pegasus

Pegasus - A system to run, manage and debug complex workflows on top of Condor

Karan Vahi (vahi@isi.edu)

Collaborative Computing Group
USC Information Sciences Institute

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

Why Scientific Workflows?

- Workflows can be portable across platforms and scalable
- Workflows are easy to reuse, support reproducibility
- Can be shared with others
  - Gives a leg-up to new staff, GRAs, PostDocs, etc
- Workflow Management Systems (WMS) can help recover from failures and optimize overall application performance
- WMS can capture provenance and performance information
- WMS can provide debugging and monitoring tools

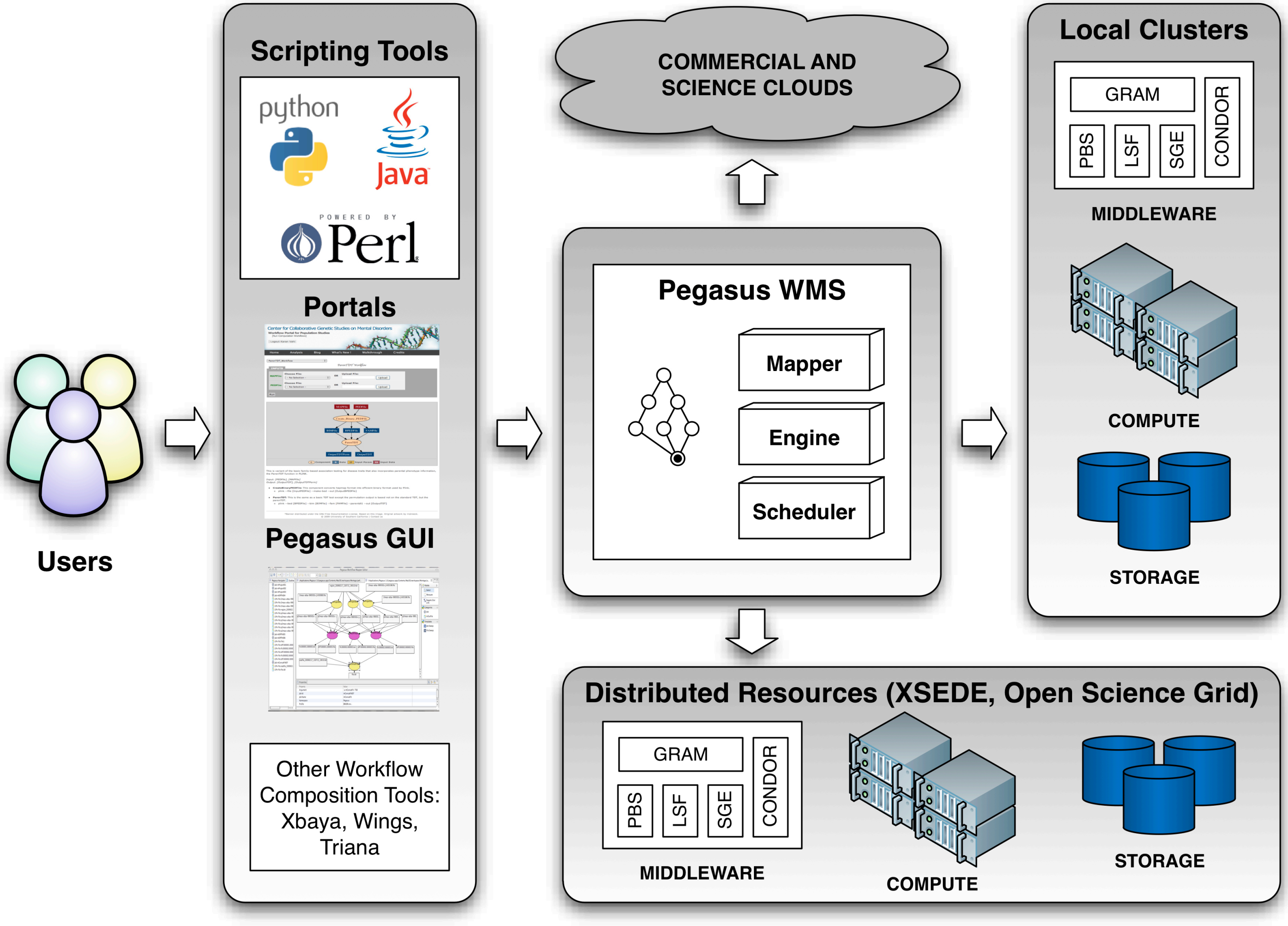
Pegasus Workflow Management System

- Takes in a workflow description and can map and execute it on wide variety of environments
  - Local desktop
  - Local Condor Pool
  - Local Campus Cluster
  - Grid
  - Commercial or Academic Clouds

