



# Simplify Your Science with Workflow Tools

International HPC Summer School  
June 27, 2017

Scott Callaghan  
Southern California Earthquake Center  
University of Southern California  
scottcal@usc.edu





# Overview

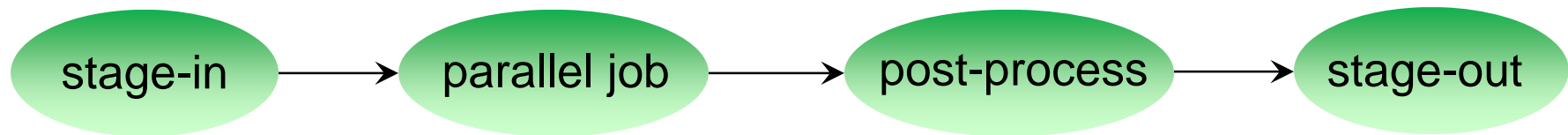
- What are scientific workflows?
- What problems do workflow tools solve?
- Overview of available workflow tools
- CyberShake (seismic hazard application)
  - Computational overview
  - Challenges and solutions
- Ways to simplify your work
- Goal: Help you figure out if this would be useful



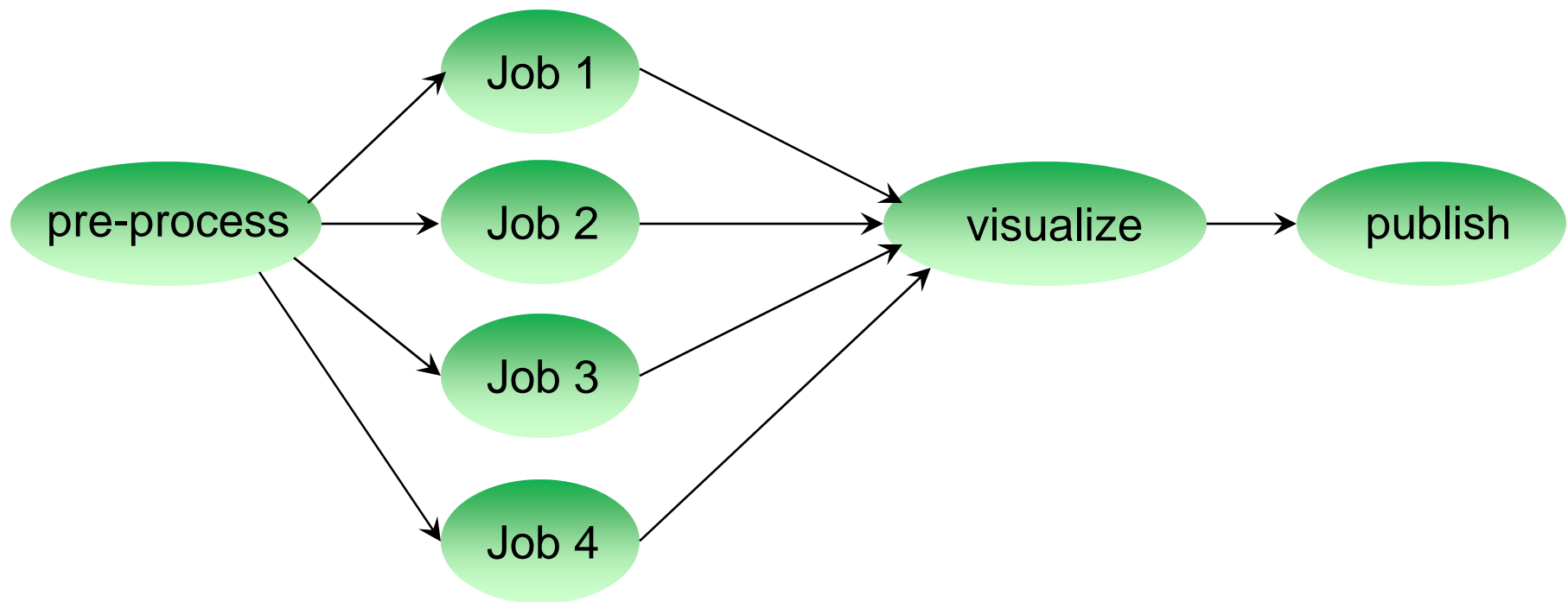
## Scientific Workflows

- Formal way to express a scientific calculation
- Multiple tasks with dependencies between them
- No limitations on tasks
- Capture task parameters, input, output
- Independence of workflow process and data
  - Often, run same workflow with different data
- You use workflows all the time...

