SCEC

Simplify Your Science with Workflow Tools

Scott Callaghan
scottcal@usc.edu

2023 IHPCSS
July 11, 2023

I've got a problem.

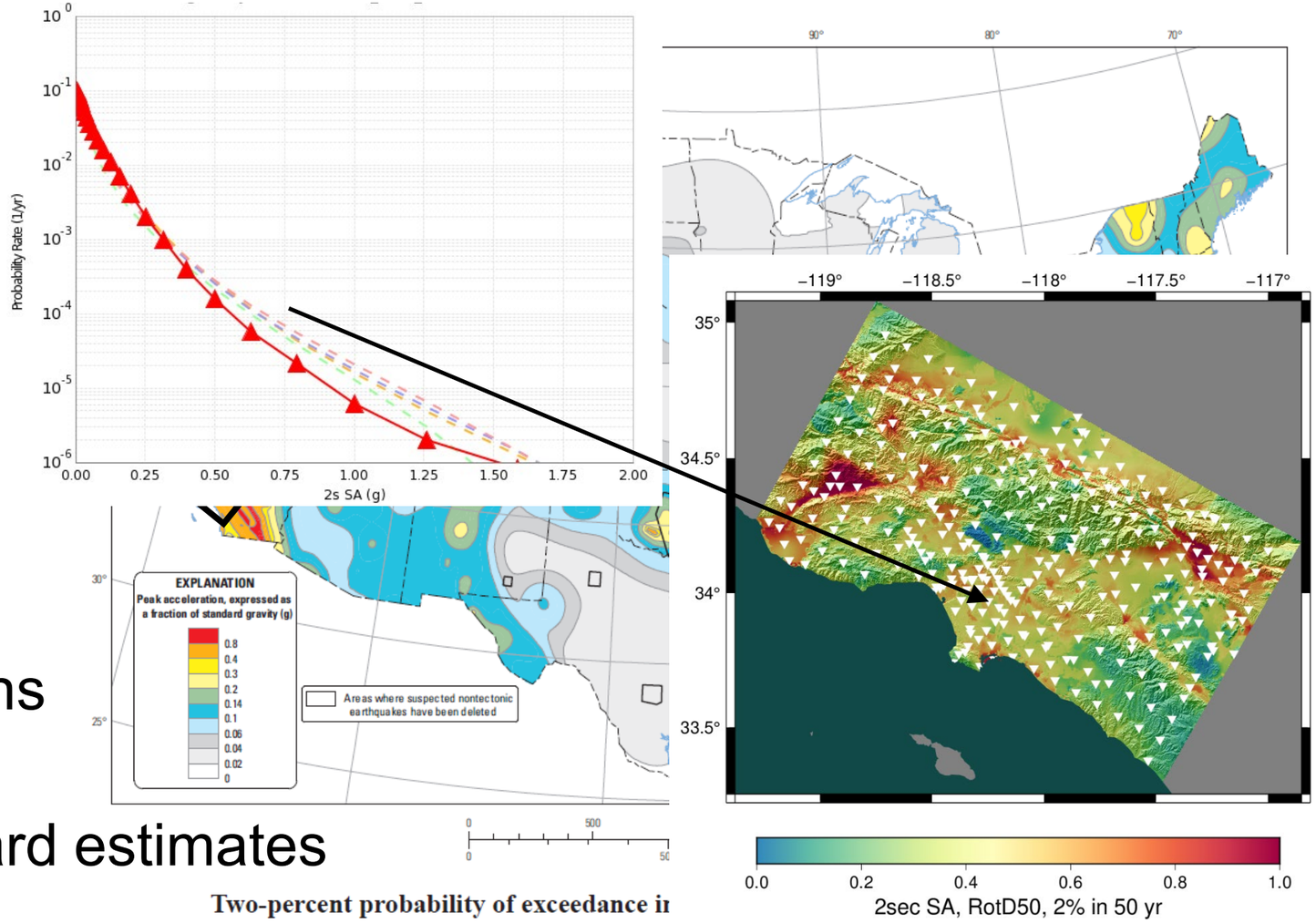
600,000 earthquakes

420 million seismograms

26 million computational tasks

Why am I doing this?

