Workflow Tools

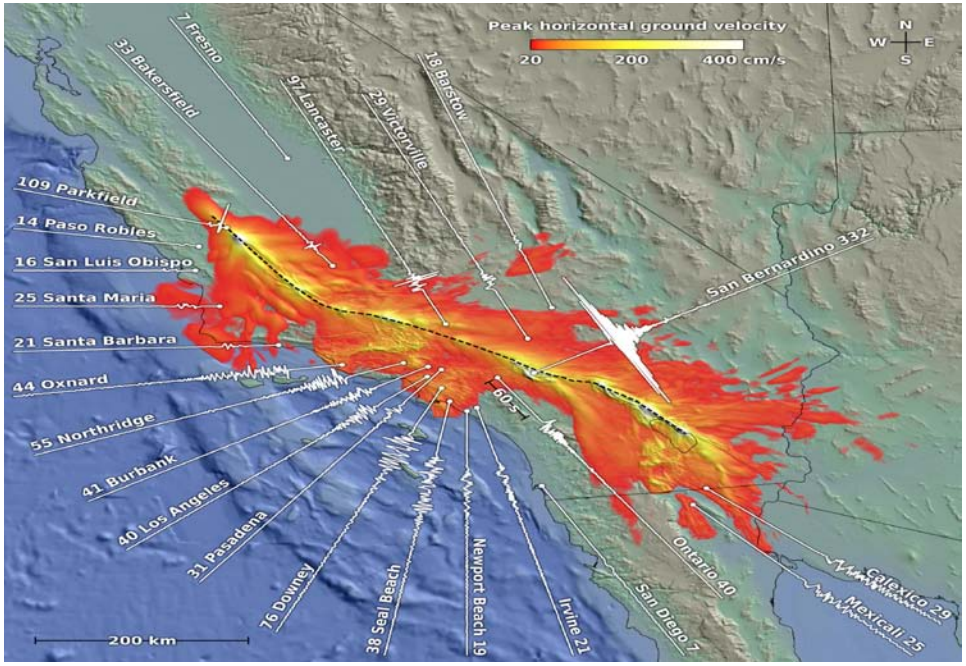
EU/US HPC Summer School
June 27, 2012

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

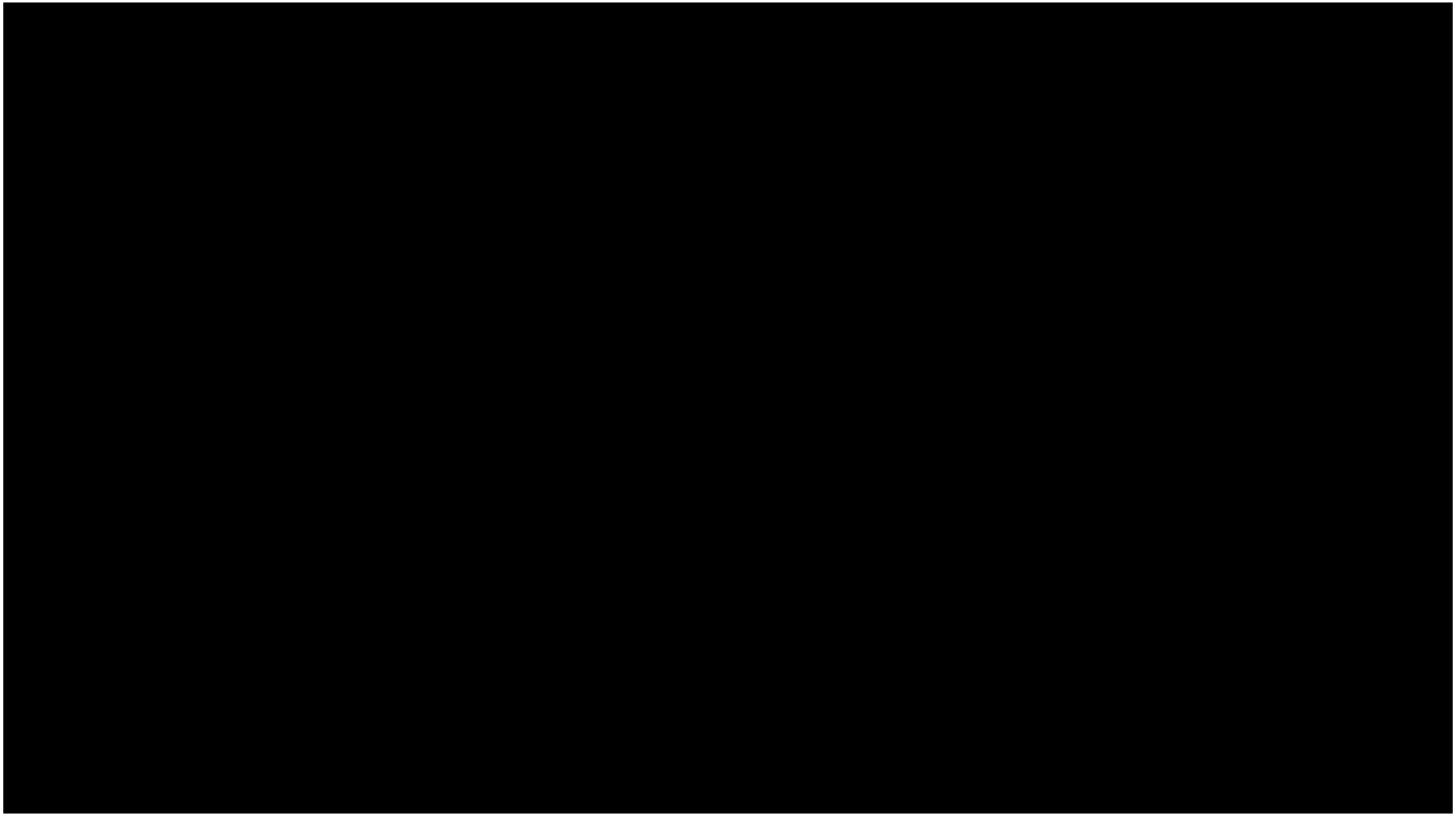
SCEC
an NSF+USGS center

Southern California Earthquake Center

- Collaboration of 600+ scientists at 60+ institutions
- Studying earthquake system science
- Focusing on Southern California
- Includes computational models of earthquake processes
  - Wide range of scales
  - Single earthquakes
  - Hazard at locations

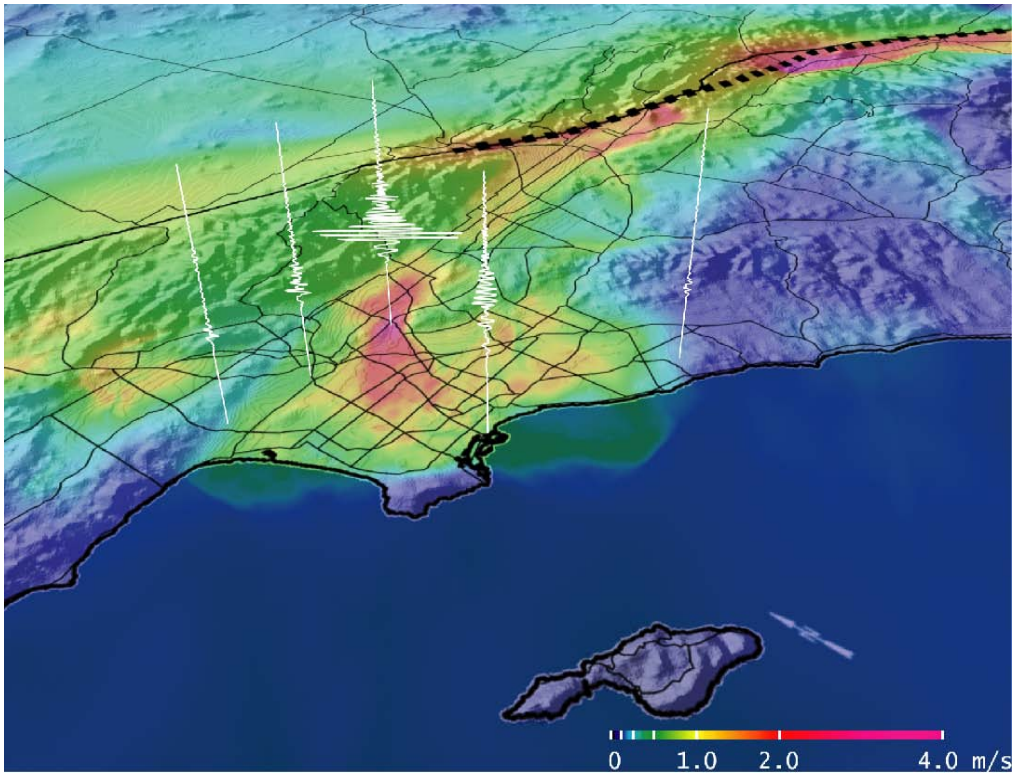


For example



What do I do?

