SC/EC

Simplify Your Science with Workflow Tools

Scott Callaghan
scottcal@usc.edu

2021 IHPCSS
July 28, 2021

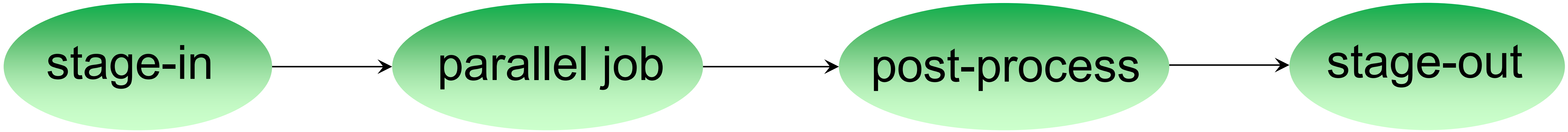
Overview

- What are scientific workflows?
- What problems do workflow tools solve?
- Overview of available workflow tools
- Real-world seismic hazard application (SCEC CyberShake)
  - Computational overview
  - Challenges and solutions
- Ways to simplify your work
- Goal: Help you figure out if this would be useful

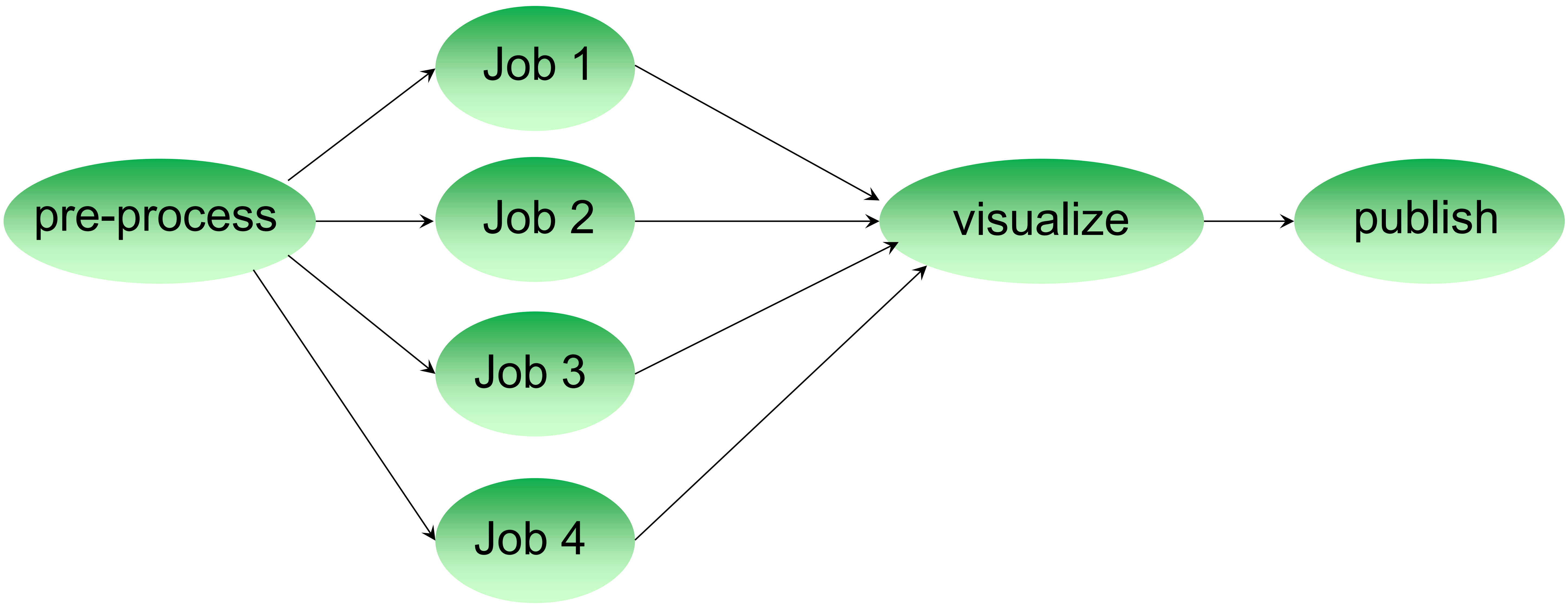
Scientific Workflows

- Formal way to express a scientific calculation
- Multiple tasks with dependencies between them
  - No limitations on tasks
- Capture task parameters, input, output
- Workflow process and data are independent
  - Often, run same workflow with different data
  - Could take the same workflow and run it on different systems
- Actually, you use workflows all the time…