- Want to know expected earthquake shaking over the next 50 years
  - Building codes
  - Insurance rates
  - Disaster planning
- Interested in Southern California, near Los Angeles
- Want to use best available science
  - Run wave propagation simulations of many possible earthquakes
  - Use shaking information for hazard estimates

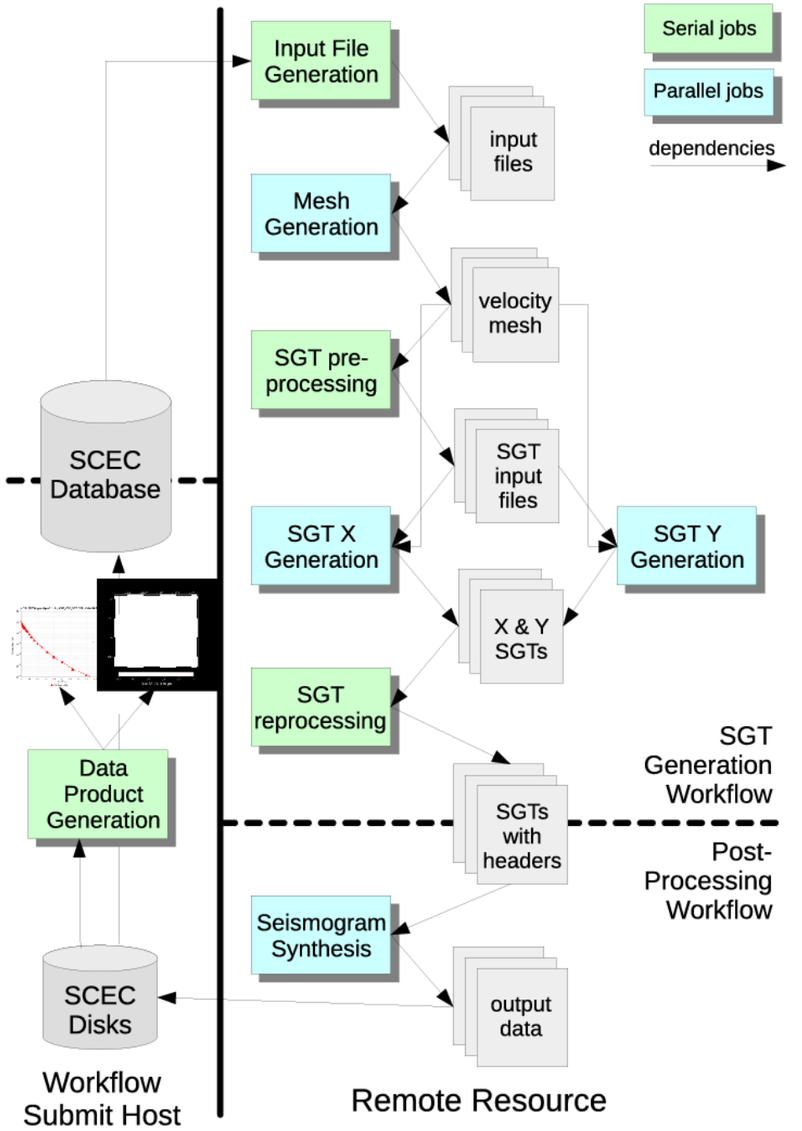


What are my challenges?

- Automation
  - Too many jobs to run by hand
- Data management
  - Millions of input and output files to track
- Job execution
  - Heterogenous job types (serial, parallel, CPU, GPU)
  - Millions of tasks
- Error recovery
  - Resiliency to common problems