- Integrate codes and scale up earthquake simulations
- Run high throughput workflows on large clusters



Overview

- What are scientific workflows?
- Available workflow tools
  - GUI tools
  - Scripting tools
- CyberShake (geoscience application)
  - Computational overview
  - Challenges and solutions
- Conclusions for 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
  - Short or long
  - Loosely or tightly coupled
- Independence of workflow process and data
  - Often, run same workflow with different data
  - Workflow could be data-dependent
- You use workflows all the time…

Sample Workflow

```
#!/bin/bash
```

1) Stage-in input data to compute environment

```
scp myself@datastore.com:/data/input.txt /scratch/input.txt
```

2) Run a serial job with an input and output

```
bin/pre-processing in=input.txt out=tmp.txt
```

3) Run a parallel job with the resulting data

```
mpiexec bin/parallel-job in=tmp.txt out_prefix=output
```

4) Run a set of independent serial jobs in parallel – scheduling by hand

```
for i in `seq 0 $np`; do
       bin/integrity-check output.$i &
done
```

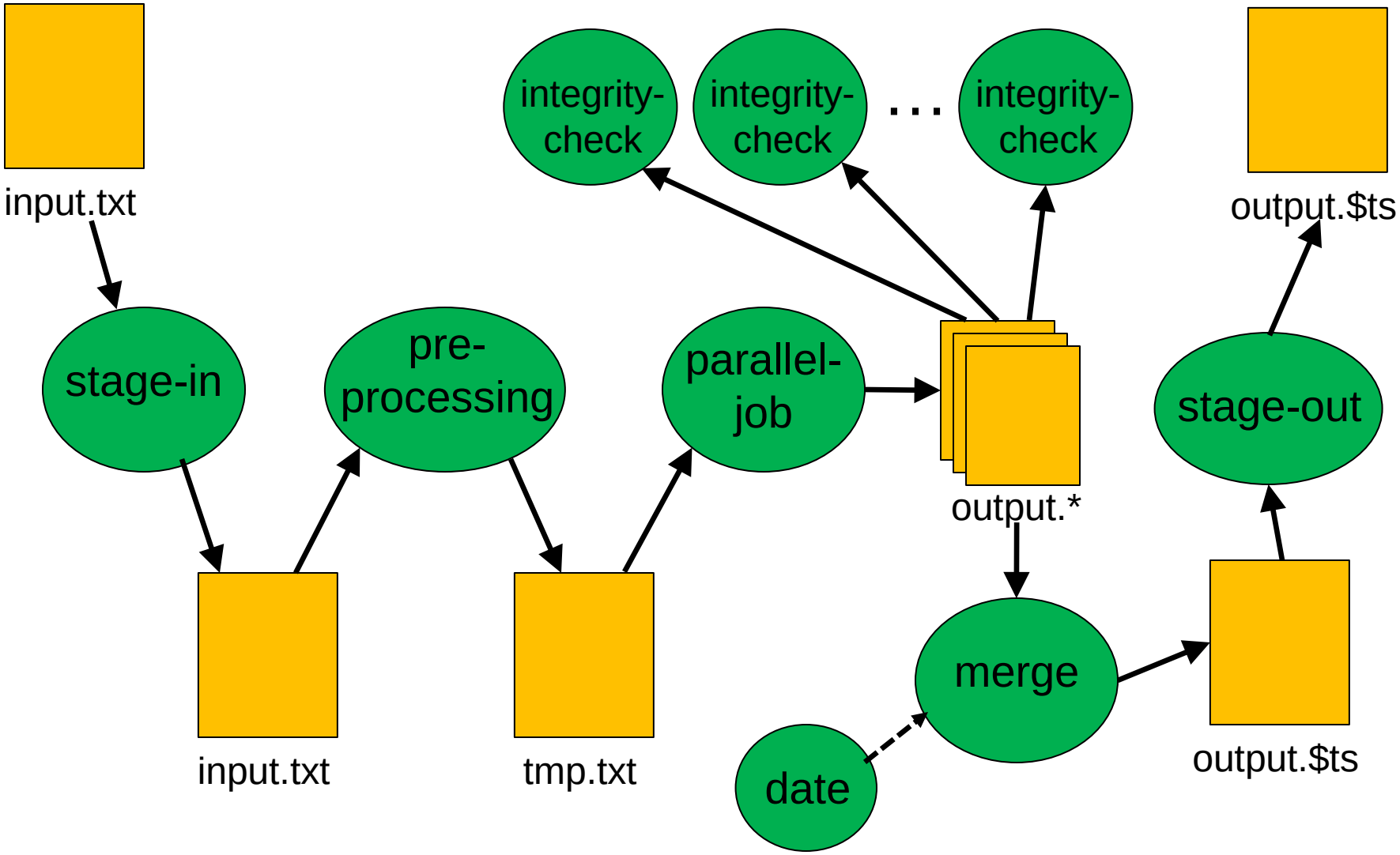
5) While those are running, get metadata and run another serial job

```
ts=`date +%s`
bin/merge prefix=output out=output.$ts
```

6) Finally, stage results back to permanent storage

```
scp /scratch/output.$ts myself@datastore.com:/data/output.$ts
```

Could think of shell script like…



Workflow Components

- Task executions
  - Provide 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
  - May have complicated dependencies
- Task scheduling
  - Minimize execution time while preserving dependencies
- Metadata
  - Automatically track
- Getting cores

Types of Tools

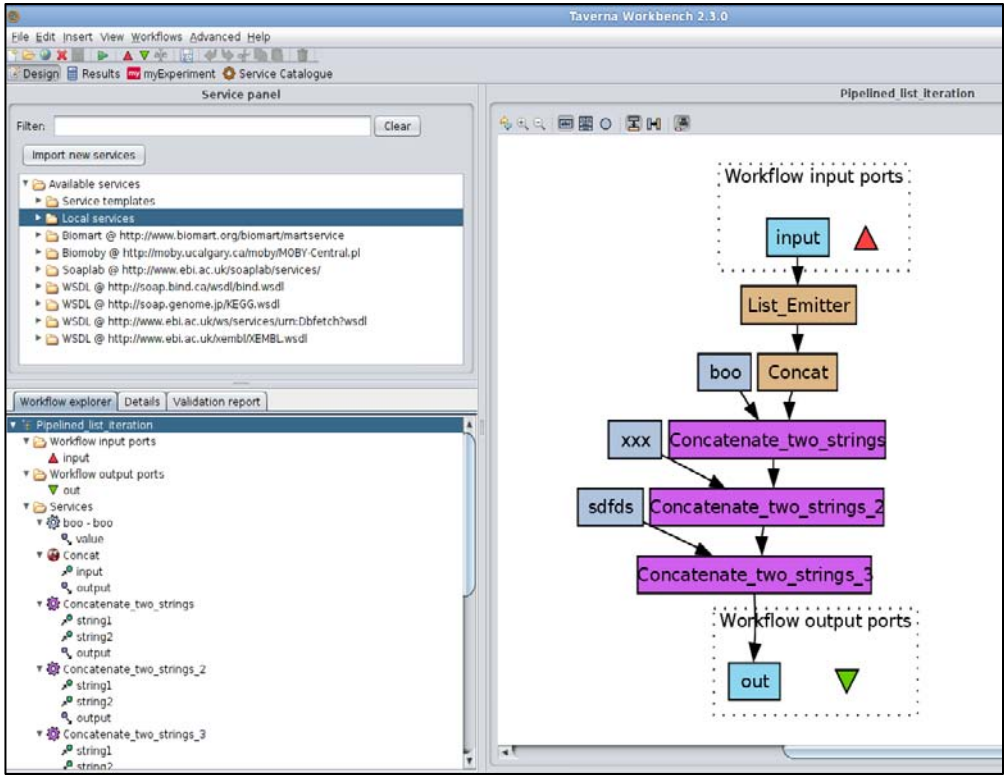
- Workflow management systems take care of these concerns
- GUI-based (generally targeted at medium-scale)
  - Kepler
  - Taverna, Triana, VisTrails
- Scripting (generally better scalability)
  - Pegasus (and Condor)
  - Swift
- Most tools are free and open source
- Not a complete list!

