

Simplify Your Science with Workflow Tools

Scott Callaghan (SCEC)

July 17, 2026

2026 IHPCSS

scottcal@usc.edu



I've got a problem.

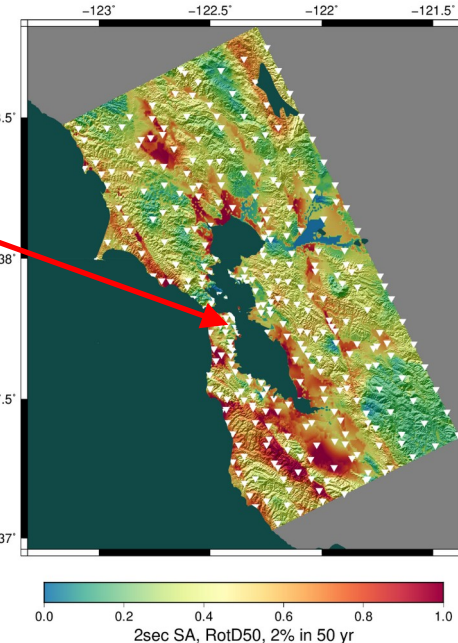
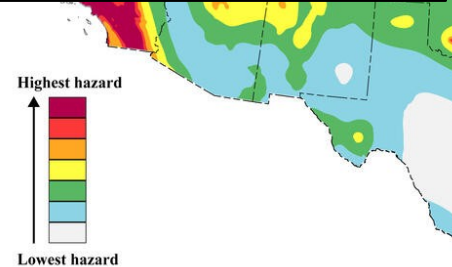
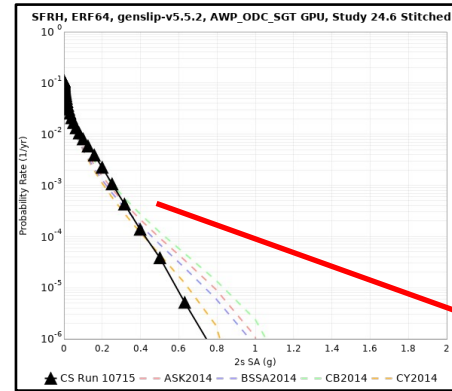
300,000 earthquakes

127 million simulated seismograms

5 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 Northern California, near San Francisco (9M residents)
- Want to use best available science
 - Run wave propagation simulations of many possible earthquakes
 - Use shaking information for hazard estimates





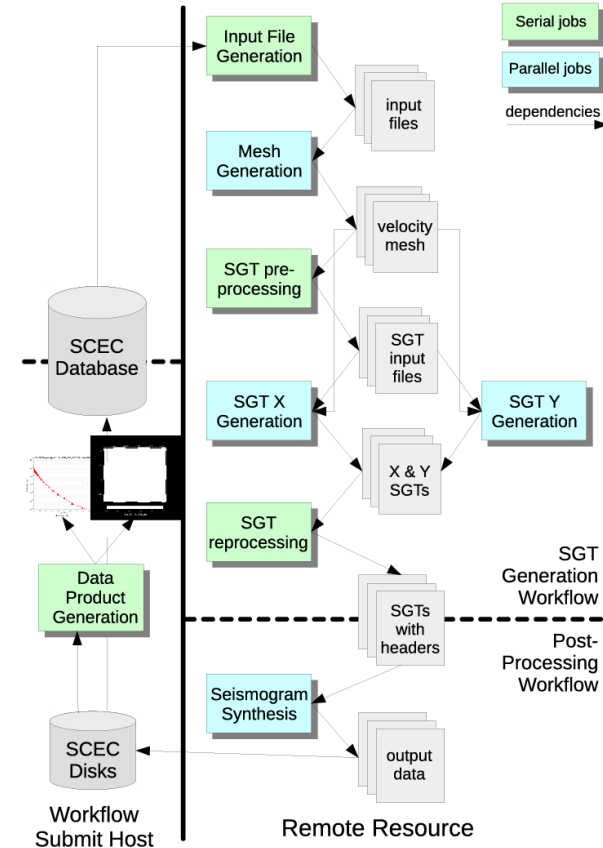
What are my challenges?