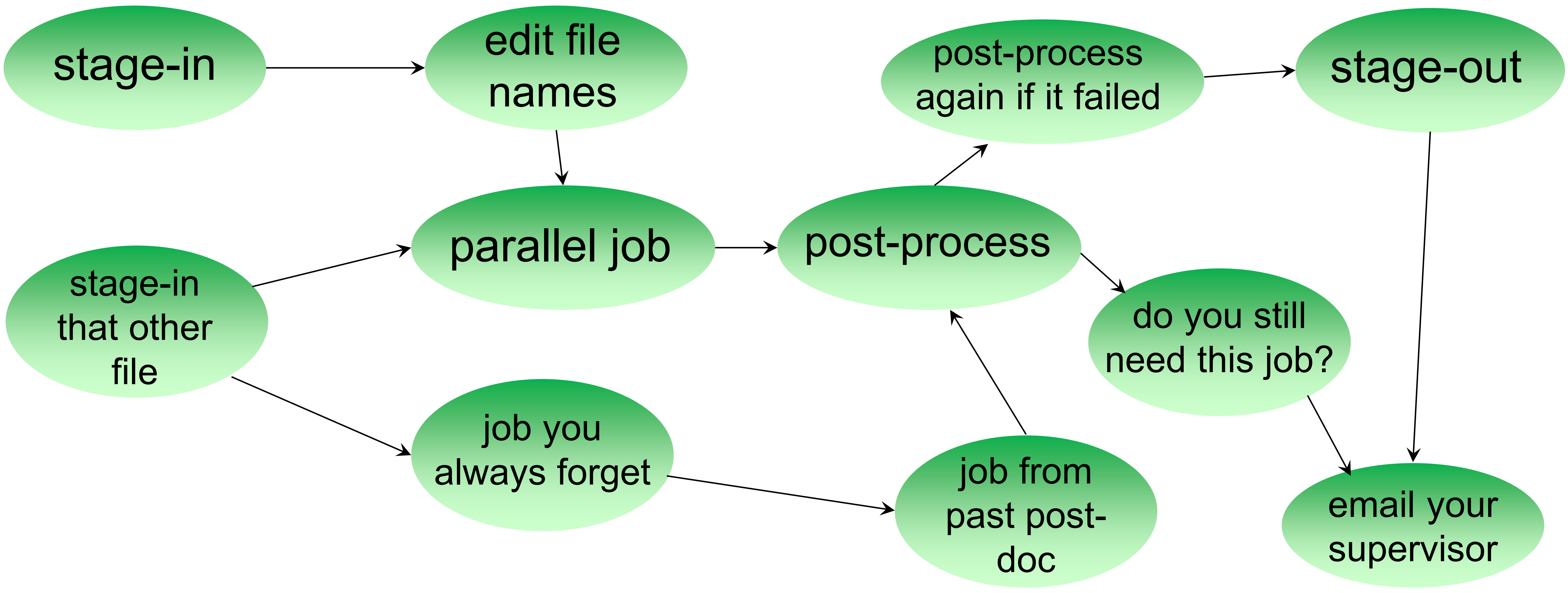
Perhaps your workflow is simple…



…or maybe a bit more complicated…



…or maybe just confusing



Workflow Components

- Task executions with dependencies
  - Specify a series of tasks to run
  - Outputs from one task may be inputs for another
- Task scheduling
  - Some tasks may be able to run in parallel with other tasks
- File management
  - Inputs must be present for tasks to run
- Metadata
  - Track when a task was run, key parameters
- Resource provisioning (getting processors)
  - Computational resources are needed to run jobs

What do we need help with?

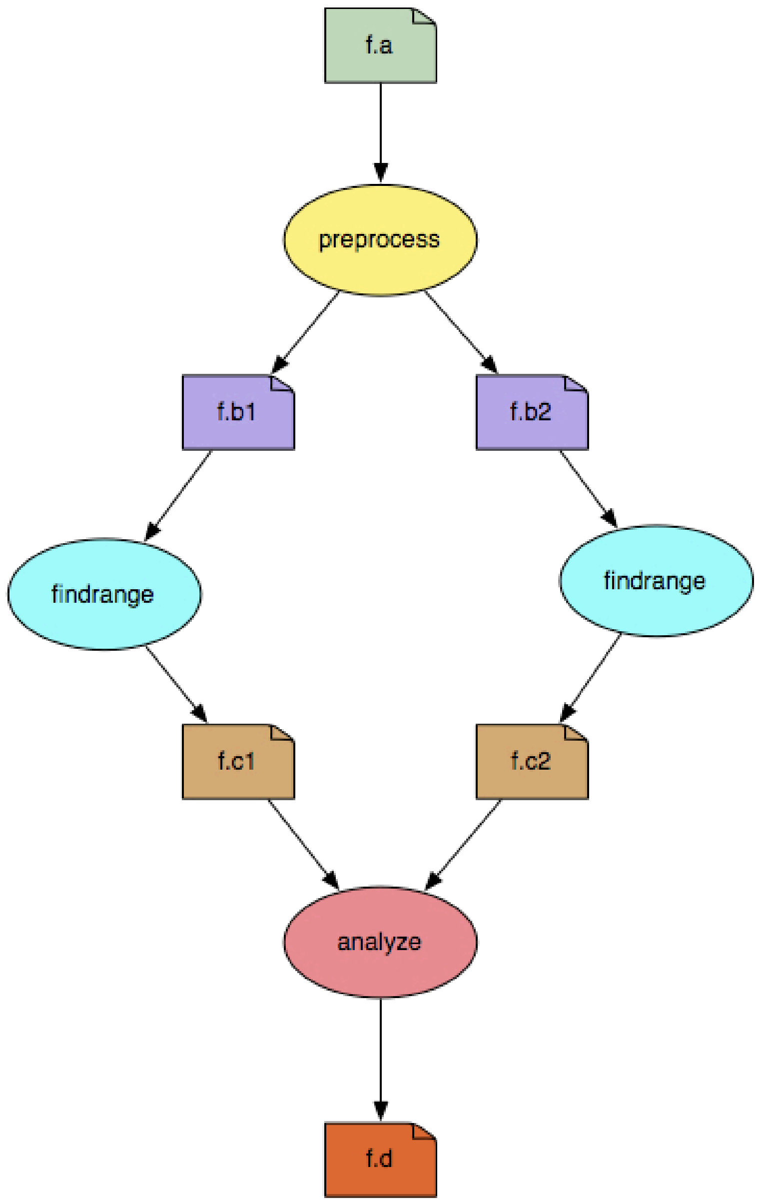
- Task executions with dependencies
  - What if something fails in the middle?
  - Dependencies may be complex
- Task scheduling
  - Minimize execution time while preserving dependencies
- File management
  - Make sure inputs are available for tasks
- Metadata
  - Automatically capture and track
- Matching jobs to processors (across multiple systems?)

Workflow tools can help!

