Workflow Tools

International HPC Summer School

June 26, 2015

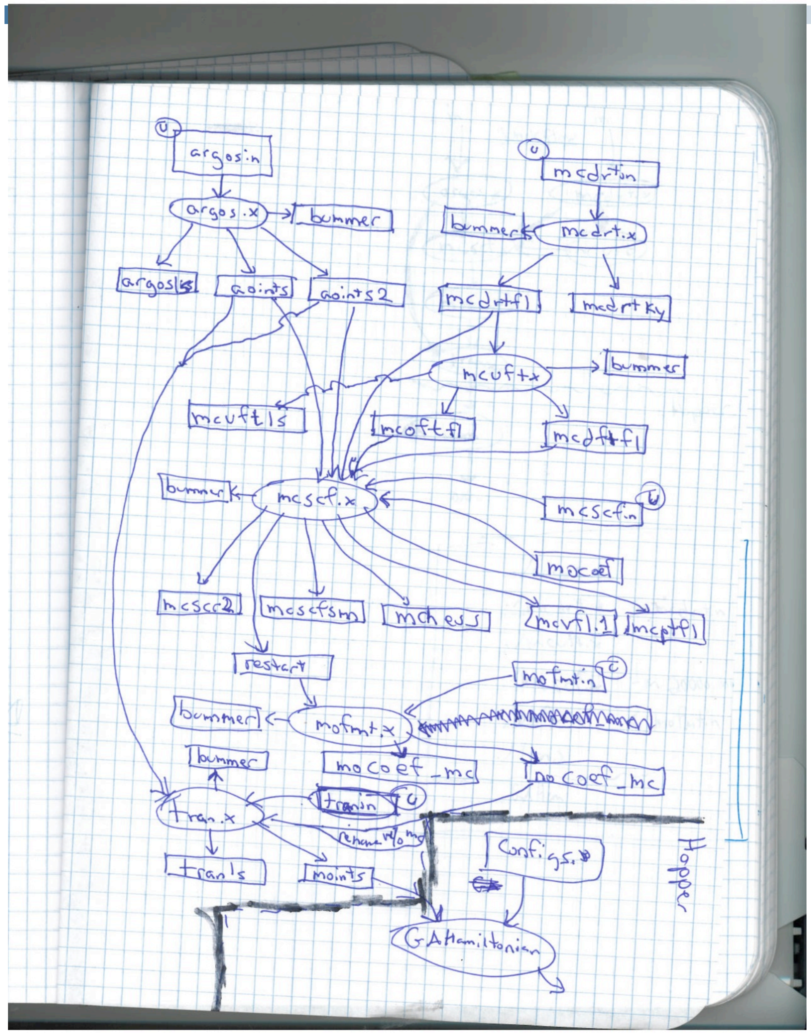
Gideon Juve

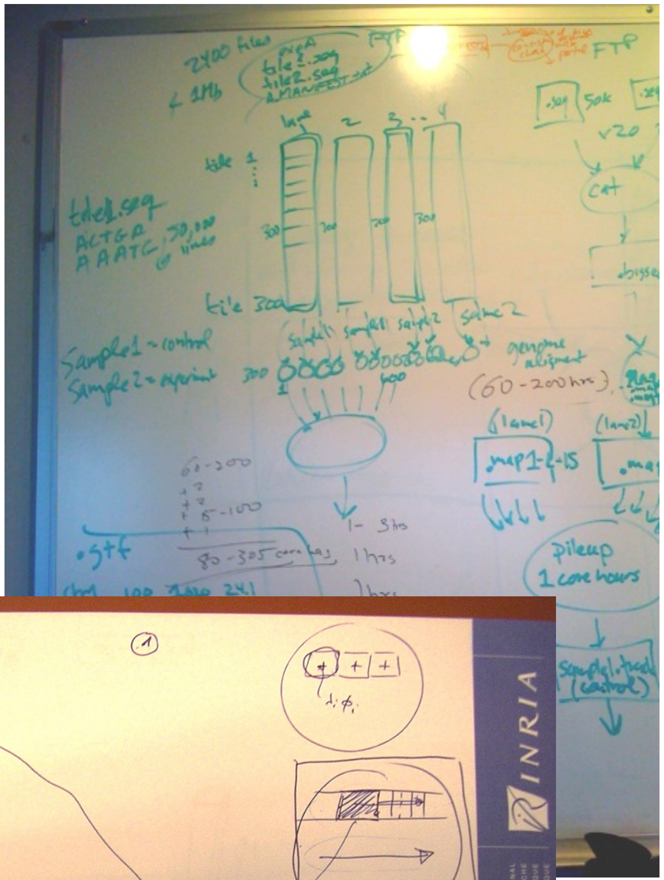
gideon@isi.edu

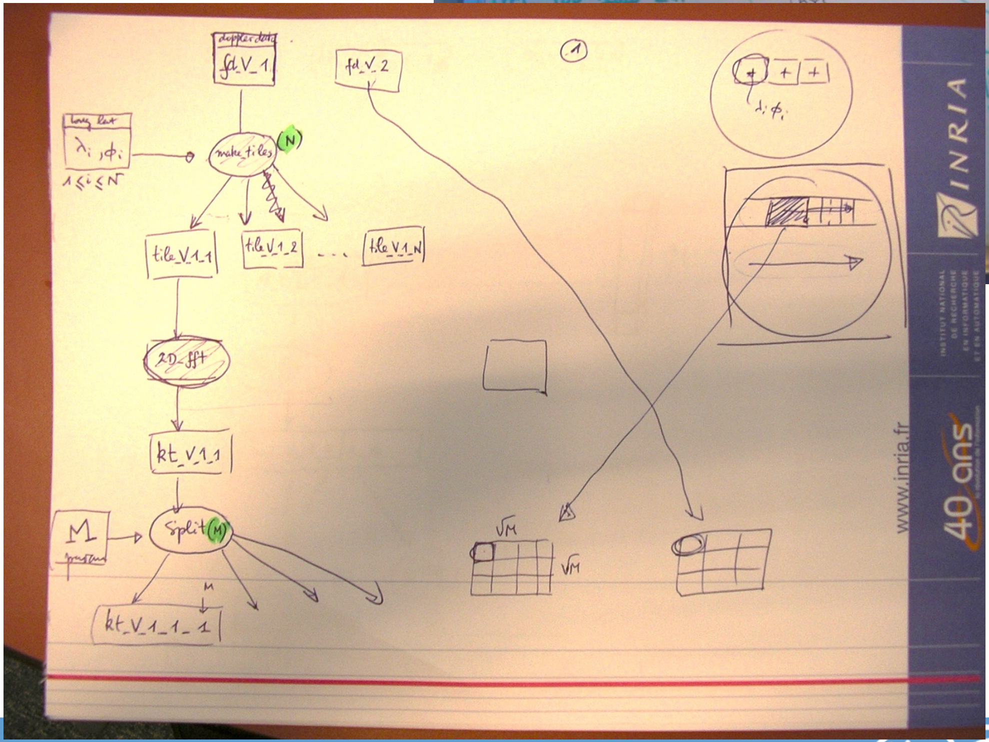
Overview

- What are scientific workflows and why use them?
- Example workflow applications
- Overview of available workflow systems
- Introduction to Pegasus WMS

What are workflows?

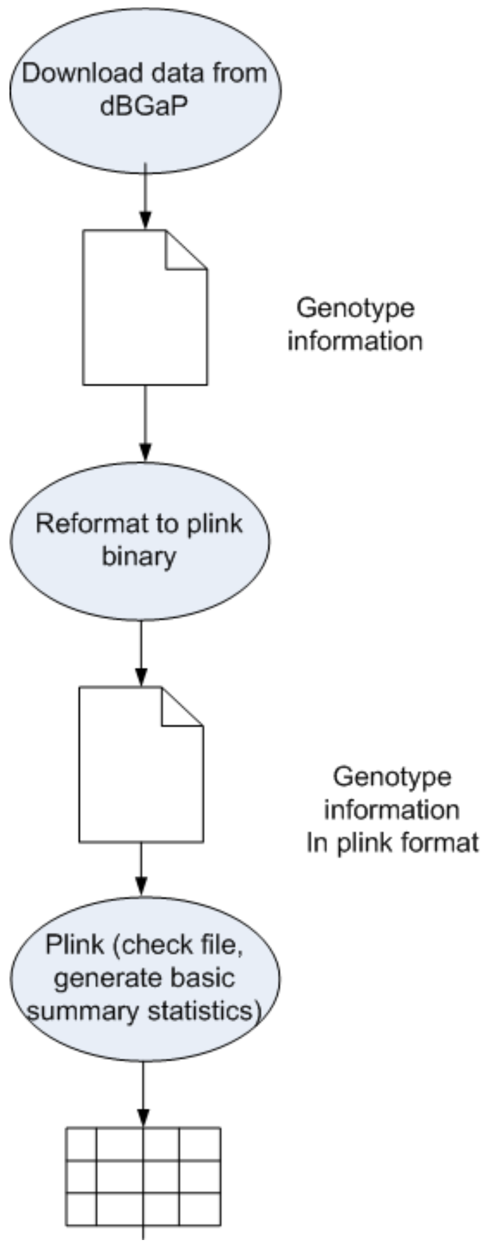




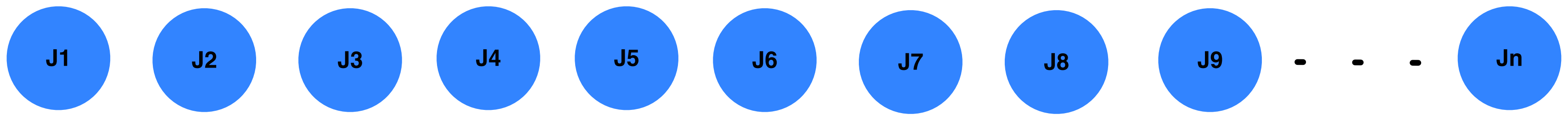


Scientific Workflows

- Formal way to capture multi-step computations
- Relatively coarse grained
- Capture the steps and their parameters
- Define the input/output data of each step
- Describe dependencies between steps


Download data from dBGaP
Genotype information
Reformat to plink binary
Genotype information In plink format
Plink (check file, generate basic summary statistics)

Workflows can be simple!

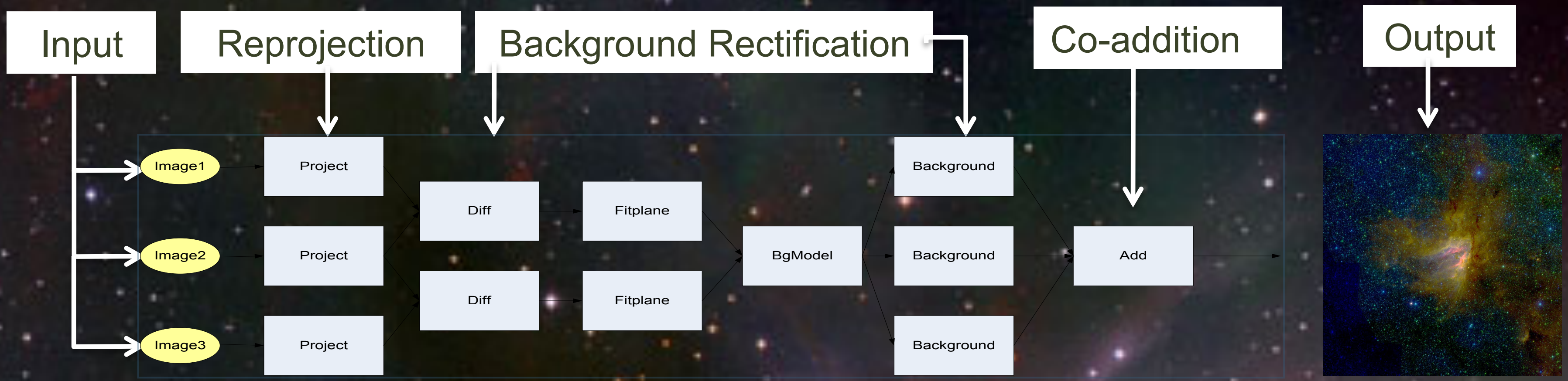


Why Scientific Workflows?

- Automate complex, multi-stage processing pipelines
- Enable parallel, distributed computations
- Use existing code, no rewrites
- Simple to construct and modify
- Reusable, aid reproducibility
- Can be shared with others
- Record how data was produced (provenance)
- Handle failures with to provide reliability
- Keep track of data and files

Science-grade Mosaic of the Sky

Science-grade Mosaic of the Sky



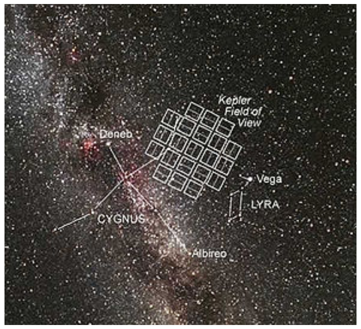
Montage Workflow

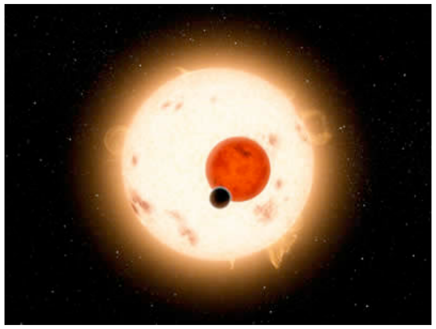
montage.ipac.caltech.edu

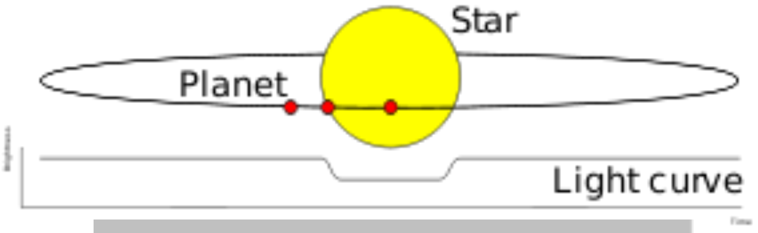
Size of mosaic in degrees square	Number of input data files	Number of tasks	Number of intermediate files	Total data footprint	Cummulative wall time
1	84	387	850	1.9 GB	21 mins
2	300	1442	3176	6.8 GB	54 mins
4	685	3738	8258	18 GB	3 hours, 18 mins
6	1461	7462	16458	37 GB	7 hours, 7 mins
8	2565	12757	28113	64 GB	11 hours, 44 mins

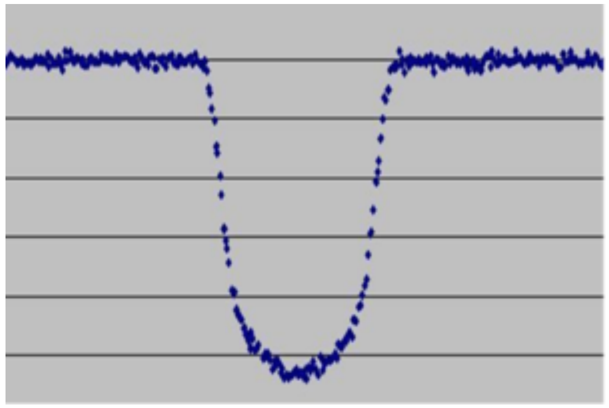
Bag of Tasks: Periodogram Workflow

- Kepler continuously monitors the brightness of over 175,000 stars
  - Search for periodic dips in signals as Earth-like planets transit in front of host star.
- For each star, Kepler data is used to create a “light curve”
- Need to perform a bulk analysis of all the data to search for these periodic signals