Pegasus
Workflow Management System

- Developed since 2001
- A collaboration between USC and the Condor Team at UW Madison (includes DAGMan)
- 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)
- Automatically captures provenance information
- Provides workflow monitoring and debugging tools to allow users to debug large workflows

Pegasus WMS



Users

Scripting Tools

python

Java

POWERED BY Perl

Portals

Pegasus GUI

Other Workflow Composition Tools: Xbaya, Wings, Triana

COMMERCIAL AND SCIENCE CLOUDS

Pegasus WMS

Mapper

Engine

Scheduler

Local Clusters

GRAM

PBS

LSF

SGE

CONDOR

MIDDLEWARE

COMPUTE

STORAGE

Distributed Resources (XSEDE, Open Science Grid)

GRAM

PBS

LSF

SGE

CONDOR

MIDDLEWARE

COMPUTE

STORAGE

Abstract Workflow (DAX)

- Pegasus workflow description—DAX
  - workflow “high-level language”
  - devoid of resource descriptions
  - devoid of data locations
  - refers to codes as logical transformations
  - refers to data as logical files

- You can use Java, Perl, Python APIs to generate DAXes

DAX: http://pegasus.isi.edu/wms/docs/4.0/creating_workflows.php#abstract_workflows

Understanding DAX

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- Section 1: Files - Acts as a Replica Catalog (can be empty) →
  <file name="f.a">
    <pfn url="file:///scratch/tutorial/inputdata/diamond/f.a" site="local"/>
  </file>

<!-- Section 2: Executables - Acts as a Transformaton Catalog (can be empty) →
  <executable namespace="pegasus" name="preprocess" version="4.0" installed="true" arch="x86"
os="linux">
    <pfn url="file:///opt/pegasus/default/bin/keg" site="local"/>
  </executable>
…
<!-- Section 4: Job's, DAX's or Dag's - Defines a JOB or DAX or DAG (Atleast 1 required) -->

  <job id="j1" namespace="pegasus" name="preprocess" version="4.0">
    <argument>-a preprocess -T 60 -i  <file name="f.a"/> -o  <file name="f.b1"
/>   <file name="f.b2"/></argument>
    <uses name="f.a" link="input" transfer="true" register="true"/>
    <uses name="f.b1" link="output" transfer="false" register="false"/>
    <uses name="f.b2" link="output" transfer="false" register="false"/>
  </job>
….
<!-- Section 5: Dependencies - Parent Child relationships (can be empty) -->