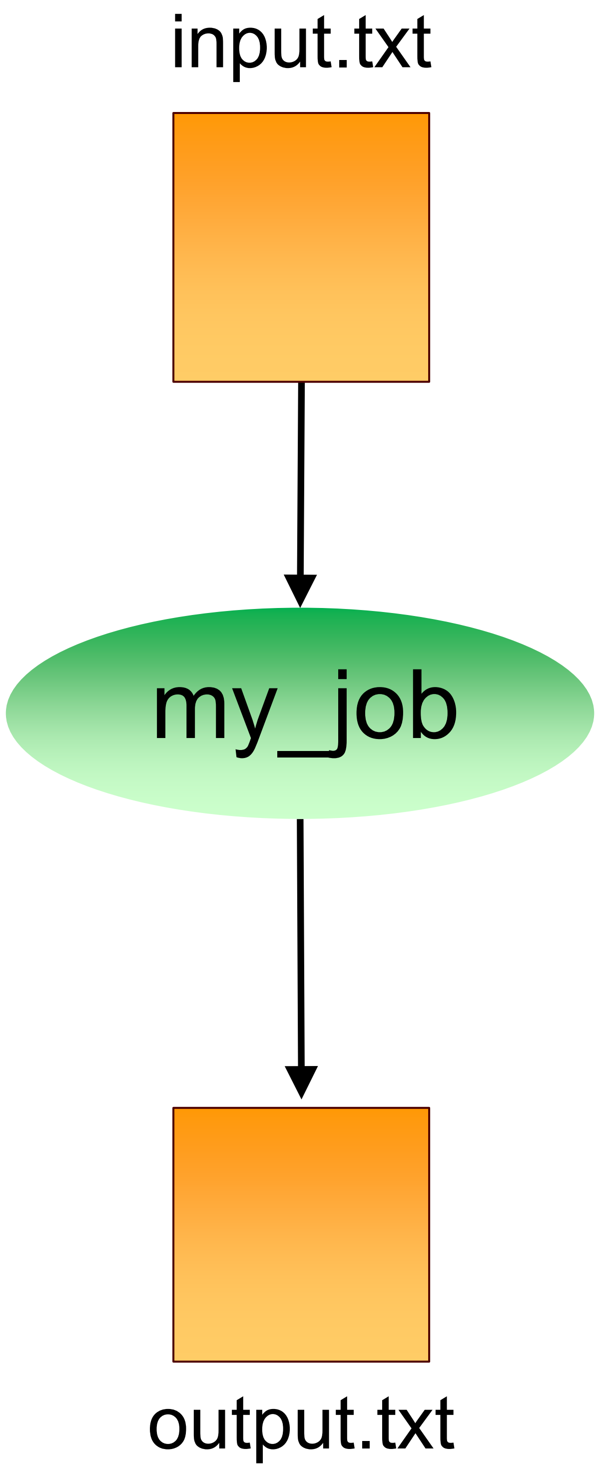
- Automate your pipeline
- Define your workflow via programming or GUI
- Run workflow on local or remote system
- Can support all kinds of workflows
- Use existing code (no changes)
- Provide many kinds of fancy features and capabilities
  - Flexible, but can be complex
- Will discuss one set of tools (Pegasus) as example, but concepts are shared

Pegasus-WMS

- 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 ("***create***")
  - Python (recommended), Java, or R
  - Define tasks with parent / child relationships
  - Describe files and their roles
- Pegasus creates XML file of workflow called a "DAX"
- Workflow represented by directed acyclic graph



Sample Workflow Creation

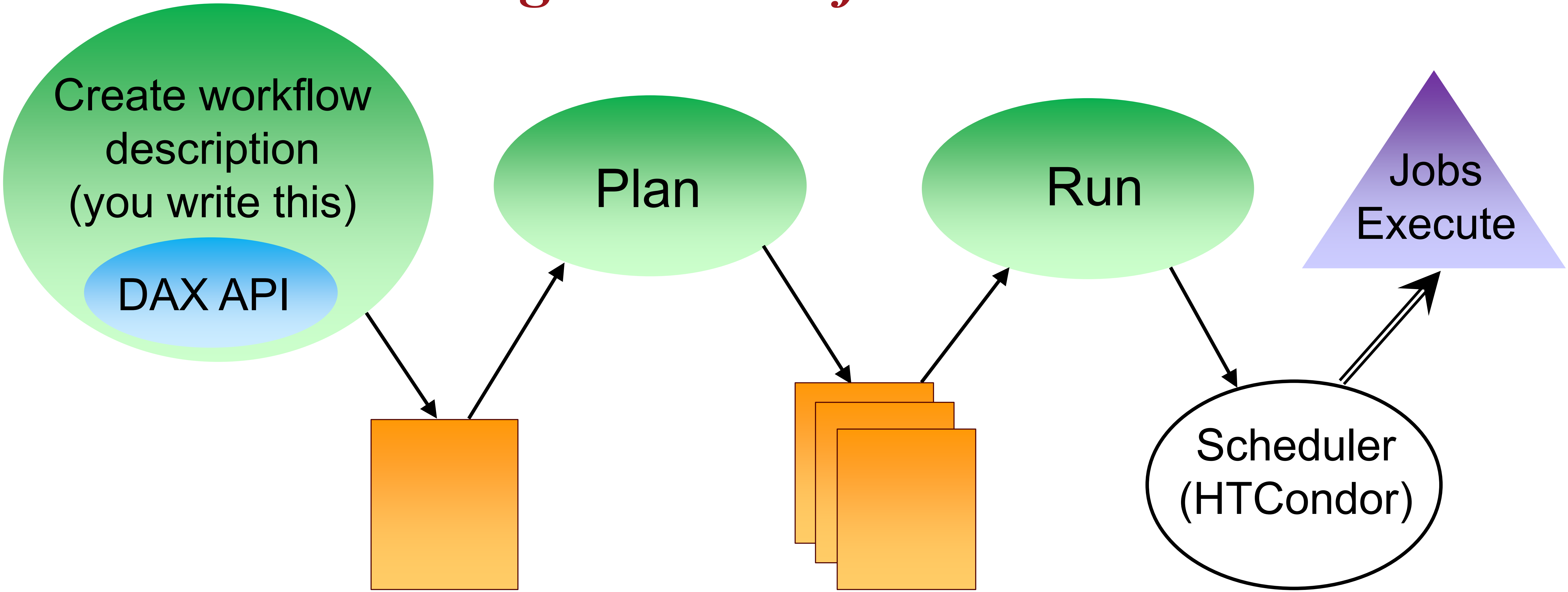



```
//Create DAX object
dax = ADAG("test_dax")
//Define my job
myJob = Job(name="MyGreatJob")
//Input and output files to my job
inputFile = File("input.txt")
outputFile = File("output.txt")
//Arguments to my_job (./my_job input=input.txt output=output.txt)
myJob.addArgument("input=input.txt", "output=output.txt")
//Role of the files for the job
myJob.uses(inputFile, link=Link.INPUT)
myJob.uses(outputFile, link=Link.OUTPUT)
//Add the job to the workflow
dax.addJob(myJob)
//Write to file
fp = open("test.dax", "w")
dax.writeXML(fp)
fp.close()
```

