Pegasus WMS: A workflow system for running large scale workflows on national cyberinfrastructure

Karan Vahi

Science Automation Technologies Group

USC Information Sciences Institute

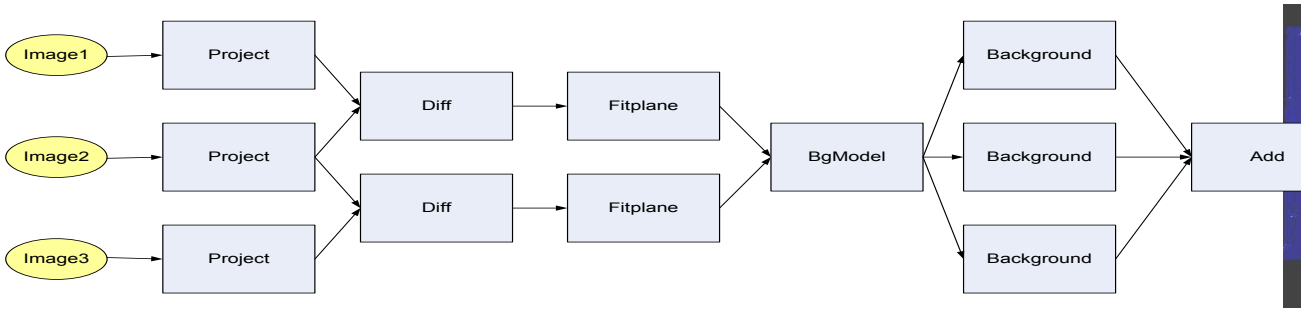
Outline of Talk

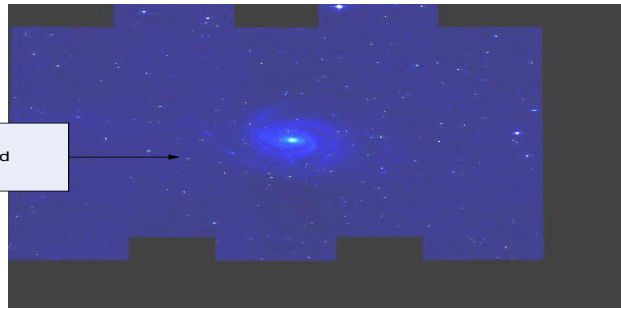
- Introduction to Scientific Workflows and Pegasus
- Data Management in Pegasus
- Workflow Monitoring and Debugging

Scientific Workflows

- Capture individual data transformation and analysis steps
- Large monolithic applications broken down to smaller jobs
  - Smaller jobs can be independent or connected by some control flow/ data flow dependencies
  - Usually expressed as a Directed Acyclic Graph of tasks
- Allows the scientists to modularize their application
- Scaled up execution over several computational resources
- Provide automation
- Foster Collaborations

Generating mosaics of the sky (Bruce Berriman, Caltech)





Size of the mosaic in degrees square*	Number of jobs	Number of input data files	Number of Intermediate files	Total Data Footprint	Approx. execution time (20 procs)
1	232	53	588	1.2GB	40 mins
2	1,444	212	3,906	5.5GB	49 mins
4	4,856	747	13,061	20GB	1hr 46 mins
6	8,586	1,444	22,850	38GB	2 hrs. 14 mins
10	20,652	3,722	54,434	97GB	6 hours

*The full moon is 0.5 deg. sq. when viewed form Earth, Full Sky is ~ 400,000 deg. sq.

Workflows – Launch and Forget

- A single workflow can take days, weeks or even months
- Automates tasks user *could* perform manually…

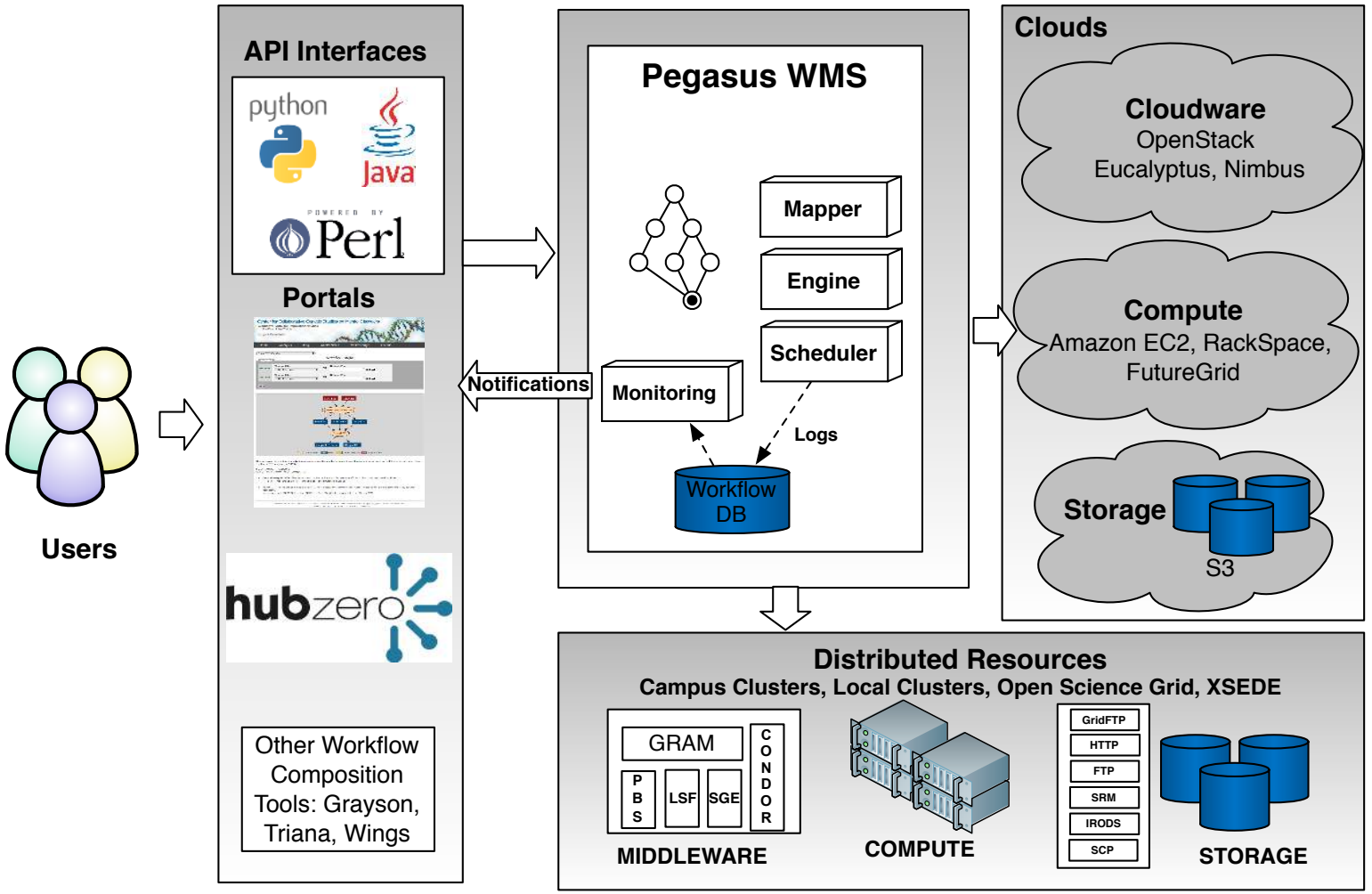
…but **WMS** takes care of automatically

- Includes features such as retries in the case of failures – avoids the need for user intervention
- The workflow itself can include error checking
- The result: one user action can utilize many resources while maintaining complex job inter-dependencies and data flows
- Maximizes compute resources / human time

Pegasus
Workflow Management System (est. 2001)

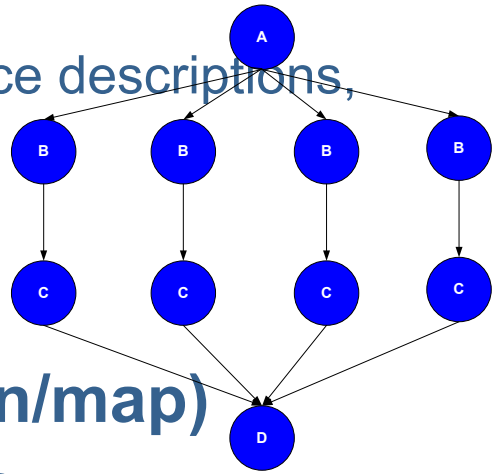
- A collaboration between USC and the Condor Team at UW Madison (includes DAGMan)
- Maps a resource-independent "abstract" workflow onto resources and executes the "executable" workflow
- Used by a number of applications in a variety of domains
- Provides reliability—can retry computations from the point of failure
- Provides scalability—can handle large data and many computations (kbytes-TB of data, 1-10^6 tasks)
- Infers data transfers, restructures workflows for performance
- Automatically captures provenance information
- Can run on resources distributed among institutions, laptop, campus cluster, Grid, Cloud

Pegasus WMS


API Interfaces
python
Java
Perl
Portals
hubzero
Other Workflow Composition Tools: Grayson, Triana, Wings
Users
Notifications
Pegasus WMS
Mapper
Engine
Scheduler
Monitoring
Logs
Workflow DB
Clouds
Cloudware
OpenStack
Eucalyptus, Nimbus
Compute
Amazon EC2, RackSpace, FutureGrid
Storage
S3
Distributed Resources
Campus Clusters, Local Clusters, Open Science Grid, XSEDE
GRAM
PBS
LSF
SGE
CONDOR
MIDDLEWARE
COMPUTE
GridFTP
HTTP
FTP
SRM
IRODS
SCP
STORAGE