- 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 executions
- Error recovery
 - Many jobs and files will lead to problems



Solution: Scientific Workflows

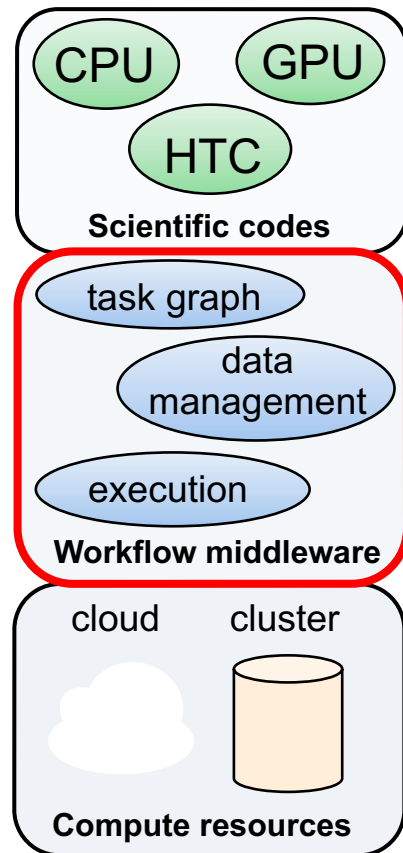
- We at SCEC decided to use scientific workflows to manage challenges
- Work is described as a directed graph of tasks with dependencies
- Separation of process from data
 - Can run the same workflow with different data
 - Can run the same workflow on different systems
- Jobs are queued and executed
- We use two open-source tools in user space:
 - Pegasus-WMS
 - HTCondor





How did workflow tools help me?

- 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
 - My seismic workflow has run on 14 systems since 2007
- Error recovery
 - Failed jobs retried
 - Workflow checkpointed if jobs keep failing
- No changes made to application codes



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.**”





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 together.
- Who would like to share something smart their partner said?

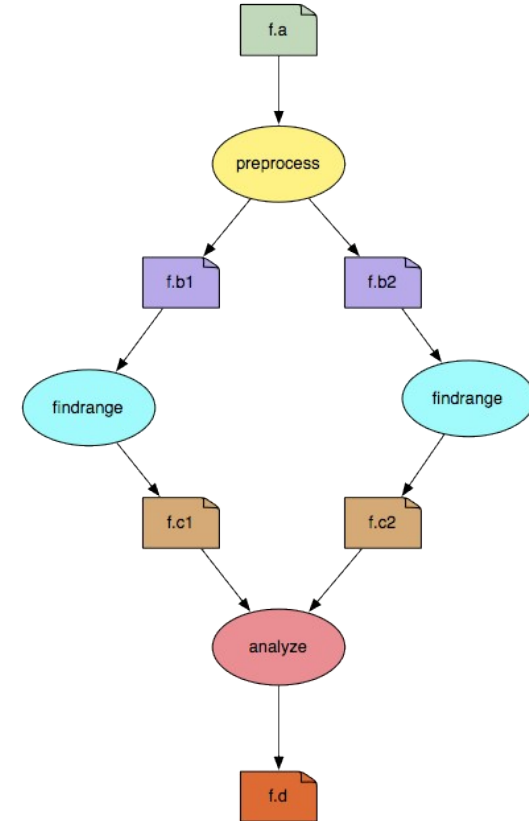


Basic tool concepts

- Many workflow tools actively supported, 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
- Tools prepare workflow 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 and provenance
- User monitors workflow execution
 - May include error handling and retry provisions

Popular Workflow Tools

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



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()