Getting ready to run

- DAX is "abstract workflow" – system-independent
  - Logical filenames and executables
  - Algorithm description
- Use Pegasus to prepare workflow for execution ("***plan***")
  - Uses catalogs to resolve logical names, compute info
  - Pegasus automatically augments workflow
    - Staging jobs (if needed) using scp, wget, Globus, Docker Hub, Amazon S3, …
    - Registers output files in a catalog to find later
    - Wraps jobs in pegasus-kickstart for detailed statistics
  - Generates a DAG
    - Top-level workflow description (tasks and dependencies)
    - Submission file for each job with specifics for the execution system

Pegasus Workflow Path



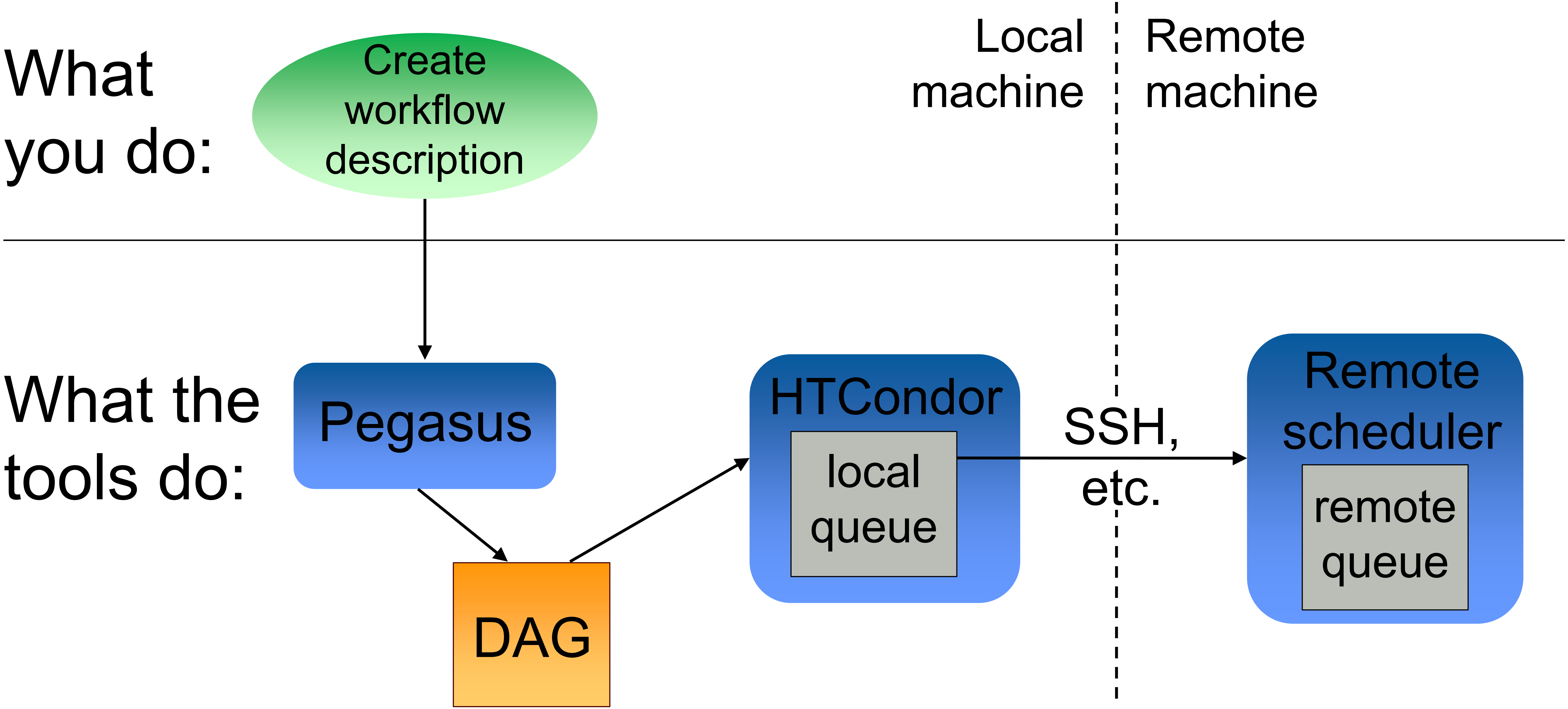
Abstract workflow (DAX)
Logical names, algorithm
`job name="MyGreatJob"`