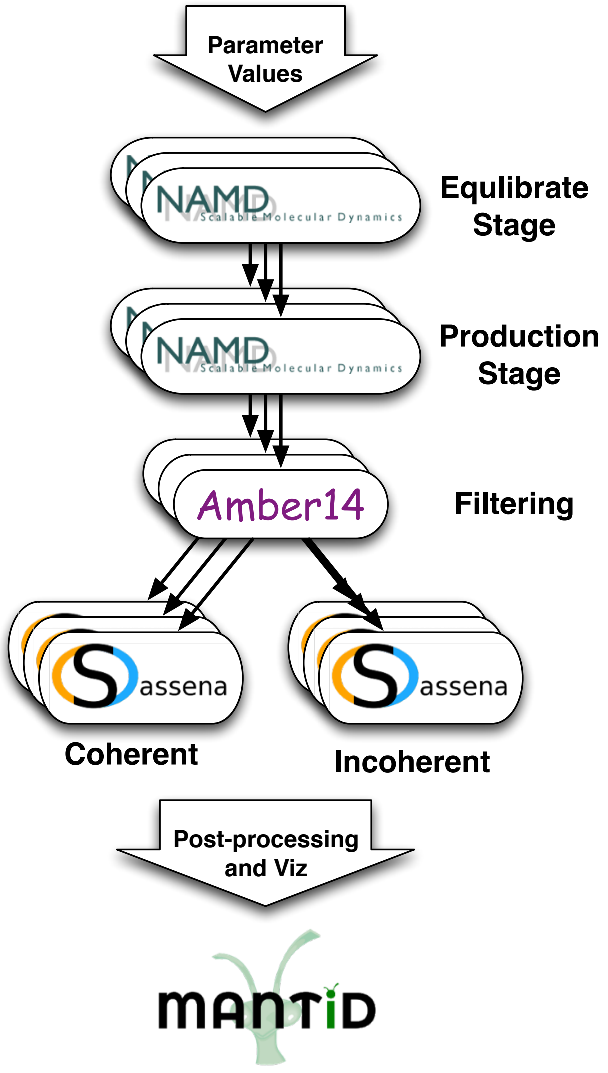



Kepler 6-b transit

2012 Run at SDSC
- 1.1M tasks, 180 jobs
- 1.1M input, 12M output files
- ~101,000 CPU hours
- 16 TB output data

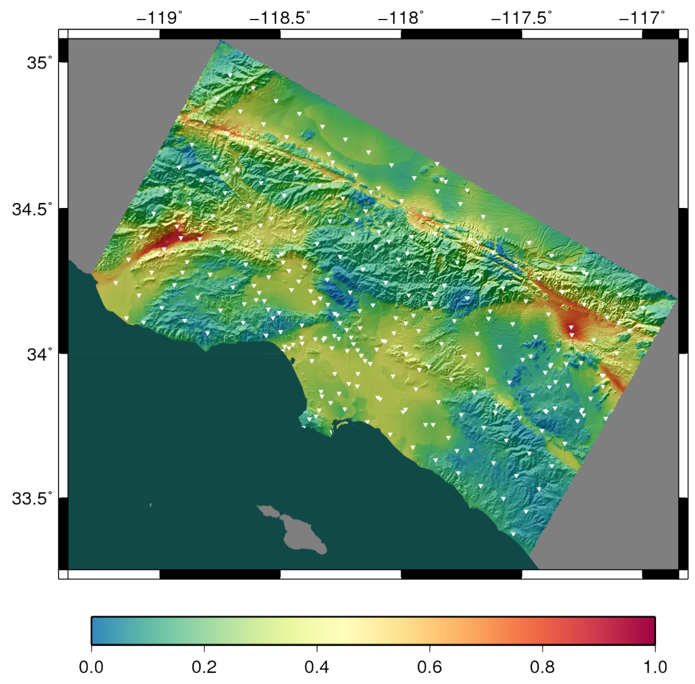
exoplanetarchive.ipac.caltech.edu

Workflows with MPI Codes: Neutron Scattering



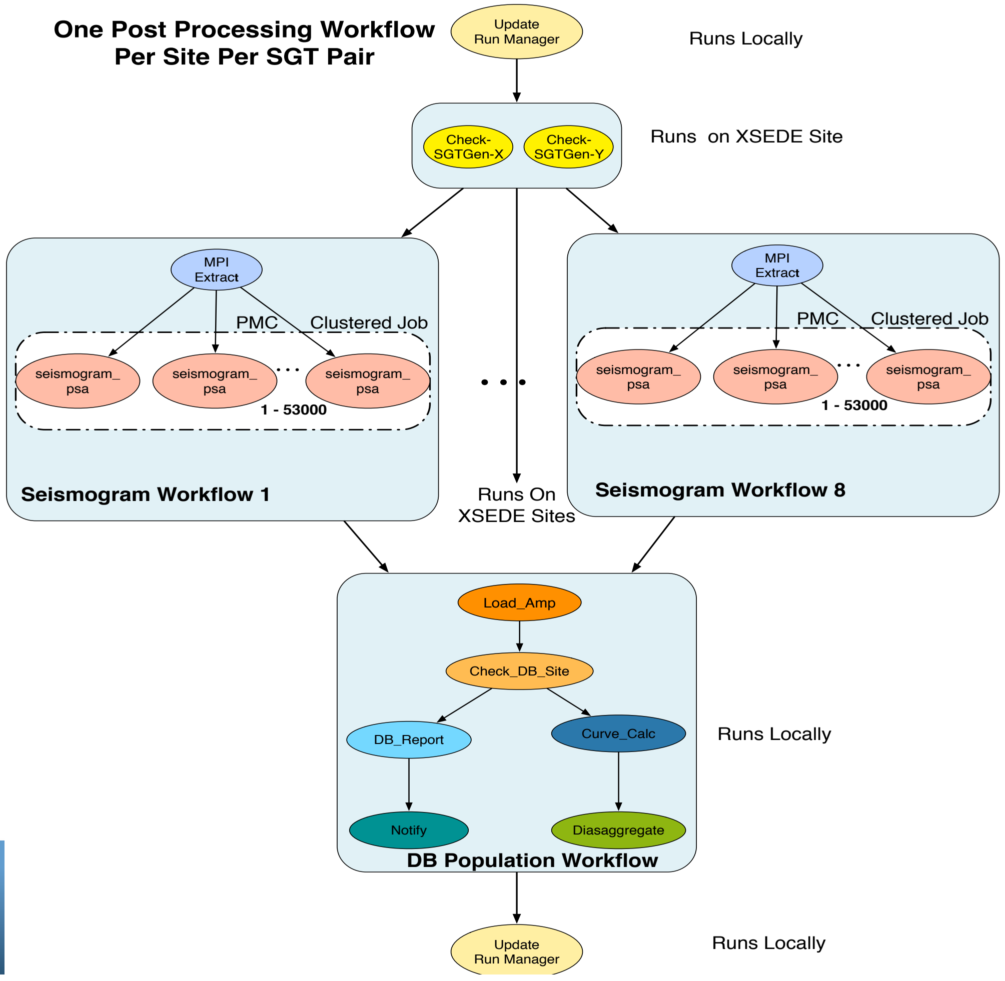
- Spallation Neutron Source at ORNL
- Parameter sweeps of MD and neutron scattering simulations
  - Fit simulation to experimental data
  - e.g. temperature, charge, force
- Nanodiamond Workflow
  - Feb 2015 on Hopper using GRAM and GridFTP
  - 19 parameter values for nonbonded interactions between ND and H2O
  - 800 core NAMD jobs x 22 hrs
  - 400 core Sassena jobs x 3 hrs
  - ~380,000 CPU hours
  - ~1/2 TB output

Large-Scale Workflows: CyberShake PSHA



CyberShake Hazard Map, 3sec SA, 2% in 50 yrs

- ✧ Builders ask seismologists: "What will the peak ground motion be at my new building in the next 50 years?"
- ✧ Seismologists answer this question using Probabilistic Seismic Hazard Analysis (PSHA)



2014: 286 Sites, 4 models

- **Each site = one workflow**
- **Each workflow has 420,000 tasks in 21 jobs using task clustering w/ PMC**
- **BlueWaters@NCSA, Stampede@TACC**

USC Viterbi School of Engineering

Workflow Management Systems

- Automate execution of workflows
- Workflow language
  - Used to describe the workflow
  - Visual with GUI or text-based
  - Frequently based on DAGs, but some provide loops and branches or more exotic semantics
- Workflow engine
  - Manages the scheduling, submission, and monitoring of tasks
  - Orchestrates the movement of data
  - Interfaces with diverse cyberinfrastructure (grids, clusters, clouds)
- There are lots of workflow management systems
  - Some are abandoned research projects

Swift (swift-lang.org)

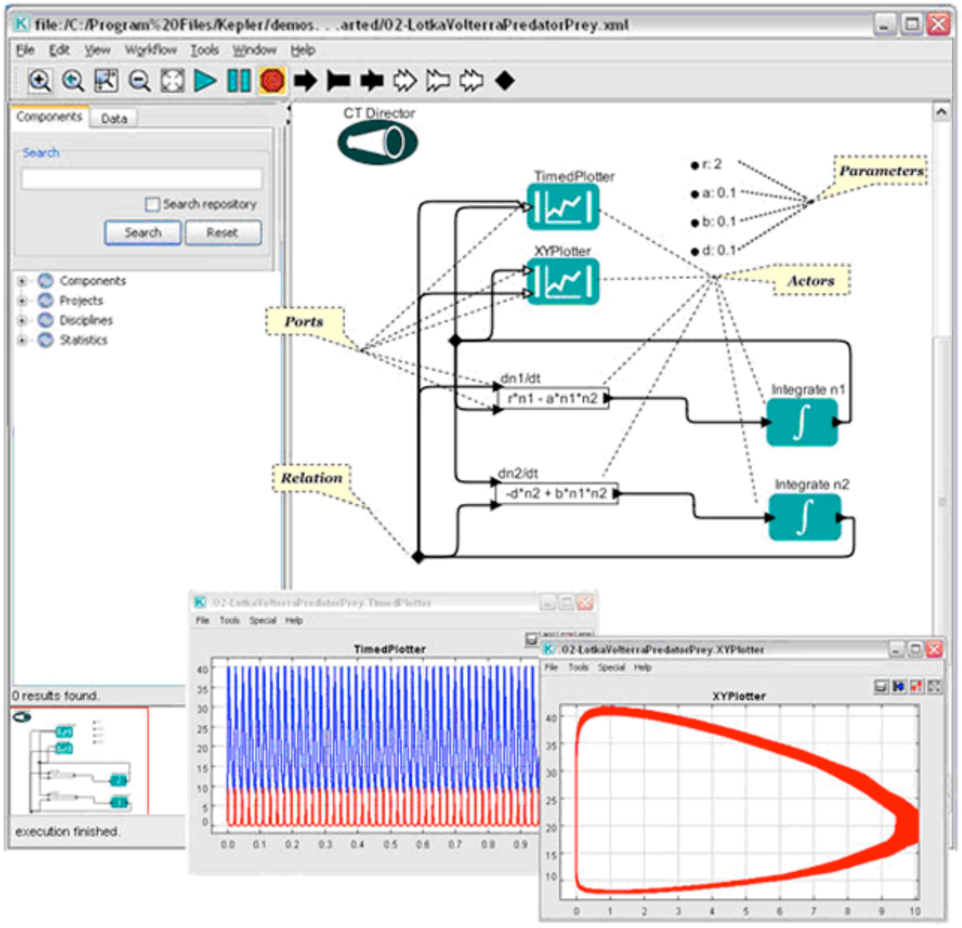
- Developed at the University of Chicago
- Workflow defined via parallel scripting language

```
//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();
```

- Supports workflows with many tasks and large data
- Interfaces with many different cluster, grid and cloud infrastructures

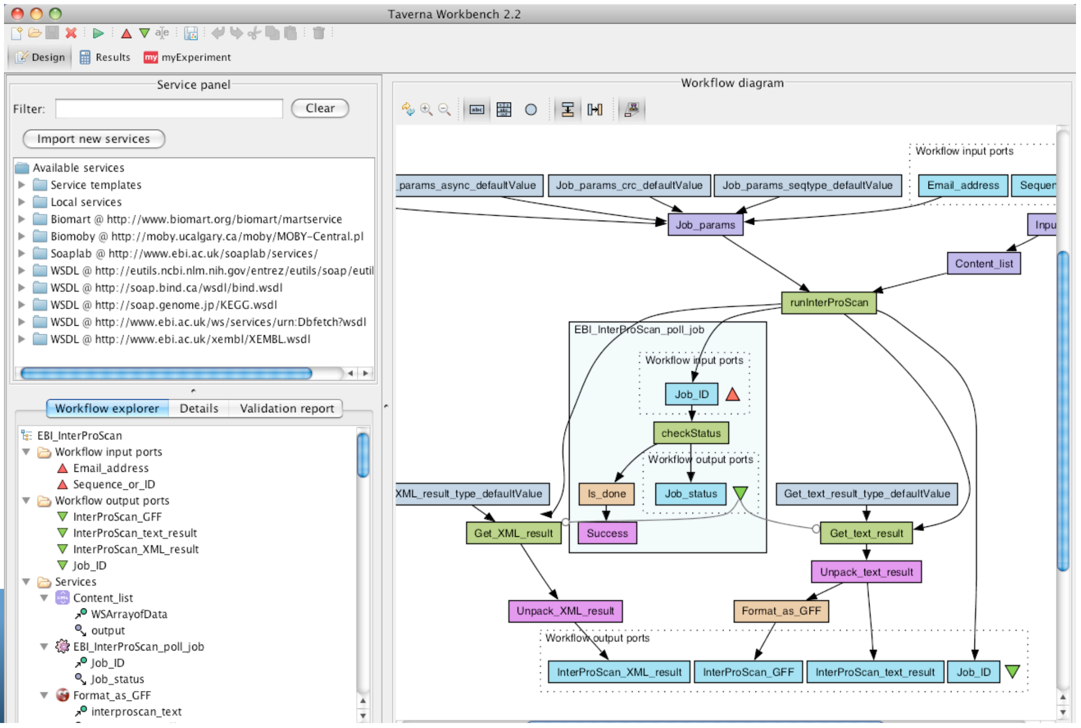
Kepler (kepler-project.org)

- Developed by a diverse group of collaborators
- GUI-based
  - Composition and execution
  - View outputs
- Many different models of computation
  - Actor model with different execution semantics
- Interfaces with grids, clusters, and web services
- Component repository for sharing and lots of built-in components



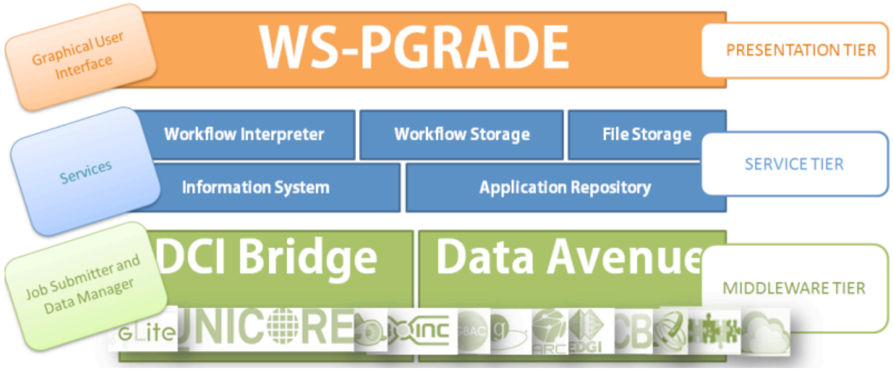
Taverna (www.taverna.org.uk)

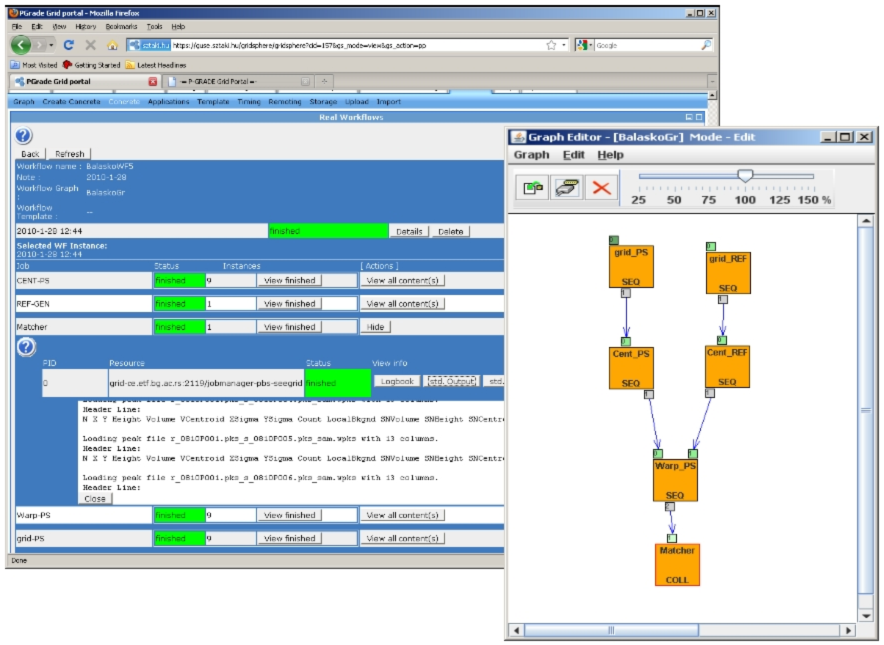
- Developed by a collaboration of UK universities
- GUI workflow composition
  - DAGs, loops, data parallel, merges
- Web services and local scripts/commands (mostly)
- Particularly good for bioinformatics
- Integrates with myExperiment for sharing workflows
- Leverages service catalogs for easy workflow composition



WS-PGRADE/gUSE (guse.hu)

- Developed at the Hungarian Academy of Sciences
- GUI interface for workflow composition
- Supports template DAGs for parameter sweep, WoW
- Integrated web portal/gateway
- Interfaces with many different infrastructures
- Extensive documentation




PGrade Grid portal - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://guse.sztaki.hu/gridsphere/gridsphere?cid=157&gs_mode=view&gs_action=pp
Most Visited Getting Started Latest Headlines
PGrade Grid portal
-= P-GRADE Grid Portal =-
Graph Create Concrete Concrete Applications Template Timing Remoting Storage Upload Import
Real Workflows
Back Refresh
Graph Editor - [BalaskoGr] Mode - Edit
Graph Edit Help
25 50 75 100 125 150 %
grid_PS SEQ
grid_REF SEQ
Cent_PS SEQ
Cent_REF SEQ
Warp_PS SEQ
Matcher COLL
Done

Workflow name : BalaskoWF5
Note : 2010-1-28
Workflow Graph : BalaskoGr
Workflow Template : --