Pegasus Workflow Management System

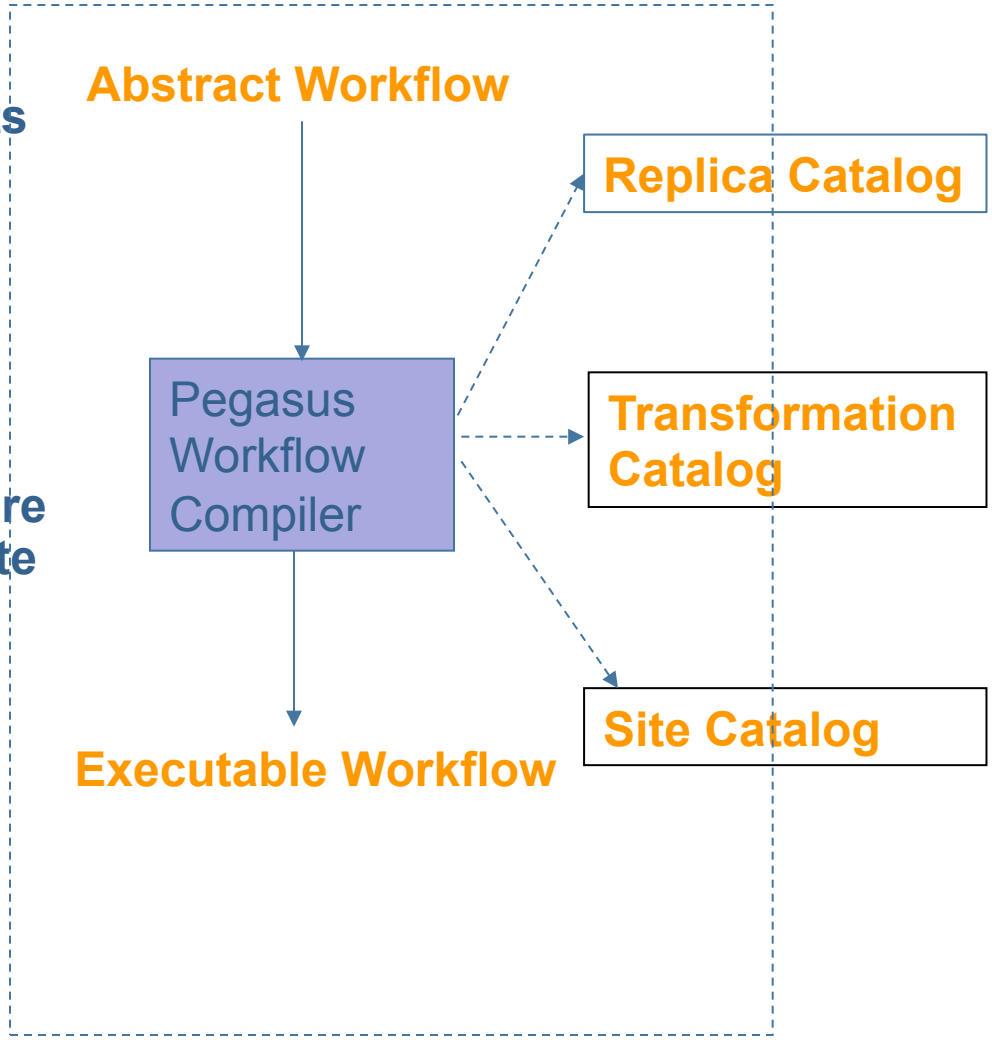
- **Abstract Workflows - Pegasus input workflow description**
  - Workflow “high-level language”
  - Only identifies the computation, devoid of resource descriptions, devoid of data locations
  - File Aware



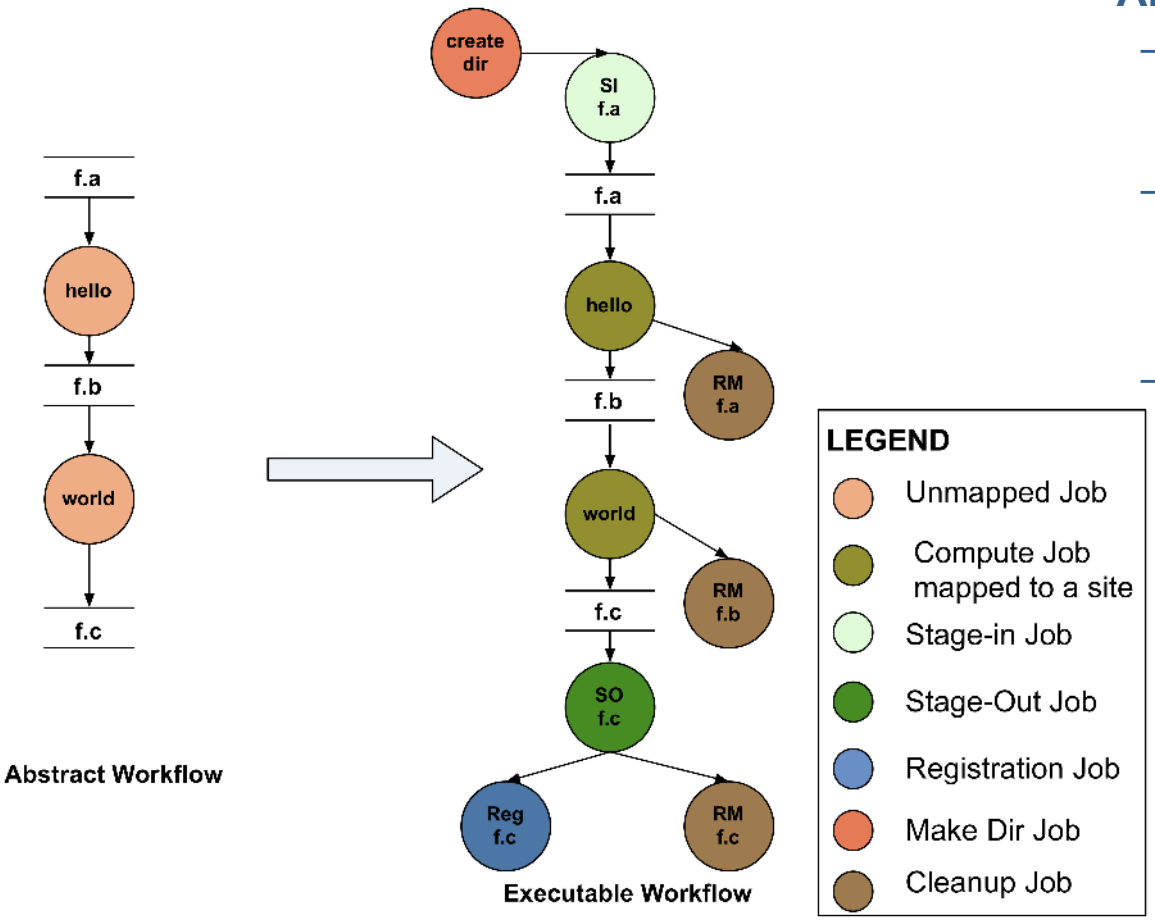
- **Pegasus is a workflow “compiler” (plan/map)**
  - Target is DAGMan DAGs and Condor submit files
  - Transforms the workflow for performance and reliability
  - Automatically locates physical locations for both workflow components and data
  - Collects runtime provenance

Abstract to Executable Workflow Mapping - Discovery

- **Data**
  - Where do the input datasets reside?
- **Executables**
  - Where are the executables installed ?
  - Do binaries exist somewhere that can be staged to remote grid sites?
- **Site Layout**
  - What does a execution site look like?



Abstract to Executable Workflow Mapping



- **Abstraction provides**
  - **Ease of Use (do not need to worry about low-level execution details)**
  - **Portability (can use the same workflow description to run on a number of resources and/or across them)**
  - **Gives opportunities for optimization and fault tolerance**
    - **automatically restructure the workflow**
    - **automatically provide fault recovery (retry, choose different resource)**

Outline of Talk

- Introduction to Scientific Workflows and Pegasus
- Data Management in Pegasus
- Workflow Monitoring and Debugging

Data Management in Pegasus

- **Data Discovery**
  - Where do input datasets and executables reside
  - Can I select amongst multiple input locations
- **Move data to where the jobs execute**
  - How do you ship in the small/large amounts data required by the workflows?
  - Can I use SRM? How about GridFTP? HTTP and Squid proxies?
  - Can I use Cloud based storage like S3 on EC2?
- **Data Reuse**
  - Reuse existing data products instead of re-computing them
- **Data Space Optimizations**
  - Remove files when no longer required by the workflow

Data Discovery - Replica Catalog

- **Replica Catalog stores mappings between logical filenames and their target locations**
- **Used to**
  - discover input files for the workflow
  - track data products created
  - Data is replicated for scalability, reliability and availability
- **Supported Types**
  - **File based Replica Catalog**
    - useful for small datasets
    - cannot be shared across users
  - **Database based Replica Catalog**
    - useful for medium sized datasets
    - can be used across users
  - **Globus Replica Location Service**
    - useful for large scale data sets across multiple users
    - LIGO's LDR deployment that scales to millions of files

Data Discovery – Replica Selection

- **Input Files maybe replicated at multiple sites**
  - How do you select the which input file to access?
  - LIGO Data Grid
    - Multiple tiers of replication
    - Central Index of locations of inputs based on RLS
    - However, not all users have access to replicas

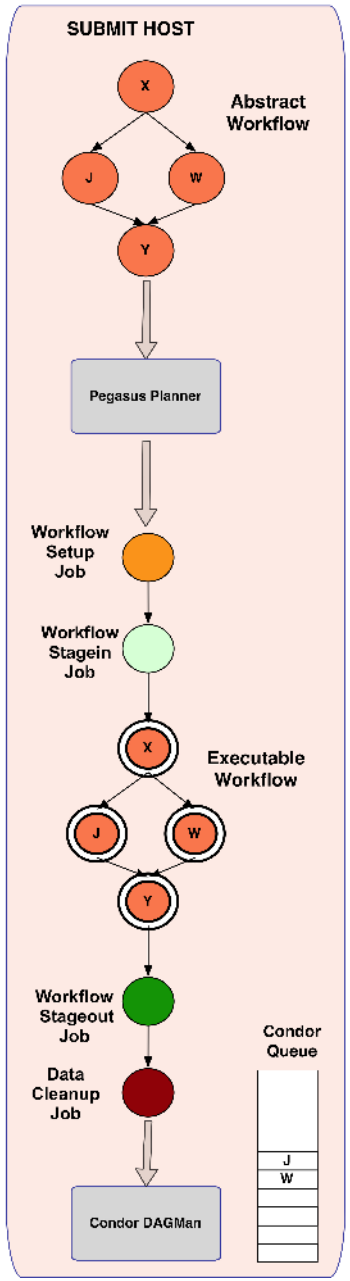
- **Supported Replica Selection Policies**
  - Prefer local files and symlink against them
  - For compute sites specify preferred locations or blacklist sites
  - User defined policies based on regular expression ranks