  <child ref="j4">
    <parent ref="j2"/>
    <parent ref="j3"/>
  </child></adag>
```

(excerpted for display) x

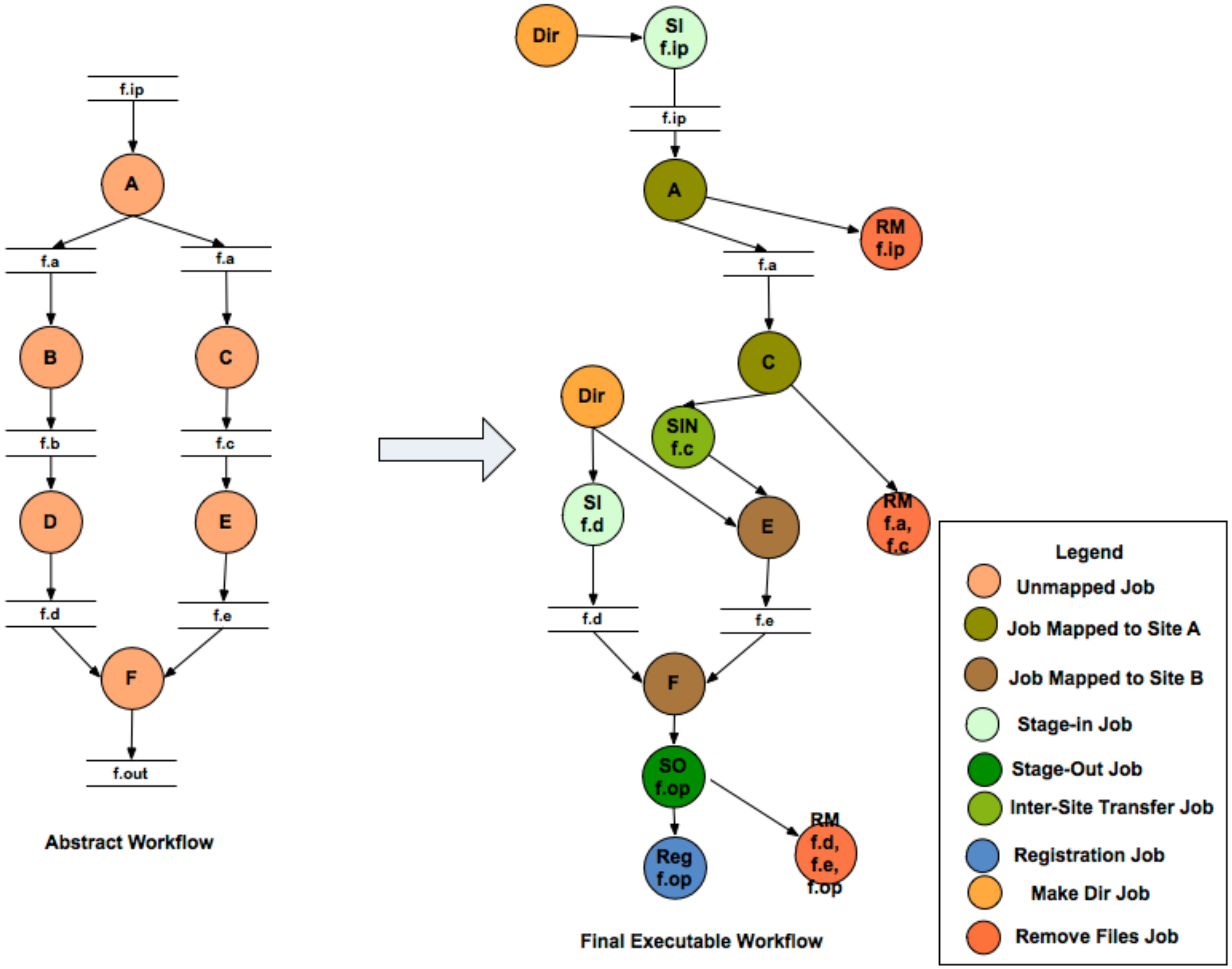
Basic Workflow Mapping

- Select where to run the computations
  - Change task nodes into nodes with executable descriptions
    - Execution location
    - Environment variables initializes
    - Appropriate command-line parameters set
- Select which data to access
  - Add stage-in nodes to move data to computations
  - Add stage-out nodes to transfer data out of remote sites to storage
  - Add data transfer nodes between computation nodes that execute on different resources

Basic Workflow Mapping

- ❖ Add nodes that register the newly-created data products
- ❖ Add nodes to create an execution directory on a remote site
- ❖ Write out the workflow in a form understandable by a workflow engine