2010-1-28 12:44 finished Details Delete

Selected WF Instance:
2010-1-28 12:44

Job	Status	Instances		[Actions]
CENT-PS	finished	9	View finished	View all content(s)
REF-GEN	finished	1	View finished	View all content(s)
Matcher	finished	1	View finished	Hide

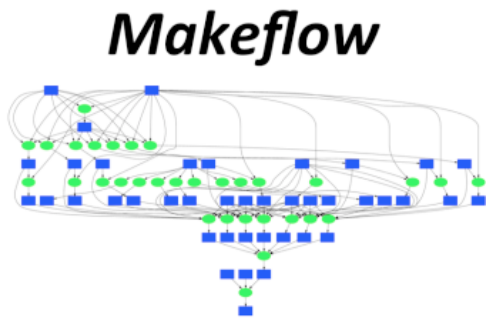
PID	Resource	Status	View info
0	grid-ce.etf.bg.ac.rs:2119/jobmanager-pbs-seegrid	finished	Logbook std. Output std.

```
Header Line:
N X Y Height Volume VCentroid XSigma YSigma Count LocalBkgnd SNVolume SNHeight SNCentr

Loading peak file r_081OP001.pks_s_081OP005.pks_sam.wpks with 13 columns.
Header Line:
N X Y Height Volume VCentroid XSigma YSigma Count LocalBkgnd SNVolume SNHeight SNCentr

Loading peak file r_081OP001.pks_s_081OP006.pks_sam.wpks with 13 columns.
Header Line:
```
Close

Job	Status	Instances		[Actions]
Warp-PS	finished	9	View finished	View all content(s)
grid-PS	finished	9	View finished	View all content(s)

Other Workflow Systems

- VisTrails (vistrails.org)
  - Used for visualization pipelines with VTK
- Galaxy (galaxyproject.org)
  - Oriented toward biomedical research
  - Interfaces with many web services
  - Web-based GUI interface
- UNICORE Workflow System (unicore.eu)
  - GUI for workflow composition, or XML
  - Branches, loops, parallel loops
- Makeflow (ccl.cse.nd.edu/software)
  - Simple, make-like workflow language
  - Targets many different grid, cluster systems



Pegasus Workflow Management System

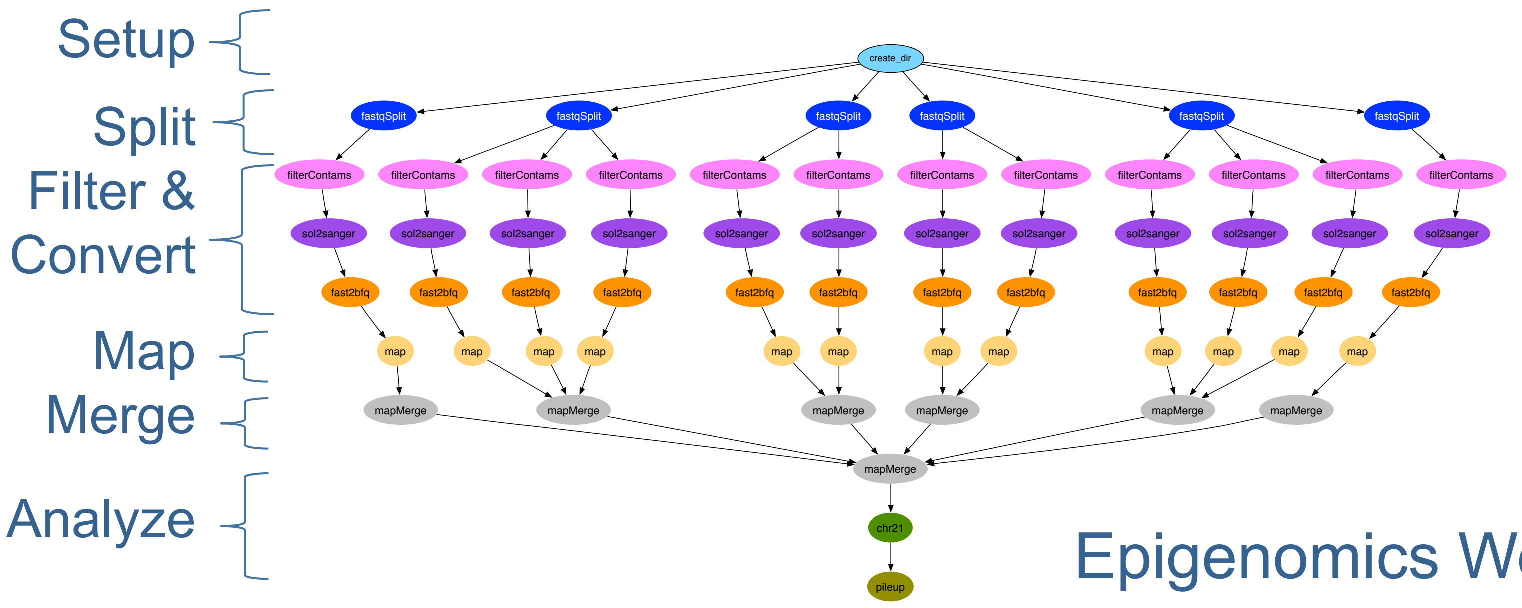
- Under development since 2001
- A collaboration between USC/ISI and the Condor Team at UW Madison
  - USC/ISI develops Pegasus
  - UW Madison develops DAGMan and Condor
- Actively used in a wide variety of domains
  - Earth science, physics, astronomy, bioinformatics, climate modeling, neutron science, and many others
  - About 600 workflows a day

Why Pegasus?

- Maps abstract workflows to diverse computing infrastructures
  - Desktop, Condor Pool, HPC Cluster, Grid, Cloud
- Supports large-scale, data-intensive workflows
  - O(1M) tasks and O(TB) of data
- Automatically plans and executes data transfers
- Manages failures to provide reliability
  - Retries and checkpointing
- Provides workflow monitoring and debugging tools to allow users to debug large workflows
- Technical support
  - full-time staff, mailing lists, public repository and bug tracker, regular releases, decent documentation

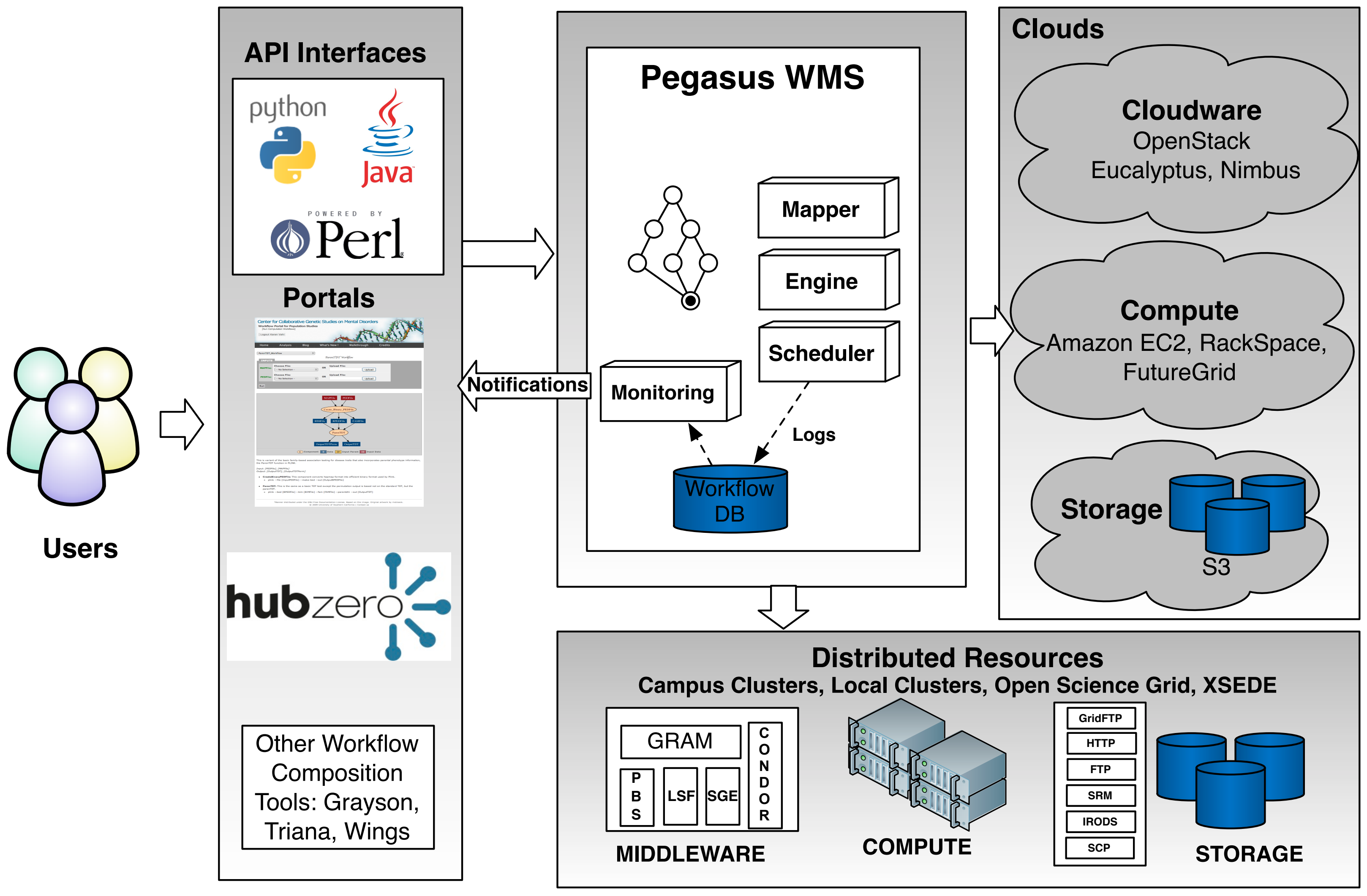
Pegasus Workflows

- Expressed as a DAG: nodes=tasks, edges=dependencies
- Tasks are command-line programs, executed as batch jobs
- Dependencies are usually data dependencies
- Data is exchanged via files

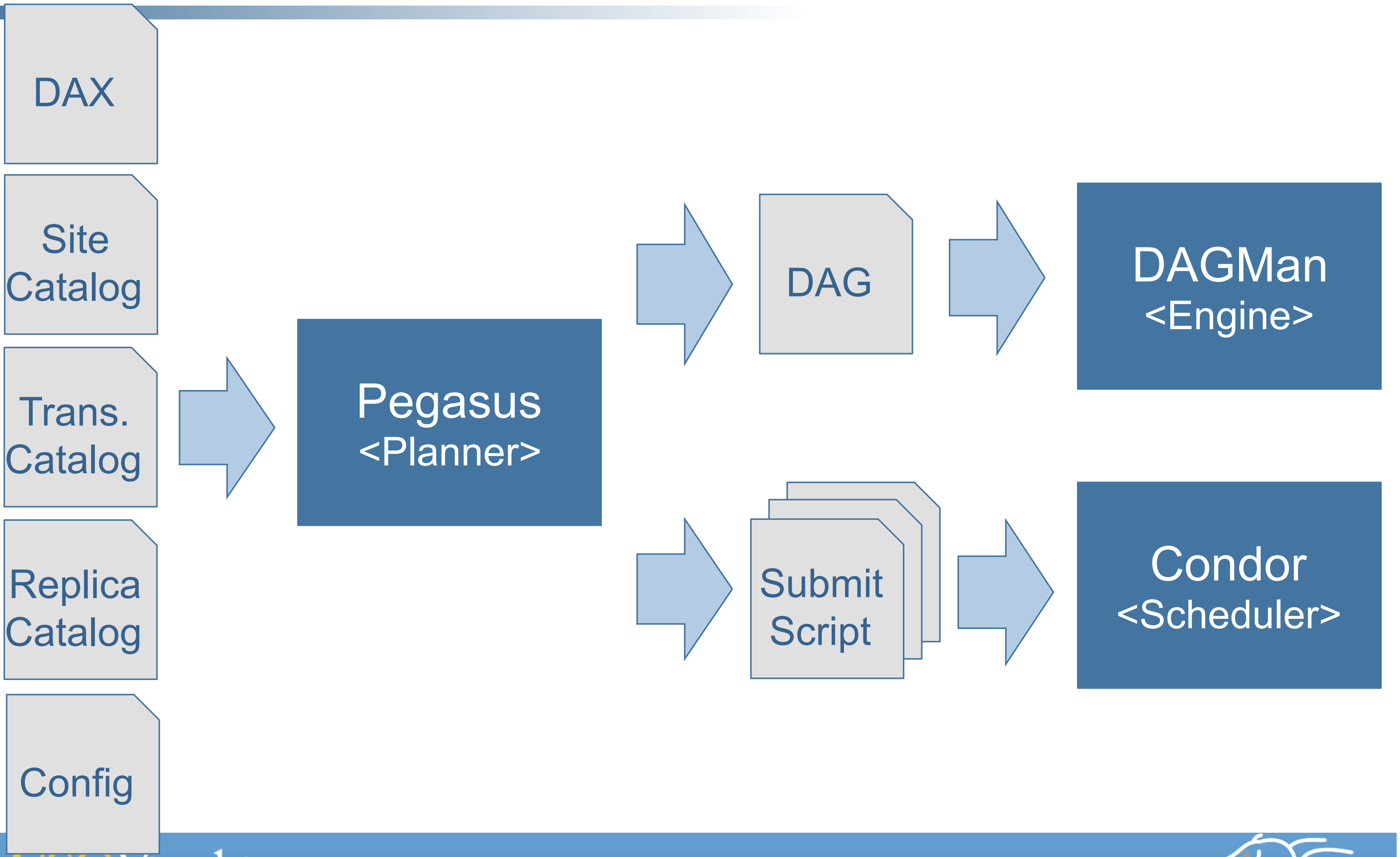


Epigenomics Workflow

Pegasus WMS Environment



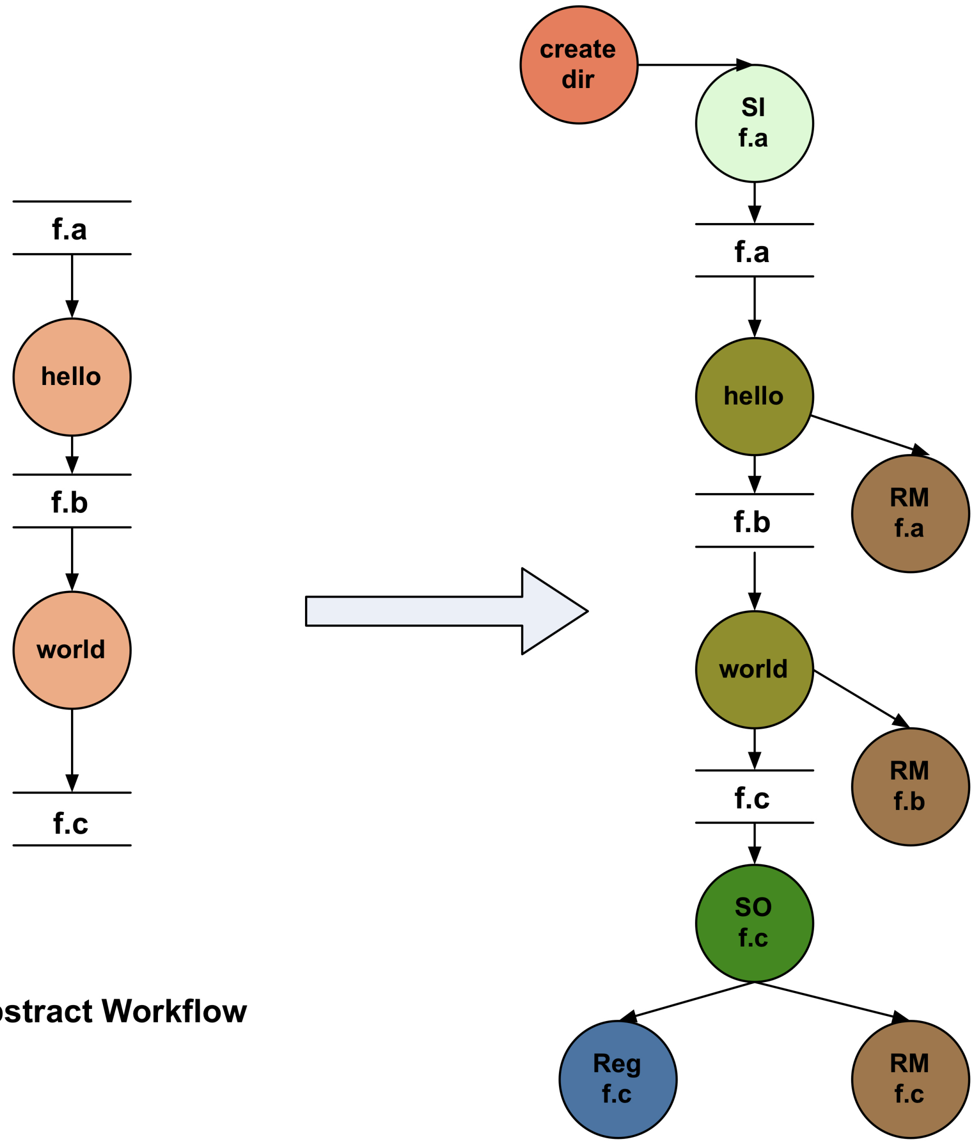
Pegasus WMS Data Flow



Workflow Planning (Mapping)