Move data to where the jobs execute

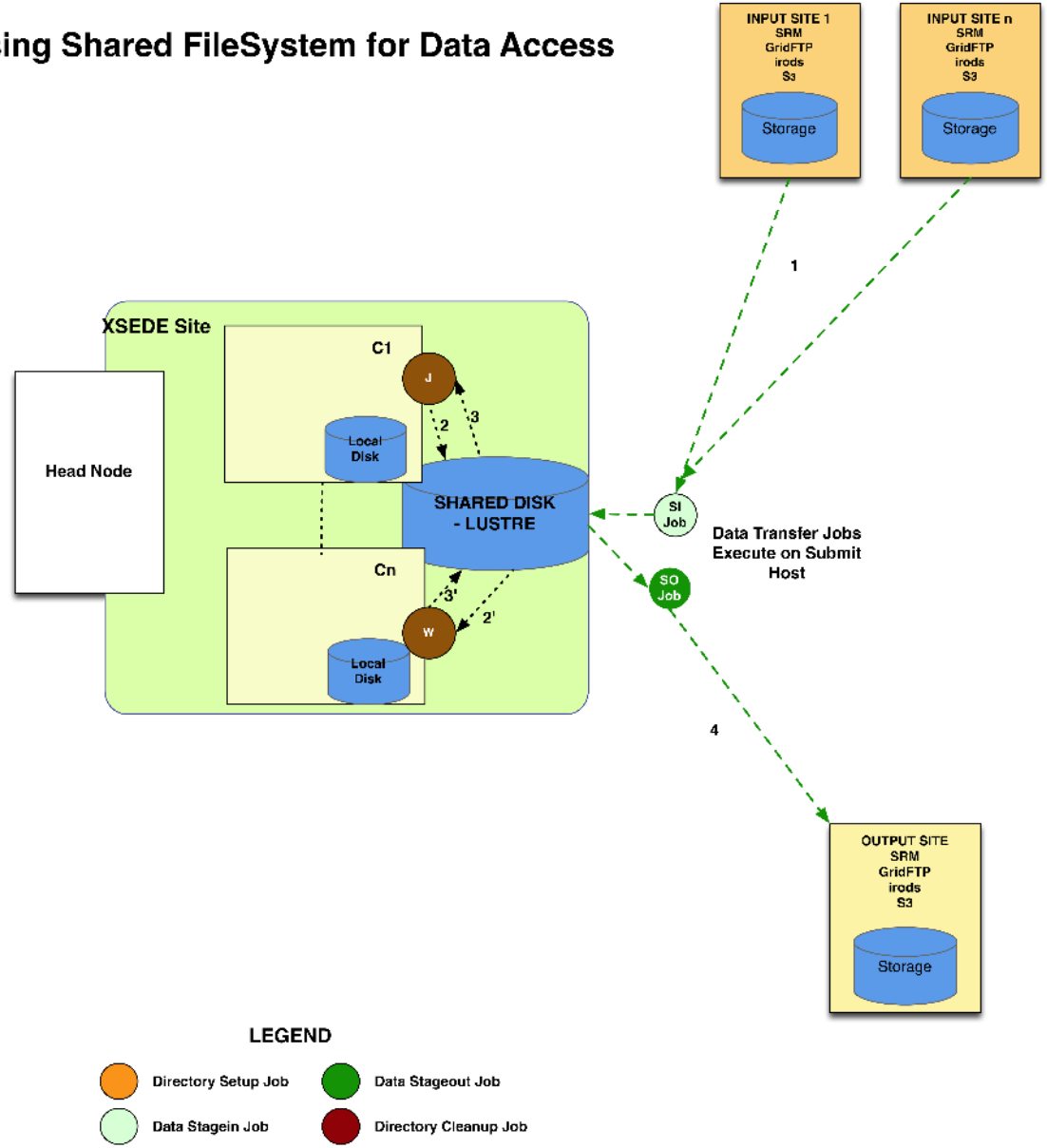
Three Main Configurations

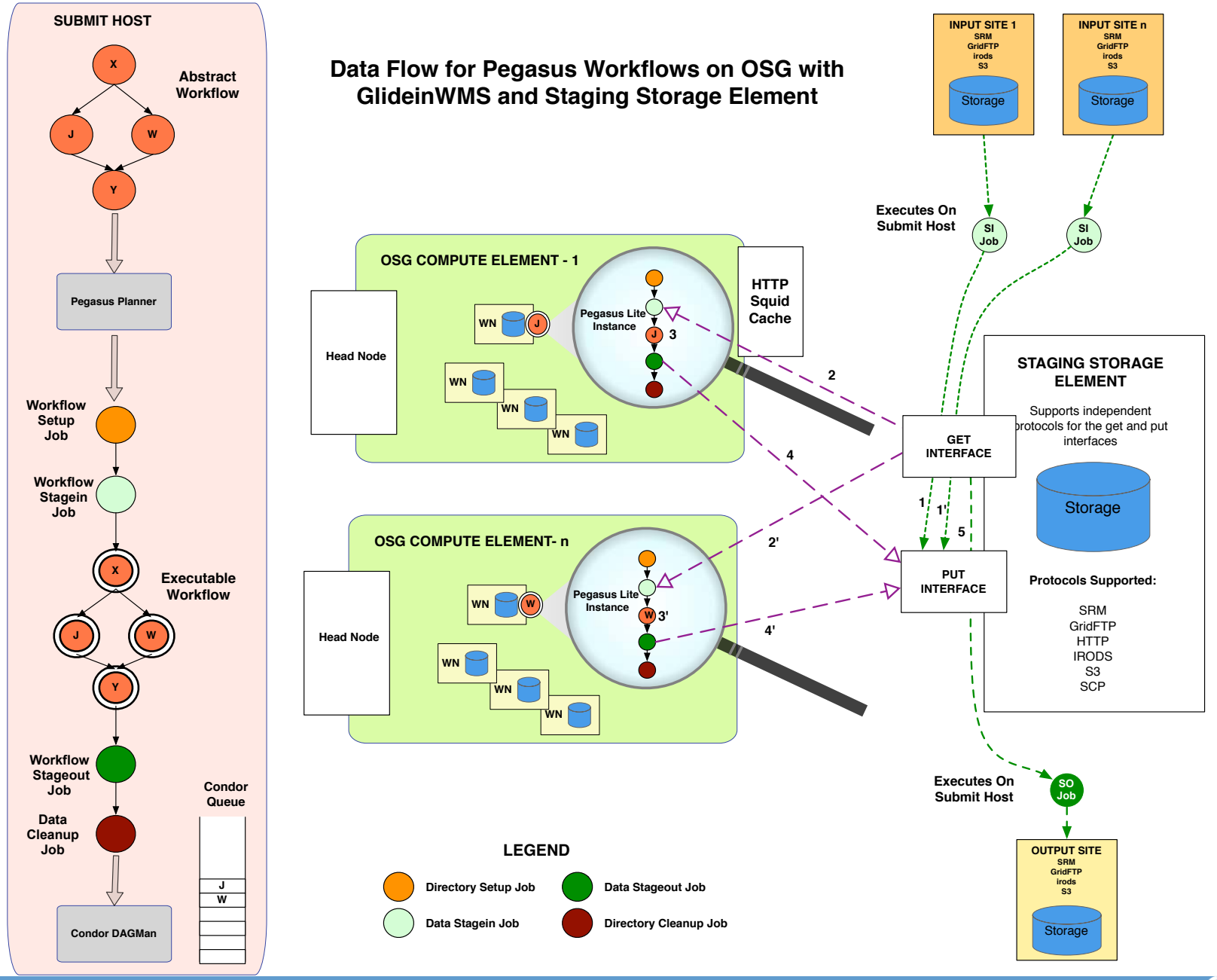
- **Shared Filesystem setup (typical of XSEDE and HPC sites)**
  - Worker nodes and the head node have a shared filesystem, usually a parallel filesystem with great I/O characteristics
  - Can leverage symlinking against existing datasets
- **NonShared filesystem setup using an existing storage element for staging (typical of OSG and campus Condor pools)**
  - Worker nodes don't share a filesystem.
  - Data is pulled from / pushed to the existing storage element.
- **Condor IO (Typical of large Condor Pools like CHTC)**
  - Worker nodes don't share a filesystem
  - Symlink against datasets available locally
  - Data is pulled from / pushed to the submit host via Condor file transfers