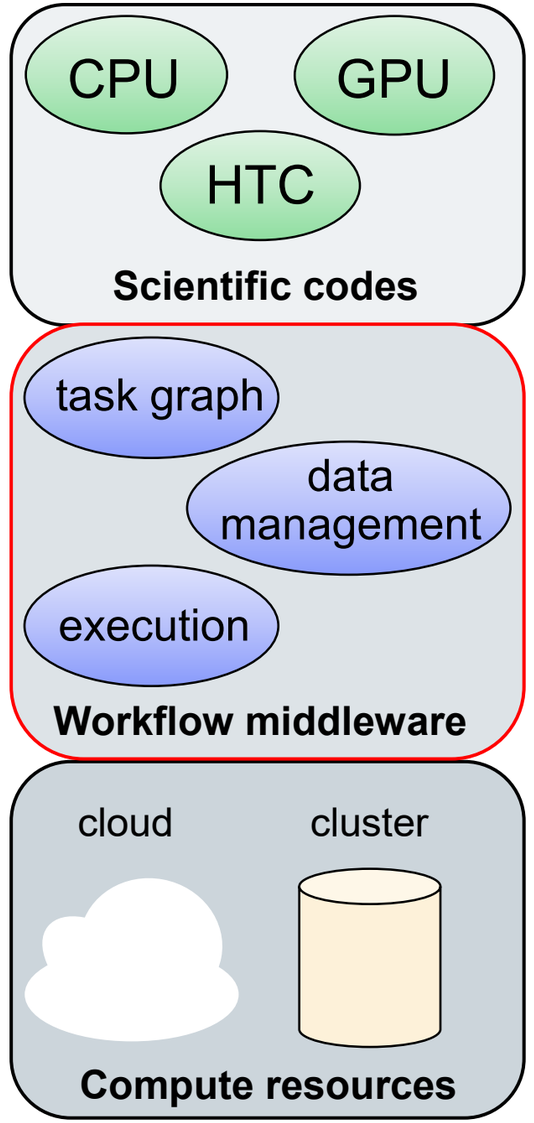
Solution: Scientific Workflows

- Decided to use scientific workflows to manage challenges
- Selected a pair of open-source tools which work together
  - Pegasus-WMS
  - HTCondor
- Express tasks as a directed graph of jobs with dependencies
  - No changes to application codes
- Jobs are queued and executed



How did workflow tools help?

- Automation
  - Jobs are automatically submitted when ready to run
  - Workflow tools able to automate jobs with 2FA
- Data management
  - Input and output files are copied automatically
- Job execution
  - Support remote, distributed execution of heterogeneous jobs
  - Workflow has run on 11 systems since 2007
- Error recovery
  - Failed jobs retried
  - Workflow checkpointed if jobs keep failing

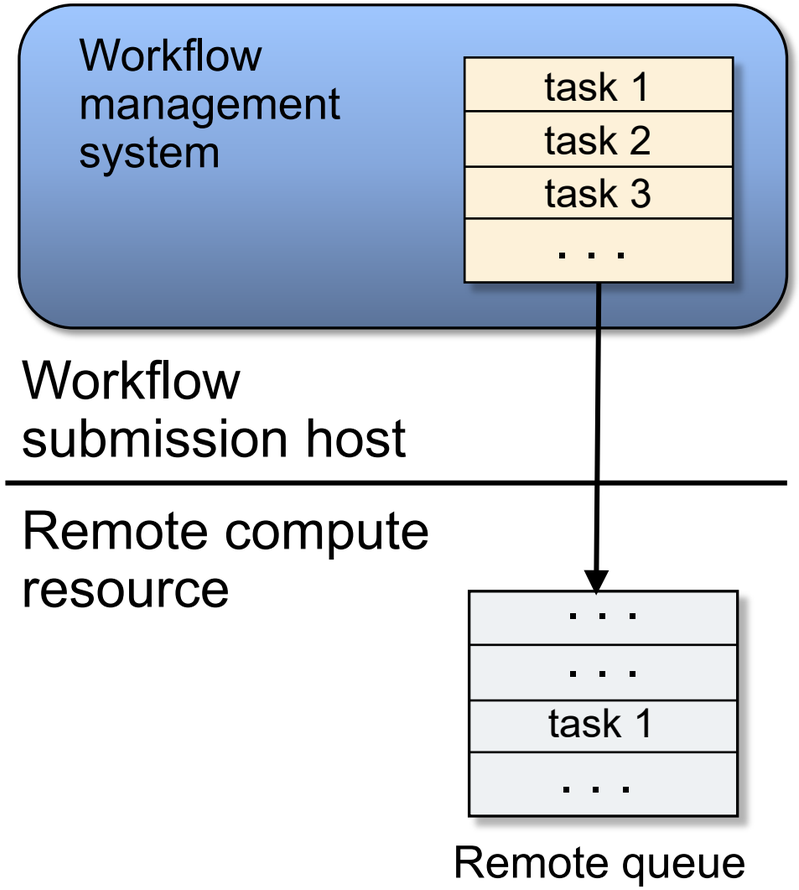


Challenge: Data Management

- Millions of data files produced and consumed
  - Pegasus provides staging
    - Symlinks files if possible, transfers files if needed
    - Transfers output back to local archival disk
    - Supports running parts of workflows on separate systems
  - Cleans up temporary files when no longer needed
  - Directory hierarchy to reduce files per directory
- We added automated checks for file correctness
  - Right number of files, NaNs, zero-value checks, correct size
  - Included as new jobs in workflow