- Pegasus converts abstract workflow descriptions into executable workflows (similar to compiler)
  - Facilitates portability
  - Separates data management from workflow composition
  - Enables workflow-level optimizations
  - Others…
- Planning process:
  - Choose a site for each job (site selection)
  - Add resource-specific information
  - Choose input files (replica selection)
  - Plan data movements and add data management jobs
  - Perform optimizations
  - Add setup and cleanup jobs
  - Generate executable workflow artifacts

Abstract to Executable Workflow Mapping



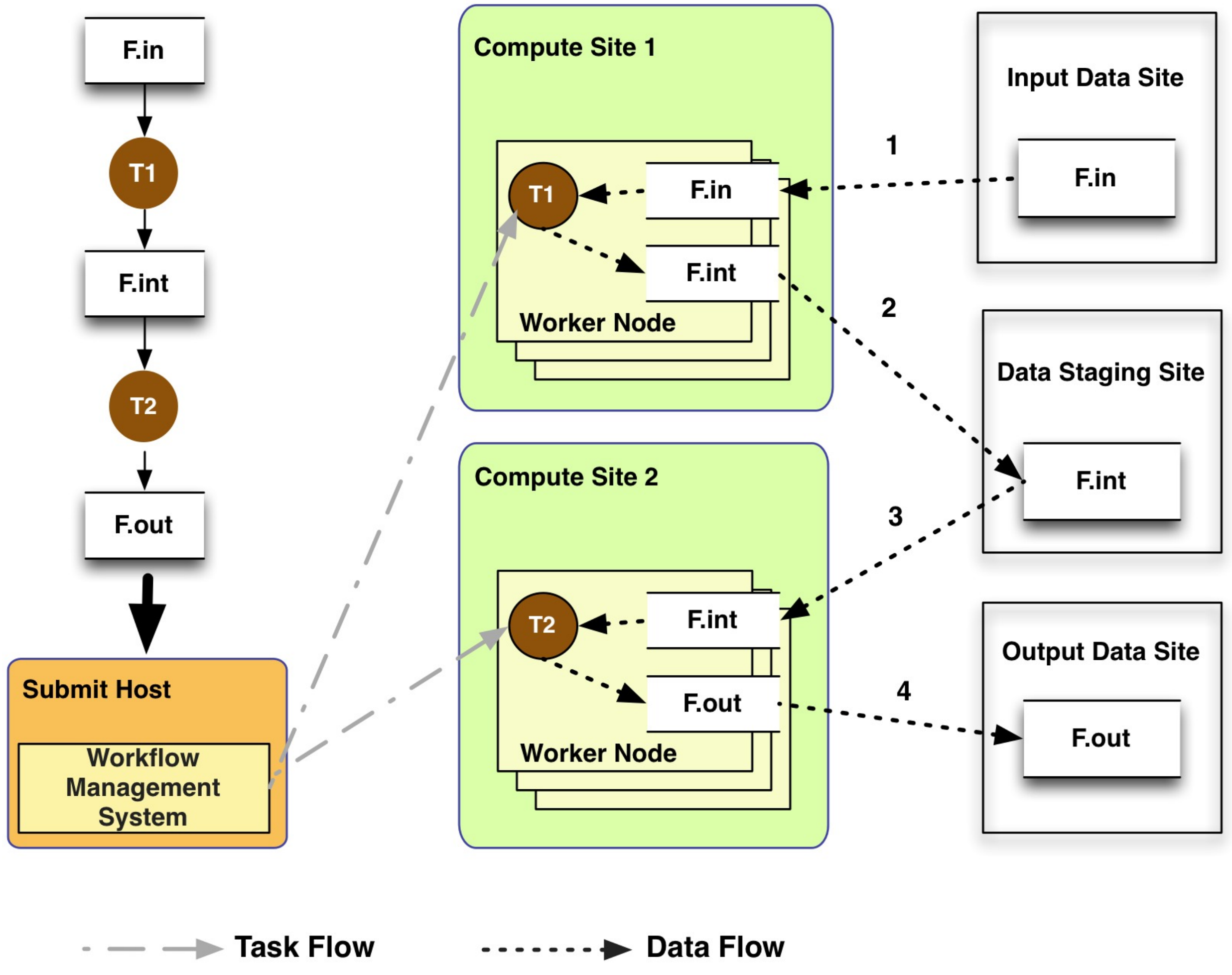
Abstract Workflow

Executable Workflow

LEGEND

- ● Unmapped Job
- ● Compute Job mapped to a site
- ● Stage-in Job
- ● Stage-Out Job
- ● Registration Job
- ● Make Dir Job
- ● Cleanup Job

Data Management



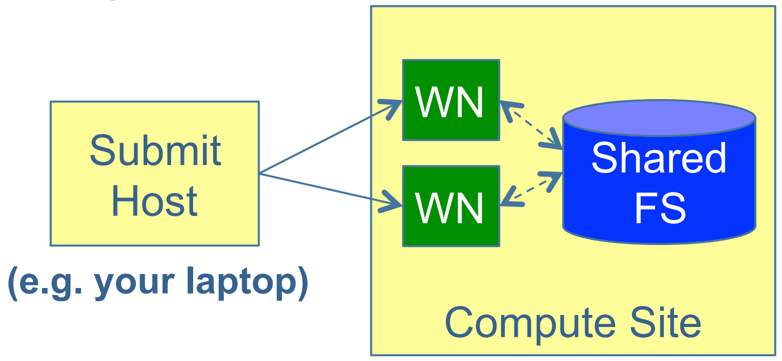
- Most of the tasks in scientific workflow applications require POSIX file semantics
  - Each task in the workflow opens one or more input files
  - Read or write a portion of it and then close the file.
- Data Staging Site can be the shared filesystem on the compute cluster!

- Input Data Site, Compute Site and Output Data Sites can be co-located
  - Example: Input data is already present on the compute site.

Data Staging Configurations

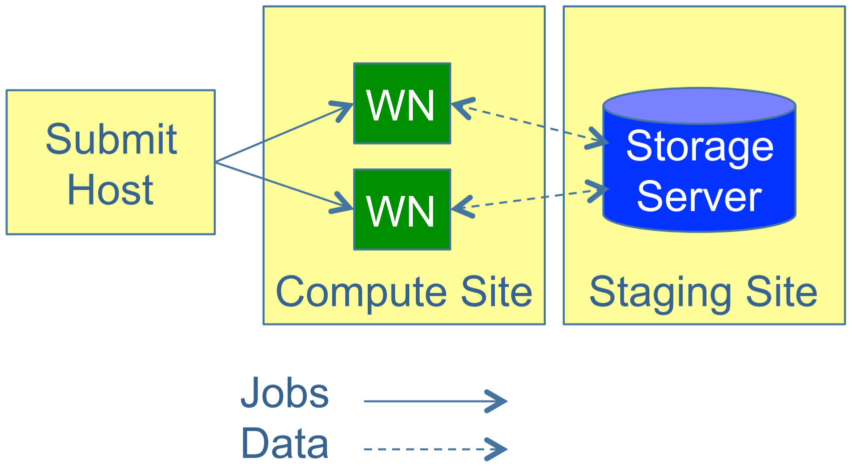
Shared File System (typical of most HPC sites)

- Worker nodes and the head node have a shared filesystem, usually a parallel filesystem with high-performance I/O
- Can leverage symlinking against pre-staged datasets
- Staging site is the compute site



Non-shared File System (typical of OSG and EC2)

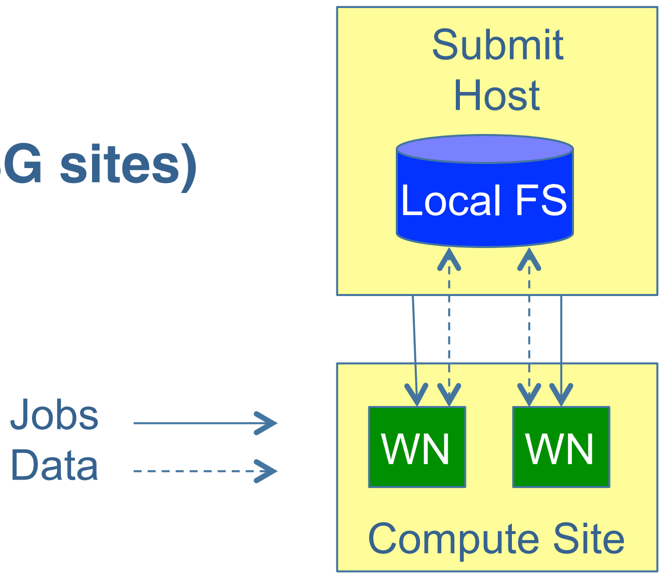
- Worker nodes don't share a file system
- Uses a staging site separate from the compute site such as Amazon S3
- Data is pulled from / pushed to the staging site
- Also known as "PegasusLite"



Data Staging Configurations

Condor I/O (Typical of Condor Pools like OSG sites)

- Worker nodes don't share a file system
- Data is pulled from / pushed to the submit host via Condor file transfers
- Staging site is the submit host



Using Pegasus allows you to move from one deployment to another without changing the workflow description

Many Data Protocols Supported:

- SCP
- HTTP
- FTP
- GridFTP
- Amazon S3
- iRODS
- SRM
- cp
- symlink
- FDT
- Google Storage
- StashCache

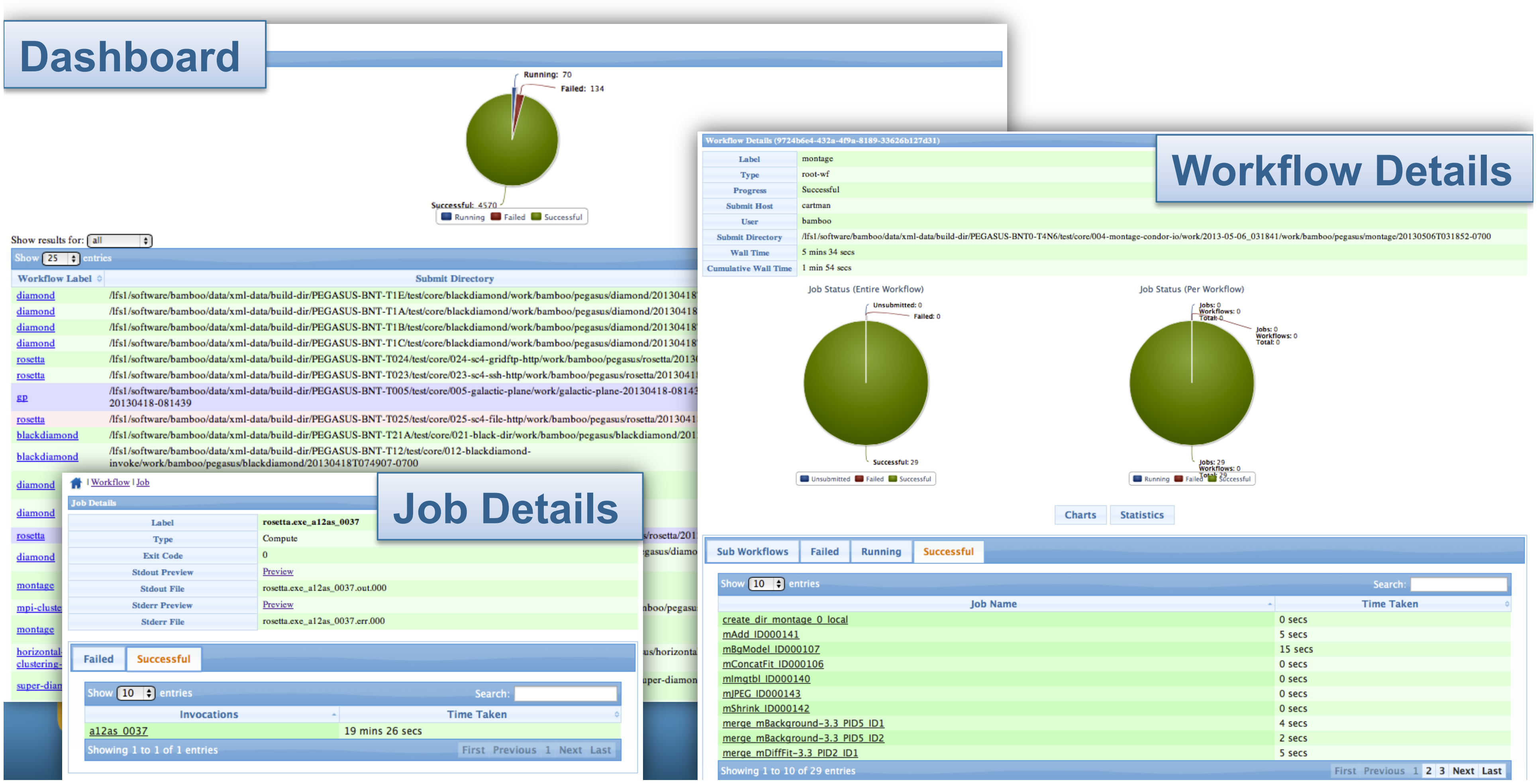
Workflow Monitoring and Reporting

- **Data collection**
  - Data extracted from log files and stored in a relational database
  - DB contains workflow structure, status information, runtimes, host info, task stdout/stderr
- **Reporting tools**
  - Status of the workflow
    - pegasus-status path/to/submit/directory
  - Detailed runtime statistics
    - pegasus-statistics -s all path/to/submit/directory

```
------------------------------------------------------------------------------
Type           Succeeded Failed  Incomplete  Total     Retries   Total+Retries
Tasks          135002    0       0           135002    0         135002
Jobs           4529      0       0           4529      0         4529
Sub-Workflows  2         0       0           2         0         2
------------------------------------------------------------------------------

Workflow wall time                                : 13 hrs, 2 mins, (46973 secs)
Workflow cumulative job wall time                 : 384 days, 5 hrs, (33195705 secs)
Cumulative job walltime as seen from submit side : 384 days, 18 hrs, (33243709 secs)
```

Pegasus Dashboard

- Web-based workflow monitoring GUI
  - Data comes from monitoring database
  - Supports monitoring, troubleshooting, and reporting



Show results for: all

Show 25 entries

Workflow Label	Submit Directory
diamond	/lfs1/software/bamboo/data/xml-data/build-dir/PEGASUS-BNT-T1E/test/core/blackdiamond/work/bamboo/pegasus/diamond/2013041
diamond	/lfs1/software/bamboo/data/xml-data/build-dir/PEGASUS-BNT-T1A/test/core/blackdiamond/work/bamboo/pegasus/diamond/2013041
diamond	/lfs1/software/bamboo/data/xml-data/build-dir/PEGASUS-BNT-T1B/test/core/blackdiamond/work/bamboo/pegasus/diamond/2013041
diamond	/lfs1/software/bamboo/data/xml-data/build-dir/PEGASUS-BNT-T1C/test/core/blackdiamond/work/bamboo/pegasus/diamond/2013041
rosetta	/lfs1/software/bamboo/data/xml-data/build-dir/PEGASUS-BNT-T024/test/core/024-sc4-gridftp-http/work/bamboo/pegasus/rosetta/2013
rosetta	/lfs1/software/bamboo/data/xml-data/build-dir/PEGASUS-BNT-T023/test/core/023-sc4-ssh-http/work/bamboo/pegasus/rosetta/201304
gp	/lfs1/software/bamboo/data/xml-data/build-dir/PEGASUS-BNT-T005/test/core/005-galactic-plane/work/galactic-plane-20130418-0814 20130418-081439
rosetta	/lfs1/software/bamboo/data/xml-data/build-dir/PEGASUS-BNT-T025/test/core/025-sc4-file-http/work/bamboo/pegasus/rosetta/201304
blackdiamond	/lfs1/software/bamboo/data/xml-data/build-dir/PEGASUS-BNT-T21A/test/core/021-black-dir/work/bamboo/pegasus/blackdiamond/201
blackdiamond	/lfs1/software/bamboo/data/xml-data/build-dir/PEGASUS-BNT-T12/test/core/012-blackdiamond-invoke/work/bamboo/pegasus/blackdiamond/20130418T074907-0700
diamond	
diamond	
rosetta	s/rosetta/201
diamond	gasus/diamo
montage	
mpi-cluste	nboo/pegasu
montage	
horizontal-clustering-	us/horizonta
super-diam	uper-diamon

Workflow Details (9724b6e4-432a-4f9a-8189-33626b127d31)

Label	montage
Type	root-wf
Progress	Successful
Submit Host	cartman
User	bamboo
Submit Directory	/lfs1/software/bamboo/data/xml-data/build-dir/PEGASUS-BNT0-T4N6/test/core/004-montage-condor-io/work/2013-05-06_031841/work/bamboo/pegasus/montage/20130506T031852-0700
Wall Time	5 mins 34 secs
Cumulative Wall Time	1 min 54 secs

Show 10 entries Search:

Job Name	Time Taken
create_dir_montage_0_local	0 secs
mAdd_ID000141	5 secs
mBgModel_ID000107	15 secs
mConcatFit_ID000106	0 secs
mImgtbl_ID000140	0 secs
mJPEG_ID000143	0 secs
mShrink_ID000142	0 secs
merge_mBackground-3.3_PID5_ID1	4 secs
merge_mBackground-3.3_PID5_ID2	2 secs
merge_mDiffFit-3.3_PID2_ID1	5 secs