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



Using Shared FileSystem for Data Access

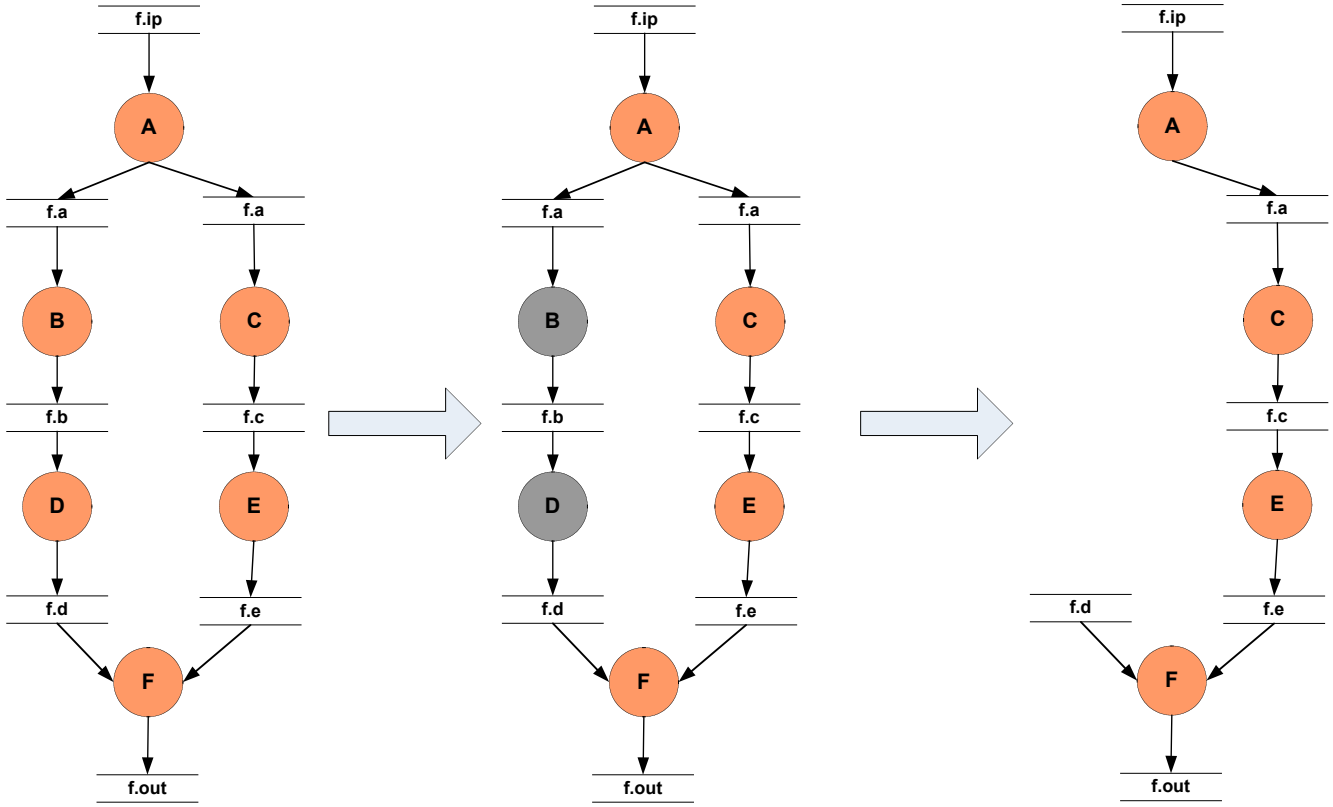



SUBMIT HOST
X
J
W
Y
Abstract Workflow
Pegasus Planner
Workflow Setup Job
Workflow Stagein Job
X
J
W
Y
Executable Workflow
Workflow Stageout Job
Data Cleanup Job
Condor DAGMan
Condor Queue
J
W
Data Flow for Pegasus Workflows on OSG with GlideinWMS and Staging Storage Element
INPUT SITE 1
SRM
GridFTP
irods
S3
Storage
INPUT SITE n
SRM
GridFTP
irods
S3
Storage
Executes On Submit Host
SI Job
SI Job
OSG COMPUTE ELEMENT - 1
Head Node
WN
J
WN
WN
WN
Pegasus Lite Instance
J
3
HTTP Squid Cache
2
4
OSG COMPUTE ELEMENT- n
Head Node
WN
W
WN
WN
WN
Pegasus Lite Instance
W
3'
2'
4'
GET INTERFACE
PUT INTERFACE
1
1'
5
STAGING STORAGE ELEMENT
Supports independent protocols for the get and put interfaces
Storage
Protocols Supported:
SRM
GridFTP
HTTP
IRODS
S3
SCP
Executes On Submit Host
SO Job
OUTPUT SITE
SRM
GridFTP
irods
S3
Storage
LEGEND
Directory Setup Job
Data Stagein Job
Data Stageout Job
Directory Cleanup Job

Key to supporting different data configurations

- Pegasus has a notion of **staging site**
- All the auxiliary jobs added by Pegasus place or retrieve data from the staging site
- In case of sharedfs approach, the shared filesystem on the compute site is the staging site
- In non-sharedfs deployments like Clouds, OSG we have a staging site separate from the compute site.
  - The jobs pull input data from staging site when they start up.
  - The jobs push output data to the staging site when they finish.

Workflow Reduction (Data Reuse)



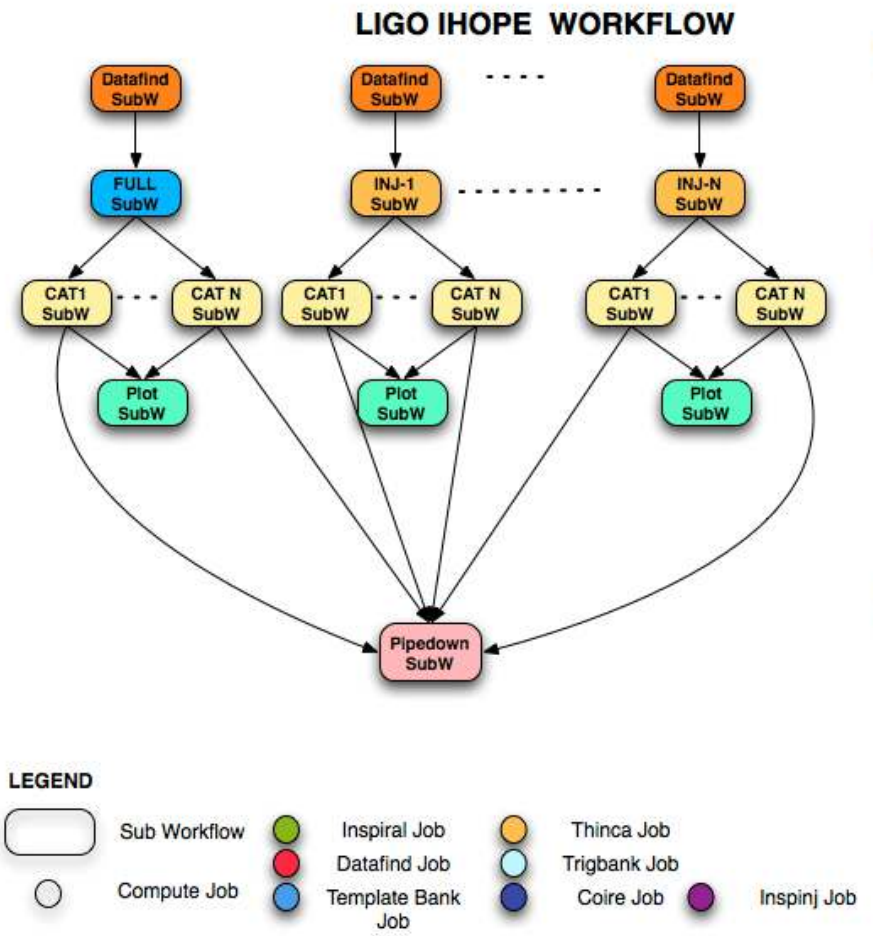
Abstract Workflow

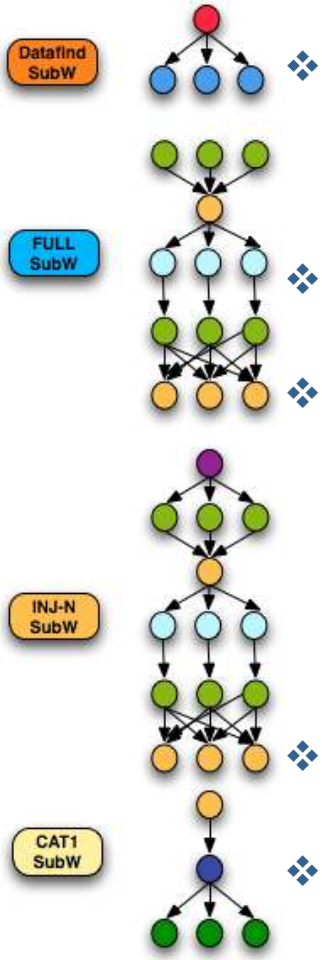
File f.d exists somewhere.
Reuse it.
Mark Jobs D and B to delete

Delete Job D and Job B

Useful when you have done a part of computation and then realize the need to change the structure. Re-plan instead of submitting rescue DAG!

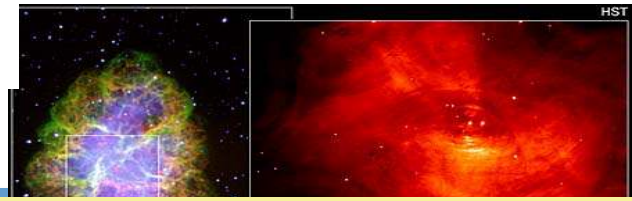
LIGO INSPIRAL WORKFLOWS





- Continuous gravitational waves are expected to be produced by a variety of celestial objects
- Only a small fraction of potential sources are known
- Need to perform blind searches, scanning the regions of the sky where we have no a priori information of the presence of a source
  - Wide area, wide frequency searches
- Search for binary inspirals collapsing into black holes.
- Usually executed on the LIGO Data Grid

- Typical LIGO Workflow
  - 185,000 nodes, 466,000 edges
  - **10TB** of input data accessed
  - Generates **1TB** of output data



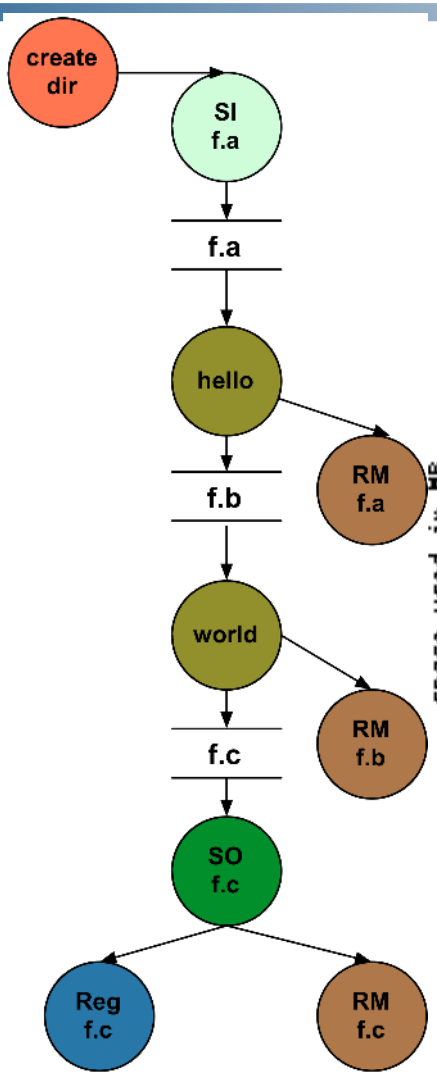
Pegasus Features Used: Data Reuse , Job Clustering, Hierarchal Workflows, Debugging tools, Run in non shared filesystem environments