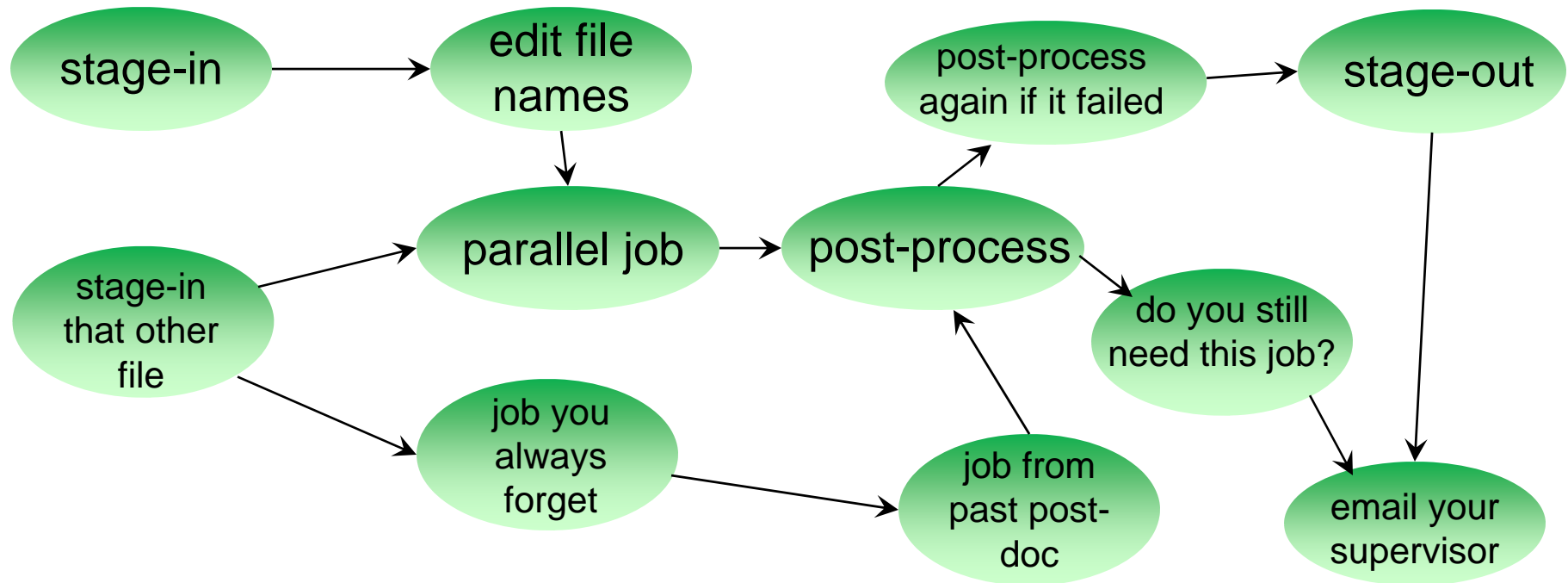
Perhaps your workflow is simple...



...or maybe a bit more complicated...



## ...or maybe just confusing





# Workflow Components

- **Task executions**
  - Specify a series of tasks to run
- **Data and control dependencies between tasks**
  - Outputs from one task may be inputs for another
- **Task scheduling**
  - Some tasks may be able to run in parallel with other tasks
- **File and metadata management**
  - Track when a task was run, key parameters
- **Resource provisioning (getting cores)**
  - Computational resources are needed to run jobs



## What do we need help with?

- **Task executions**
  - What if something fails in the middle?
- **Data and control dependencies**
  - Make sure inputs are available for tasks
- **Task scheduling**
  - Minimize execution time while preserving dependencies
- **Metadata**
  - Automatically capture and track
- **Getting cores**