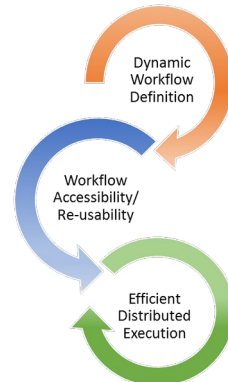
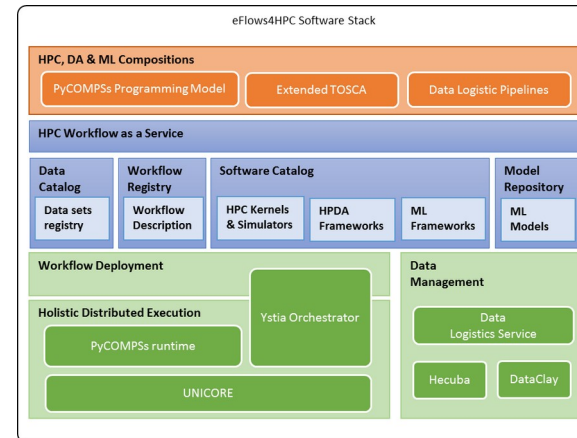
# open the output file and read the result
with open('output/p1.out', 'r') as f:
    print(f.read())
```

- 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

- Snakemake (international; most maintainers in Europe)



- Create “rules” in Pythonish language describing how to create outputs from inputs
- Good integration with Conda and containers
- Support for dynamic workflows

- 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 (Sequera 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
- Integrates well with containers
- Many more! Ask me about specific use cases

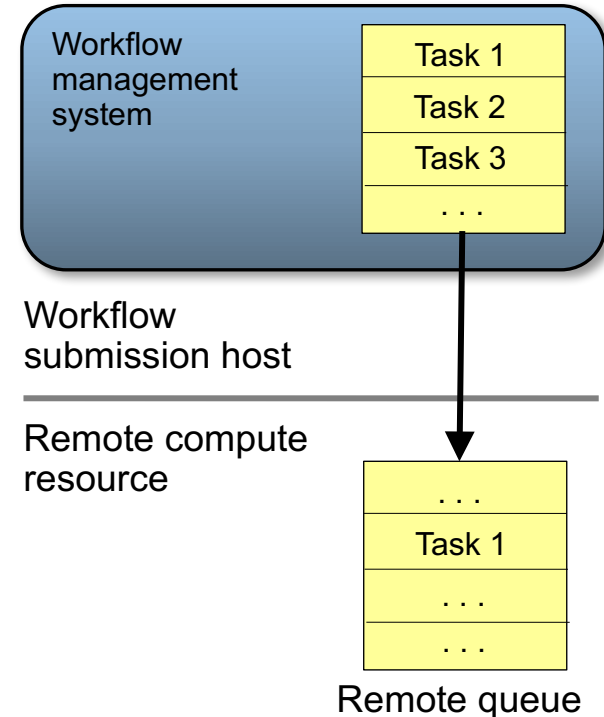


Challenge: Data Management

- 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
 - Number of files, NaNs, zero-value checks, correct size
 - Included as new jobs in workflow

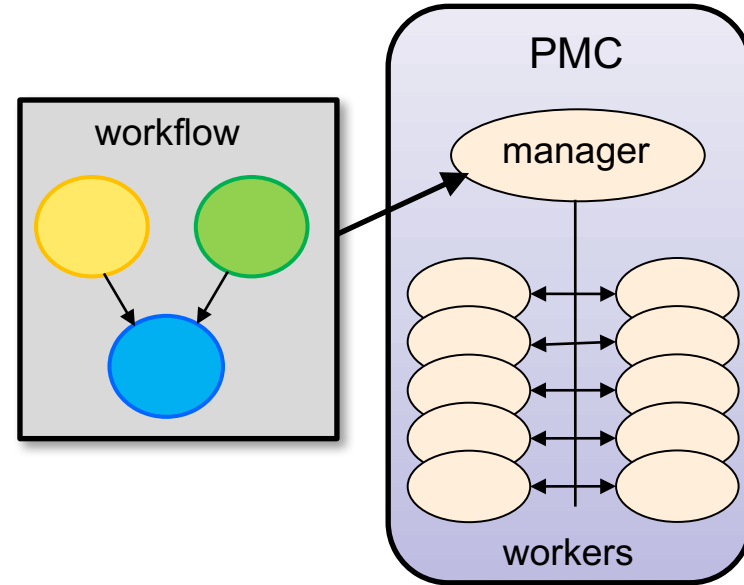
Challenge: Job Execution

- 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



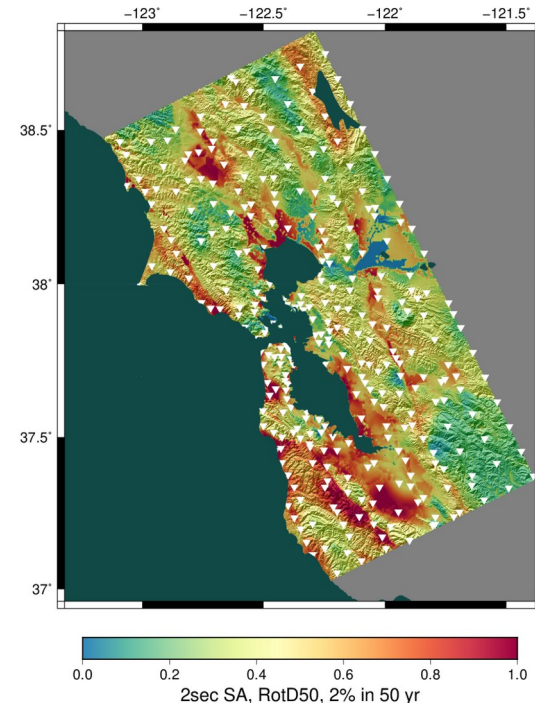
Pegasus-mpi-cluster (PMC)

- MPI wrapper around workflows of 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 Science Results

- Completed regional hazard calculation for Northern California
- 45 days of around-the-clock execution
- Used two systems:
 - 55,000 node-hrs (mostly GPU) on *Frontier*, peak 44%
 - 125,000 node-hrs (all CPU) on *Frontera*, peak 12%
- Workflow tools:
 - Ran 27,720 jobs
 - Managed 1.0 PB of data
 - Staged 350 TB from *Frontier* to *Frontera*
 - Staged 36 TB / 9 M files to long-term storage
 - Inserted 2.3 billion shaking metrics into local database