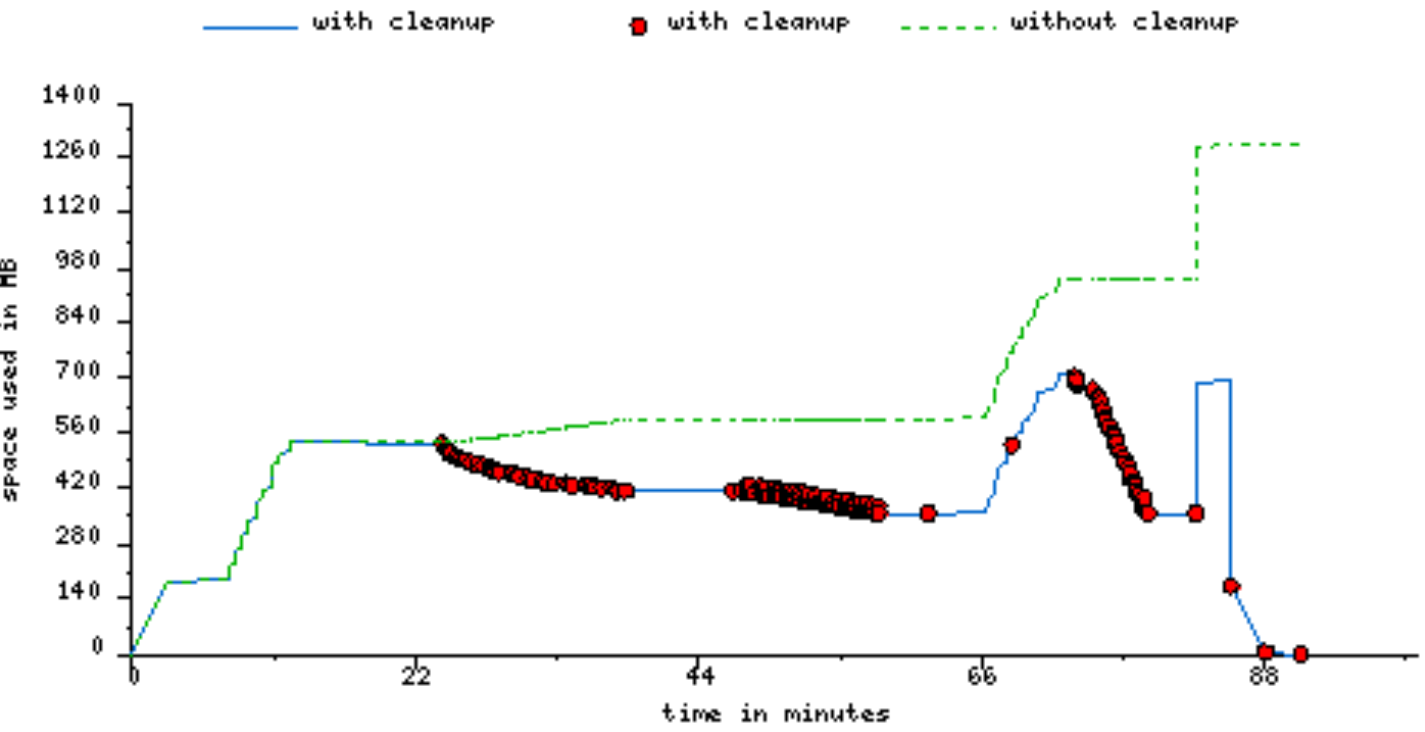
File cleanup

- **Problem: Running out of disk space during workflow execution**
- **Why does it occur**
  - Workflows could bring in huge amounts of data
  - Data is generated during workflow execution
  - Users don't worry about cleaning up after they are done
- **Solution**
  - **Do cleanup after workflows finish**
    - Does not work as the scratch may get filled much before during execution
  - **Interleave cleanup automatically during workflow execution.**
    - Requires an analysis of the workflow to determine, when a file is no longer required
  - **Cluster the cleanup jobs by level for large workflows**

Real Life Example: Used by a UCLA genomics researcher to delete TB's of data automatically for long running workflows!!

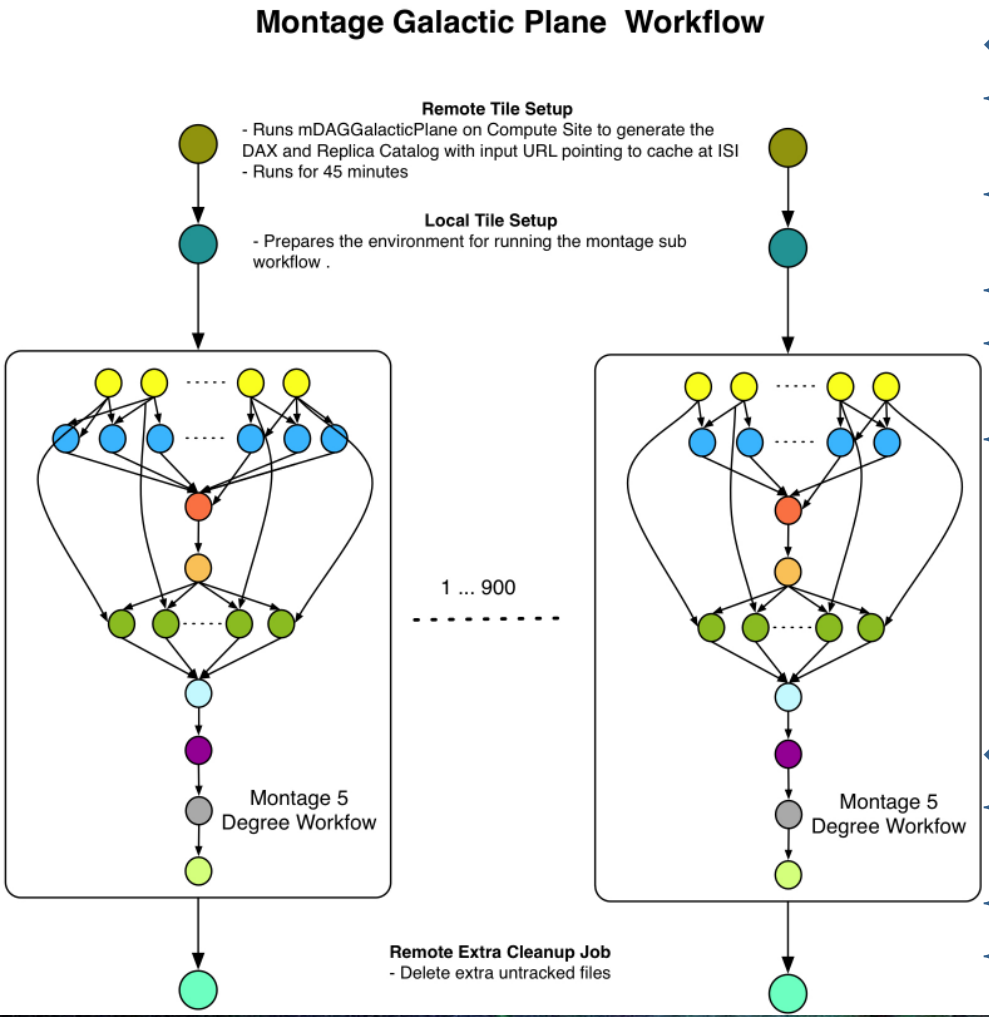
File cleanup (cont)





Montage 1 degree workflow run with cleanup

GALACTIC PLANE WORKFLOWS



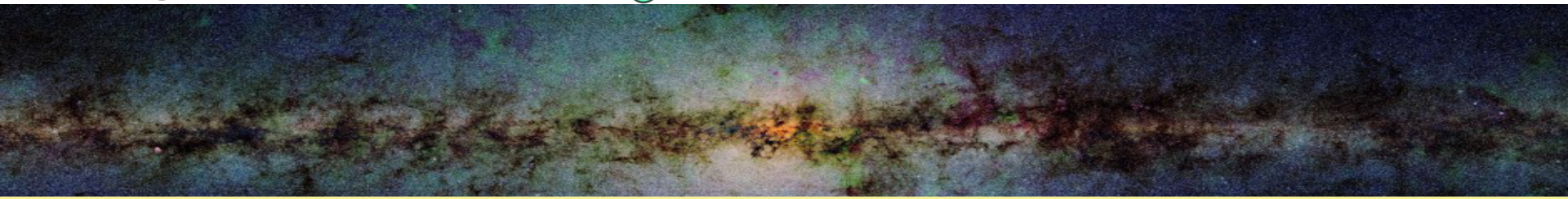
❖ Description

- ✧ Galactic Plane for generating mosaics from the NASA Telescope Missions like Spitzer etc.
- ✧ Used to generate tiles 360 x 40 around the galactic equator
- ✧ A tile 5 x 5 with 1 overlap with neighbors
- ✧ Output datasets to be potentially used in NASA Sky and Google Sky
- ✧ Each per band workflow
  - ➢ **1.6 million** input files
  - ➢ **10.5 million** tasks
  - ➢ Consumes **34,000 CPU hours**
  - ➢ Generates **900 tiles** in FITS format

× 17

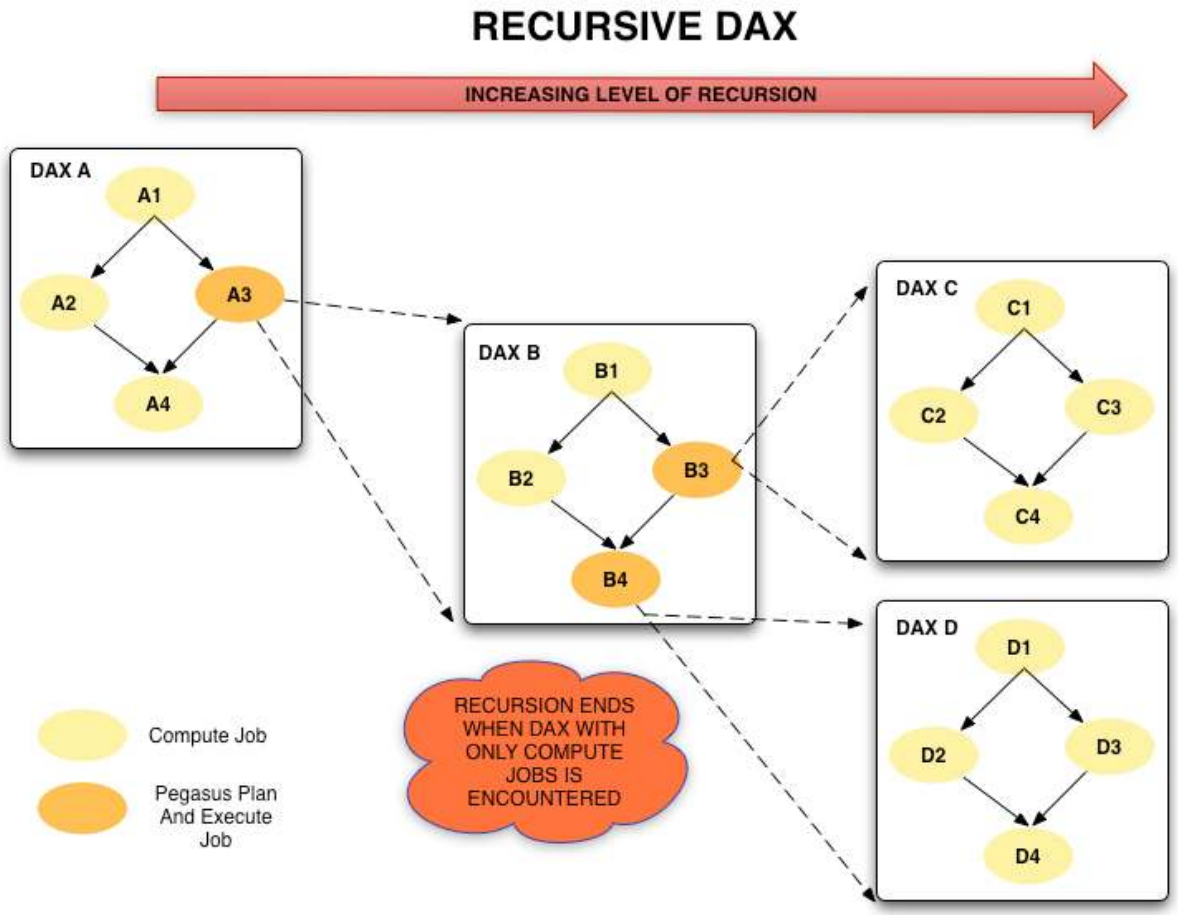
❖ Ongoing Runs on XSEDE, Amazon and OSG

- ✧ Run workflows corresponding to each of the 17 bands (wavelengths)
- ✧ Total Number of Data Files – **18 million**
- ✧ Potential Size of Data Output – **86 TB**