## Workflow tools can help!

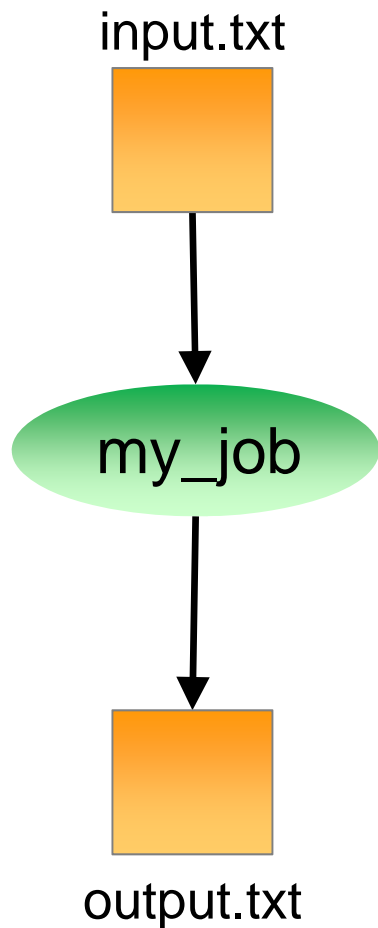
- Automate your pipeline
- Define your workflow via programming or GUI
- Run workflow on local or remote system
- Can support all kinds of workflows
- Use existing code (no changes)
- Provide many kinds of fancy features and capabilities
  - Flexible but can be complex
- Will discuss one set of tools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



# Pegasus-WMS

- Developed at USC's Information Sciences Institute
- Used in many 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, Java, Perl
  - Tasks with parent / child relationships
  - Files and their roles
- Pegasus creates XML file of workflow called a DAX
- Workflow represented by directed acyclic graph