Kepler

- Developed by NSF-funded Kepler/CORE team (UCs)
- Actor and Director model
  - Actors = tasks
  - Director = controls execution of tasks
    - Serial, parallel, discrete time modeling
- Many built-in math and statistics modules
- Generally, execution on local machine
- More commonly used with ecology, geology
- Extensive documentation

Other GUI tools

- Taverna
  - Developed by myGrid (funded by OMII-UK)
  - Runs workflow to minimize completion time
  - Commonly used in life science community
- Triana
  - Signal analysis, image manipulation
- VisTrails
  - Visualization



Scripting Tools

- Define workflow via programming
- Can support large numbers of tasks
- Provide many kinds of fancy features and capabilities
  - More flexibility
  - More complex
- Today, simple overview
- Will focus on Pegasus, but concepts are shared

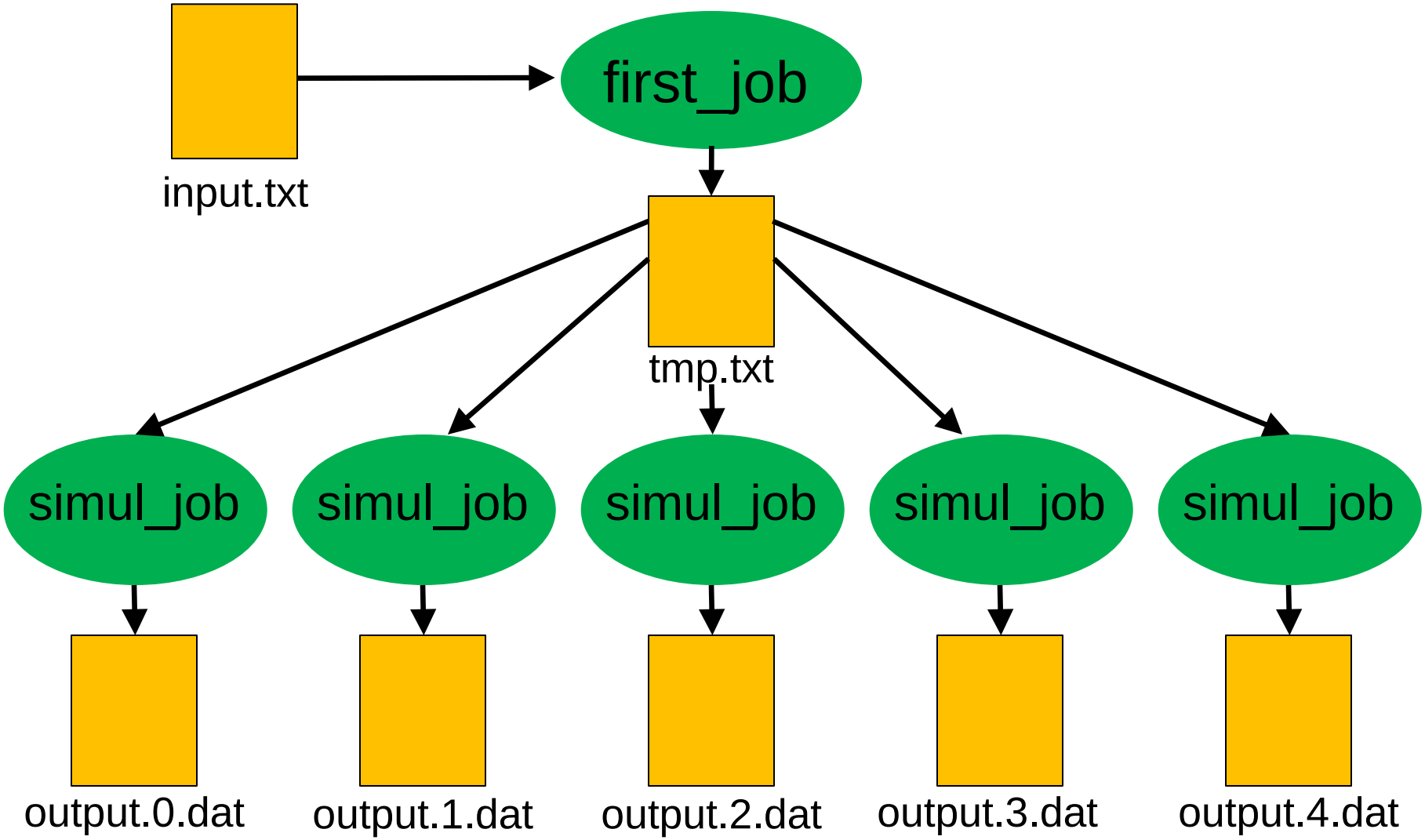
Pegasus

- Developed at USC's Information Sciences Institute
- Designed to address our earlier problems:
  - Task execution
  - Data and control dependencies
  - Data and metadata management
  - Error recovery
- Uses Condor DAGMan for
  - Task scheduling
  - Resource provisioning

Pegasus Concepts

- Separation of “submit host” and “execution site”
  - Create workflow using code on your local machine
  - Can run on local machine or on distributed resources
- Workflow represented with directed acyclic graphs
- You use API to write code describing workflow
  - Python, Java, Perl
  - Tasks with parent / child relationships
  - Files and their roles
  - Can have nested workflows
- Pegasus creates XML file of workflow called a DAX

Sample Workflow



Sample DAX Generator

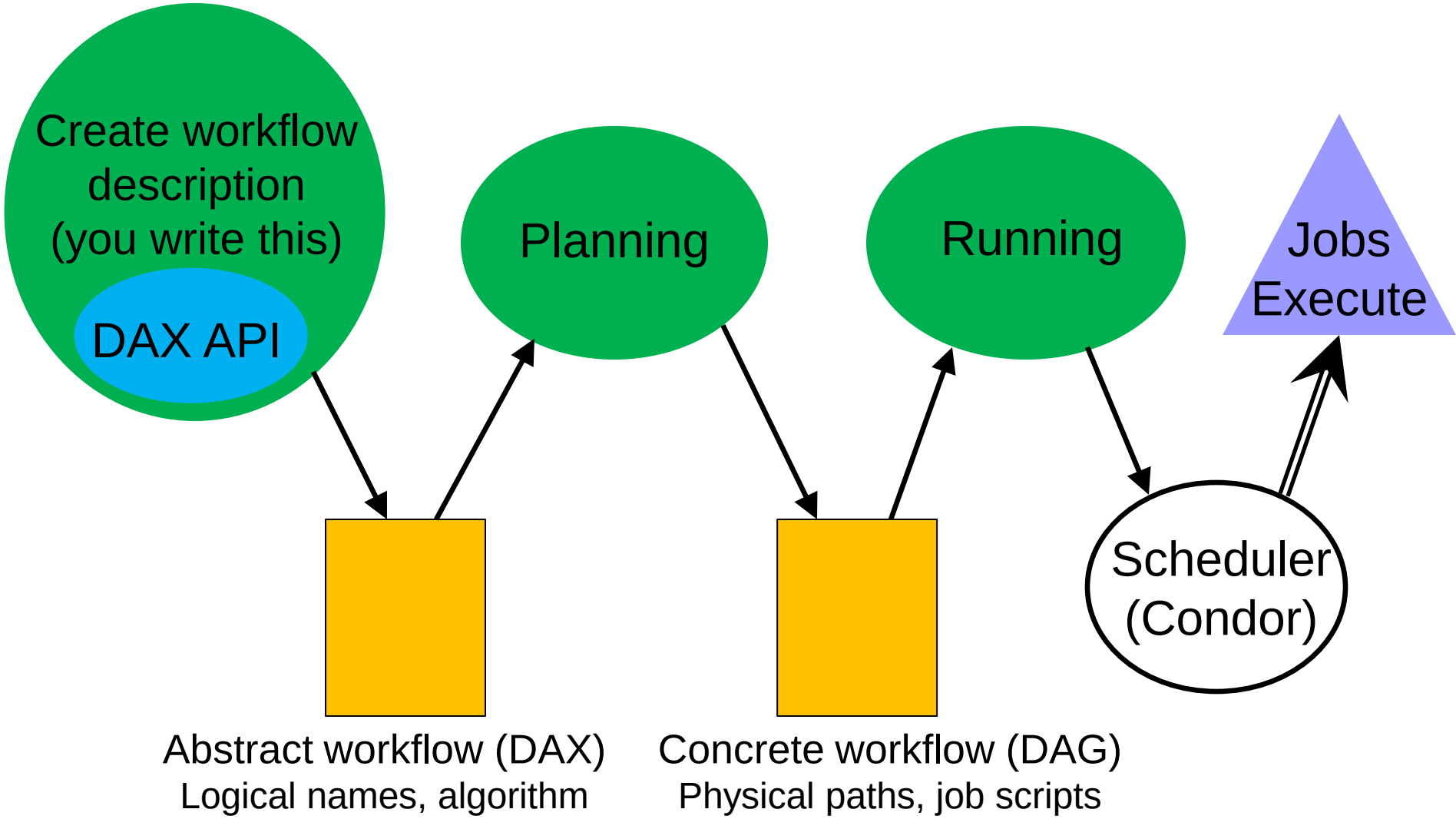
```
public static void main(String[] args) {
    //Create DAX object
    ADAG dax = new ADAG("test_dax");
    //Define first job
    Job firstJob = new Job("0", "my_namespace", "first_job", "v1.0");
    //Input and output files to first job
    File firstInputFile = new File("input.txt");
    File firstOutputFile = new File("tmp.txt");
    //Arguments to first_job (first_job input=input.txt output=tmp.txt)
    firstJob.addArgument("input=input.txt");
    firstJob.addArgument("output=tmp.txt");
    //Role of the files for the job
    firstJob.uses(firstInputFile, File.LINK.INPUT);
    firstJob.uses(firstOutputFile, File.LINK.OUTPUT);
    //Add the job to the workflow
    dax.addJob(firstJob);
```


```
  for (int i=0; i<5; i++) {
    //Create simulation job
    Job simulJob = new Job(i+1, "my_namespace", "simul_job", "v2.1");
    //Define files
    File simulInputFile = new File("tmp.txt");
    File simulOutputFile = new File("output." + i + ".dat");
    //Arguments to job
    //simulJob parameter=<i> input=tmp.txt output=output<i>.dat
    simulJob.addArgument("parameter=" + i);
    simulJob.addArgument("input=tmp.txt");
    simulJob.addArgument("output=" + simulOutputFile.getName());
    //Role of files
    simulJob.uses(simulInputFile, File.LINK.INPUT);
    simulJob.uses(simulOutputFile, File.LINK.OUTPUT);
    //Add job to dax
    dax.addJob(simulJob);
    //Dependency on firstJob
    dax.addDependency(firstJob, simulJob);
  }
  //Write to file
  dax.writeToFile("test.dax");
}
```