    - ✧ Include provenance capture steps

Comparison of abstract and executable workflows



Why mapping?

- Many workflow systems support only executable workflow composition
- 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)

Discovery during the Mapping Process

- Data
  - Pegasus looks up a Replica Catalog to discover
    - input locations and track output locations.
- Executables
  - Pegasus looks up a Transformation catalog to discover
    - Where are the executables installed ?
    - Do binaries exist somewhere that can be staged to remote grid sites?
- Site Layout
  - Pegasus looks up a Site Catalog to discover
    - What does the execution environment look like?
    - Which servers to use for staging of data
    - What remote job submission interface to use

Simple Steps to Run Pegasus

1. Specify your computation in terms of DAX
   - Write a simple DAX generator
   - Java, Python and Perl based API provided with Pegasus
2. Set up your catalogs
   - Use *pegasus-sc-client* to generate site catalog and transformation catalog for your environment
   - Record the locations of your input files in a replica client using *pegasus-rc-client*
3. Plan and Submit your workflow
   - Use *pegasus-plan* to generate your executable workflow that is mapped onto the target resources and submits it for execution
4. Monitor and Analyze your workflow
   - Use *pegasus-status* | *pegasus-analyzer* to monitor the execution of your workflowMonitor and Analyze your workflow
5. Mine your workflow for statistics
   - Use pegasus-statistics

Workflow Monitoring - Stampede

- Enhanced Monitoring framework with DB backend
  - Supports SQLite or MySQL