Showing 1 to 10 of 29 entries First Previous 1 2 3 Next Last

Job Details

Label	rosetta.exe_a12as_0037
Type	Compute
Exit Code	0
Stdout Preview	Preview
Stdout File	rosetta.exe_a12as_0037.out.000
Stderr Preview	Preview
Stderr File	rosetta.exe_a12as_0037.err.000

Show 10 entries Search:

Invocations	Time Taken
a12as_0037	19 mins 26 secs

Showing 1 to 1 of 1 entries First Previous 1 Next Last

Failure Management

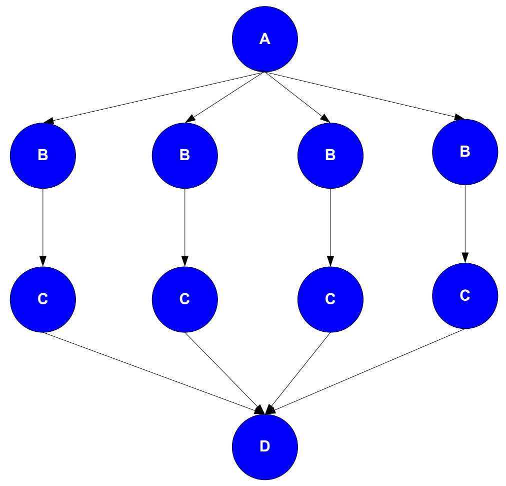
- Pegasus detects job failures
  - non-zero exit code
  - output does not contain a specified “success message”
  - output does contain a specified “failure message”
  - it exceeds a specified time limit
  - it fails to produce expected output files
- Job Retries
  - Helps with transient failures
  - Each job has a set number of retries per run
- Rescue DAGs
  - DAGMan writes a checkpoint file so workflow can be restarted
  - Can recover from almost any failure with minimal loss
- Checkpoint files
  - Job generates checkpoint files
  - Staging of checkpoint files is automatic on restarts

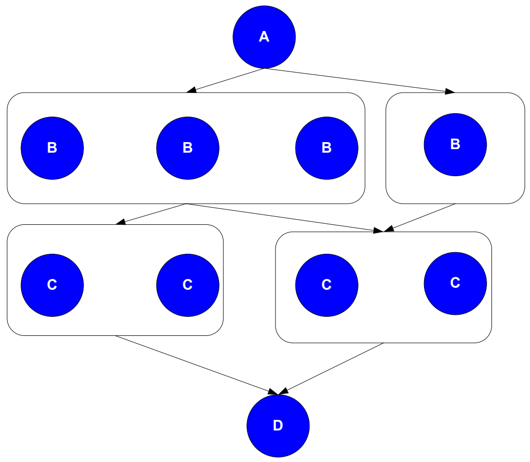
Workflow Debugging

- Problem: You have 1M tasks, and one of them fails
- pegasus-analyzer: Provides summary of workflow execution
- Outputs
  - A brief summary section
    - showing how many jobs have succeeded
    - and how many have failed
  - For each failed job:
    - showing its last known state
    - exitcode
    - working directory
    - the location of its submit, output, and error files
    - any stdout and stderr from the job

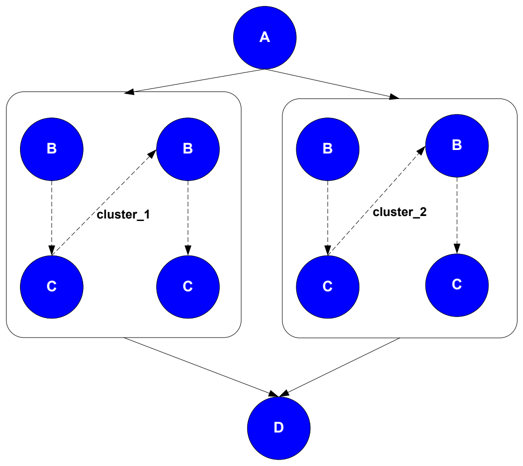
Task Clustering

- Cluster small running jobs together to achieve better performance
- Why?
  - Each job has scheduling overhead – need to make this overhead worthwhile
  - Ideally users should run a jobs that take at least 10/30/60/? minutes
  - Clustered tasks can reuse common input data – less data transfers





Horizontal clustering



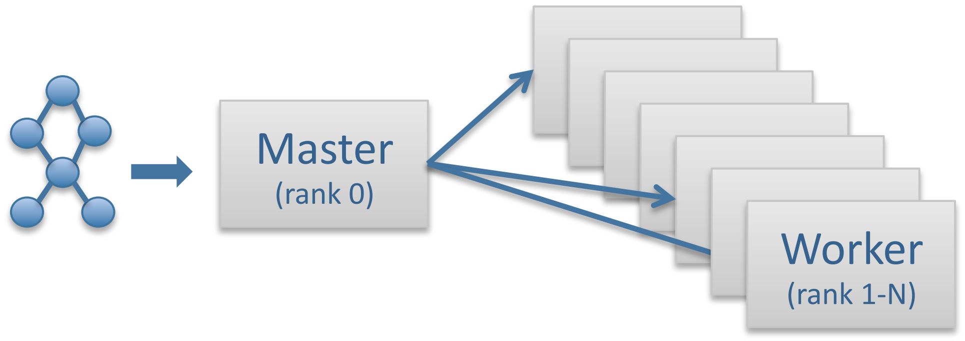
Label-based clustering

USC Viterbi School of Engineering

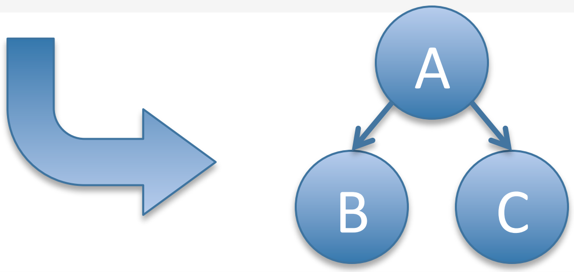
Also: time-based clustering

Pegasus-MPI-Cluster

- A master/worker task scheduler for running fine-grained workflows and ensembles on HPC systems
- Runs as an MPI job → Works on most HPC systems
- Allows sub-graphs of a Pegasus workflow to be submitted as monolithic jobs to remote resources
- Can be used on a sub-graph, or the entire workflow



```
TASK A /bin/echo I am A
TASK B /bin/echo I am B
TASK C /bin/echo I am C
EDGE A B
EDGE A C
```

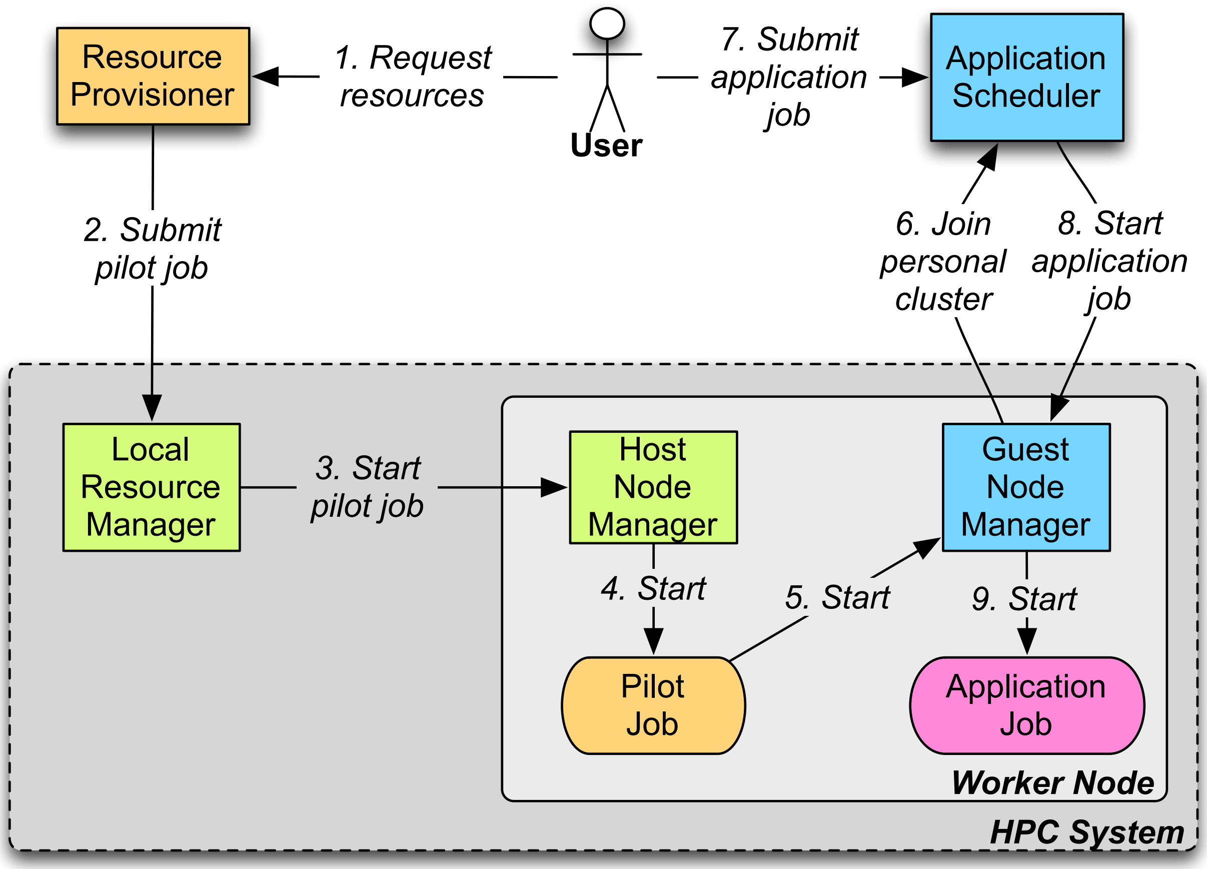


PMC Features

- Fault Tolerance
  - Retries at the task level (master resends task to another worker)
  - Retries at the workflow level (using a transaction log to record progress)
- Resource-aware scheduling
  - Many HPC machines have low memory/core
  - PMC can allocate memory and cores to a task, and force other slots on the same node to be idle
- I/O Forwarding
  - Small tasks == small I/O == poor performance
  - PMC reads data off of pipes from worker and forwards it using MPI messages to a central I/O process, which collects the data and writes it to disk
  - Writes are not interleaved, no locking required for synchronization

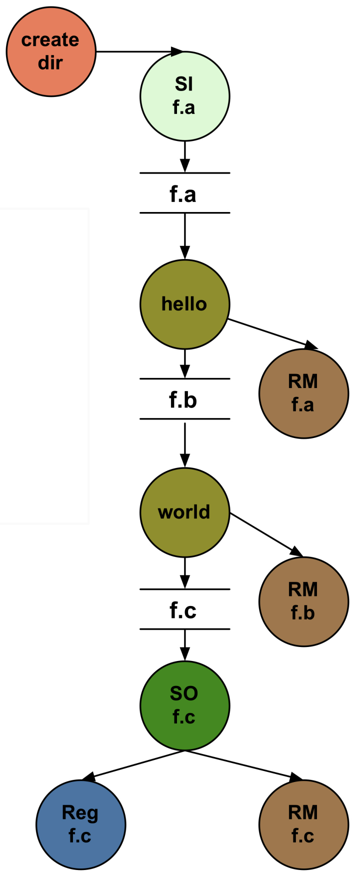
Resource Provisioning with Pilot Jobs

- Key idea: Use HPC scheduler to run application scheduler
- Parallel pilot jobs
- Amortize queue delays over many application jobs
- Apply application-specific policy

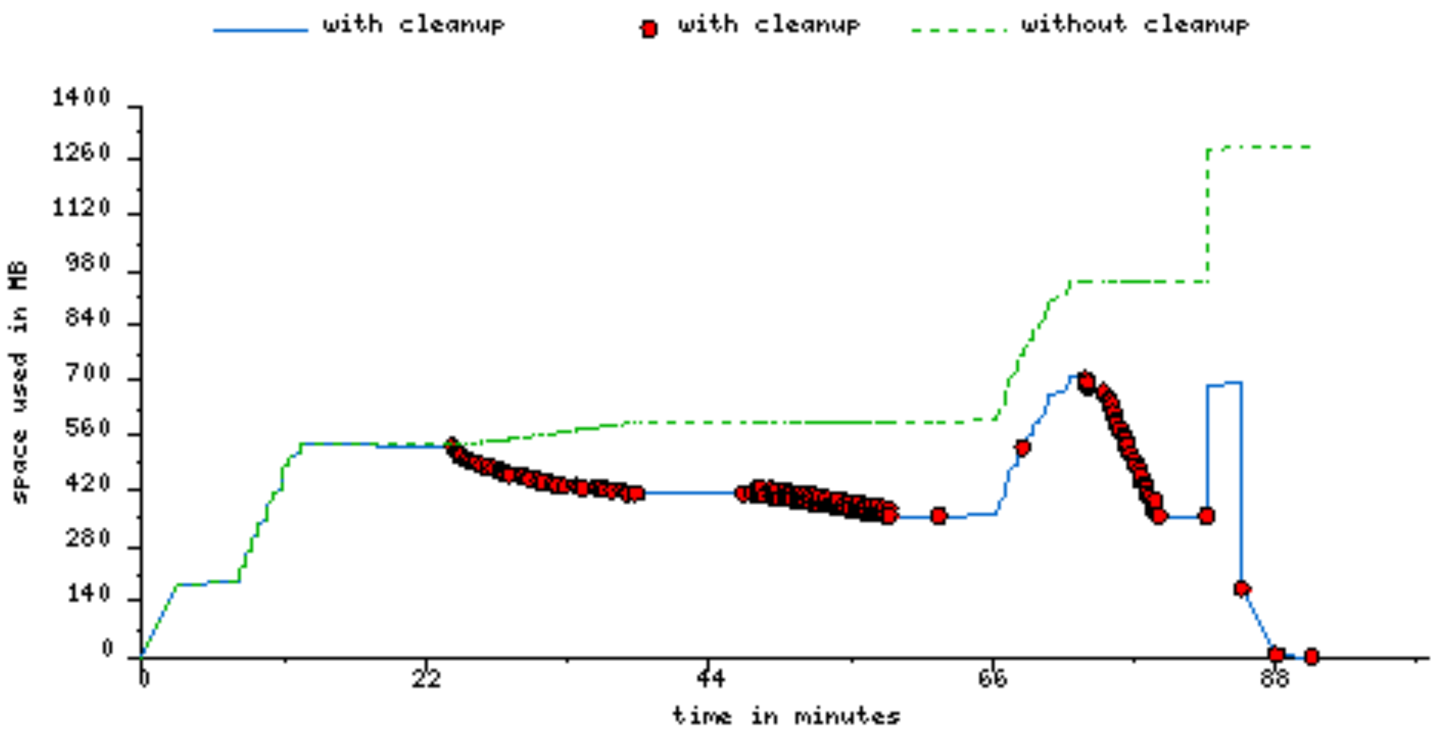


Data Cleanup

Problem: Workflow uses more disk space than quota
Solution: Add cleanup jobs to the workflow

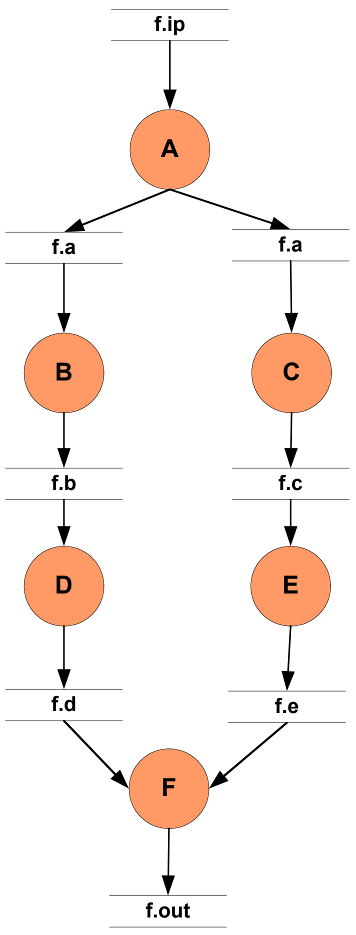


Executable Workflow

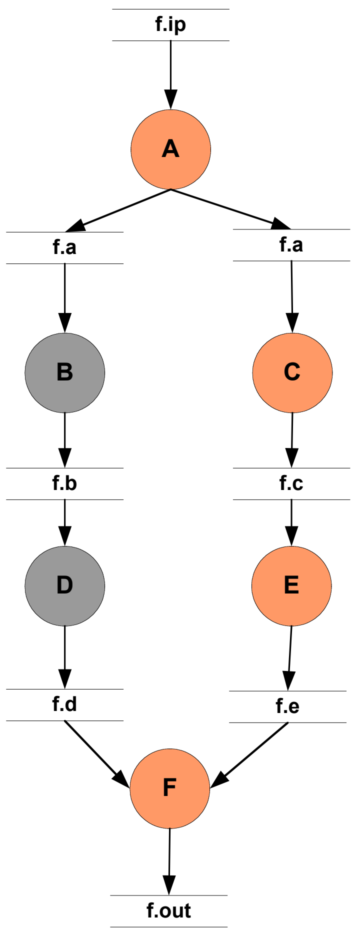


Montage 1 degree workflow run with cleanup

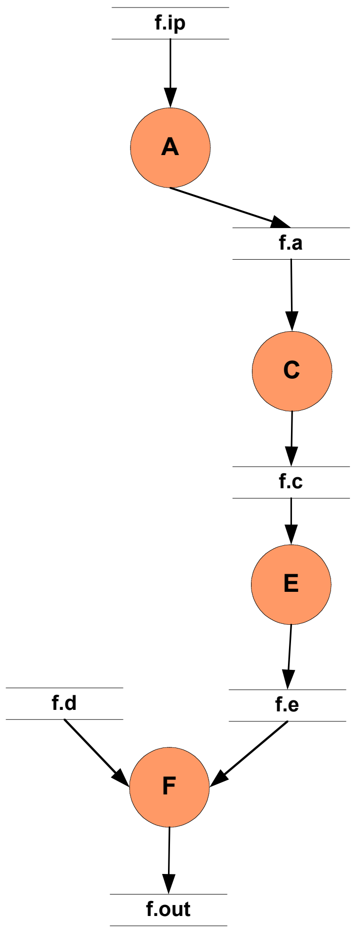
Workflow Reduction (Data Reuse, Restarts)