Concrete workflow (DAG)
Physical paths, job scripts
`executable=/path/to/directory/my_job`

Other tools in stack

- HTCondor (University of Wisconsin Madison)
  - Pegasus submits workflow to HTCondor (“***run***”)
  - Supervises runtime execution of DAG files
    - Maintains job queue
    - Monitors dependencies
    - Schedules jobs
    - Retries failures
    - Writes checkpoint
- Tools for remote job submission to clusters and clouds
  - SSH, BOSCO, CREAMCE, glideins/pilot jobs, …
  - Condor uses these tools to match jobs to resources

Full workflow stack



Other Workflow Tools

- Regardless of the tool, same basic elements
  - Describe your high-level workflow (Pegasus "Create")
  - Prepare your workflow for the execution environment (Pegasus "Plan")
  - Schedule and run your workflow (HTCondor)
  - Send jobs to remote resources (SSH, pilot jobs)
  - Monitor the execution of the jobs (HTCondor DAGMan)
- Brief overview of some other available tools
  - All support large-scale workflows

Other 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

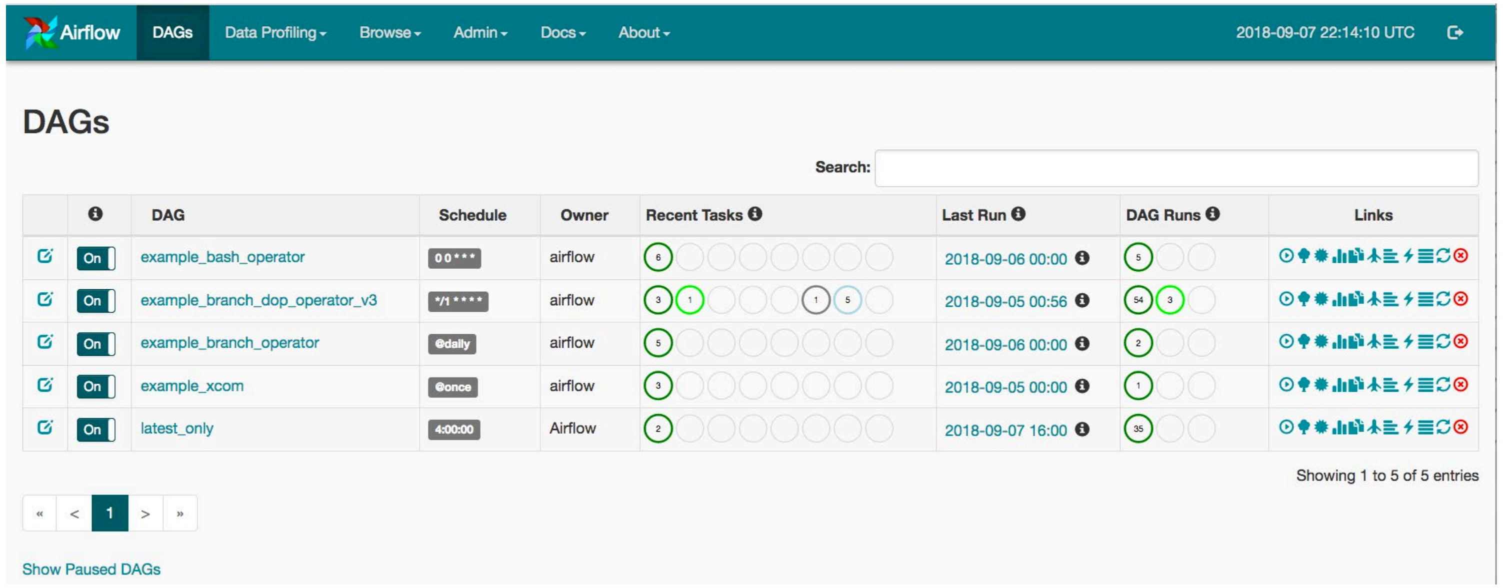
```
@bash_app
def mysim(stdout=("output/p1.out", "w"),
  stderr=("output/p1.err", "w")):
    #Call a bash command-line app ‘simulate’
    return "app/simulate"

# 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

- Airflow (Apache project / Airbnb)
  - Workflow defined with Python API
  - Description used to write DAGs
  - Internal scheduler triggers tasks
  - Support for Kubernetes (deploys containers)
  - Web UI for getting status

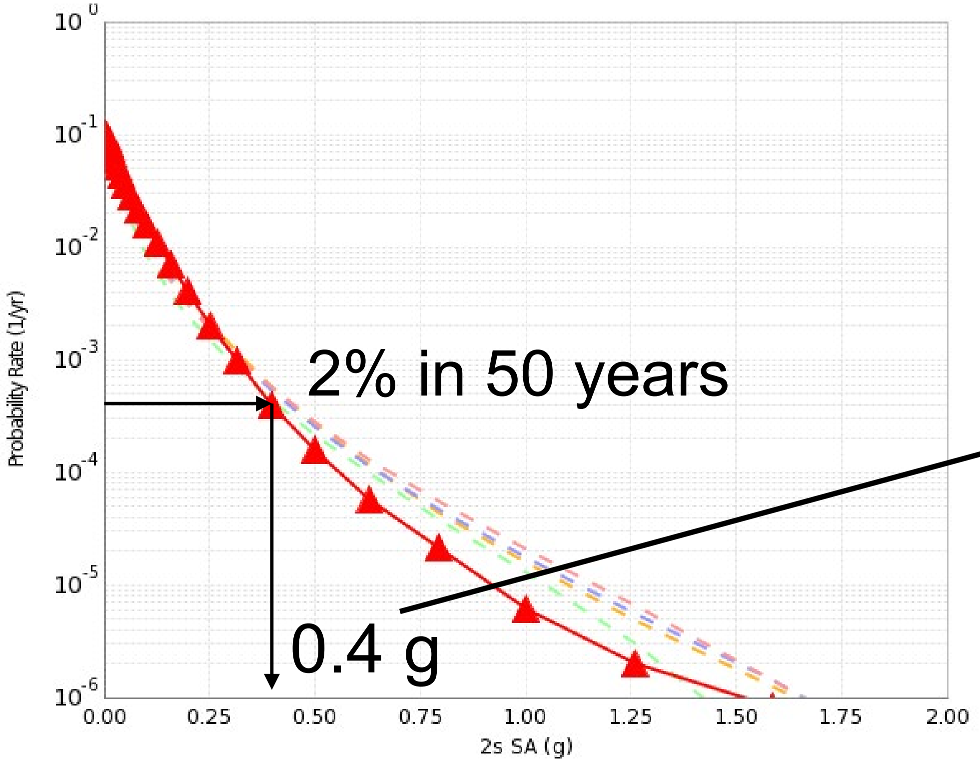


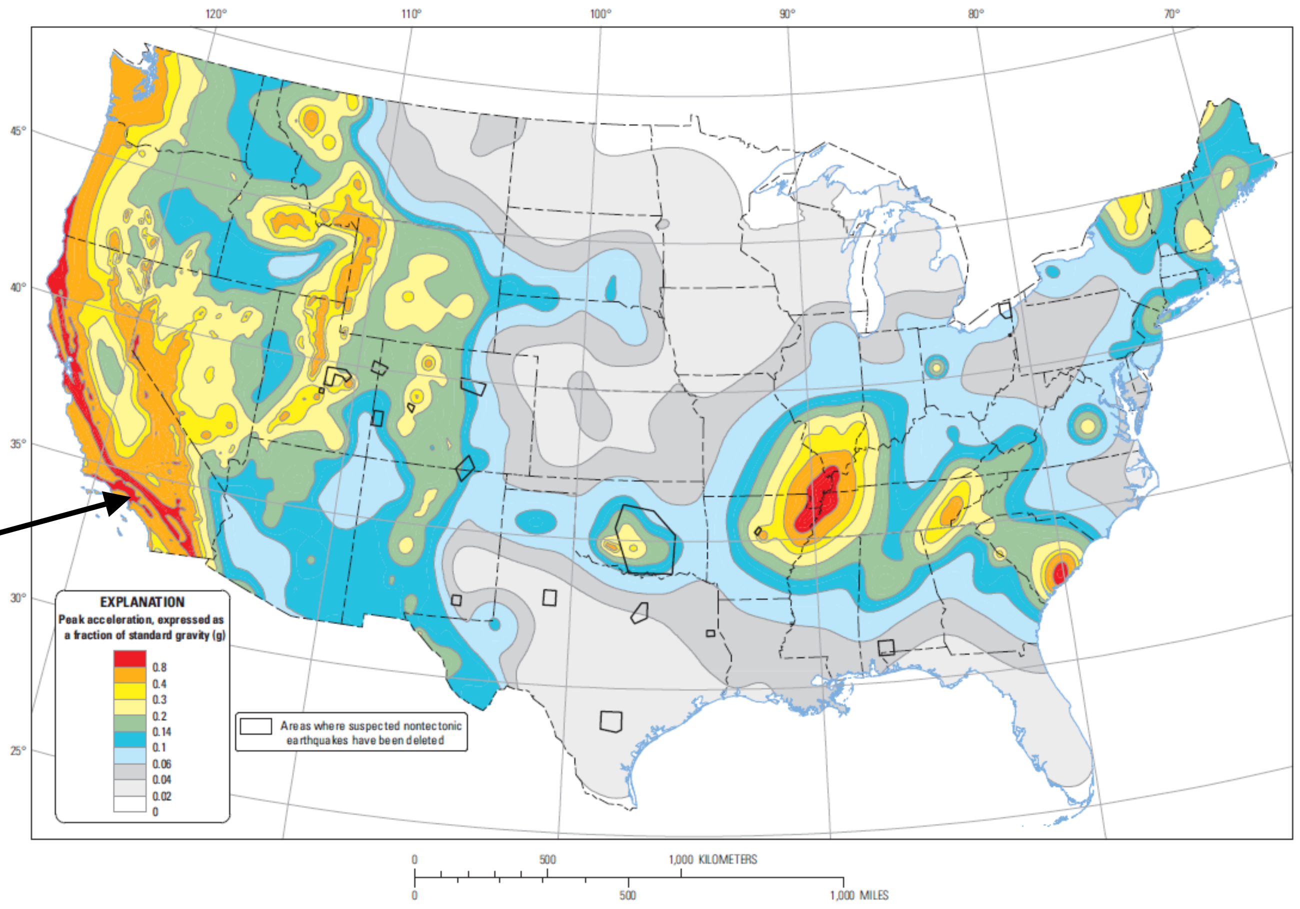