  - Python API to query the framework
  - 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 workflow
  - pegasus-plots
    - Visualization of your workflow execution

Statistics: http://pegasus.isi.edu/wms/docs/4.0/monitoring_debugging_stats.php

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.

Workflow and Task Notifications

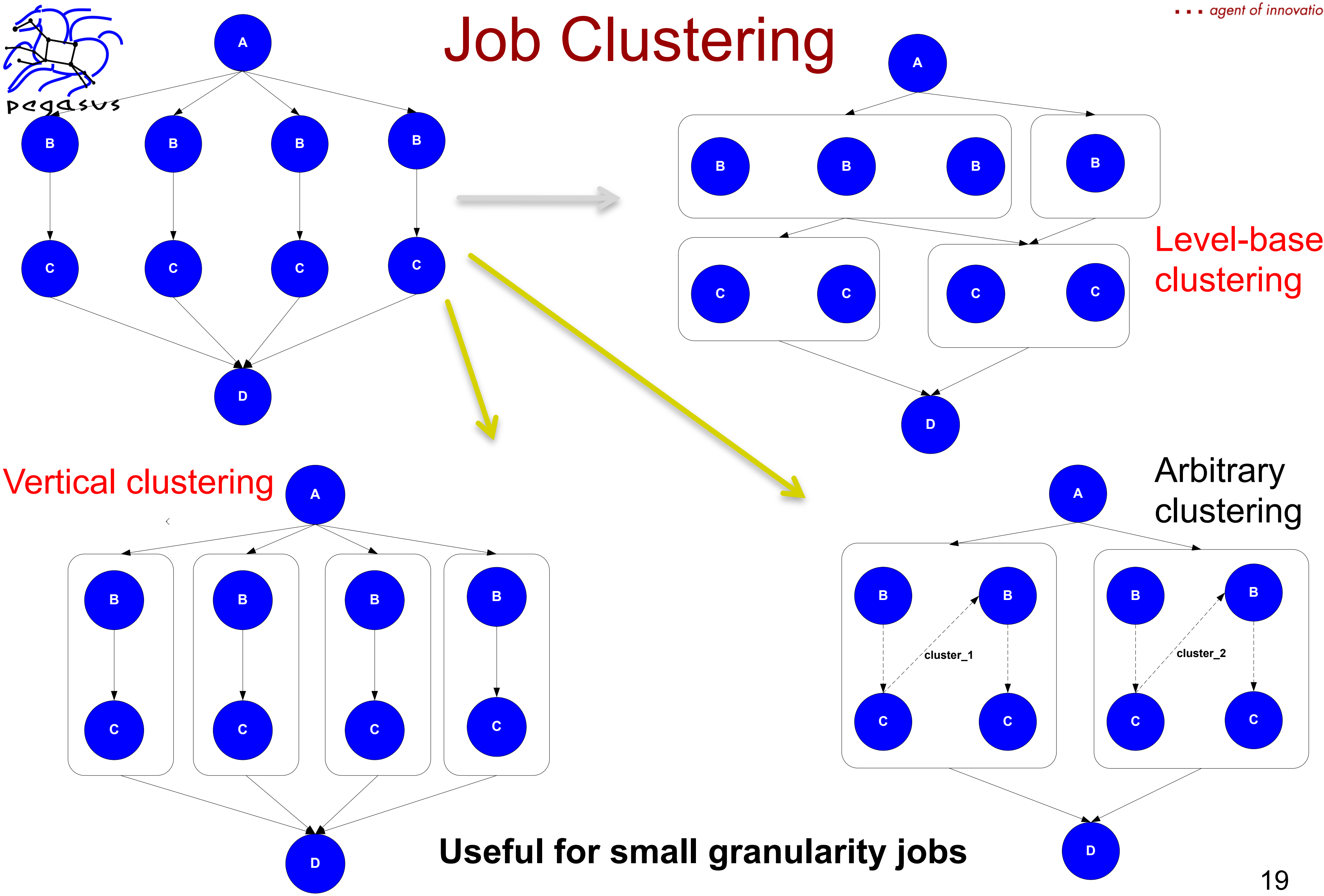
- Support for adding Notification to Workflow and Tasks
  - Event based callouts
    - On Start, On End, On Failure, On Success
  - Provided with email and jabber notification scripts
  - Can run any user provided script as notification.
  - Defined in the DAX.

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 minutes to execute

Job Clustering



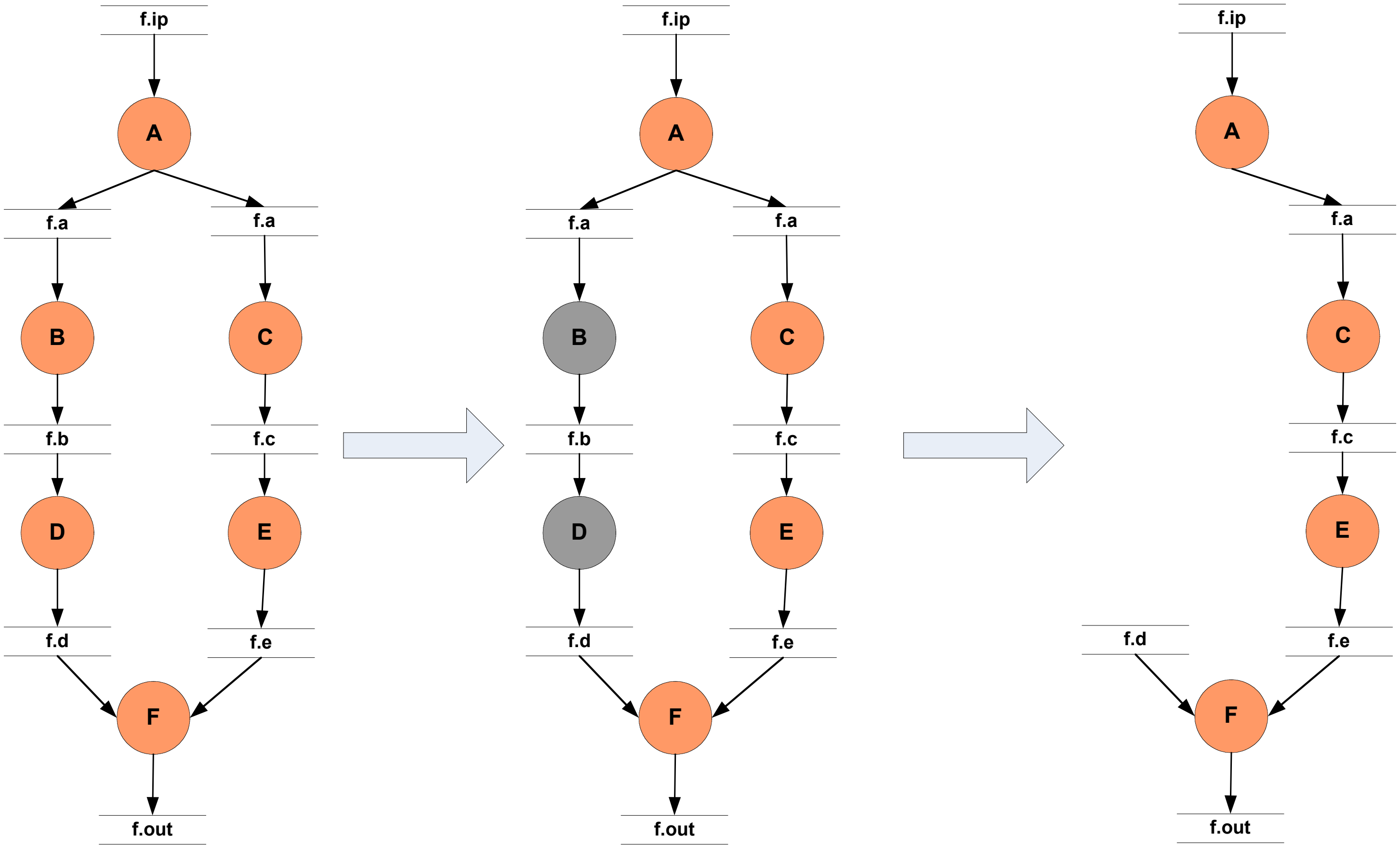
Level-based clustering

Vertical clustering

Arbitrary clustering

