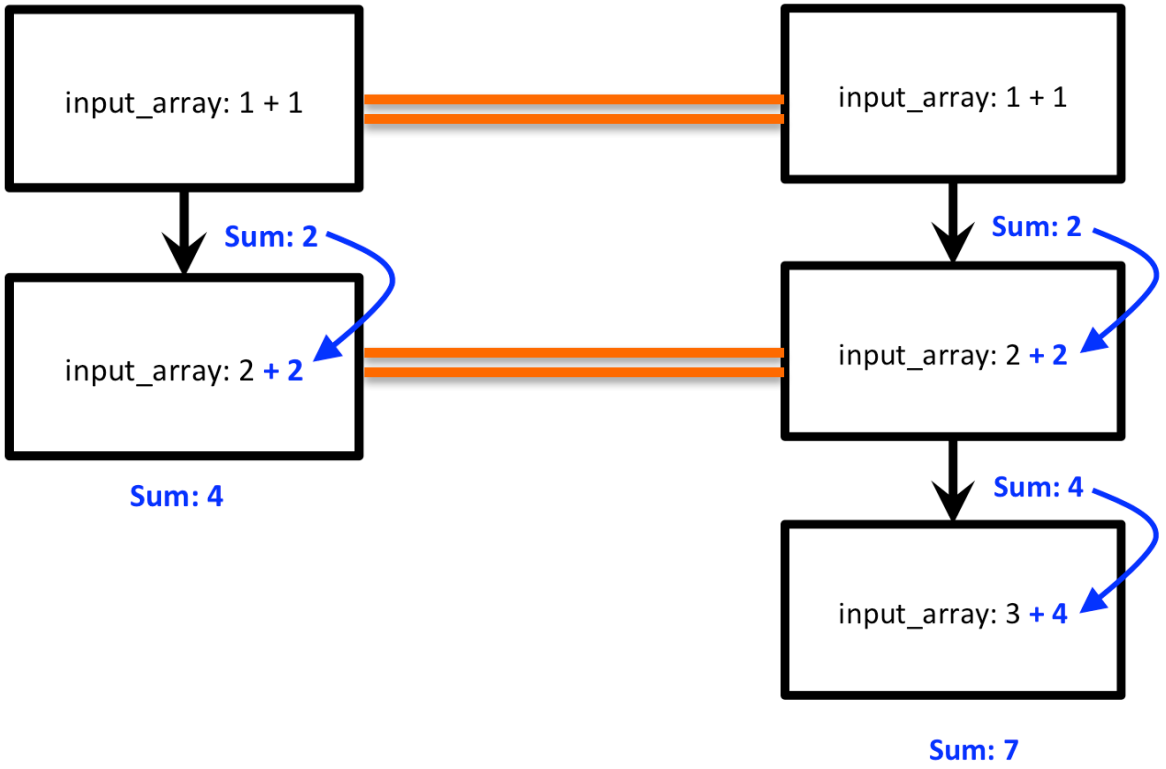
Just some of your search options:

- simple matches
- match in array
- greater than/less than
- regular expressions
- match subdocument
- Javascript function
- MapReduce…

All for free, and all on the native workflow format!

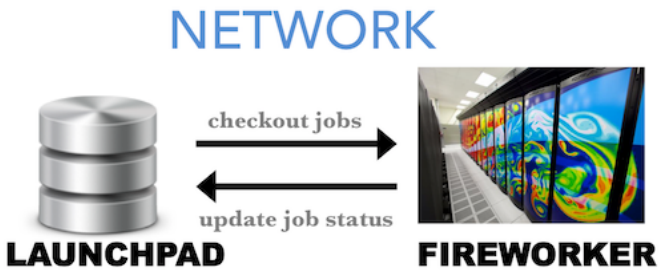
Argonne NATIONAL LABORATORY

JSON DUPLICATE CHECKING SIMPLE AND AUTOMATIC



"PULL" STRUCTURE (NOT PUSH) MAKES LIFE EASY: EXECUTION MODES ARE ALL VARIATIONS ON A THEME

- Theme: Worker machine pulls a job & runs it



- Variation 1:
  - different workers can be configured to pull different types of jobs via config + MongoDB
- Variation 2:
  - worker machines sort the jobs by a priority key and pull matching jobs the highest priority

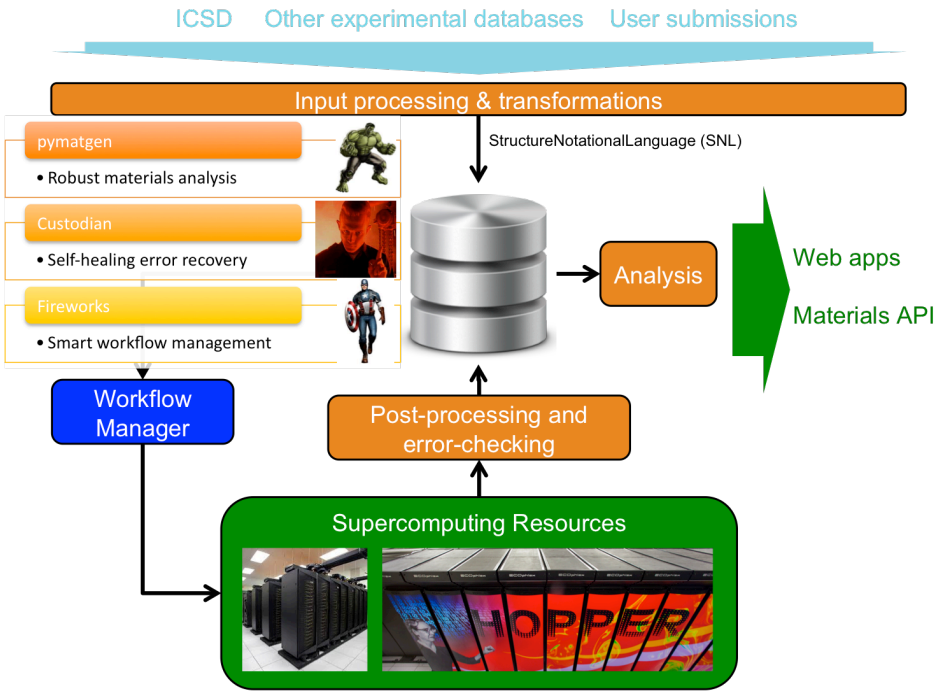
JCESR'S USAGE

JCESR leverages ALCF, MIT, and NERSC resources in a distributed fashion with the high-throughput (HT) infrastructure of the Materials Project to:

- Execute an electronic structure code (VASP, Qchem, Gaussian, AbInit, etc) with reasonable efficiency
- Monitor the calculations using the Fireworks workflow software to enable automatic restarts, saving of intermittent data, etc

Issues:

- ALCF does not host Mongo DB databases requiring users to setup and secure remote instances
- No method for telling a task to run outside the job scheduler even if there's no compute process



Automatic workflow of first-principles calculations

Argonne NATIONAL LABORATORY