## Sample Workflow



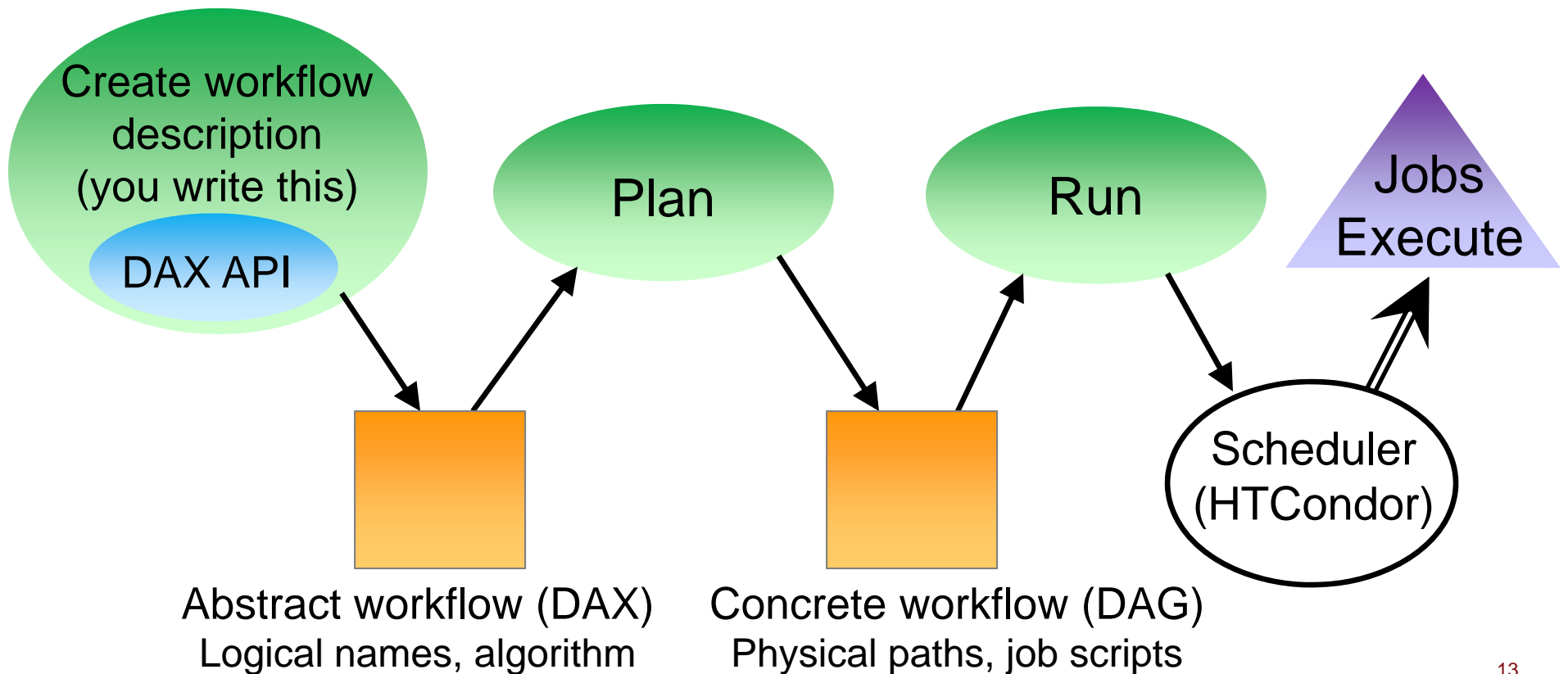
```
//Create DAX object
dax = ADAG("test_dax")
//Define my job
myJob = Job(name="my_job")
//Input and output files to my job
inputFile = File("input.txt")
outputFile = File("output.txt")
//Arguments to my_job (./my_job input=input.txt output=output.txt)
myJob.addArgument("input=input.txt", "output=output.txt")
//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 (“Planning”)

- DAX is “abstract workflow”
  - Logical filenames and executables
  - Algorithm description
- Use Pegasus to “plan” workflow for execution
  - Uses catalogs to resolve logical names, compute info
  - Pegasus automatically augments workflow
    - Staging jobs (if needed) with GridFTP or Globus Online
    - 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

# Pegasus Workflow Path

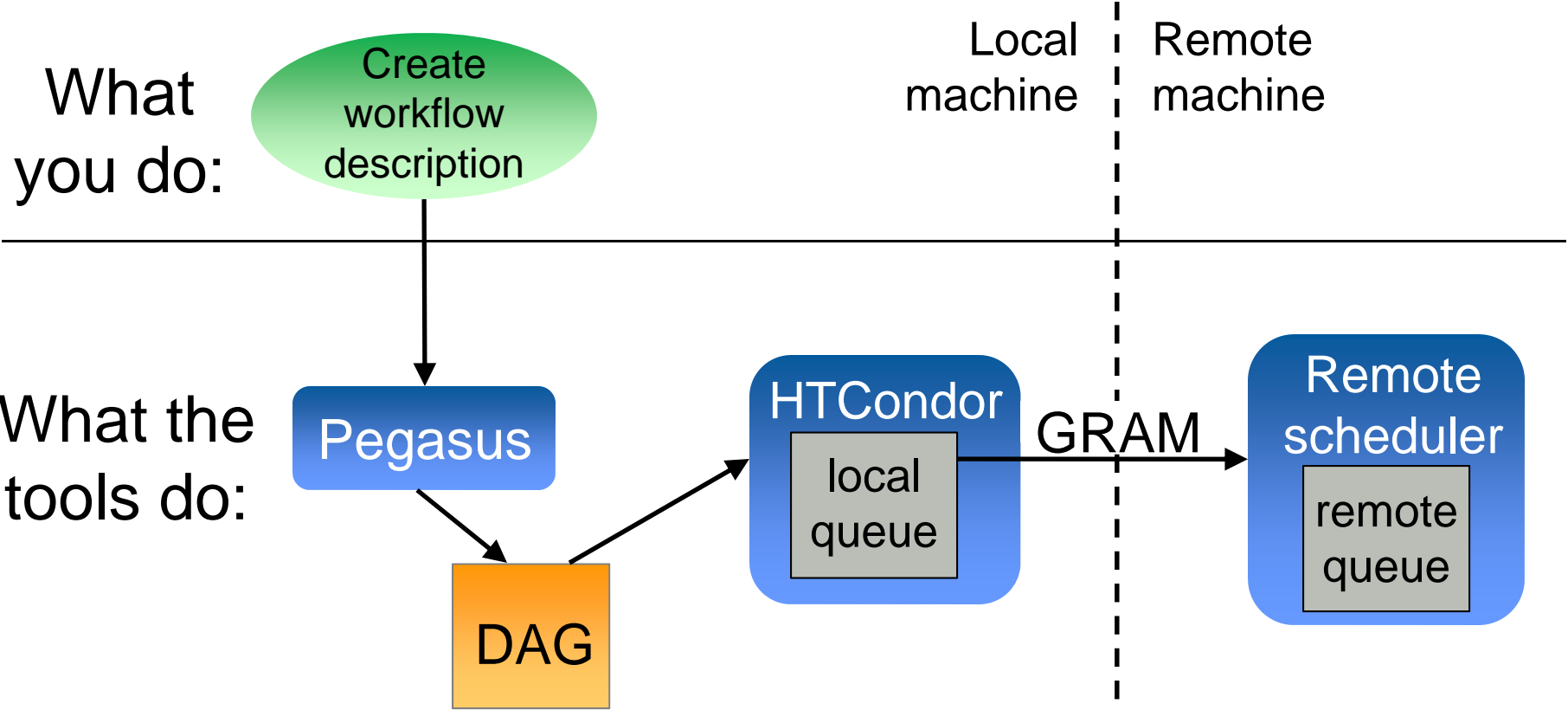




## Other tools in stack

- **HTCondor (UW Madison)**
  - Pegasus ‘submits’ workflow to HTCondor
  - Supervises runtime execution of DAG files
    - Maintains queue
    - Monitors dependencies
    - Schedules jobs
    - Retries failures
    - Writes checkpoint
- **GRAM (Globus Toolkit)**
  - Uses certificate-based authentication for remote job submission
  - Supported by many HPC resources

# Full workflow stack





## Other Workflow Tools

- **Regardless of the tool, trying to solve same problems**
  - Describe your workflow (Pegasus “Create”)
  - Prepare your workflow for the execution environment (Pegasus “Plan”)
  - Send jobs to resources (HTCondor, GRAM)
  - Monitor the execution of the jobs (HTCondor DAGMan)
- **Brief overview of some other available tools**



## Other Workflow Tools

- **Swift (U of Chicago)**

- Workflow defined via scripting language