Other Workflow Tools

- JUBE (Jülich Supercomputing Center)
  - Designed for automating benchmarks and testing on multiple systems
  - Workflow described via XML files
- 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 (Barcelona Centre for Genomic Regulation)
  - Uses dataflow paradigm: tasks write/read from channels (not always a file)
  - Custom scripting language for defining workflows
- Pachyderm (commercial software, free version with limited size/runtime)
  - Data-driven: “Git for Data Science”
  - Provides data version control and tools for building and running data science workflows
- Many more! Ask me about specific use cases

Workflow Application: CyberShake

- What will peak earthquake shaking be over the next 50 years?
  - Used in building codes, insurance rates, disaster planning
  - Answered via Probabilistic Seismic Hazard Analysis (PSHA)
    - Get list of all possible earthquakes
    - Figure out how much shaking each causes
    - Combine shaking levels with probabilities





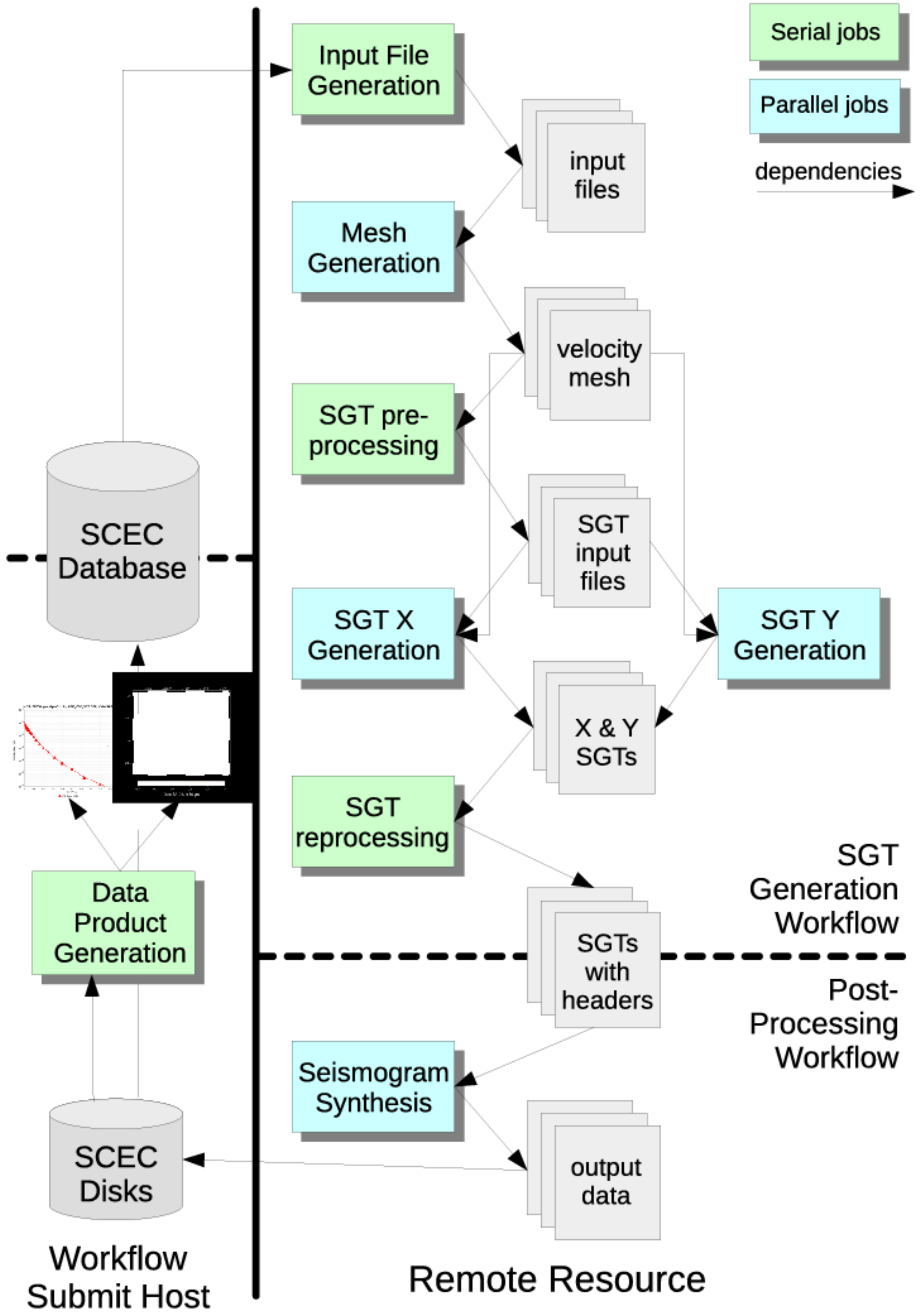
Two-percent probability of exceedance in 50 years map of peak ground acceleration

CyberShake Computational Requirements

- Determine shaking of ~500,000 earthquakes per site
- 2 large parallel jobs
  - Perform wave propagation
  - 400 GPUs x 30 min
  - 1.5 TB total output
- 500,000 small serial jobs
  - Calculate seismograms and shaking measures
  - 1 core x 4 min
  - 17 GB total output
- Need ~900 sites for hazard map

CyberShake Challenges

- Automation
  - Too much work to run by hand
- Data management
  - Input files need to be moved to cluster