Planning

- DAX is “abstract workflow”
  - Logical filenames and executables
  - Algorithm description
- Prepare workflow to execute on certain system
- Use Pegasus to “plan” workflow
  - Uses catalogs to resolve logical names, compute info
  - Pegasus automatically augments workflow
    - Stages jobs (if needed) with GridFTP
    - 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 (Condor format)

Pegasus Workflow Path



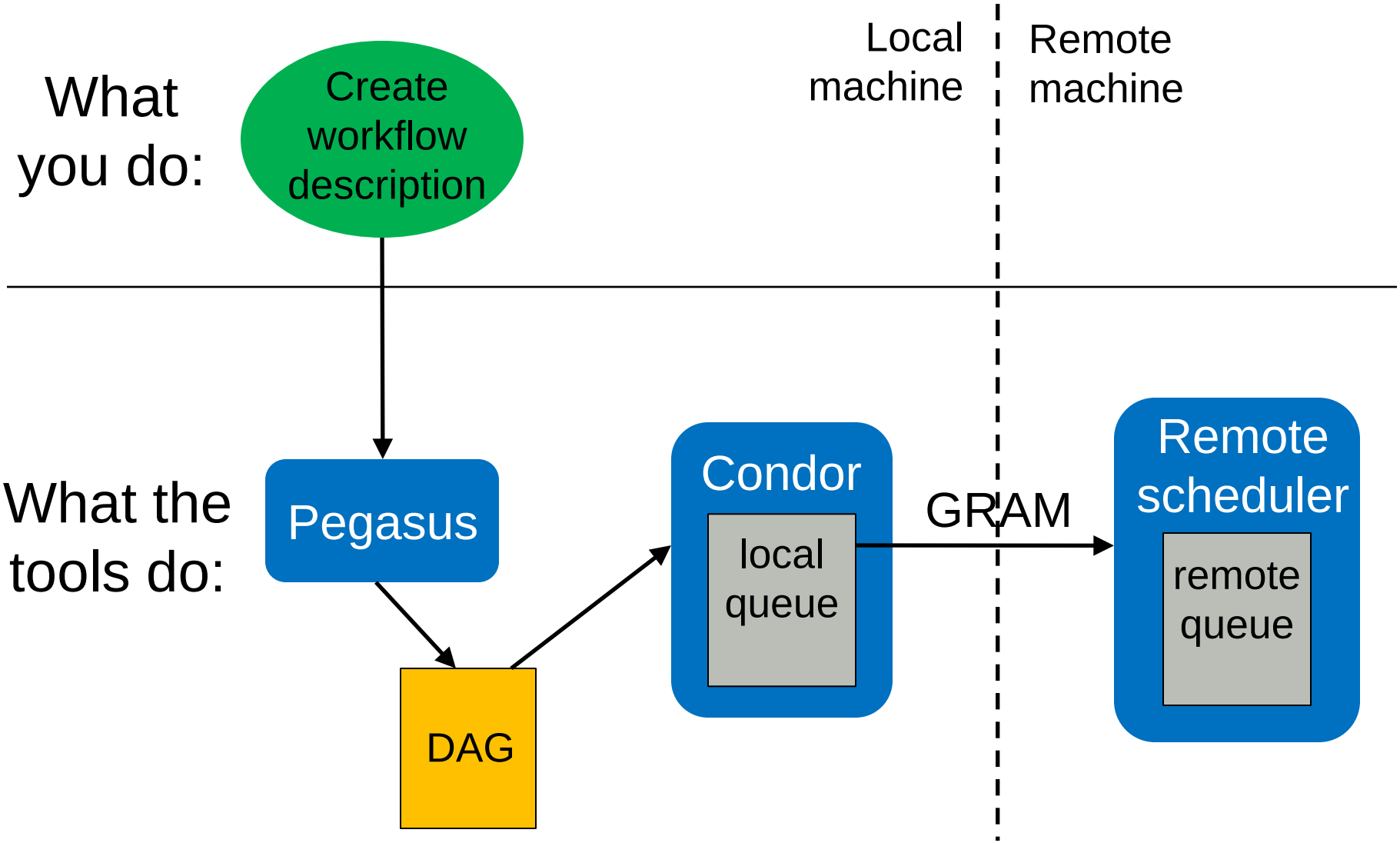
Running with Condor

- Developed by Condor group at U of Wisconsin
- Pegasus “submits” workflow to Condor DAGMan
  - Contains local queue of jobs
  - Monitors dependencies
  - Schedules jobs to resources
  - Automatically retries failed jobs
    - Writes rescue DAG to restart if job keeps failing
  - Updates status (jobs ready, complete, failed, etc.)
- Can run locally or on remote system
  - Condor-G uses GRAM to submit jobs to remote scheduler

GRAM

- Part of the Globus Toolkit
- Uses certificate-based authentication
  - Like gsissh
  - Requires X509 certificate and account on remote machine
- Enables submission of jobs into a remote queue
- Supported by many university and XSEDE resources
  - Lonestar, Ranger, Kraken (for example)