```
//Create new type
type messagefile;
//Create app definition, returns messagefile
app (messagefile t) greeting() {
    //Print and pipe stdout to t
    echo "Hello, world!" stdout=@filename(t);
}
//Create a new messagefile, linked to hello.txt
messagefile outfile <"hello.txt">
//Run greeting() and store results
outfile = greeting();
```

- Workflow compiled internally and executed
- Focus on large data, many tasks

- **Askalon (U of Innsbruck)**

- Create workflow description
  - Use workflow language
  - Or use UML editor to graphically create
- Conversion: like planning, to prep for execution
- Submit jobs to Enactment Engine, which distributes jobs for execution at remote grid or cloud sites
- Provides monitoring tools

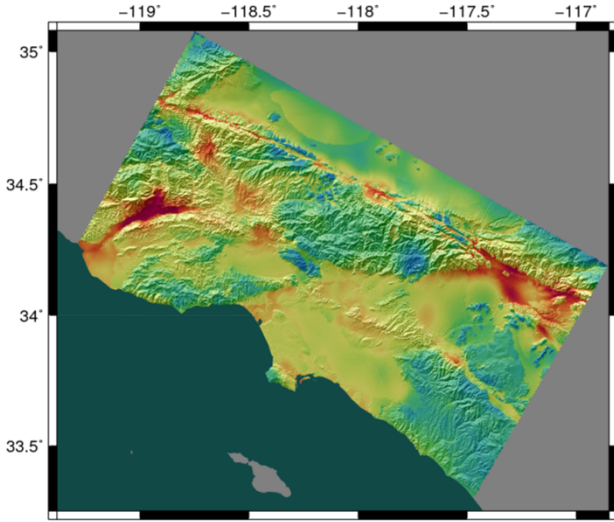
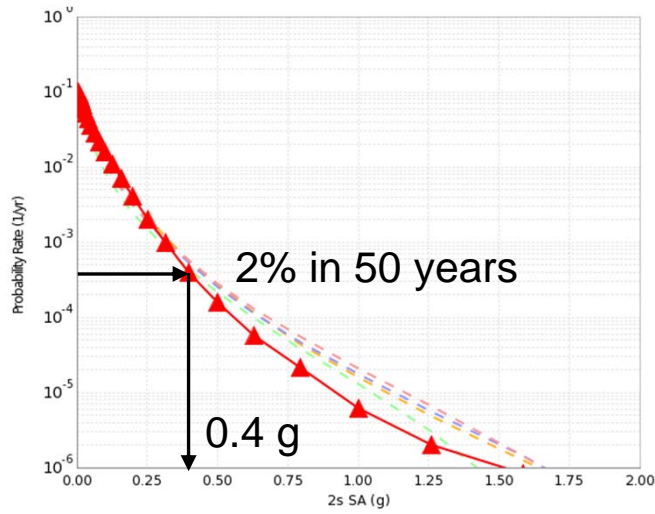


## More Workflow Tools

- **Kepler (diverse US collaboration)**
  - GUI interface
  - Many models of computation ('actors') with built-in components (tasks)
- **RADICAL Cybertools**
  - Sits atop SAGA, a Python API for submitting remote jobs
  - Uses pilot jobs to provision resources
- **UNICORE (Jülich Supercomputing Center)**
  - GUI interface to describe workflow
  - Branches, loops, parallel loops
- **Many more: ask me about specific use cases**
- **NCSA Blue Waters has webinars on several tools**

# Workflow Application: CyberShake

- What will peak ground motion be over the next 50 years?
  - Used in building codes, insurance, government, planning
  - Answered via Probabilistic Seismic Hazard Analysis (PSHA)
  - Communicated with hazard curves and maps





# CyberShake Computational Requirements

- Determine shaking of ~500,000 earthquakes per site
- Large parallel jobs
  - 2 GPU wave propagation jobs, 800 nodes x 1 hr, 1.5 TB output
- Small serial jobs
  - 500,000 seismogram calculation jobs, 1 core x 4.7 min, 30 GB
- Need ~300 sites for hazard map
- Decided to use scientific workflows
  - Automation
  - Data management
  - Error recovery



## Challenge: Resource Provisioning

- For large parallel jobs, submit to remote scheduler
  - GRAM (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 too
- Solution: Pegasus-mpi-cluster (PMC)



## Pegasus-mpi-cluster

- **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

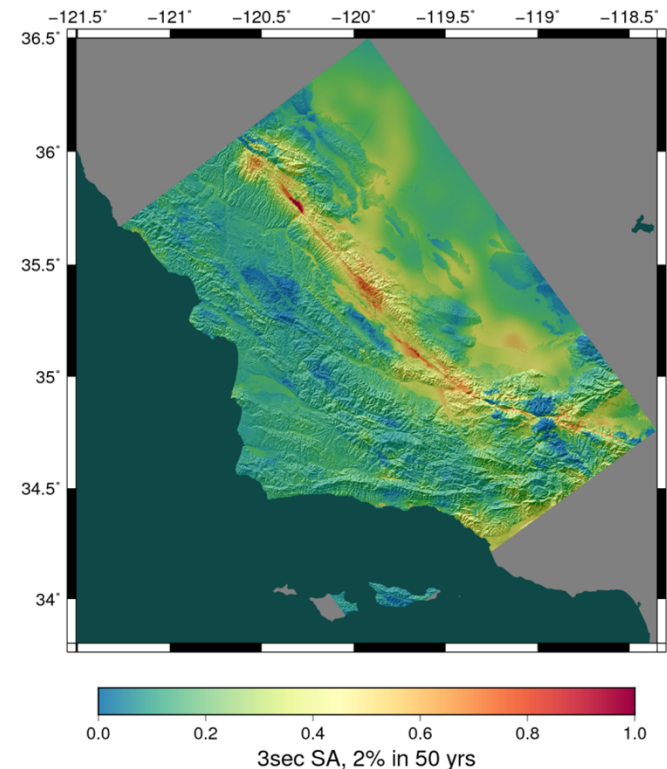


## Challenge: Data Management

- 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
  - Included as new jobs in workflow

## CyberShake Study 17.3

- Hazard curves for 876 sites
- Used OLCF Titan and NCSA Blue Waters
- Averaged 1295 nodes (CPUs and GPUs) for 31 days
  - Workflow tools scheduled 15,581 jobs
  - 23.9 workflows running concurrently
- Generated 285 million seismograms
- Workflow tools managed 777 TB of data
  - 308 TB of intermediate data transferred
  - 10.7 TB (~17M 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 cores**
  - 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
- Provides framework to assemble community codes
- Scales from local computer to large clusters
- Provide portable algorithm description independent of data
  - CyberShake 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/CC, 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/>
- Globus: <http://www.globus.org/>
- Swift: <http://swift-lang.org>
- Askalon: <http://www.dps.uibk.ac.at/projects/askalon/>
- Kepler: <https://kepler-project.org/>
- RADICAL Cybertools: <https://radical-cybertools.github.io/>
- UNICORE: <http://www.unicore.eu/>
- CyberShake: <http://scec.usc.edu/scecpedia/CyberShake>



# Questions?



# CyberShake Computational Requirements