  - Output files transferred back for archiving
- Error recovery
  - Detect and recover from basic errors automatically
- Resource provisioning
  - How to execute large numbers of small tasks?
- Decided to use scientific workflows
  - Selected Pegasus-WMS



Challenge: Resource Provisioning

- For large parallel GPU jobs, 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
- 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
- Solution: Pegasus-mpi-cluster (PMC)

Pegasus-mpi-cluster (PMC)

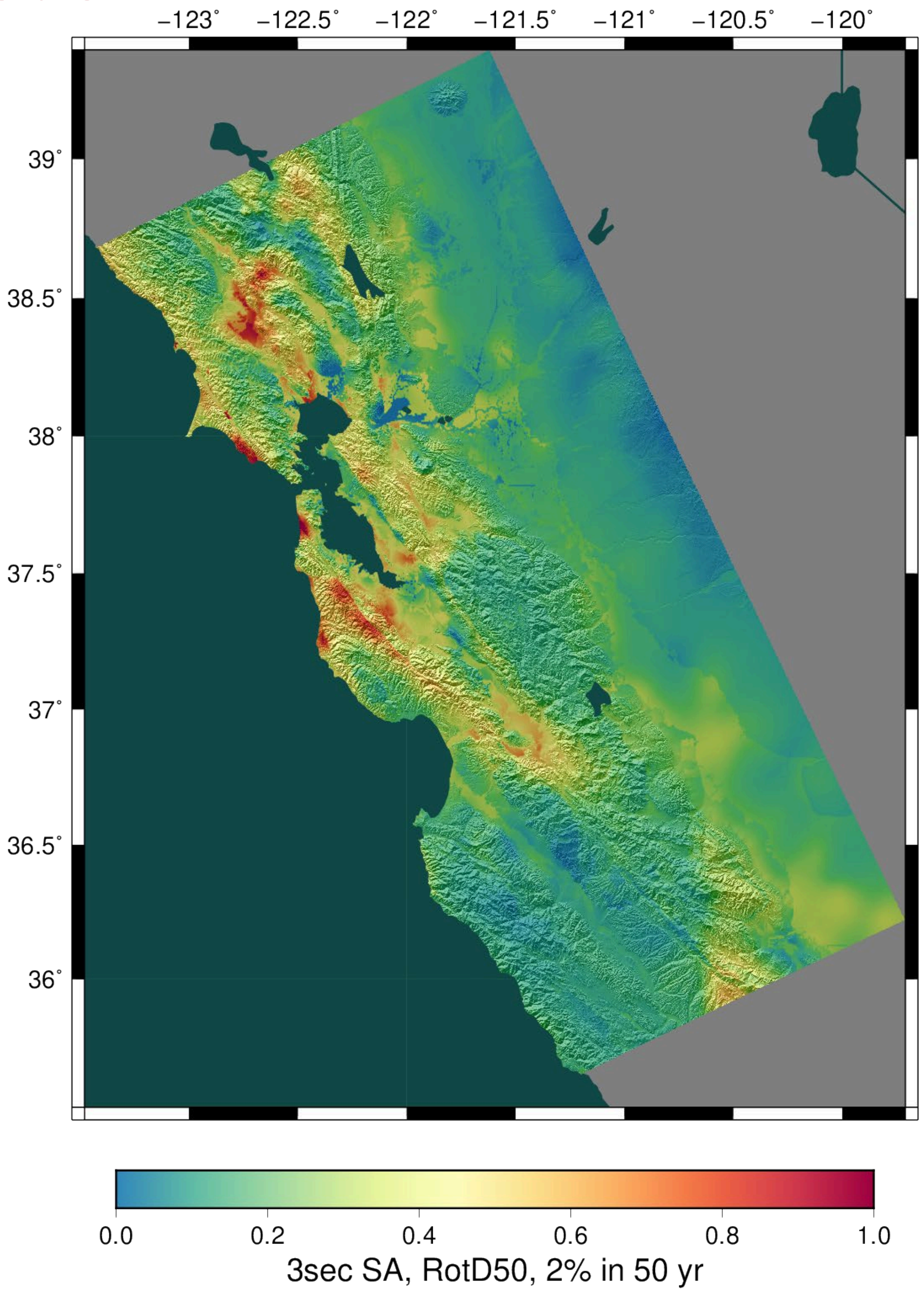
- MPI wrapper around serial or thread-parallel jobs
  - Master-worker paradigm
  - Preserves dependencies
  - HTCondor submits job to multiple nodes, starts PMC
  - Specify jobs as usual, Pegasus does wrapping
- Uses intelligent scheduling
  - Core counts
  - Memory requirements
- Can combine writes
  - Workers write to master, master aggregates to fewer files
- Developed for our application

Challenge: Data Management

- Millions of data files
  - 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 machines
  - Cleans up temporary files when no longer needed
  - Directory hierarchy to reduce files per directory
- We added automated checks to check integrity
  - Correct number of files, NaN, zero-value checks, correct size
  - Included as new jobs in workflow

CyberShake Study 18.8

- Hazard results for 869 sites
- Used OLCF *Titan* and NCSA *Blue Waters*
  - 1/3 of workflows ran jobs on both systems
- 6.2 million node-hours (120M core-hours)
  - Averaged 2,018 nodes (CPUs & GPUs) for 128 days
  - Max of 16,219 nodes (~280,000 cores)
- Workflow tools scheduled 17,921 jobs