Pegasus/Condor/GRAM stack



Other Tools

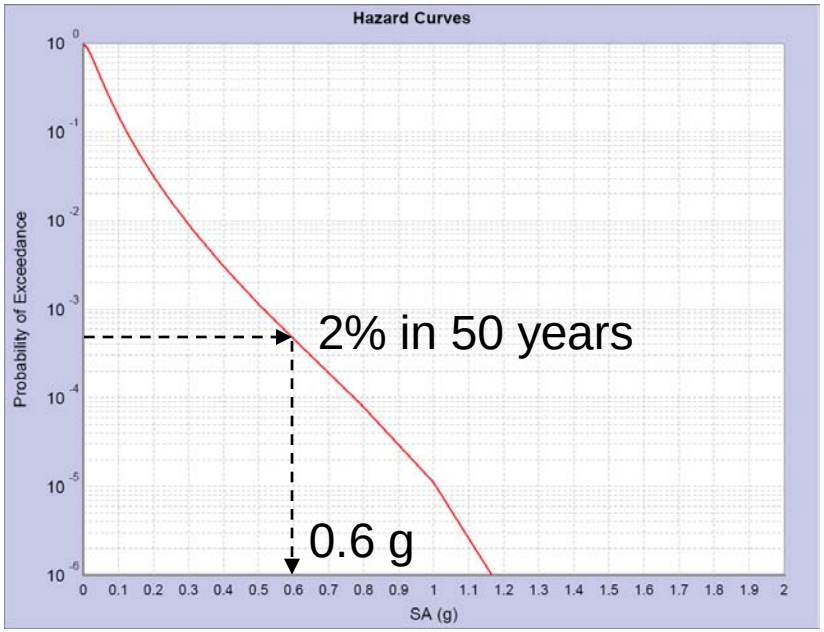
- Swift
  - Similar, but workflow defined via scripting language

```
type messagefile;

app (messagefile t) greeting() {
    echo “Hello, world!” stdout=@filename(t);
}
messagefile outfile <“hello.txt”>

outfile = greeting();
```

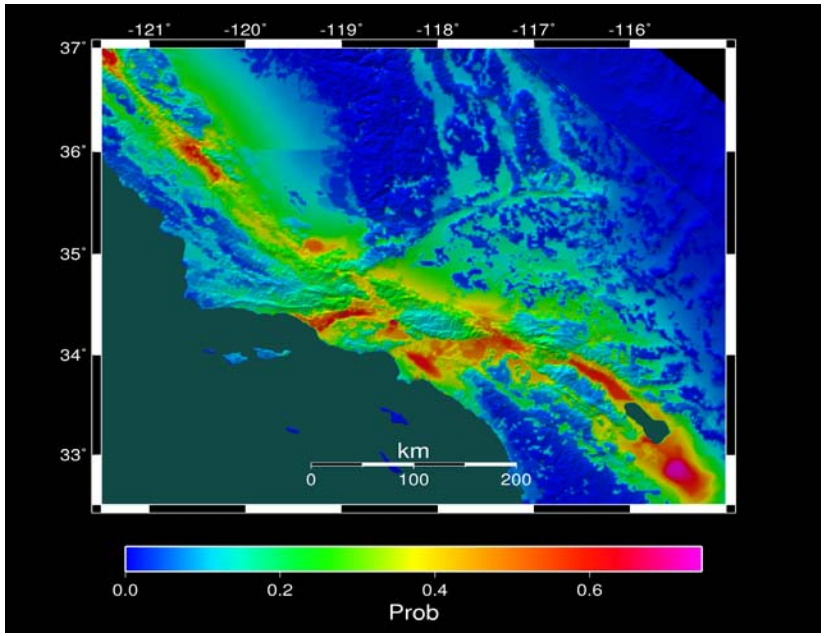
- Catalogs used to resolve executables and resources
- Workflow compiled internally and executed
- Which tool is better depends on the app and you

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



Hazard curve for downtown LA



Probability of exceeding 0.1g in 50 yrs

How to do PSHA

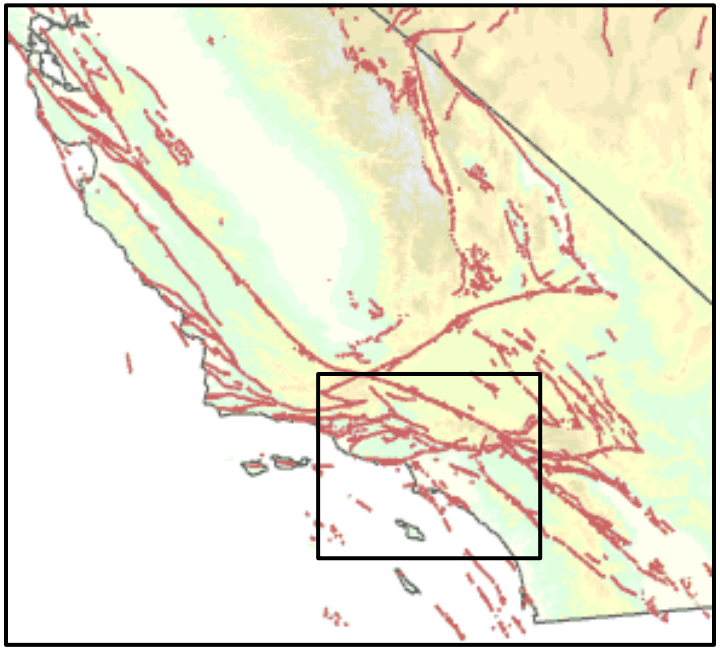
1. Pick a location of interest.
2. Define what future earthquakes might happen.
3. Estimate the magnitude and probability for each earthquake.
4. Determine the shaking caused by each earthquake at the site of interest.
5. Combine the shaking levels with the probabilities to produce a hazard curve.

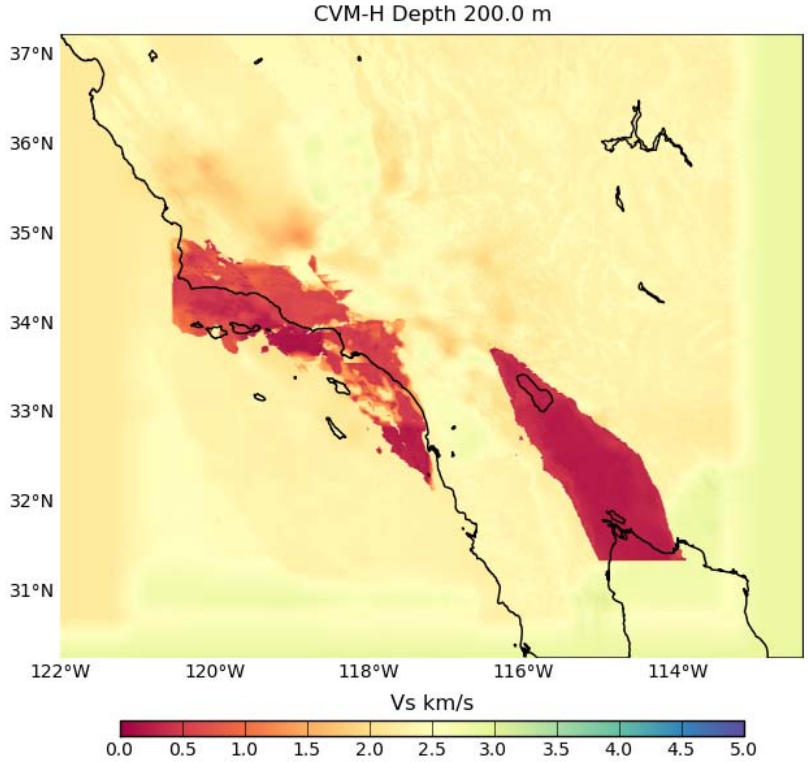
Repeat for multiple sites for a hazard map.

Typically performed with attenuation relationships created by fitting existing data.

Tensor Creation

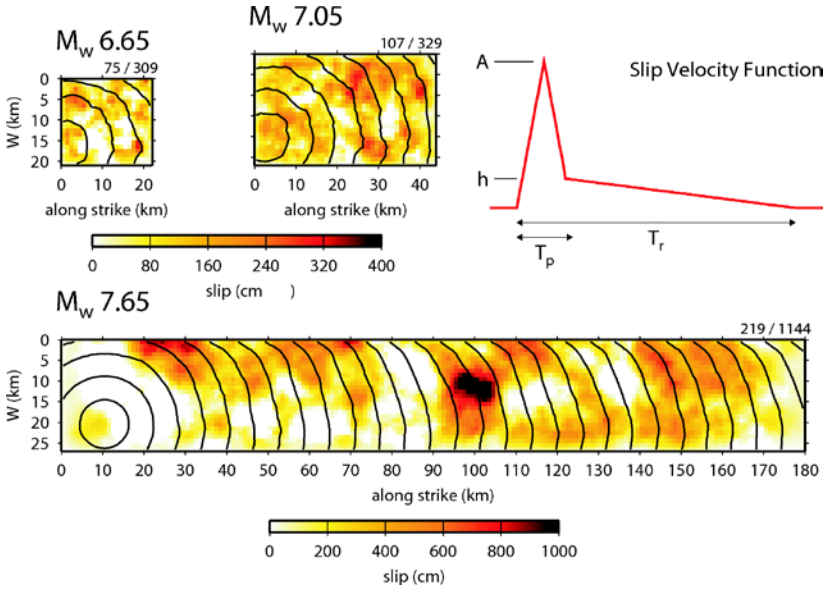
- Wave propagation simulation
  - Create 1.5 billion point mesh with material properties
  - Generate Strain Green Tensors for volume
  - Parallel, ~12,000 CPU-hrs