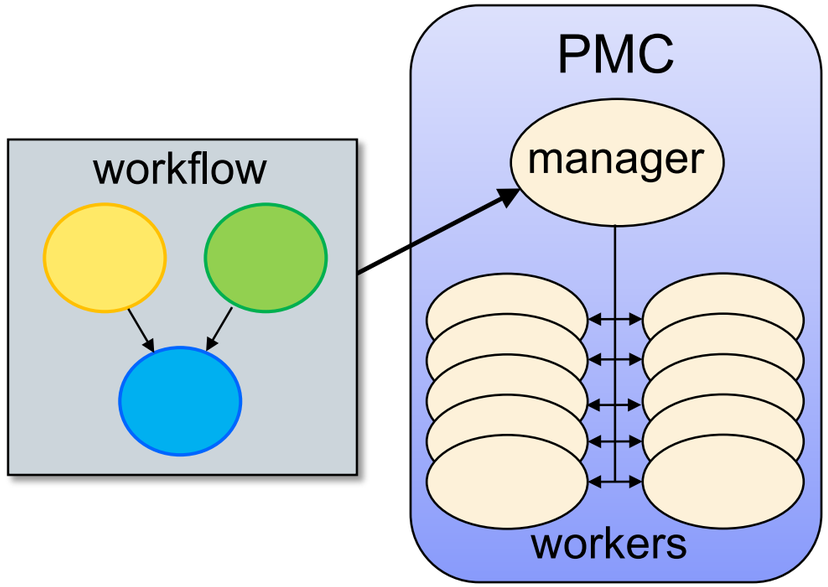
Challenge: Job Execution

- For large parallel jobs, workflow tools submit to remote scheduler
  - SSH (or other tool) puts jobs in remote queue
  - Runs like a normal batch job
  - Can specify either CPU or GPU nodes
- Workflow tools support job bundling
- For small serial jobs, need high throughput
  - Putting lots of jobs in the batch queue is ill-advised
    - Scheduler isn't designed for heavy job load
    - Scheduler cycle is ~5 minutes
    - Policy limits number of job submissions



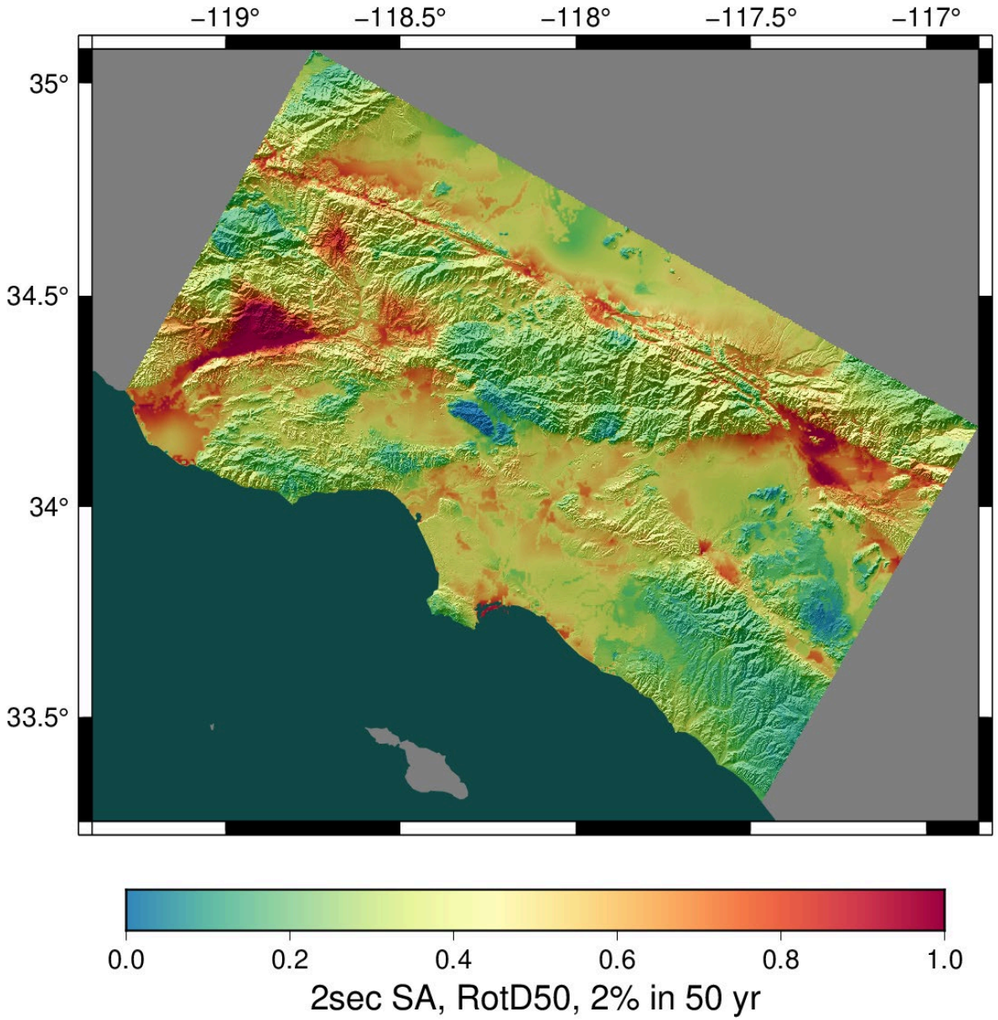
Pegasus-mpi-cluster (PMC)