- Workflow tools managed 1.2 PB of data
  - 157 TB of data automatically transferred
  - 14.4 TB (~12M files) staged back to local disk
- Workflow tools scale!



Problems Workflows Solve

- Task executions
  - Workflow tools will retry and checkpoint if needed
- Data management
  - Stage-in and stage-out data for jobs automatically
- Task scheduling
  - Optimal execution on available resources
- Metadata
  - Automatically track runtime, environment, arguments, inputs
- Getting computational resources
  - Whether large parallel jobs or high throughput

Should you use workflow tools?

- Probably using a workflow already
  - Replaces manual hand-offs and polling to monitor status
- Provides framework to assemble community codes
  - Also useful for training new teammates
- Scales from local computer to large clusters
- Provide portable algorithm description independent of data
  - CyberShake has run on 9 systems since 2007 with same workflow
- Does add additional software layers and complexity
  - Some development time is required

Final 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 compute environments (dream big!)
  - Larger clusters, XSEDE/PRACE/RIKEN/SciNet, Amazon EC2
- 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/
- Apache Airflow: https://airflow.apache.org/
- JUBE: https://www.fz-juelich.de/ias/jsc/EN/Expertise/Support/Software/JUBE/_node.html
- Makeflow: http://ccl.cse.nd.edu/software/makeflow/
- Work Queue: http://ccl.cse.nd.edu/software/workqueue/
- Nextflow: https://www.nextflow.io/
- Pachyderm: https://www.pachyderm.com/
- CyberShake: http://scec.usc.edu/scecpedia/CyberShake

Questions?