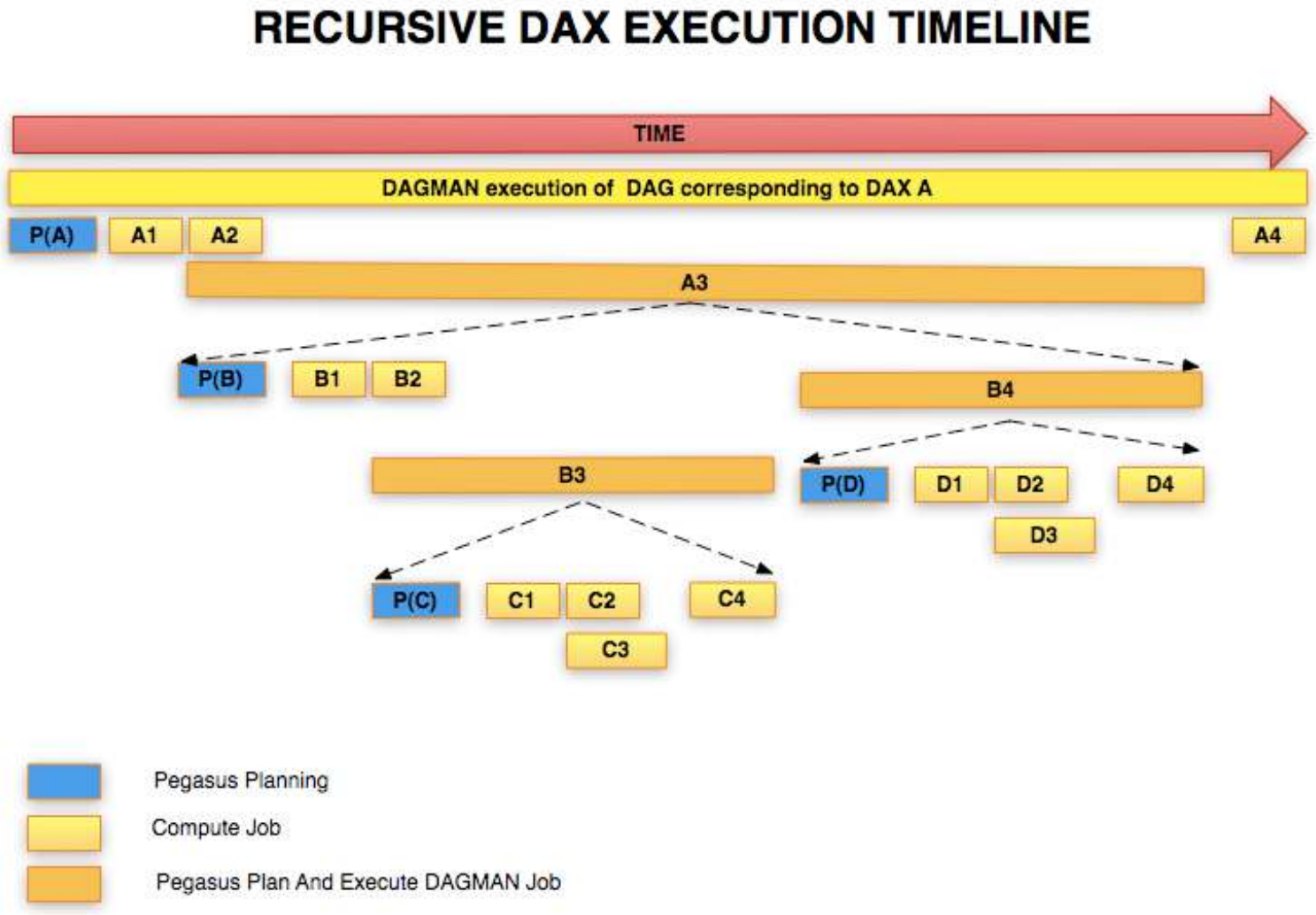


Pegasus Features Used: Hierarchal Workflows, Job Clustering , Cleanup

Hierarchal Workflows – Scaling upto million node workflows

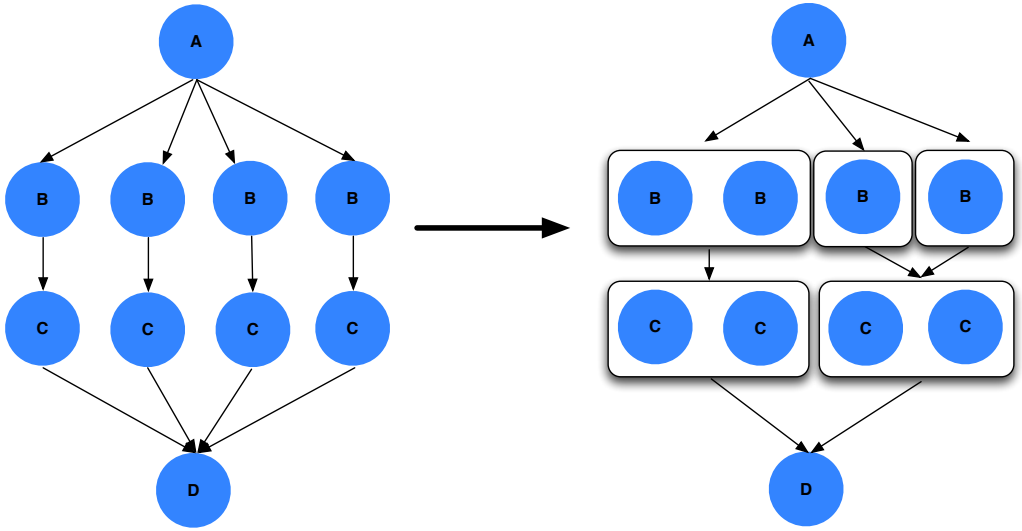


Hierarchal Workflows - Scaling upto million node workflows



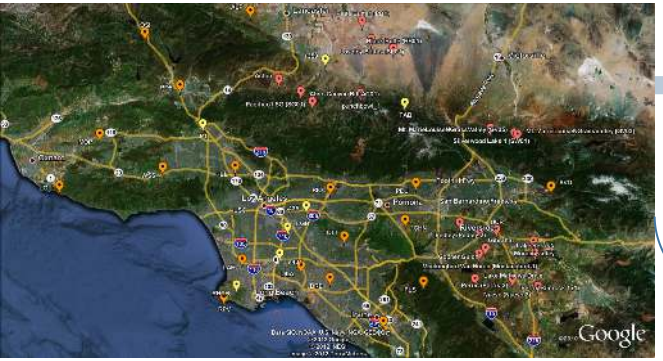
Workflow Restructuring to improve application performance

- **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 job on the grid that takes at least 10/30/60/? minutes to execute
  - Clustered tasks can reuse common input data – less data transfers

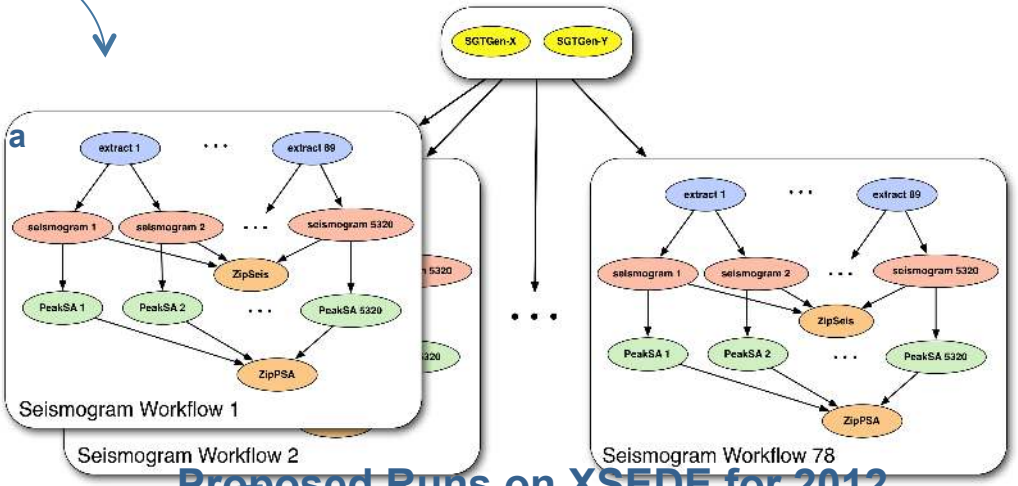


Level-based clustering

SCEC CYBERSHAKE WORKFLOWS

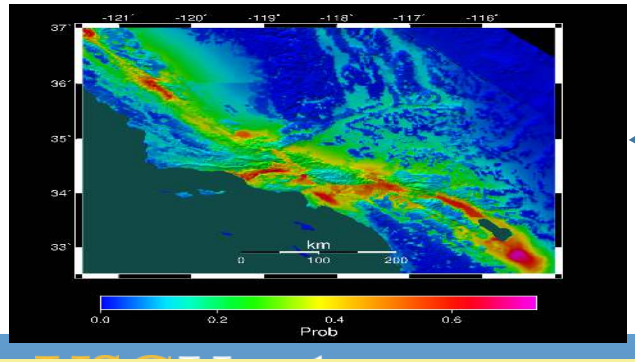


- **Description**
  - Builders ask seismologists: "What will the peak ground motion be at my new building in the next 50 years?"
  - Seismologists answer this question using Probabilistic Seismic Hazard Analysis (PSHA)



Post Processing Workflows

- **For each site in the input map, generate a hazard curve**
- **Each per site workflow has**
  - **820,000 tasks in the workflow**
  - **Input Strain Green Tensor 40 GB**
  - **Outputs about 10GB per site**
  - **CPU Time used : 38 days, 23 hrs**



Proposed Runs on XSEDE for 2012

- 3 Hazard maps each covering 200 sites
- To be run mainly on **Kraken** using MPI (PMC)
- Inputs SGT : approx **15.6 TB** (40 * 400 GB)
- Outputs: **500 million files** (820000/site x 600 sites) approx **5.8 TB** (600 * 10 GB)
- Number of Output Files : **= about 500 million**