Useful for small granularity jobs

WF 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

How to: *Files need to be cataloged in replica catalog at runtime. The registration flags for these files need to be set in the DAX.*

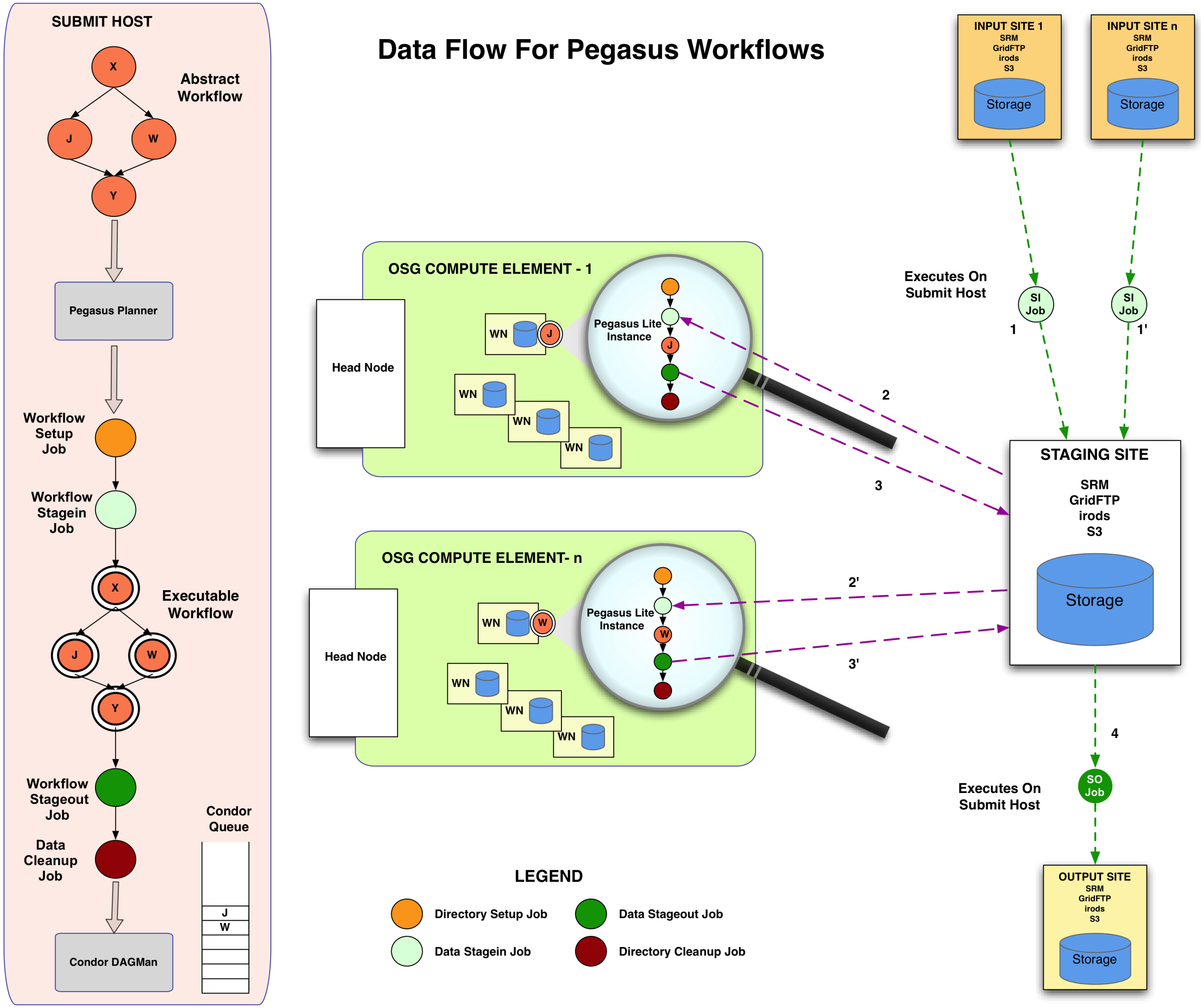
Transfer of Executables

- Allows the user to dynamically deploy scientific code on remote sites
- Makes for easier debugging of scientific code
- The executables are transferred as part of the workflow
- Currently, only statically compiled executables can be transferred
- Also we transfer any dependant executables that maybe required. In your workflow, the mDiffFit job is dependant on mDiff and mFitplane executables

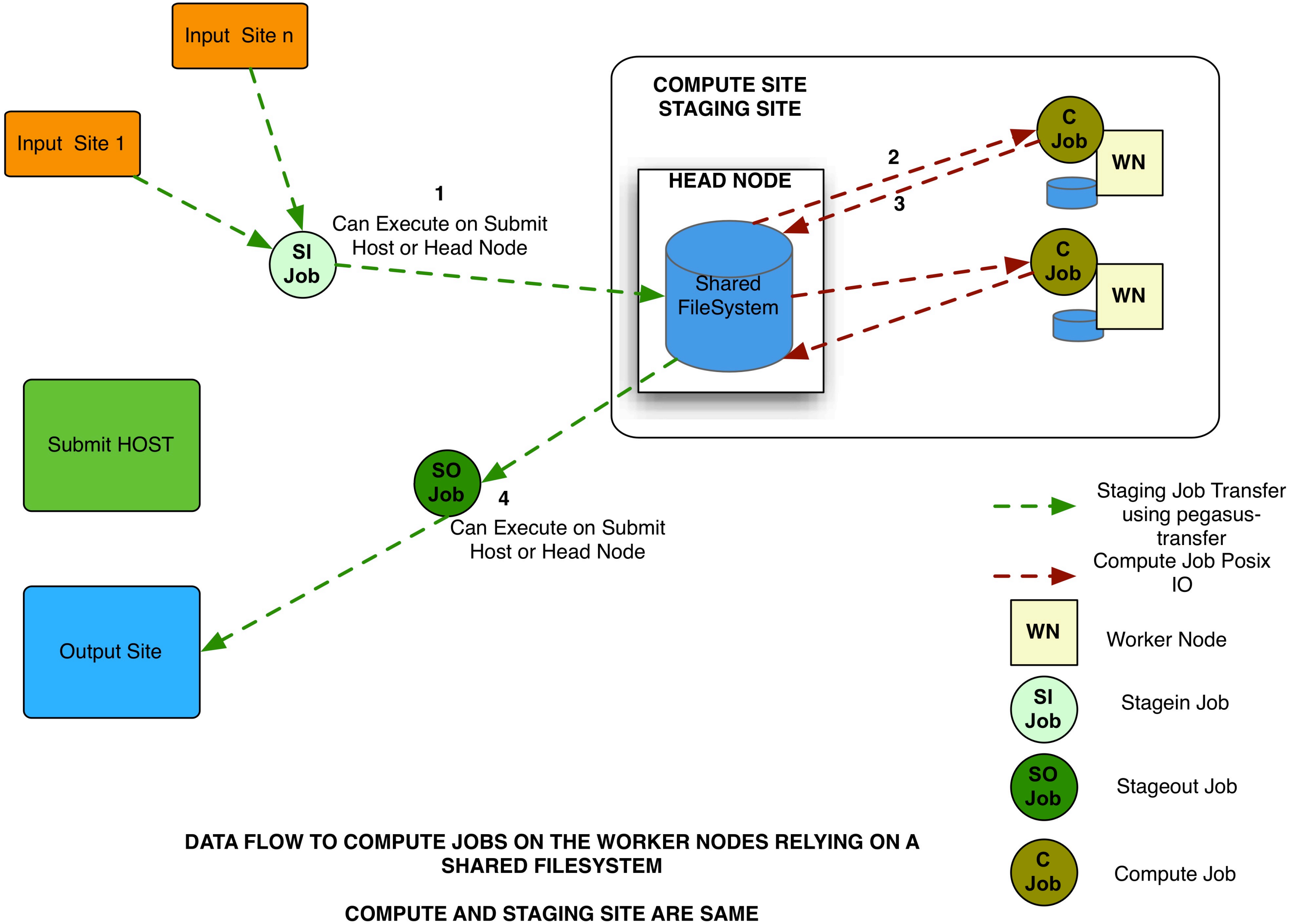
Supported Data Staging Configurations

- ❖ Three General Configurations Supported
  - ✧ Shared Filesystem setup