Abstract Workflow



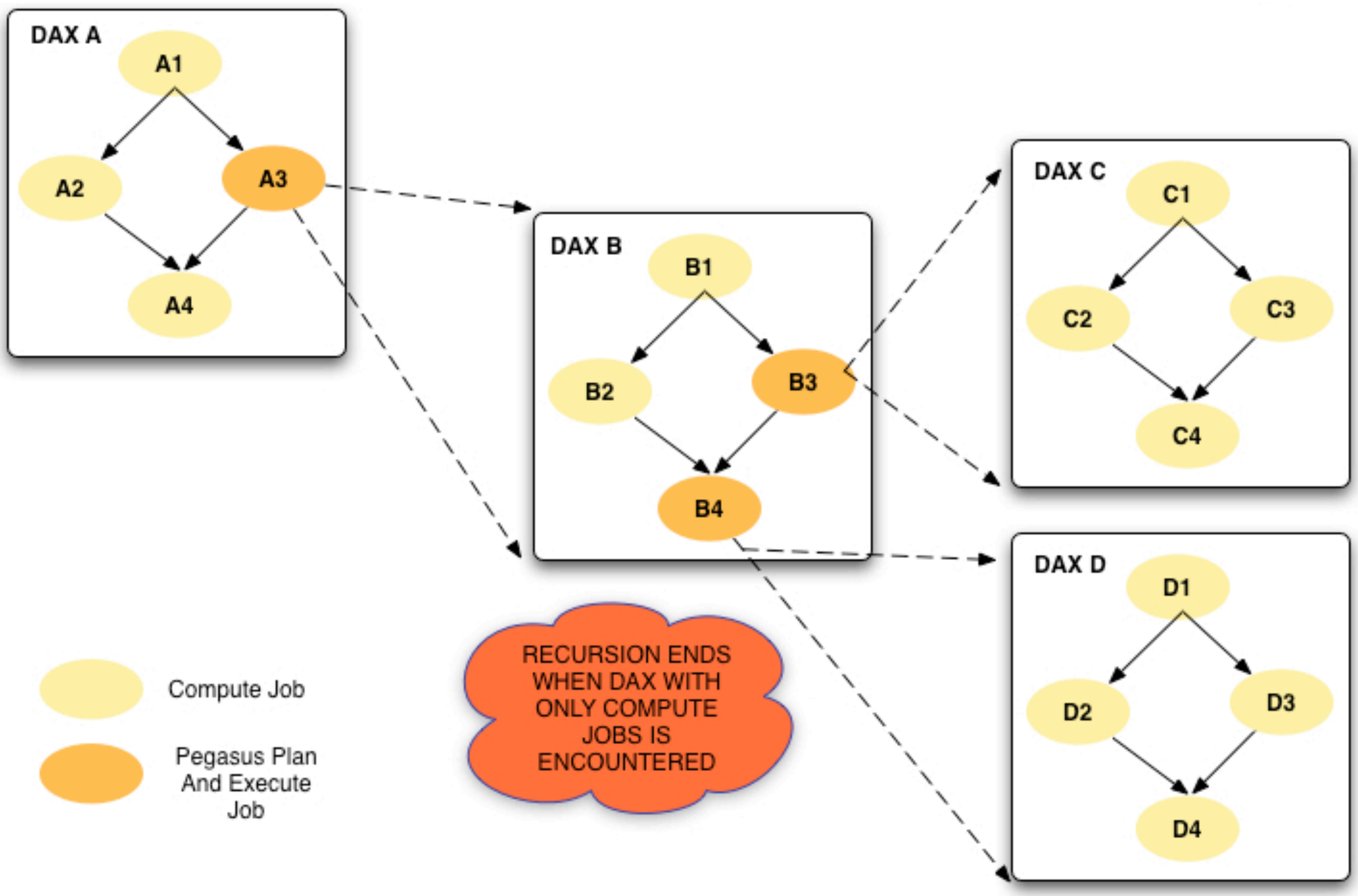
File f.d exists somewhere.
Reuse it.
Mark Jobs D and B to delete



Delete Job D and Job B

Data reuse happens automatically when output files are found in the replica catalog

Large-scale, Hierarchical Workflows


DAX A
A1
A2
A3
A4
DAX B
B1
B2
B3
B4
DAX C
C1
C2
C3
C4
DAX D
D1
D2
D3
D4
Compute Job
Pegasus Plan And Execute Job
RECURSION ENDS WHEN DAX WITH ONLY COMPUTE JOBS IS ENCOUNTERED

Other Features

- Job and Transfer Throttling
  – Prevents too many jobs/transfers from overloading system
- Notifications
  – System calls a script when certain events occur: send email, text, etc.
- Executable and Worker Package Staging
  – Enables dynamic deployment of code on remote sites
  – Planner matches the executable in the TC to the site in the SC
- Kickstart Job Wrapper
  – Records detailed information about job execution (execution host, environment, memory usage, I/O, files accessed, CPU time, etc.)
- Shell planner mode
  – Generate a shell script of your workflow

Final Thoughts

- Probably using a workflow already
  - Replaces scripts, manual hand-offs and polling to monitor
- Automation is vital
  - Eliminate babysitting your jobs: your time is valuable!
  - Able to recover from failures without losing work
- Put ALL processing steps in the workflow
  - Include validation, visualization, data publishing, notifications
- Does add additional software layers and complexity
  - Some development time is required
- Choose workflow system carefully
  - Consider required features, target environment, maturity, support
- We want to help you!

Questions?

Some Computational Science Challenges

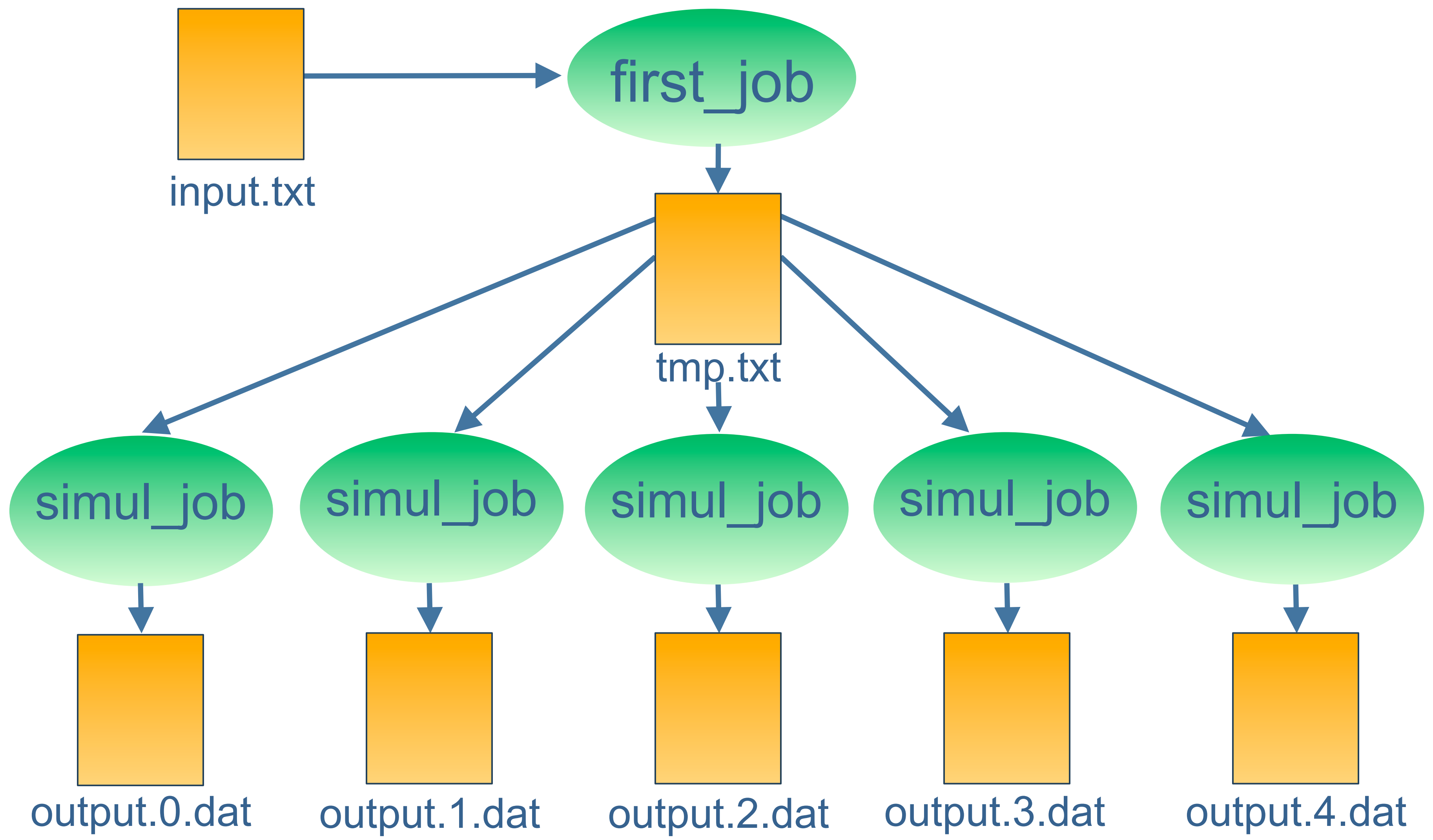
- Integrate several programs into one pipeline
- Run an ensemble of simulations
- Repeat processing steps on new data or parameters
- Reproduce previous results, or similar results
- Share analysis steps with other researchers
- Recreate the history of data products
- Run code on hundreds or thousands of inputs
- Execute analyses in parallel on distributed resources
- Reliably execute pipelines on unreliable infrastructure

Scientific workflows can help with these problems

Workflow Management System Functionality

- **Job execution**
  - Interfaces with middleware and batch systems to submit and monitor jobs
- **Data and control dependencies between jobs**
  - Tracks dependencies and makes sure jobs are executed in the right order
- **Scheduling**
  - Some jobs may be able to run in parallel with others
  - Ordering and placement can improve performance
- **Data management**
  - Transfers of input and output files to/from machine
- **Provenance**
  - Track when a job was run, where it was run, what data it produced, key parameters, metadata
- **Reliability**
  - Keeps track of what finished successfully, and what did not
- **Resource provisioning**
  - Allocating resources to run jobs

Example Workflow



Example DAX Generator in Python

```
# Create DAX object
dax = ADAG("test_dax")
# Define first job
firstJob = Job(name="first_job")
# Input and output files to first job
firstInputFile = File("input.txt")
firstOutputFile = File("tmp.txt")
# Args to first_job (first_job input=input.txt output=tmp.txt)
firstJob.addArgument("input=input.txt", "output=tmp.txt")
# Role of the files for the job
firstJob.uses(firstInputFile, link=Link.INPUT)
firstJob.uses(firstOutputFile, link=Link.OUTPUT)
# Add the job to the workflow
dax.addJob(firstJob)
```

```
for i in range(0, 5):
    # Create simulation job
    simulJob = Job(id="%s" % (i+1), name="simul_job")
    # Define files
    simulInputFile = File("tmp.txt")
    simulOutputFile = File("output.%d.dat" % i)
    # Arguments to job
    # simulJob parameter=<i> input=tmp.txt output=output<i>.dat
    simulJob.addArgument("parameter=%d" % i, "input=tmp.txt",
        "output=%s" % simulOutputFile.getName())
    # Role of files
    simulJob.uses(simulInputFile, link=Link.INPUT)
    simulJob.uses(simulOutputFile, line=Link.OUTPUT)
    # Add job to dax
    dax.addJob(simulJob)
    # Dependency on firstJob
    dax.depends(parent=firstJob, child=simulJob)
# Write to file
fp = open("test.dax", "w")
dax.writeXML(fp)
fp.close()
```

Site Catalog

- Stores details about each target execution/storage site
  - Job submission endpoints (GRAM URL, etc.)
  - Paths to storage/scratch directories
  - Data transfer services (GridFTP servers, etc.)
  - Paths to credentials (X509 proxy, ssh key, etc.)
  - Site-level configuration (environment variables, etc.)
  - “local” site is special—refers to submit host

```
<!-- Example site catalog -->
<sitecatalog>
  <site handle="example" arch="x86_64" os="LINUX">
    <grid type="gt5" contact="example.isi.edu/jobmanager-fork” jobtype="auxillary"/>
    <grid type="gt5" contact="example.isi.edu/jobmanager-pbs” jobtype="compute"/>
    <directory type="shared-scratch" path="/scratch">
      <file-server operation="all" url="gsiftp://example.isi.edu/scratch"/>
    </directory>
    <profile namespace="env" key="GLOBUS_LOCATION">/usr/local/globus</profile>
    <profile namespace=“pegasus” key=“style”>globus</profile>
    <profile namespace=“pegasus” key=“X509_USER_PROXY”>/tmp/x509_u40001</profile>
  </site>
</sitecatalog>
```

Transformation Catalog

- Maps transformations to executables on each site
  - Physical path or URL of executable and dependent data/ configuration files
  - Executable characteristics (OS, architecture, glibc, etc.)
  - Job-level configuration (e.g. environment variables, profiles)

```
# Example transformation catalog
tr example::date {

  profile env ”TZ" ”America/Los_Angeles”

  site example {
    pfn "/bin/date"
    os "linux"
    arch "x86_64"
    type "INSTALLED"
  }
}
```

Replica Catalog

- Maps logical files to physical files
  - LFN (name) to PFN (path or URL)
  - Mappings annotated with metadata (e.g. site/pool, size, etc.)
- Enables Pegasus to choose “best” replica (replica selection phase of planner)
- Where Pegasus registers workflow output locations
- Support file-based or DB-based RC (also callout)

```
# Example replica catalog
f.1     gsiftp://example.isi.edu/inputs/f.1 pool=”example”
f.1     file:///inputs/f.1                  pool=”example”
f.2     file:///inputs/f.2                  pool=“example”
f.2     file:///inputs/f.2                  pool=“local”
```

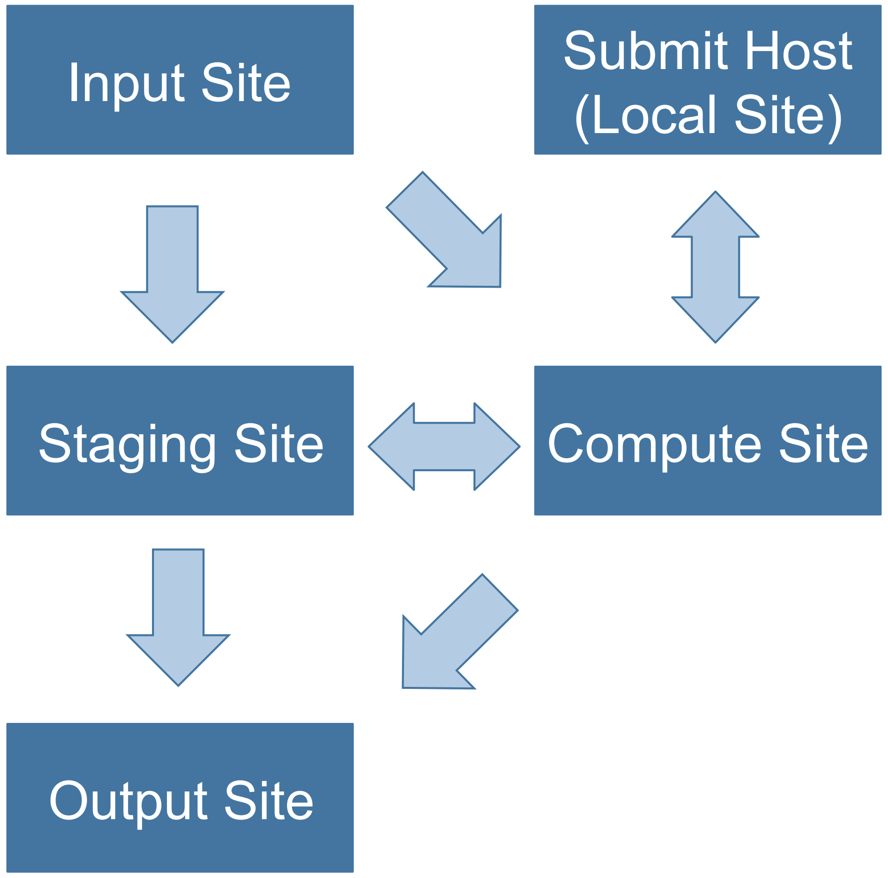
“pool” == site

Configuration Properties and Profiles

- Specify all the tuning knobs for Pegasus
- Unification of properties and profiles several years ago
- Often in a "pegasus.properties" file (or command-line)
- Some are global and apply to all sites and jobs
- Some (profiles) can also be specified in the TC, SC and DAX with different scopes
- Examples
  - pegasus.data.configuration = sharedfs
  - pegasus.style = condor
  - dagman.retry = 2
  - pegasus.exitcode.successmsg = "All data processed"