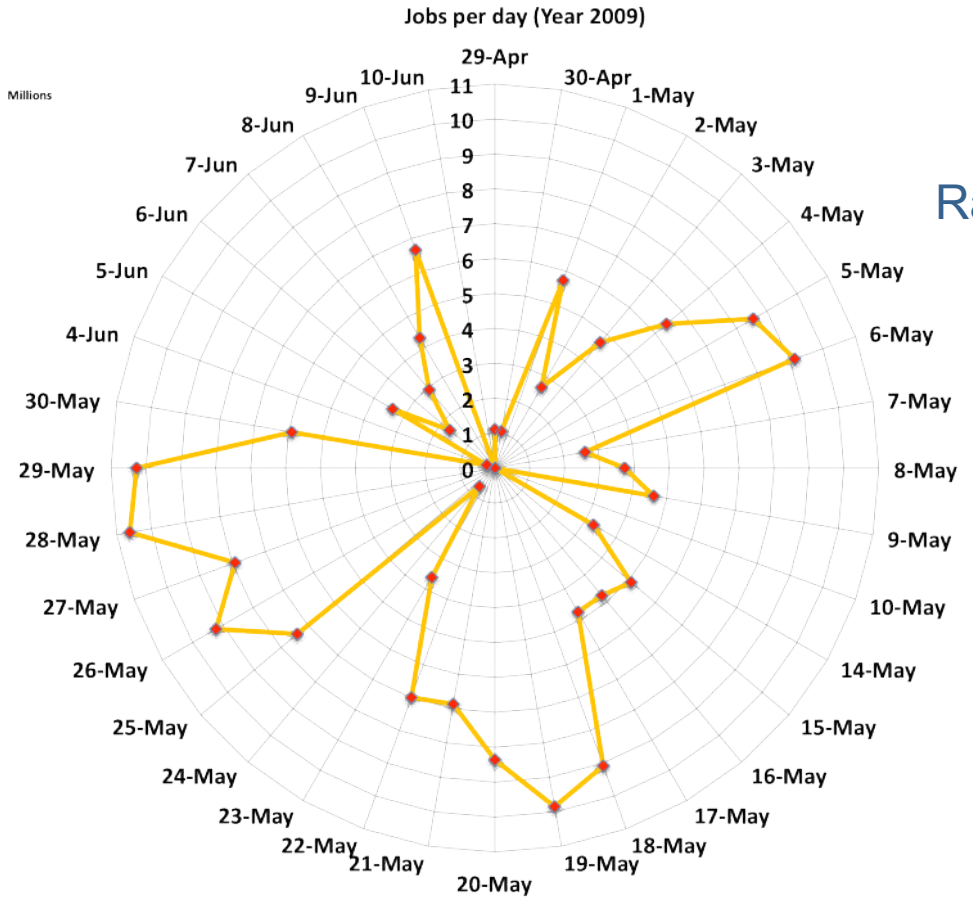
Pegasus Features Used: Hierarchal Workflows, Job Clustering , Cleanup, Symlinking against existing datasets

Outline of Talk

- Introduction to Scientific Workflows and Pegasus
- Data Management in Pegasus
- Workflow Monitoring and Debugging

Final Piece in the Puzzle – Tracking Workflows

SCEC-2009: Millions of tasks completed per day



Radius = 11 million

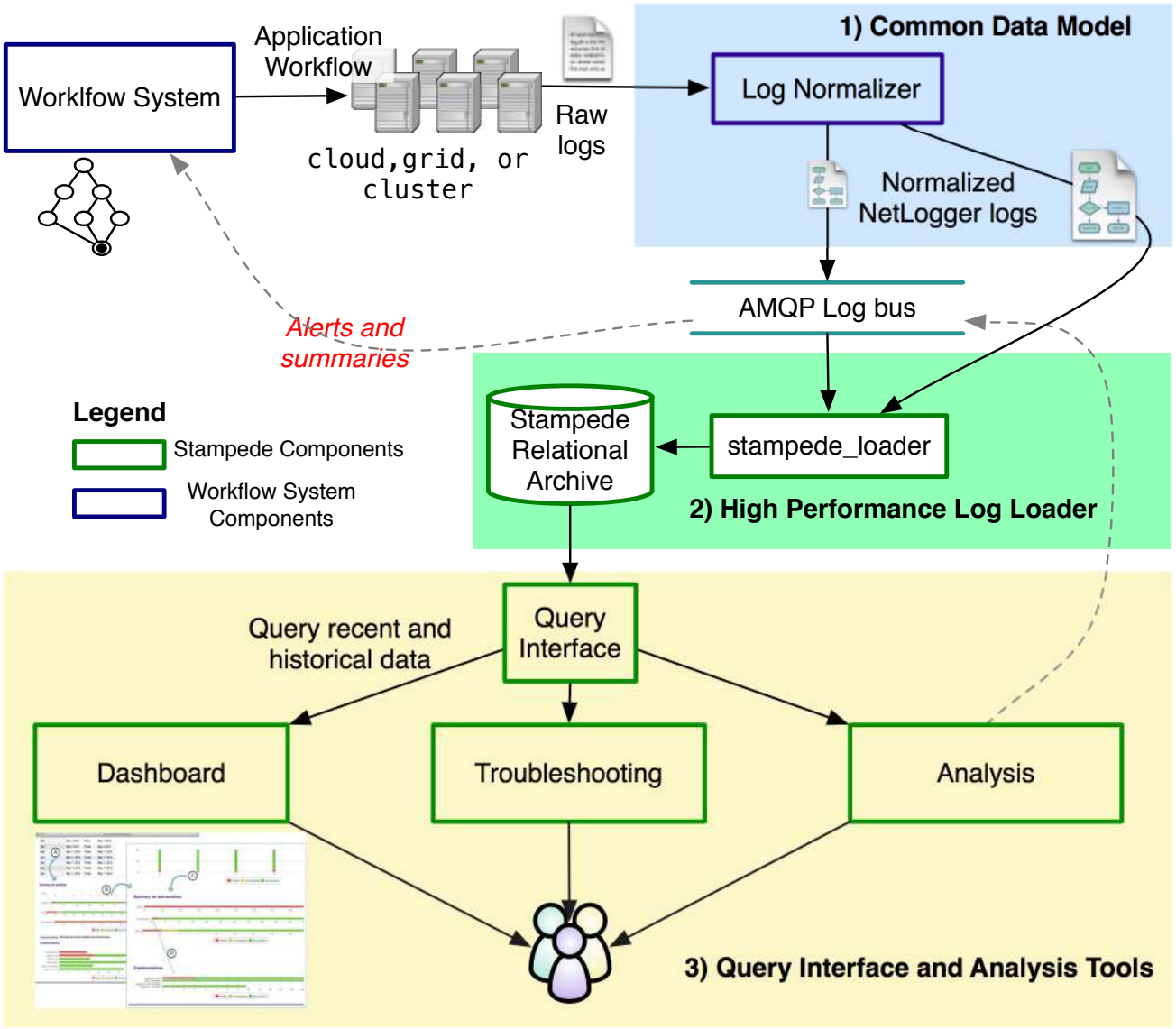
Final Piece in the Puzzle – Tracking Workflows

- **Pegasus can be used to run large workflows.**
- **Does the workflow system provide insight to the workflow runs**
  - Monitor the workflows
  - Debug their workflows when things go wrong
    - Imagine going through millions of job log files!
  - Generate statistics about your workflow run to determine resources consumed.
  - Notifications when things go wrong?

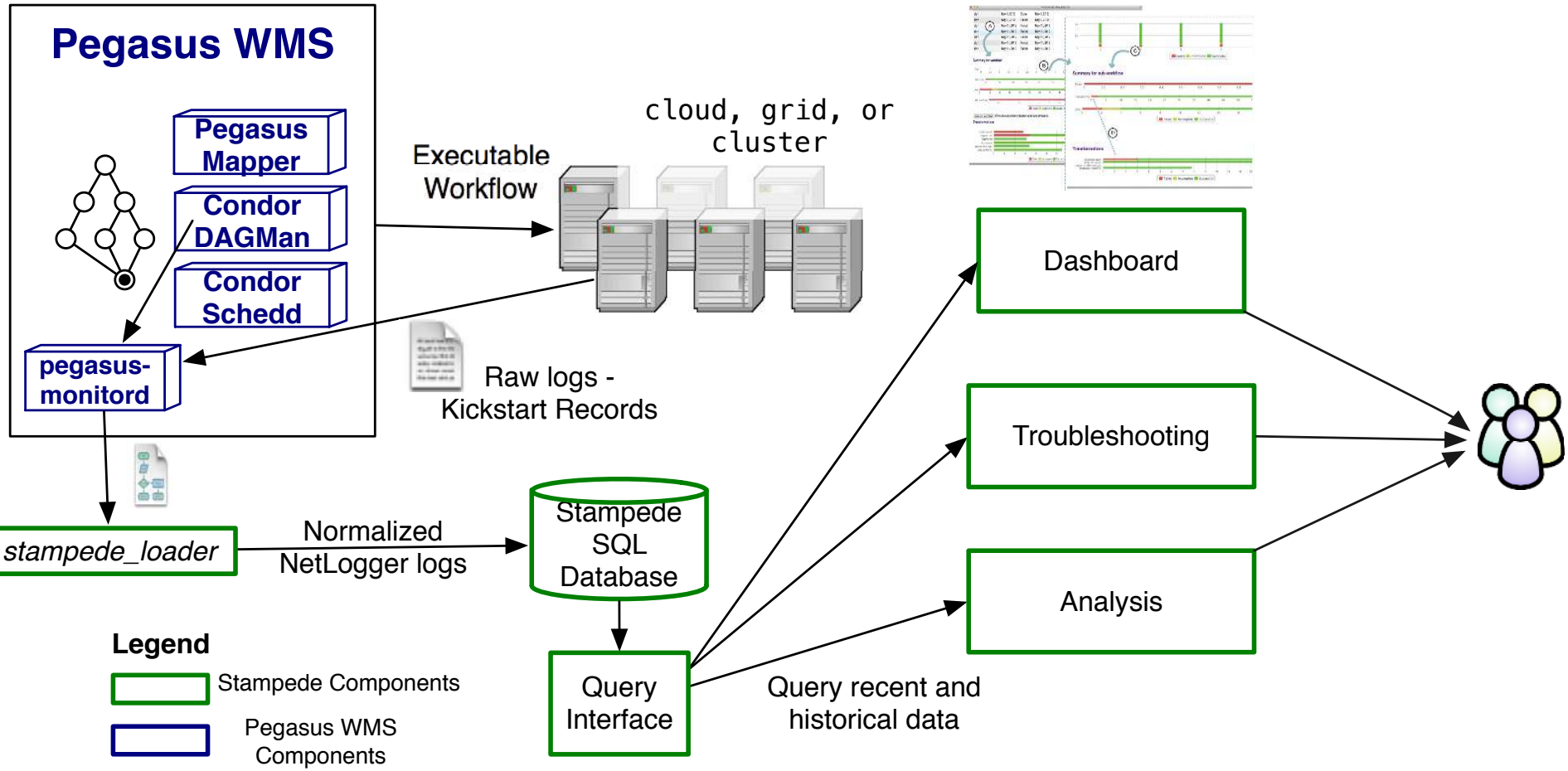
Goal: Real-time Monitoring and Analysis

1. Monitor Workflows in real time
   - Scientific workflows can involve many sub-workflows and millions of individual tasks
   - Need to correlate across workflow and job logs
   - Imagine going through hundred of thousands of log files!
   - Provide realtime updates on the workflow – how many jobs completed, failed etc
2. Troubleshoot Workflows
   - Provide users with tools to debug workflows, and provide information of why a job failed
3. Visualize Workflow performance and mine performance data
   - Provide a workflow monitoring dashboard that shows the various workflows run
   - Provide statistics about your workflow run.
4. Does the system provide notifications when things go wrong?
5. **Do all of this as generally as possible: Can we provide a solution that can apply to all workflow systems?**

How Does Stampede Provide Interoperability



Pegasus Integration with Stampede



Workflow Monitoring - Stampede

- **Leverage Stampede Monitoring framework with DB backend**
  - Populates data at runtime. A background daemon monitors the logs files and populates information about the workflow to a database
  - Stores workflow structure, and runtime stats for each task.