- MPI wrapper around serial or thread-parallel jobs
  - Manager-worker paradigm
  - Preserves dependencies
  - Job is submitted to multiple nodes, starts PMC
  - Specify jobs as usual, Pegasus does wrapping
- Uses intelligent scheduling
  - Core counts
  - Memory requirements
- Developed for our application



Recent Results

- Completed regional hazard calculation for Southern California
- 95 days of around-the-clock execution
- Used 772,000 node-hours on *Summit*
  - Peak of 73% of the system (3382 nodes)
- Workflow tools:
  - Ran 28,130 jobs
  - Managed 2.5 PB of data
  - Staged 74 TB / 19 M files to long-term storage
  - Inserted 12.5 billion shaking metrics into local database



Your turn!

- What bottlenecks are you experiencing in your work that workflow tools might be able to help with?
- Take a minute to reflect and come up with a few items.
- Now, turn to a neighbor and talk about your bottlenecks with each other.
- Who would like to share something smart their partner said?

Scientific workflow tools can help!

- Mature community of tool developers to improve your efficiency
- Describe your workflow as tasks with dependencies between them
  - “Tasks” are anything you can run on a computer
- Separation of process from data
  - Can run the same workflow with different data
  - Can run the same workflow on different systems
- Provide a variety of features to help address bottlenecks