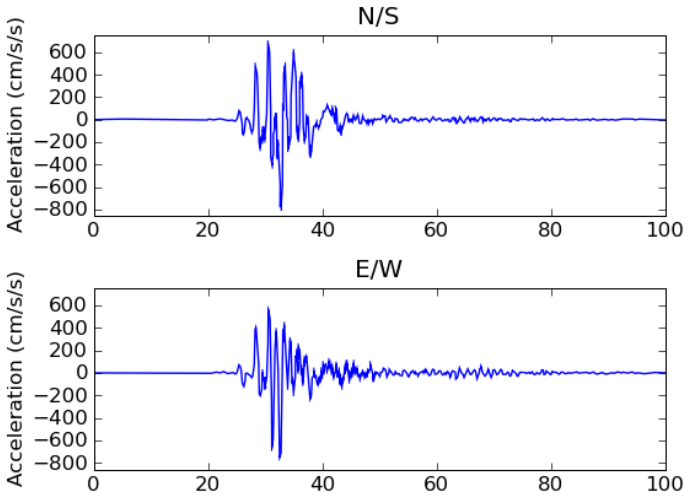




Post-Processing

- Individual earthquake contributions
  - Use “seismic reciprocity” to simulate seismograms for each of 400,000 earthquakes
  - Calculate peak shaking, combine for hazard curve
  - Loosely-coupled, short-running serial jobs





Computational Requirements

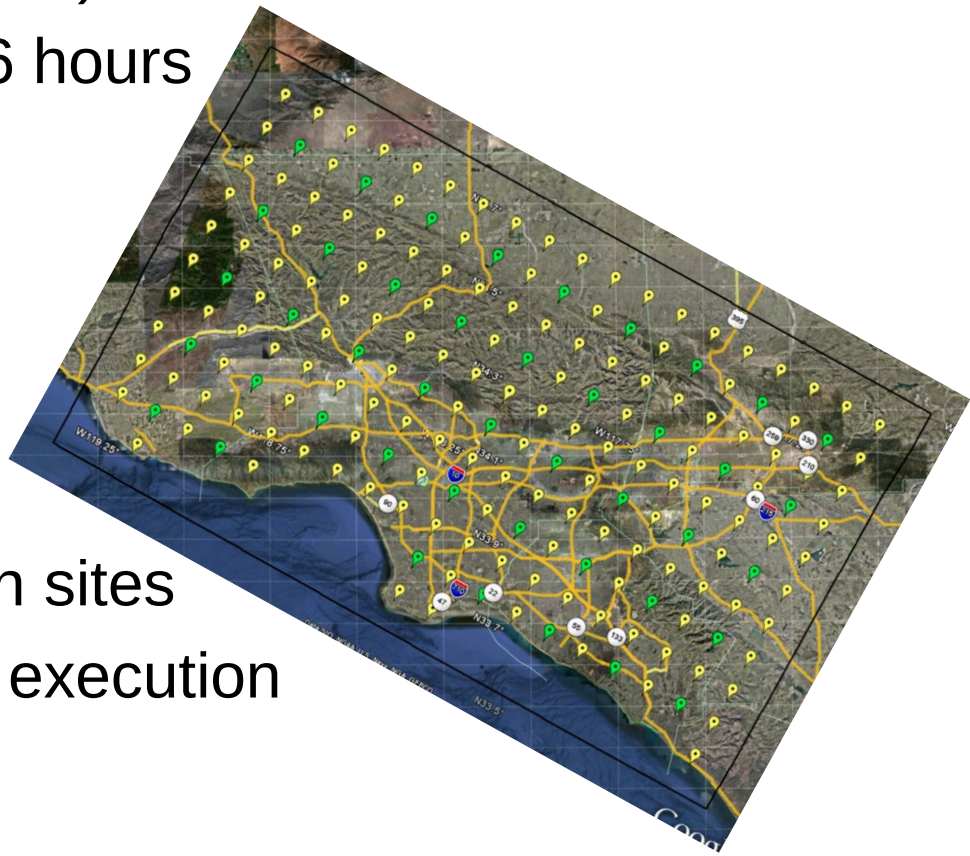
	Component	Data	Executions	Cores/exec	CPU hours
Tensor Creation	Mesh generation	15 GB	1	160	150
	Tensor simulation	40 GB	2	400	12,000
Post Processing	Tensor extraction	690 GB	7,000	1	250
	Seismogram synthesis	10 GB	**415,000**	1	1,600
	PSA calculation	90 MB	**415,000**	1	100
	Curve generation	1 MB	1	1	< 1
	Total	755 GB	837,000		14,100

This is for **one** location of interest; we wanted to run hundreds

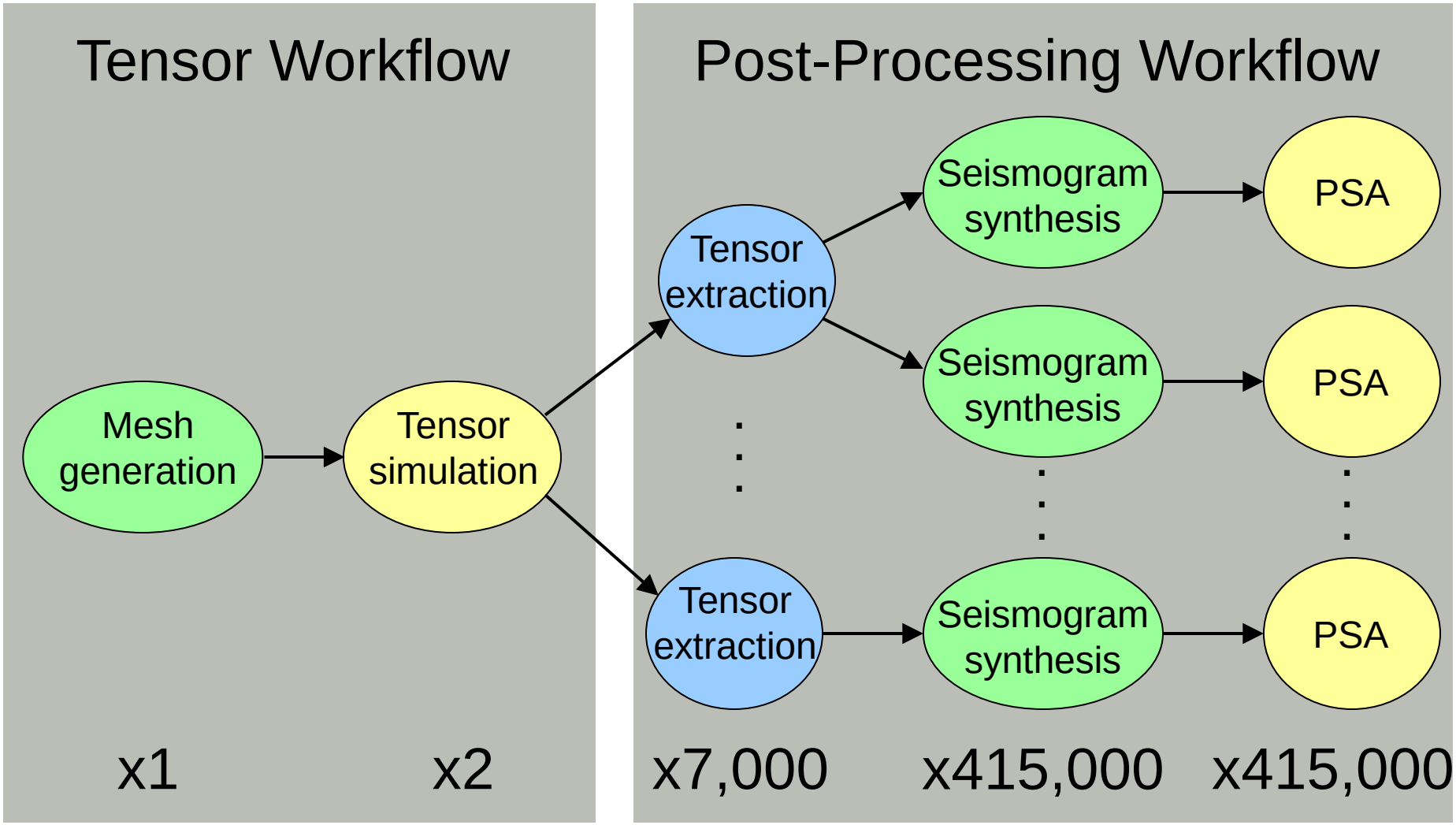
Recently decided to double the number of tasks

Why Scientific Workflows?