- **Tools for querying the monitoring framework**
  - **pegasus-status**
    - Status of the workflow
  - **pegasus-statistics**
    - Detailed statistics about your finished workflow

```
------------------------------------------------------------------------------
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)
```


Workflow Debugging Through Pegasus

- **After a workflow has completed, we can run pegasus-analyzer to analyze the workflow and provide a summary of the run**

- **pegasus-analyzer's output contains**
  - **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.

Alleviates the need for searching through large DAGMan and Condor logs!

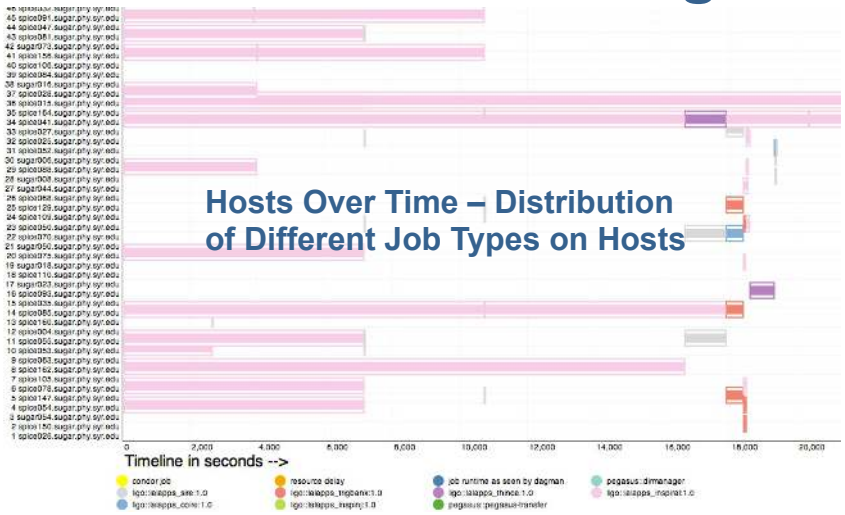
Workflow Monitoring Dashboard: pegasus-dashboard

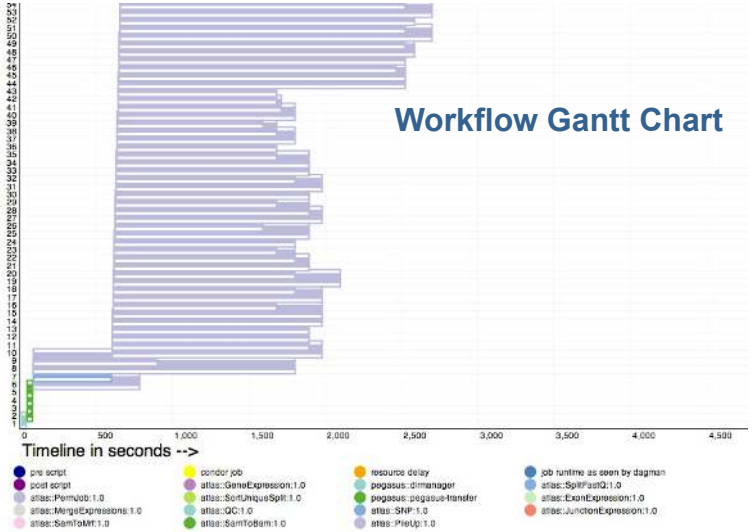
- **A python based online workflow dashboard**
  - Uses the FLASK framework
  - Beta version released in 4.2
  - Queries the STAMPEDE database
- **Lists all the user workflows on the home page and are color coded.**
  - Green indicates a successful workflow,
  - Red indicates a failed workflow
  - Blue indicates a running workflow
- **Explore Workflow and Troubleshoot (Workflow Page)**
  - Has identifying metadata about the workflow
  - Tabbed interface to
    - List of sub workflows
    - Failed jobs
    - Running jobs
    - Successful jobs.

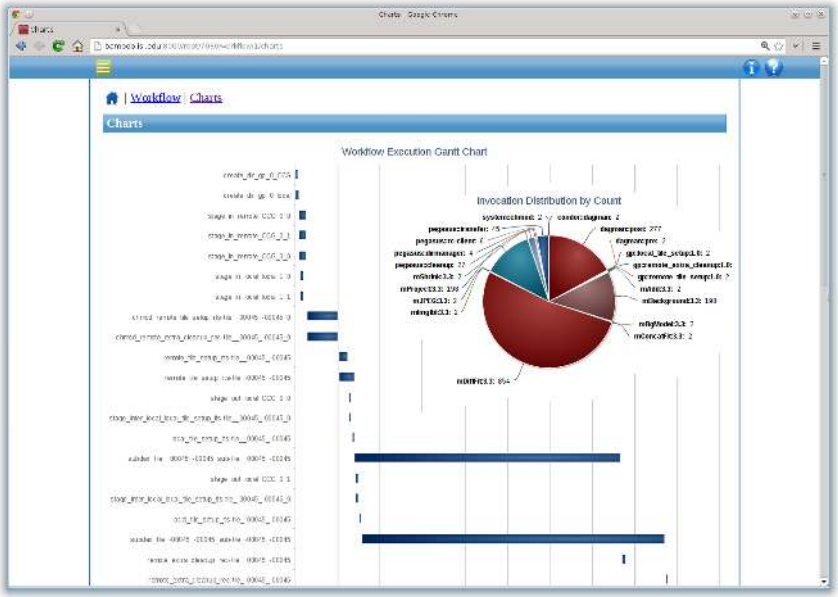
Workflow Monitoring Dashboard: pegasus-dashboard

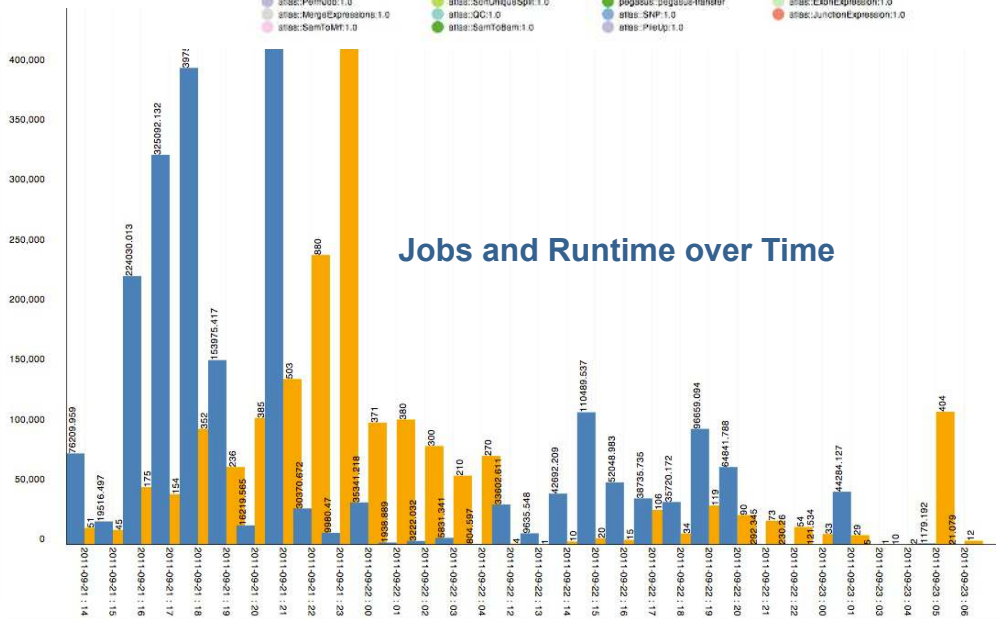
- **Job Page**
  - Lists information captured in kickstart record for the job.
  - Will show the various retries of the job
- **Statistics Page for the Workflow**
  - Generates Statistics for the workflow, similar to pegasus-statistics command line tool
- **Charts Page For the Workflow**
  - Workflow Gantt Chart
  - Job Distribution by Count/Time
  - Time Chart by Job/Invocation

Workflow Monitoring Dashboard – pegasus-dashboard









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
  - Provided with email and jabber notification scripts
  - Can run any user provided scripts
  - Defined in the DAX

Summary – What Does Pegasus provide an Application - I

- **All the great features that DAGMan has**
  - Scalability / hierarchal workflows
  - Retries in case of failure.
- **Portability / Reuse**
  - User created workflows can easily be mapped to and run in different environments without alteration.
- **Performance**
  - The Pegasus mapper can reorder, group, and prioritize tasks in order to increase the overall workflow performance.

Summary – What Does Pegasus provide an Application - II

- **Provenance**
  - Provenance data is collected in a database, and the data can be summaries with tools such as pegasus-statistics, pegasus-plots, or directly with SQL queries.

- **Reliability and Debugging Tools**
  - Jobs and data transfers are automatically retried in case of failures. Debugging tools such as pegasus-analyzer helps the user to debug the workflow in case of non-recoverable failures.

- **Data Management**
  - Pegasus handles replica selection, data transfers and output registrations in data catalogs. These tasks are added to a workflow as auxiliary jobs by the Pegasus planner.

Relevant Links

- **Pegasus: http://pegasus.isi.edu**
- **Tutorial and documentation: http://pegasus.isi.edu/wms/docs/latest/**
- **Support: pegasus-users@isi.edu**
  pegasus-support@isi.edu

Acknowledgements

Pegasus Team – Ewa Deelman, Gideon Juve, Rajiv Mayani, Mats Rynge