Basic tool concepts

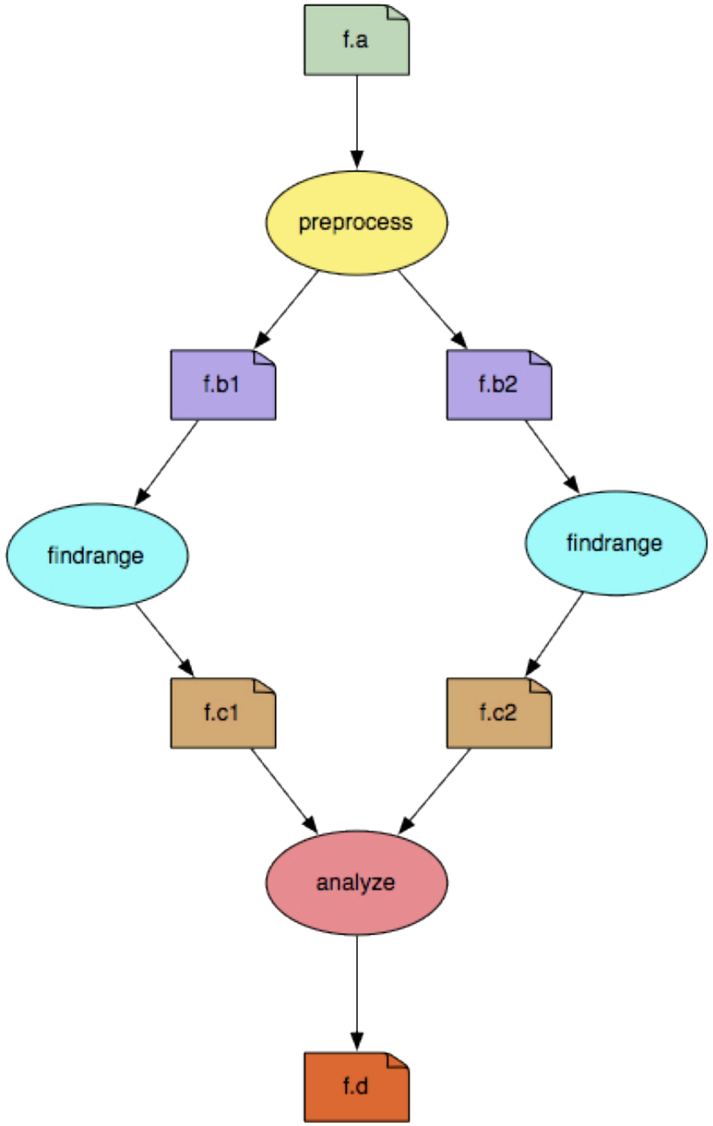
- Many workflow tools, but shared concepts between them
- Way to represent workflow tasks and their data
  - Can be specified through API, annotations, GUI
  - Explicit or implicit data roles
- Workflow prepared to run on certain hardware
- Schedule and run the workflow, honoring dependencies
  - May include remote job submission and data transfer
  - Some tools support interactivity and notebooks
  - Capture of metadata
- User monitors workflow execution
  - May include error handling and retry provisions

NAS report

- US National Academy of Sciences commissioned a report on automated research workflows (ARWs) in 2020
- "The common goal of researchers implementing ARWs is to **accelerate scientific knowledge generation, potentially by orders of magnitude, while achieving greater control and reproducibility in the scientific process.**"
- "The tools and techniques being developed under the large umbrella of ARWs promise to transform the centuries-old serial method of research investigation... Simultaneously, ARWs provide a way to satisfy pressing demands across fields to increase interoperability, reproducibility, replicability, and trustworthiness by **better tracking results, recording data, establishing provenance, and creating more consistent metadata than even the most dedicated researchers can provide themselves.**"

Popular Workflow Tools

- Pegasus-WMS (what I use)
  - Developed at USC's Information Sciences Institute
- Used in many science domains, including LIGO project
- Workflows are executed from local machine
  - Jobs can run on local machine or on distributed resources
- You use API to write code describing workflow
  - Python (recommended), Java, or R
  - Define tasks with parent / child relationships
  - Describe files and their roles
- Pegasus creates YAML file describing workflow
- Workflow represented by directed acyclic graph