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, improve your efficiency, and aid in reproducibility
- 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
- Snakemake: <https://snakemake.github.io/>
- Makeflow: <http://ccl.cse.nd.edu/software/makeflow/>
- Work Queue: <http://ccl.cse.nd.edu/software/workqueue/>
- Nextflow: <https://www.nextflow.io/>
- CyberShake (my project): <https://www.scec.org/software/cybershake>



Thanks!



U.S. National
Science
Foundation



U.S. DEPARTMENT OF
ENERGY



Information Sciences Institute

TACC

TEXAS ADVANCED COMPUTING CENTER



LEADERSHIP
COMPUTING
FACILITY



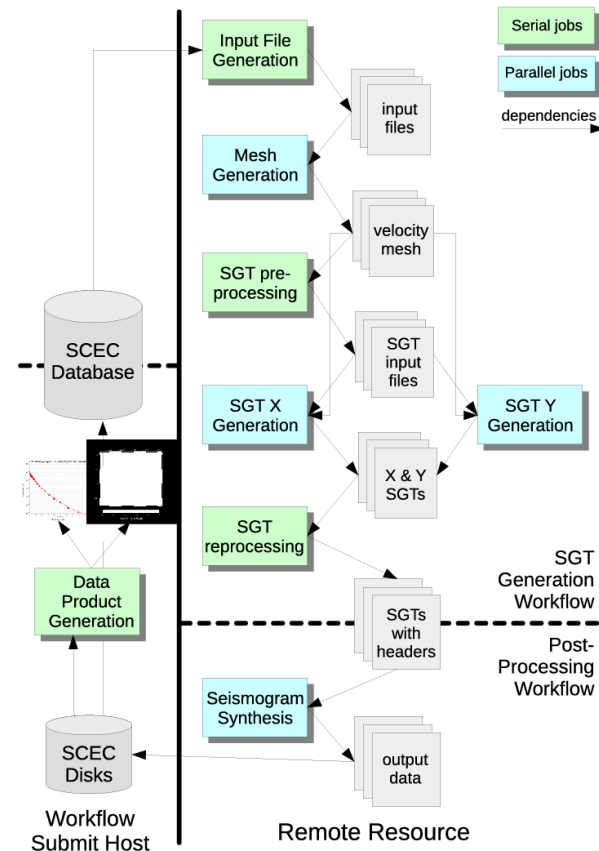
USC

Advanced Research Computing
Enabling scientific breakthroughs at scale



Solution: Scientific Workflows

- We at SCEC decided to use scientific workflows to manage challenges
- Describe your workflow as a directed graph of tasks with dependencies
 - No changes to application codes
- Separation of process from data
 - Can run the same workflow with different data
 - Can run the same workflow on different systems
- Jobs are queued and executed
- We use two open-source tools in user space:
 - Pegasus-WMS
 - HTCondor





Solution: Scientific Workflows

- We at SCEC decided to use scientific workflows to manage challenges
- Describe your workflow as a directed graph of tasks with dependencies
 - No changes to application codes
- Separation of process from data
 - Can run the same workflow with different data
 - Can run the same workflow on different systems
- Jobs are queued and executed
- We use two open-source tools in user space:
 - Pegasus-WMS
 - HTCondor