Data Management

- Pegasus supports several different data configurations
  - Many protocols
  - Complex data flows
- Workflow file types
  - Input
  - Intermediate
  - Output
- Sites
  - Local site: Pegasus WMS
  - Storage site: inputs and outputs
  - Staging site: intermediate
  - Compute site: compute jobs



Checkpoint Files

- A job can specify that it uses one or more checkpoint files
- Checkpoint files are both input files and output files
- Pegasus will stage-out these files in the case that job fails
  - Typically due to a timeout on long-running jobs
  - Jobs must periodically write checkpoint files (no signals are given)
- Pegasus will stage-in these files before retrying the job
  - They will appear in the working directory of the job

Workflow and Task Notifications

- Users want to be notified at certain points in the workflow or on certain events
- Support for adding notification to workflow and tasks
- Event based callouts
  - On Start, On End, On Failure, On Success
  - Examples contain email and jabber notification scripts
  - Can run any user provided scripts
  - Defined in the DAX

Pegasus clients for data management

- pegasus-transfer, pegasus-create-dir, pegasus-cleanup
- Support many different protocols
  - HTTP
  - SCP
  - GridFTP
  - IRODS
  - Amazon S3
  - SRM
  - cp
  - ln -s
- Remote directory creation and removal
- Support client discovery, parallel transfers, retries, and many other things to improve transfer performance and reliability

Different Directories used by Pegasus

1. **Submit Directory**
   - The directory where pegasus-plan generates the executable workflow i.e HTCondor DAGMan and job submit files.
   - Specified by ***--dir*** option to pegasus-plan
2. **Input Directory**
   - Mostly input file locations are catalogued in the Replica Catalog.
   - However, if inputs are on the submit host, then you can pass ***–input-dir*** option to pegasus-plan
3. **Scratch Directory**
   - Workflow specific directory created on the staging site by the **create-dir** job. This is where all the workflow inputs and outputs are gathered.
   - The base directory specified in the site catalog entry in ***sites.xml*** file.
4. **Output Directory**
   - The output directory where the outputs of the workflow appear.
   - Specified in the output site entry in the ***sites.xml*** file.
   - Can also be optionally specified by ***–output-dir*** option to pegasus-plan

Planning and Submitting workflows

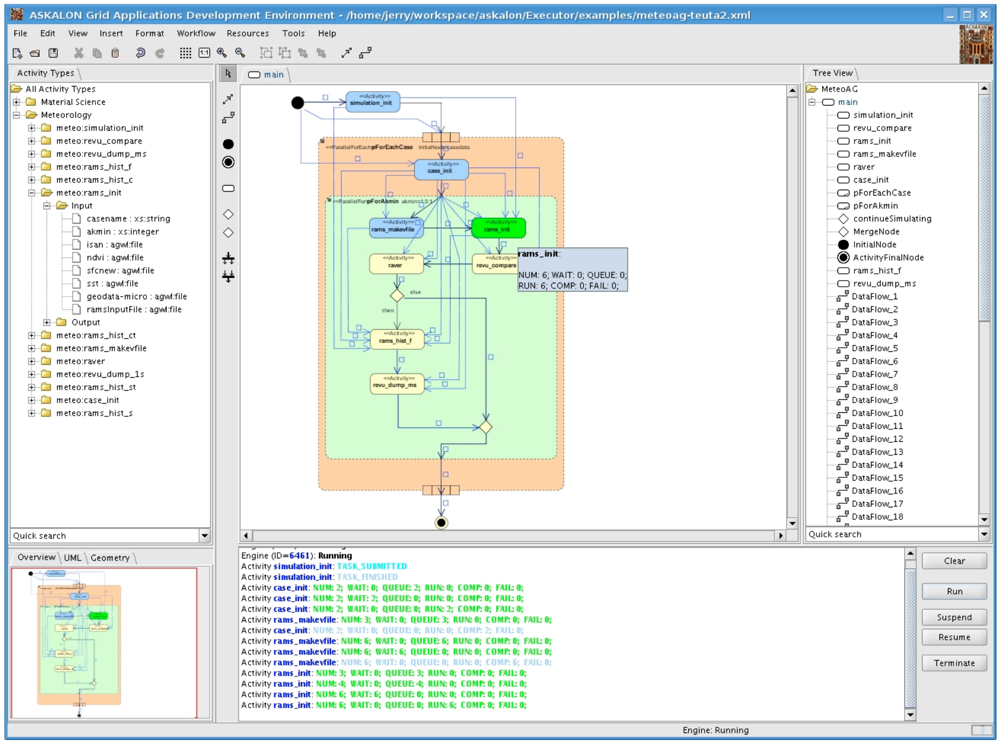
- **pegasus-plan**
  - Interface to the Pegasus planner
  - Specify input dir
  - Specify compute site(s)
  - Specify staging site(s)
  - Specify output dir or output site
- **Pegasus-run**
  - Start and restart the workflow

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, inputs
- Getting cores
  - Whether large parallel jobs or high throughput

Askalon (askalon.org)

- Developed at University of Innsbruck in Austria
- Create workflow description in AGWL (XML) or UML
  - if, for, parallelFor, DAGs
- Conversion: like planning, to bind to specific execution
- Submit jobs to Enactment Engine, which distributes jobs for execution at remote cluster, grid or cloud sites
- GUI for composition and monitoring



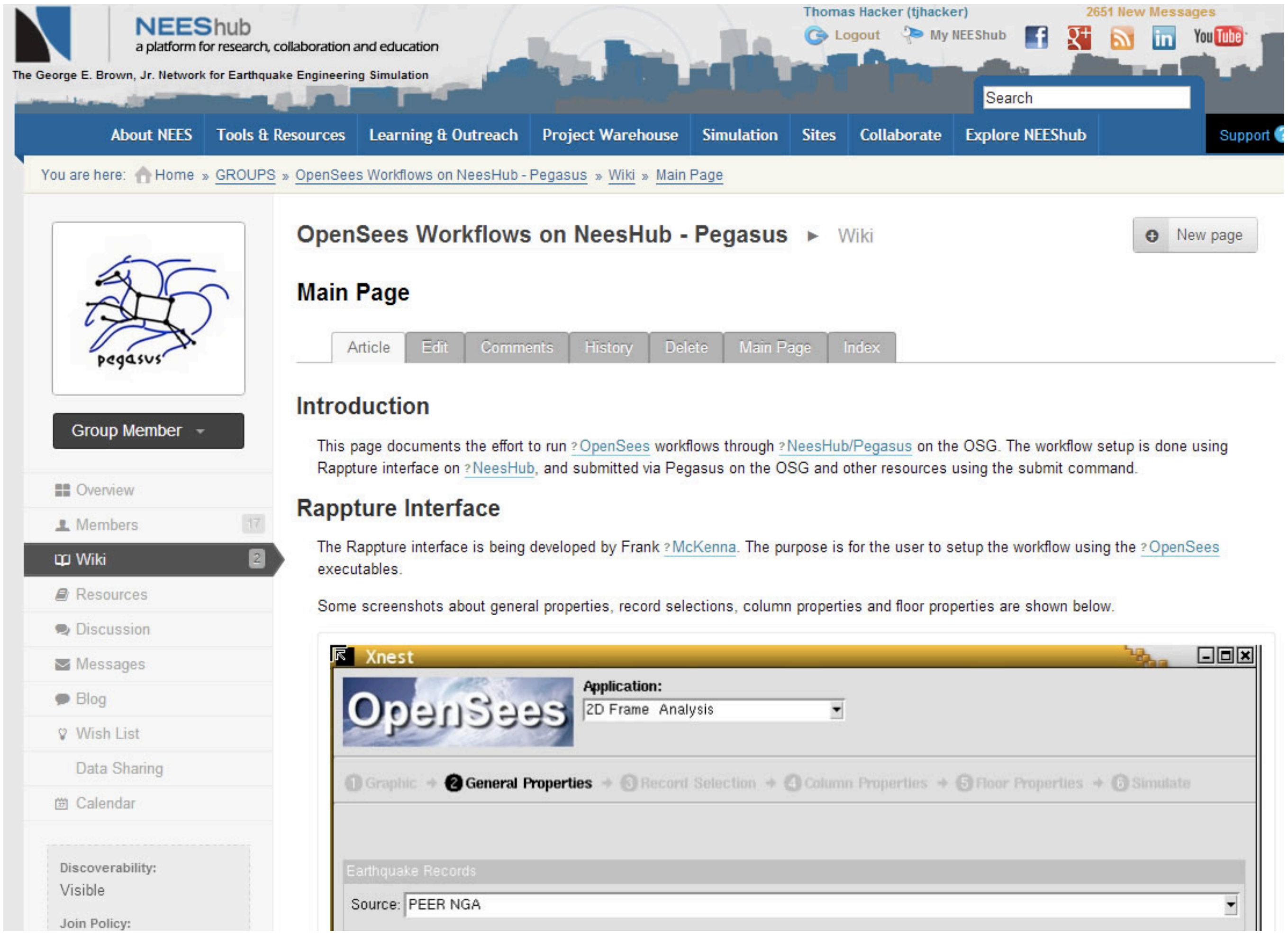
```
Engine (ID=6461): Running
Activity simulation_init: TASK_SUBMITTED
Activity simulation_init: TASK_FINISHED
Activity case_init: NUM: 2;  WAIT: 0;  QUEUE: 2;  RUN: 0;  COMP: 0;  FAIL: 0;
Activity case_init: NUM: 2;  WAIT: 2;  QUEUE: 0;  RUN: 0;  COMP: 0;  FAIL: 0;
Activity case_init: NUM: 2;  WAIT: 0;  QUEUE: 0;  RUN: 2;  COMP: 0;  FAIL: 0;
Activity rams_makevfile: NUM: 3;  WAIT: 0;  QUEUE: 3;  RUN: 0;  COMP: 0;  FAIL: 0;
Activity case_init: NUM: 2;  WAIT: 0;  QUEUE: 0;  RUN: 0;  COMP: 2;  FAIL: 0;
Activity rams_makevfile: NUM: 6;  WAIT: 0;  QUEUE: 6;  RUN: 0;  COMP: 0;  FAIL: 0;
Activity rams_makevfile: NUM: 6;  WAIT: 6;  QUEUE: 0;  RUN: 0;  COMP: 0;  FAIL: 0;
Activity rams_makevfile: NUM: 6;  WAIT: 0;  QUEUE: 0;  RUN: 0;  COMP: 6;  FAIL: 0;
Activity rams_init: NUM: 3;  WAIT: 0;  QUEUE: 3;  RUN: 0;  COMP: 0;  FAIL: 0;
Activity rams_init: NUM: 4;  WAIT: 0;  QUEUE: 4;  RUN: 0;  COMP: 0;  FAIL: 0;
Activity rams_init: NUM: 6;  WAIT: 6;  QUEUE: 0;  RUN: 0;  COMP: 0;  FAIL: 0;
Activity rams_init: NUM: 6;  WAIT: 0;  QUEUE: 0;  RUN: 6;  COMP: 0;  FAIL: 0;
```

Example Hierarchical Workflow

- <dax> element behaves like <job>
  - Arguments are for pegasus-plan (most are inherited)
- Planner is invoked when DAX job is ready to run

```
<?xml version="1.0" encoding="UTF-8"?>
<adag version="3.4" name="multi-level">
        <jobid="ID0000001"namespace="example"name="sleep">
                <argument>5</argument>
        </job>
        <dax id="ID0000002" file="sub.dax">
                <argument>--output-site local</argument>
        </dax>
        <jobid="ID0000003"namespace="example"name="sleep">
                <argument>5</argument>
        </job>
        <child ref="ID0000002">
                <parent ref="ID0000001"/>
        </child>
        <child ref="ID0000003">
                <parent ref="ID0000002"/>
        </child>
</adag>
```

Integration with HUBzero



NEEShub
a platform for research, collaboration and education
The George E. Brown, Jr. Network for Earthquake Engineering Simulation

Thomas Hacker (tjhacker) 2651 New Messages
Logout My NEEShub

About NEES | Tools & Resources | Learning & Outreach | Project Warehouse | Simulation | Sites | Collaborate | Explore NEEShub | Support

You are here: Home » GROUPS » OpenSees Workflows on NeesHub - Pegasus » Wiki » Main Page

OpenSees Workflows on NeesHub - Pegasus ▸ Wiki

Main Page

Article | Edit | Comments | History | Delete | Main Page | Index

Introduction

This page documents the effort to run ?OpenSees workflows through ?NeesHub/Pegasus on the OSG. The workflow setup is done using Rappture interface on ?NeesHub, and submitted via Pegasus on the OSG and other resources using the submit command.

Rappture Interface

The Rappture interface is being developed by Frank ?McKenna. The purpose is for the user to setup the workflow using the ?OpenSees executables.

Some screenshots about general properties, record selections, column properties and floor properties are shown below.

Group Member

Overview
Members 17
Wiki 2
Resources
Discussion
Messages
Blog
Wish List
Data Sharing
Calendar

Discoverability:
Visible

Join Policy:

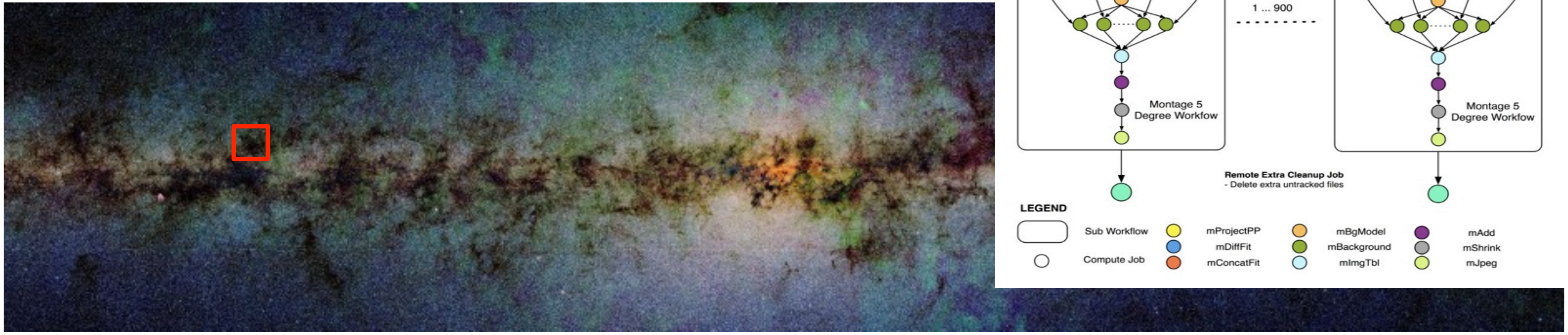
Xnest
Application: 2D Frame Analysis
1 Graphic → 2 General Properties → 3 Record Selection → 4 Column Properties → 5 Floor Properties → 6 Simulate
Earthquake Records
Source: PEER NGA

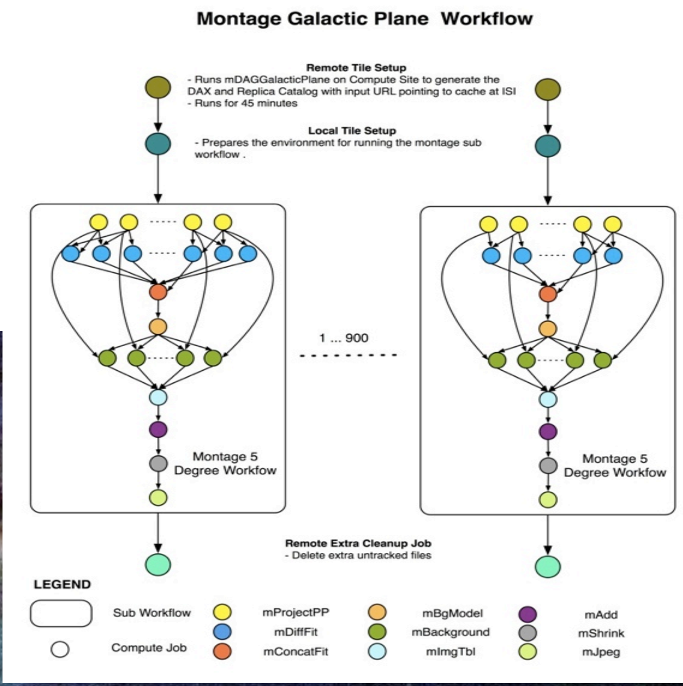
Credit: Frank McKenna UC Berkeley, NEES, HUBzero