- Large-scale, heterogeneous, high throughput
  - Parallel and many (~840,000) serial tasks
  - Task duration 100 ms – 16 hours
- Automation
- Data management
- Error recovery
- Resource provisioning
  - Possibly multiple execution sites
  - Acquire grid resources for execution
- Scalable



CyberShake workflows

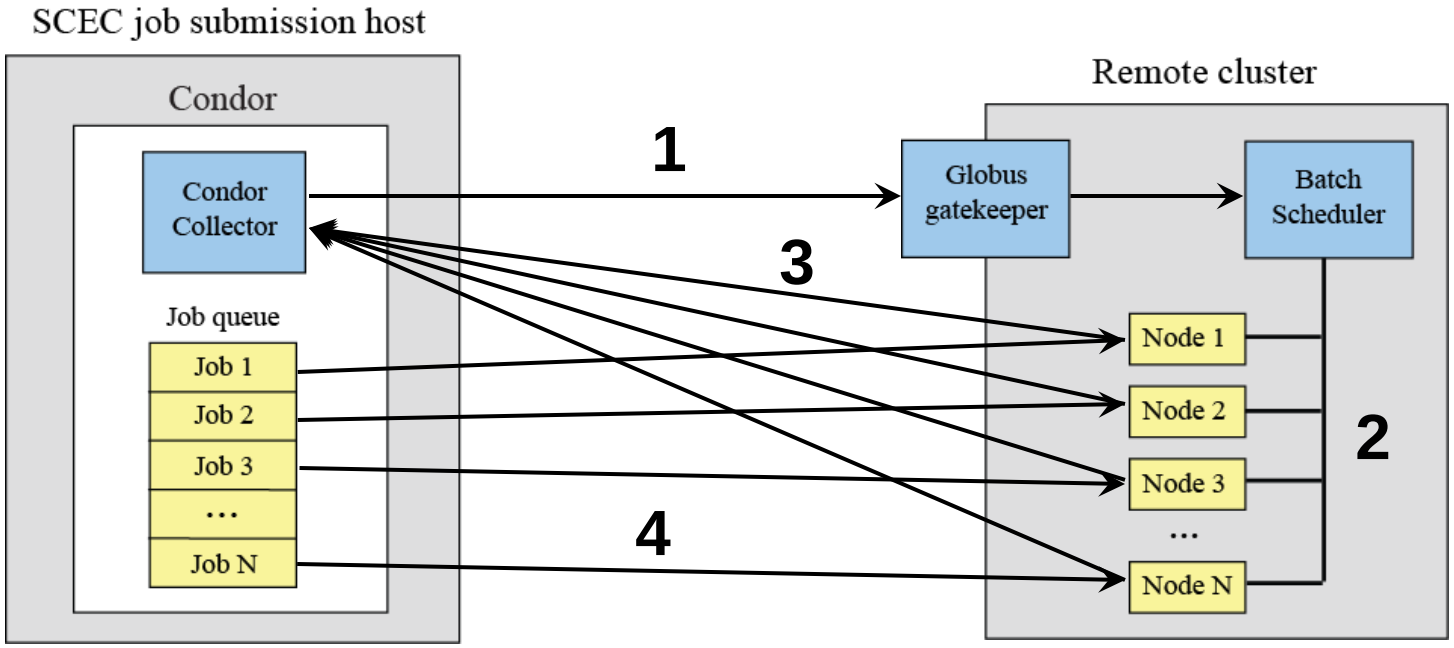


Challenge: Resource Provisioning

- In first workflow, submit job to remote scheduler
  - GRAM puts jobs in remote queue
  - Runs like a normal batch job
- For post-processing workflow, need high throughput
  - Putting a million jobs in the batch queue is ill-advised
    - Scheduler isn't designed for that many jobs
    - Scheduler cycle is ~5 minutes
    - Policy limits too
- Condor glideins
- Pegasus-mpi-cluster

Condor Glideins

1. Request group of nodes; wait in remote queue
2. Job starts up
3. Acquired nodes call back to local submit host
4. Local submit host schedules directly to nodes



Local Load

- High Condor load caused by short serial runtimes
- Pegasus feature called clustering
  - Groups instances of the same task into 1 Condor job
  - Tasks execute serially inside job
  - Condor sees fewer jobs
- Clusters dependencies too
  - Balance size of clusters with potential parallelism
- Adjusted Condor scheduling parameters

Pegasus-mpi-cluster

- Wanted to target NICS Kraken
  - No public IPs
  - Nodes have minimal kernels (no shared libraries)
  - Can't use Glideins
- MPI wrapper around serial or thread-parallel jobs
  - Master-worker paradigm
  - Specify jobs in same manner, Pegasus does wrapping
- Uses intelligent scheduling
  - Core counts, memory requirements, priorities

Challenge: Data Management

- Millions of data files
  - Pegasus provides staging
    - Symlinks files if possible, transfers files if needed
    - Supports running parts of workflows on separate machines
  - Transfers output back to SCEC disk
  - Pegasus registers data products in catalog
- Added automated checks to find corruption
  - Correct number of files, NaN, zero-value checks
  - Included as new jobs in workflow

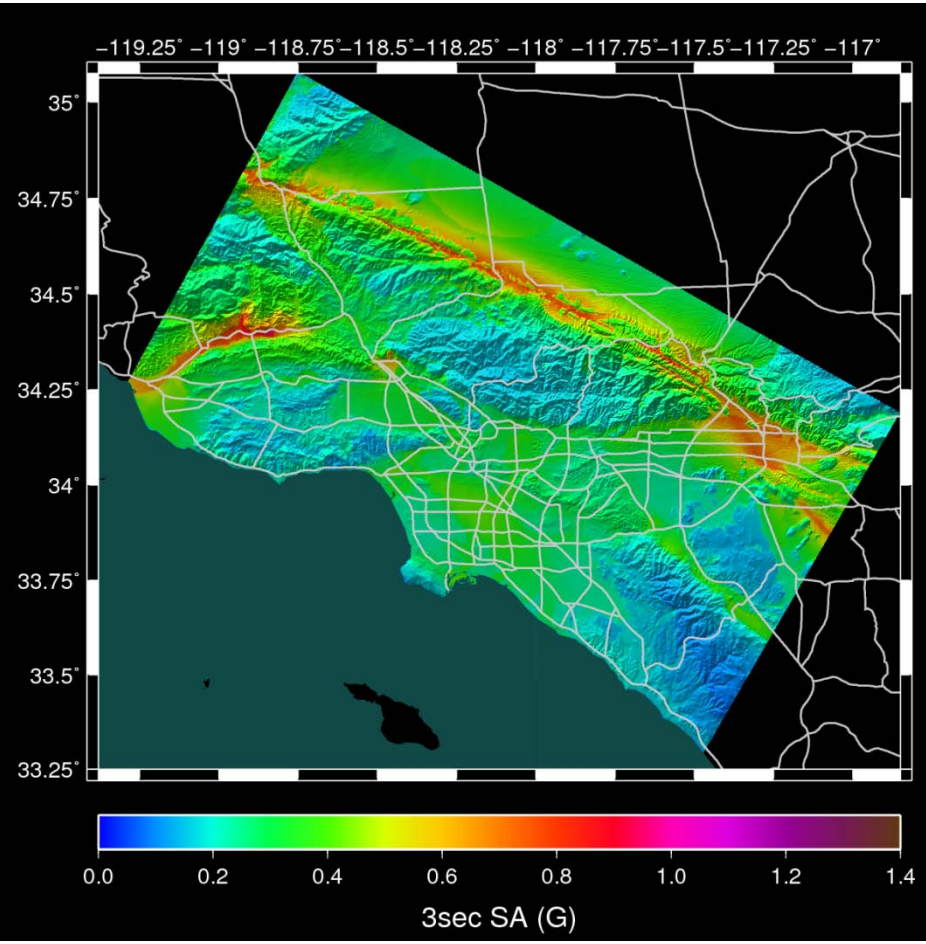
Challenge: File System Load

- Like large parallel jobs, multiple issues limit performance
- All 7000 extraction jobs read from same file
  - Read a 10 MB header, not big enough for striping
  - Used Pegasus to throttle extraction jobs
- Added memcached to cache file header
  - Local memory cache on compute node
  - Modified glidein startup script to start memcache daemon
  - Pegasus-mpi-cluster hook for custom startup script
- Expanded to use cache whenever possible