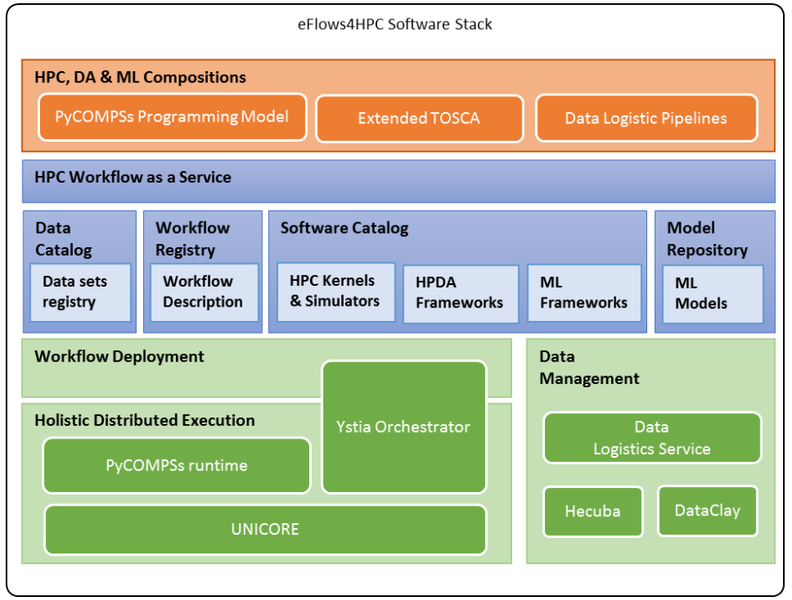
Popular Workflow Tools

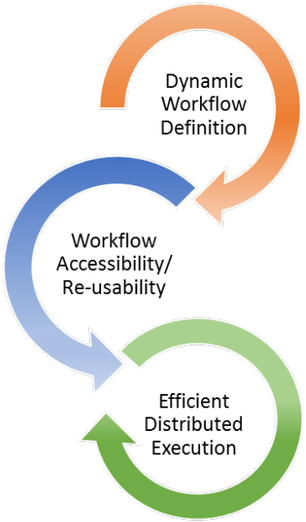
- Parsl (U of Chicago/Argonne NL)
  - Parallelize Python by annotating functions or external apps
  - Integrated with Jupyter notebooks
  - Link outputs and inputs of annotations to describe workflow

```
@bash_app
def mysim(stdout=("output/p1.out", "w"),
  stderr=("output/p1.err", "w")):
    #Call a bash command-line app 'simulate'
    return "app/simulate"

# call the mysim app and wait for the result
mysim().result()

# open the output file and read the result
with open('output/p1.out', 'r') as f:
    print(f.read())
```

  - Focus on large data, many tasks

- eFlows4HPC (BSC)
  - Workflow platform integrating different workflow software elements
  - PyCOMPS responsible for runtime
  - Workflows-as-a-service paradigm





Popular Workflow Tools

- FireWorks (LBNL/NERSC)
  - Workflows described through Python, JSON, or YAML, stored in MongoDB database
  - Submit batch scripts to launch workflow
  - Monitor through web interface
- Makeflow (Notre Dame)
  - Makefile-type syntax to specify workflow
    - Targets are output files, dependent on input files, with execution string to run
  - Can work with Work Queue for management of compute resources (workers)
    - Multiple clusters, pilot jobs, dynamic worker pool
- Nextflow (Seqera Labs in Barcelona, but open source)
  - Uses dataflow paradigm: tasks write/read from channels (not always a file), can be piped
  - Custom scripting language for defining workflows
- Many more! Ask me about specific use cases

But what about Python scripts?

- You can reproduce some features with custom scripts
- Cluster scheduler supports basic features
  - Can enforce job dependencies
  - Email notifications when complete
- Can check exit status and retry on failure
- Copy files in and out when required

What approach should you use?

- What are some pros and cons of using an established tool or custom scripts?

What did we learn?

- Workflow tools exist!
  - Designed to resolve bottlenecks and improve your efficiency
- Work with any computational tasks
  - No changes required to application codes
- Help manage jobs, files, and metadata
- Separate process from data
  - Distributed execution
  - Migrate workflow to new systems
  - Easy to explain your process to new users

Closing Thoughts

- Automation is vital, even without workflow tools
  - Eliminate human polling
  - Get everything to run automatically if successful
  - Be able to recover from common errors
- Put ALL processing steps in the workflow
  - Include validation, visualization, publishing, notifications
- Avoid premature optimization
- Consider new, larger, compute environments (dream big!)
  - Larger clusters, clouds
- Tool developers want to help you!

Links

- SCEC: http://www.scec.org
- Pegasus: http://pegasus.isi.edu
- Pegasus-mpi-cluster: http://pegasus.isi.edu/wms/docs/latest/cli-pegasus-mpi-cluster.php
- HTCondor: http://www.cs.wisc.edu/htcondor/
- Parsl: https://parsl-project.org/
- eFlows4HPC: https://eflows4hpc.eu/
- pyComps: https://eflows4hpc.readthedocs.io/en/latest/Sections/01_Software_Stack/02_Runtime_Components/COMPSs.html
- FireWorks: https://github.com/materialsproject/fireworks
- Makeflow: http://ccl.cse.nd.edu/software/makeflow/
- Work Queue: http://ccl.cse.nd.edu/software/workqueue/
- Nextflow: https://www.nextflow.io/
- CyberShake: http://scec.usc.edu/scecpedia/CyberShake

Thanks!


USGS
science for a changing world


National
Science
Foundation


PG&E


U.S. DEPARTMENT OF
ENERGY


SCEC
AN NSF+USGS CENTER


ISI
Information Sciences Institute


Pegasus


OAK RIDGE
National Laboratory
LEADERSHIP
COMPUTING
FACILITY


USC
Advanced Research Computing
Enabling scientific breakthroughs at scale