Key Pegasus Concepts

- Workflows are DAGs (or hierarchical DAGs)
  - No loops, no conditional branches
- Pegasus WMS == Pegasus planner (mapper) + DAGMan workflow engine + Condor scheduler
  - The planner does not schedule jobs
- Planning occurs ahead of execution
  - (Except hierarchical workflows)
- Planning converts an abstract workflow into a concrete, executable workflow
  - Planner is like a compiler

Data-intensive Workflows




Montage Galactic Plane Workflow
Remote Tile Setup
- Runs mDAGGalacticPlane on Compute Site to generate the DAX and Replica Catalog with input URL pointing to cache at ISI
- Runs for 45 minutes
Local Tile Setup
- Prepares the environment for running the montage sub workflow .
1 ... 900
Montage 5 Degree Workfow
Montage 5 Degree Workfow
Remote Extra Cleanup Job
- Delete extra untracked files
LEGEND
Sub Workflow
Compute Job
mProjectPP
mDiffFit
mConcatFit
mBgModel
mBackground
mImgTbl
mAdd
mShrink
mJpeg

John Good (Caltech)

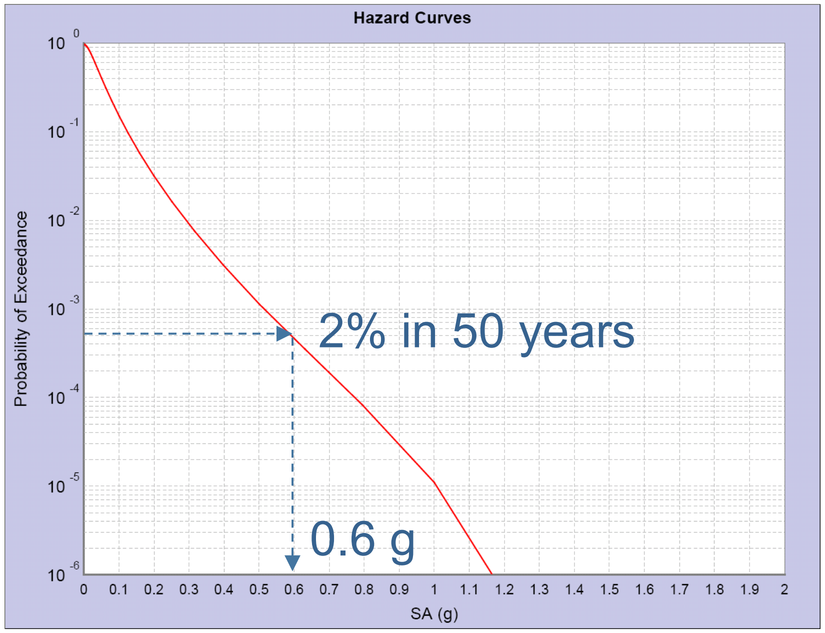
- **Montage Galactic Plane Workflow**
  - **18 million input images (~2.5 TB)**
  - **900 output images (2.5 GB each, 2.4 TB total)**
  - **10.5 million tasks (34,000 CPU hours)**
  - **Run on Amazon EC2 2013-2014**

× 17

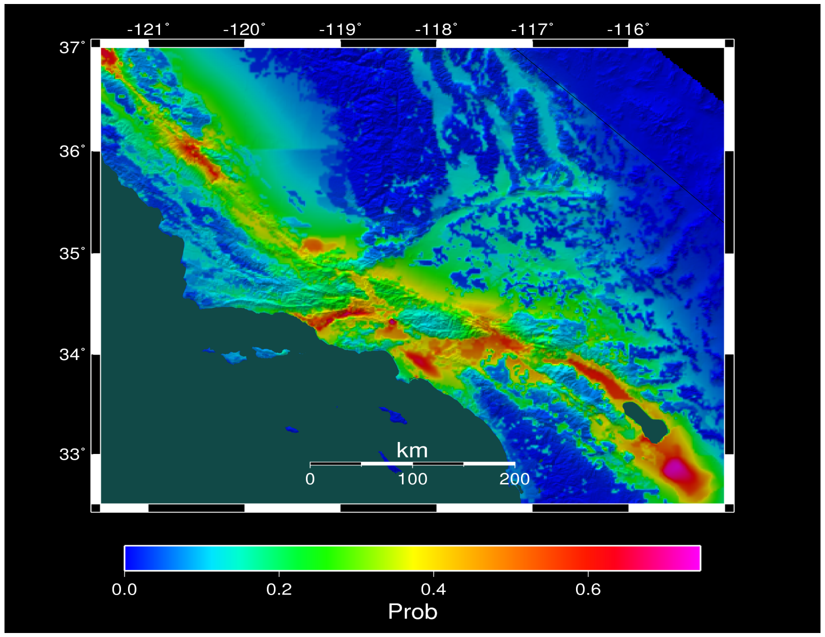
- **Need to support hierarchical workflows and scale**

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



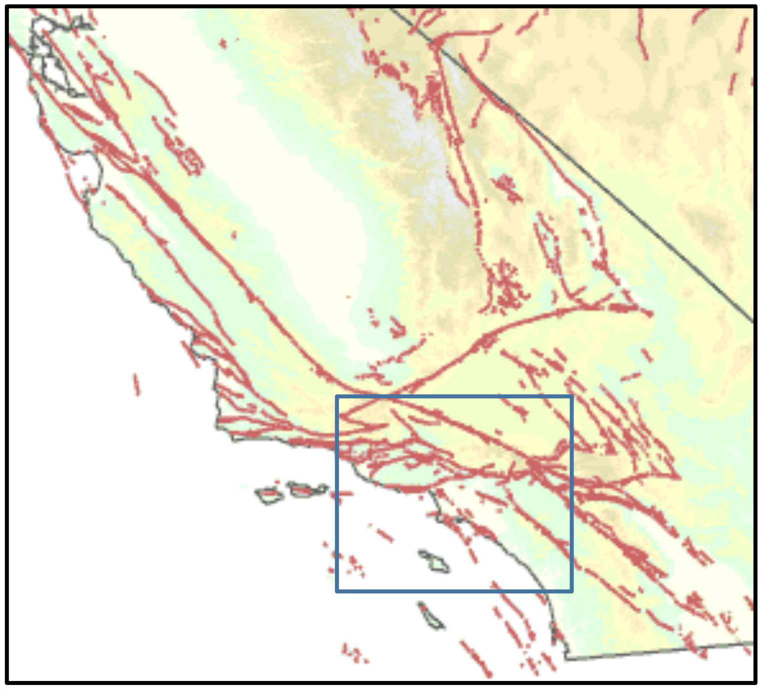
Hazard curve for downtown LA

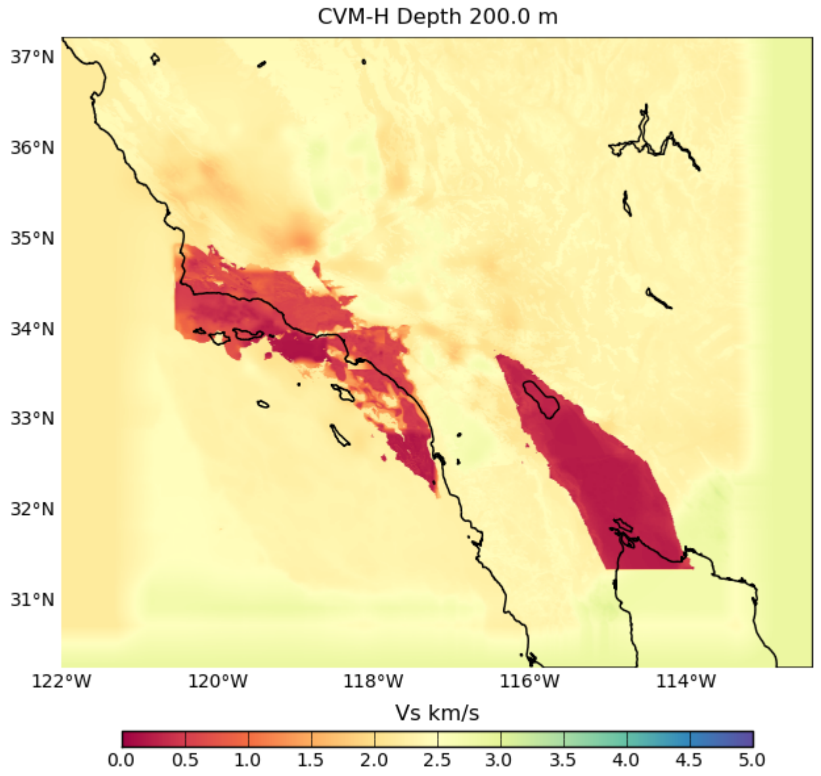


Probability of exceeding 0.1g in 50 yrs

Seismic Hazard Analysis Calculation

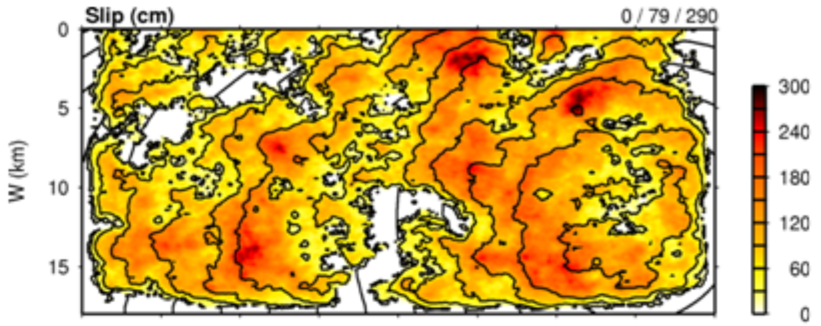
- Tensor simulation
  - Create 1.5 billion point mesh with material properties
  - Generate tensors across volume
  - Parallel, ~8,000 CPU-hrs

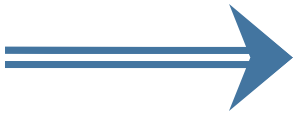


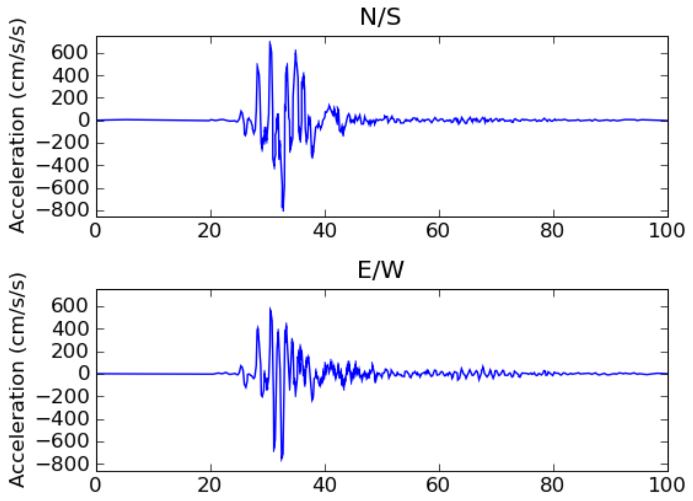


Post-Processing

- **Individual earthquake contributions**
  - Get list of earthquakes of interest (~415,000)
  - Simulate seismograms for each earthquake
  - Loosely-coupled, short-running serial jobs
- **Combine the levels of shaking with probabilities to produce a hazard curve.**







Computational Requirements

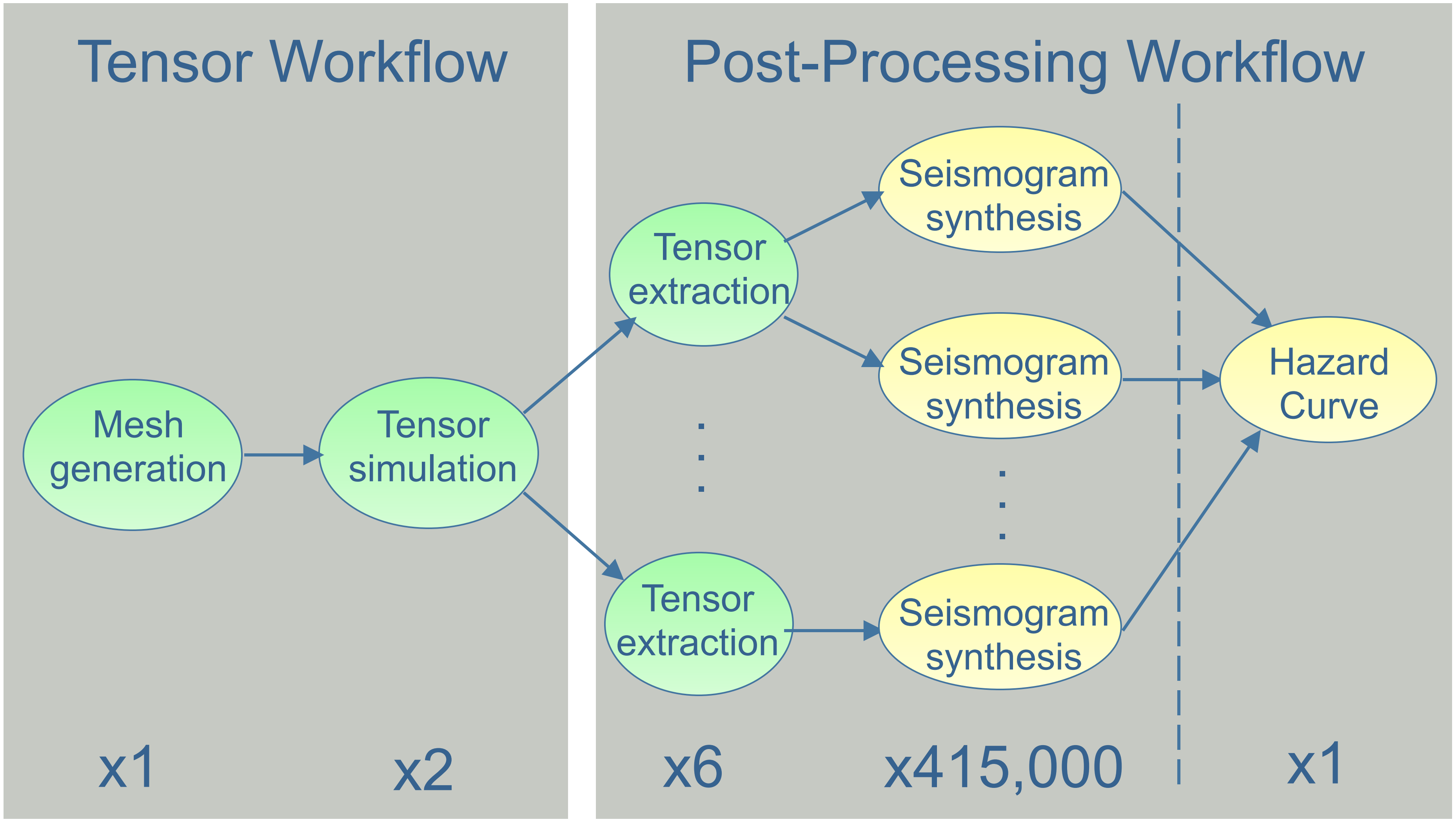
	Component	Data	Executions	Cores/exec	Core hours
Tensor Creation	Mesh generation	15 GB	1	320	50
	Tensor simulation	40 GB	2	10,000 CPU 100 GPU	16,000 CPU 1,200 GPU
Post Processing	Tensor extraction	690 GB	6	256	275
	Seismogram synthesis	12 GB	**415,000**	1	2,300
	Curve generation	1 MB	1	1	< 1
	Total	757 GB	415,000		18,625

This is for **one** location of interest; we wanted to run ~1000

Why Scientific Workflows?

- Large-scale, heterogeneous, high throughput
  - Parallel and many (~415,000) serial tasks
  - Task duration 100 ms – 2 hours
- Automation
- Data management
- Error recovery
- Resource provisioning
- Scalable
- System-independent description

CyberShake workflows



Challenge: Resource Provisioning

- In tensor workflow, submit job to remote scheduler
  - GRAM puts jobs in remote queue
  - Runs like a normal batch job
  - Can specify either CPU or GPU nodes
- For post-processing workflow, 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)

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 local archival disk
  - Pegasus registers data products in catalog
  - Cleans up temporary files when no longer needed
- Directory hierarchy to reduce files per directory
- Added automated checks to check integrity
  - Correct number of files, NaN, zero-value checks
  - Included as new jobs in workflow

Challenge: File System Load

- Seismogram tasks cause heavy I/O load
  - Reads an earthquake description
  - Writes a seismogram file
- Reduce reads
  - Generate earthquake description on the fly, from geometry
  - Added memcached to cache rupture geometry
    - Local memory cache on compute node
    - Pegasus-mpi-cluster hook for custom startup script
- Reduce writes
  - Pegasus-mpi-cluster supports “pipe forwarding”
  - Workers write to pipes, master aggregates to fewer files

CyberShake Study 14.2

- Hazard curves for 1144 sites
- 46,720 CPUs + 225 GPUs for 14 days (Blue Waters)
  - Peak of 295,040 CPUs, 1100 GPUs
- 99.8 million tasks executed
  - 81 tasks/sec
  - Only 31,463 jobs in Blue Waters queue
- On average, 26.2 workflows running concurrently
- Managed 830 TB of data
  - 57 TB output files
  - 12.3 TB staged back to local disk (~16M files)
- Workflow tools scale!