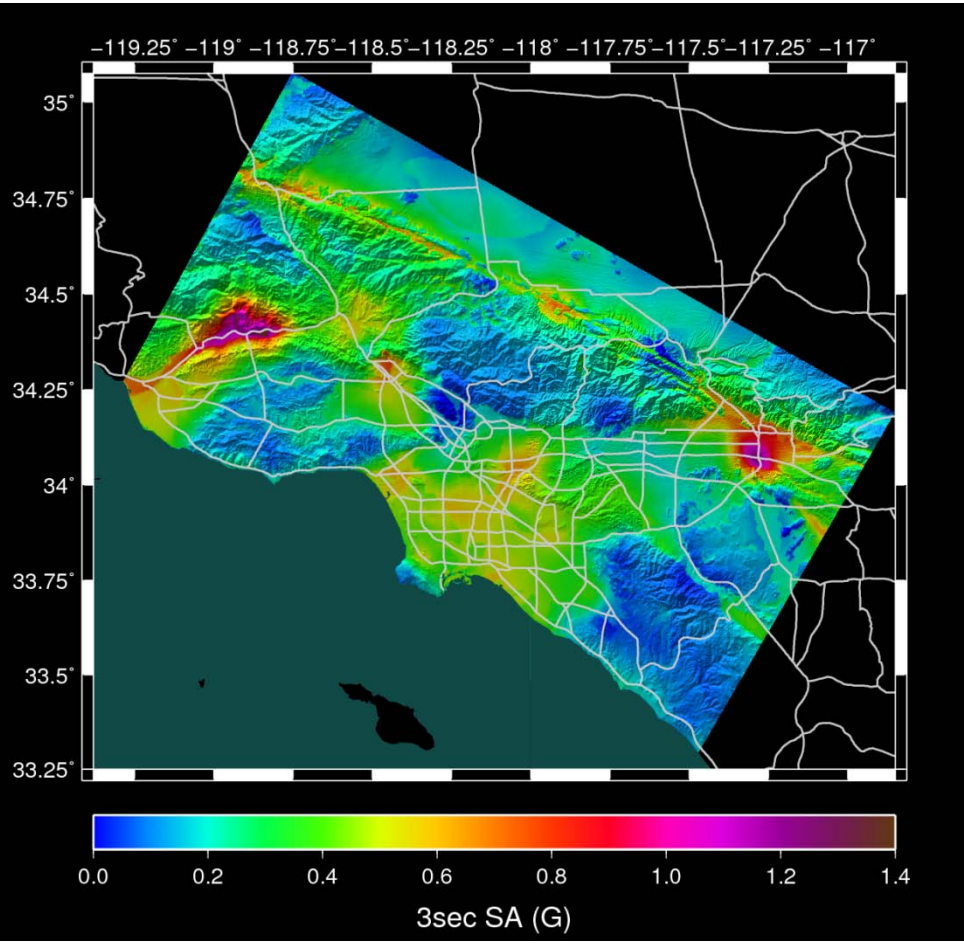
Science Run

- Hazard curves for 223 sites
- ~4500 processors for 54 days (TACC Ranger)
  - Limited by queuing policies
- 190 million tasks executed
  - 43 tasks/sec
  - 3.8 million Condor jobs, 289 failures
  - 3952 jobs in Ranger queue (including glideins)
- Managed 176 TB of data
  - 8.5 TB output files
  - 2.1 TB staged back to local disk (36,000 files)
- Future runs on Kraken
- Workflow tools scale!

Results

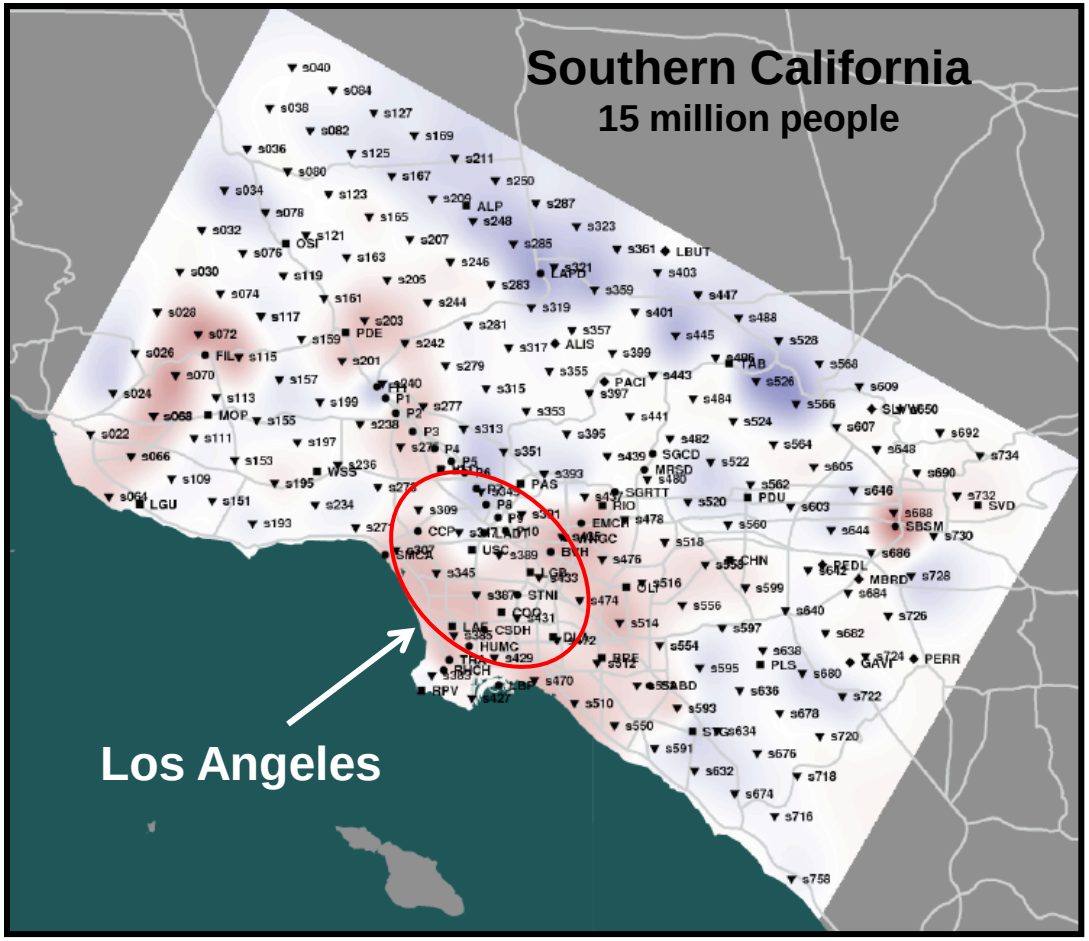


Attenuation map



CyberShake map

Results (difference)



CyberShake map compared to attenuation map – red is higher risk, blue lower

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
- Does add additional software layers and complexity
  - Some development time is required

Problems Workflows Solve

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

Things to keep in mind

- Put ALL processing steps in the workflow
  - Include validation, visualization, publishing, notifications
- Automation is vital
- Avoid premature optimization
- Consider new compute environments (dream big!)
  - Larger clusters
  - XSEDE / PRACE
  - Amazon EC2
- Tool developers want to help you!

Links

- SCEC: http://www.scec.org
- Kepler: http://kepler-project.org/
- Taverna: http://www.taverna.org.uk/
- Triana: http://www.trianacode.org/
- VisTrails: http://www.vistrails.org
- Pegasus: http://pegasus.isi.edu
- Condor: http://www.cs.wisc.edu/condor/
- Globus: http://www.globus.org/
- GridFTP: http://www.globus.org/toolkit/docs/latest-stable/gridftp/
- CyberShake: http://scec.usc.edu/scecpedia/CyberShake

Questions?

SCEC
an NSF+USGS center

ISI
Information Sciences Institute

XSEDE
Extreme Science and Engineering Discovery Environment

National Science Foundation

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

NICS

TACC

USGS
science for a changing world

OLCF
OAK RIDGE LEADERSHIP COMPUTING FACILITY

NCSA

USCGRID