    - Worker nodes and the Head Node have a shared filesystem.
  - ✧ NonShared Filesystem setup with a staging site
    - Worker Nodes don’t share a filesystem.
    - Data is pulled from an external staging site.
  - ✧ Condor IO
    - Worker Nodes don’t share a filesystem
    - Data is pulled from the submit host.

Data Flow For Pegasus Workflows



Shared Filesystem Setup

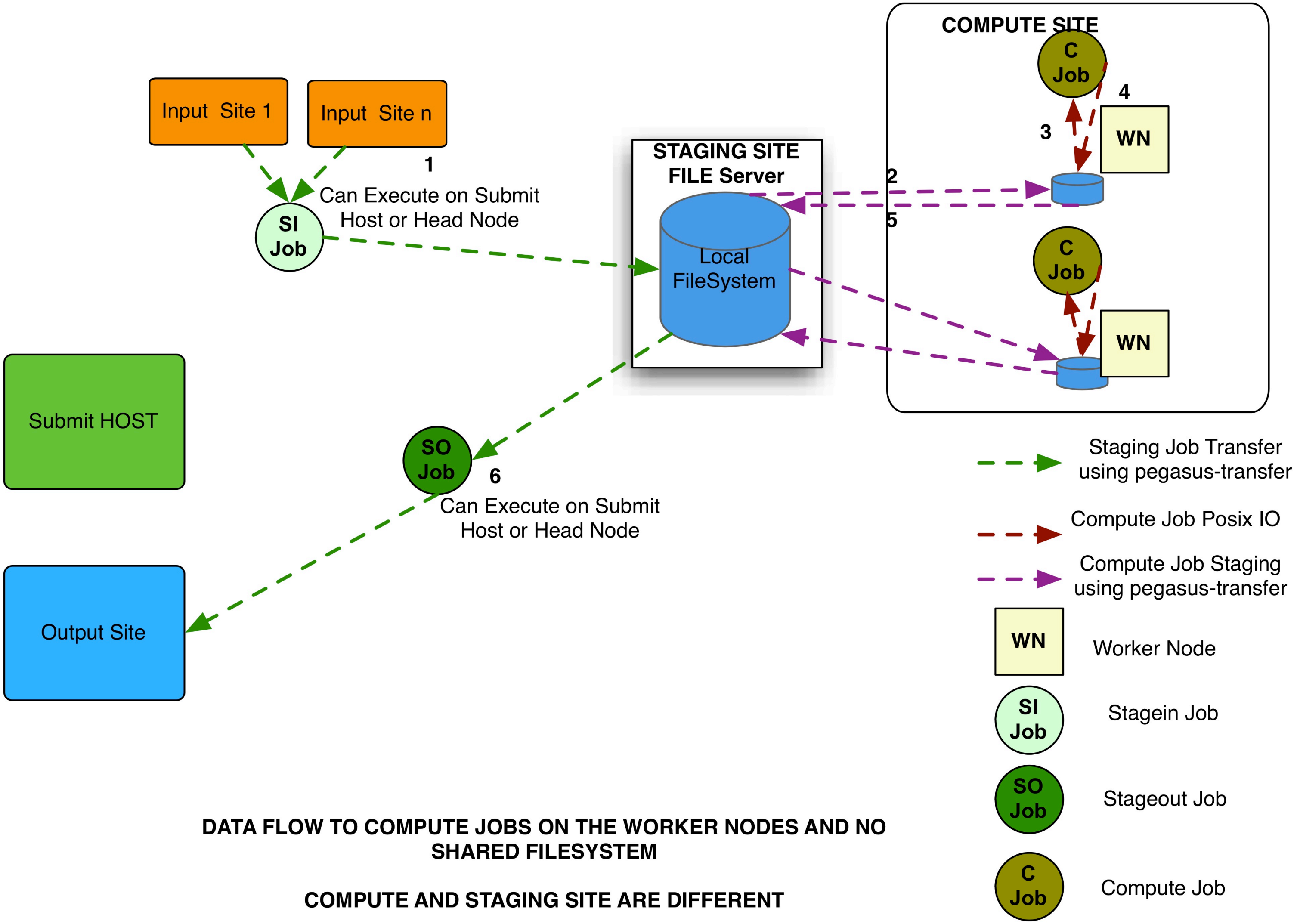


DATA FLOW TO COMPUTE JOBS ON THE WORKER NODES RELYING ON A SHARED FILESYSTEM

COMPUTE AND STAGING SITE ARE SAME

Tip: Set pegasus.data.configuration = sharedfs

Nonshared filesystem Setup

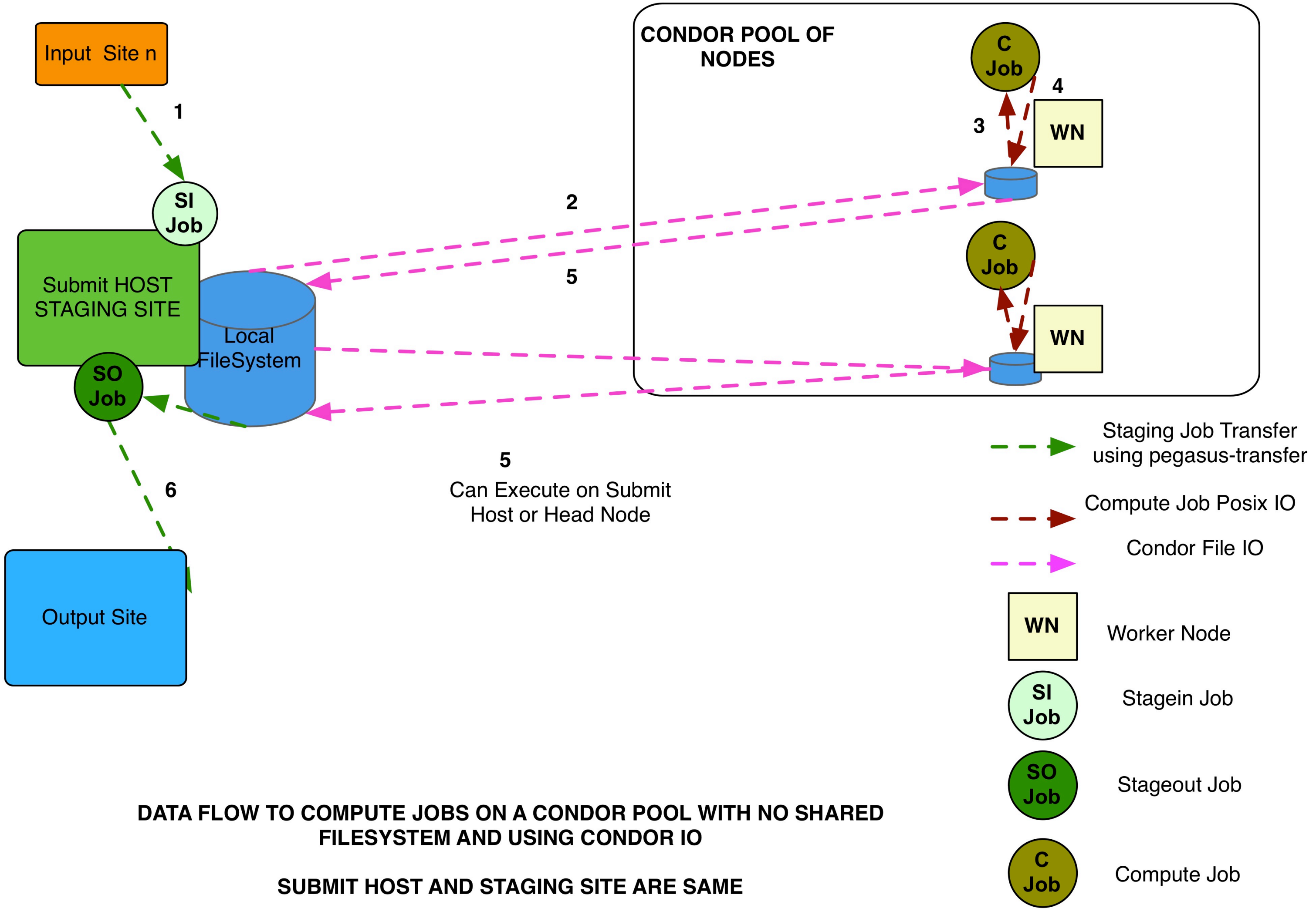


DATA FLOW TO COMPUTE JOBS ON THE WORKER NODES AND NO SHARED FILESYSTEM

COMPUTE AND STAGING SITE ARE DIFFERENT

Tip: Set pegasus.data.configuration = nonsharedfs

Condor IO

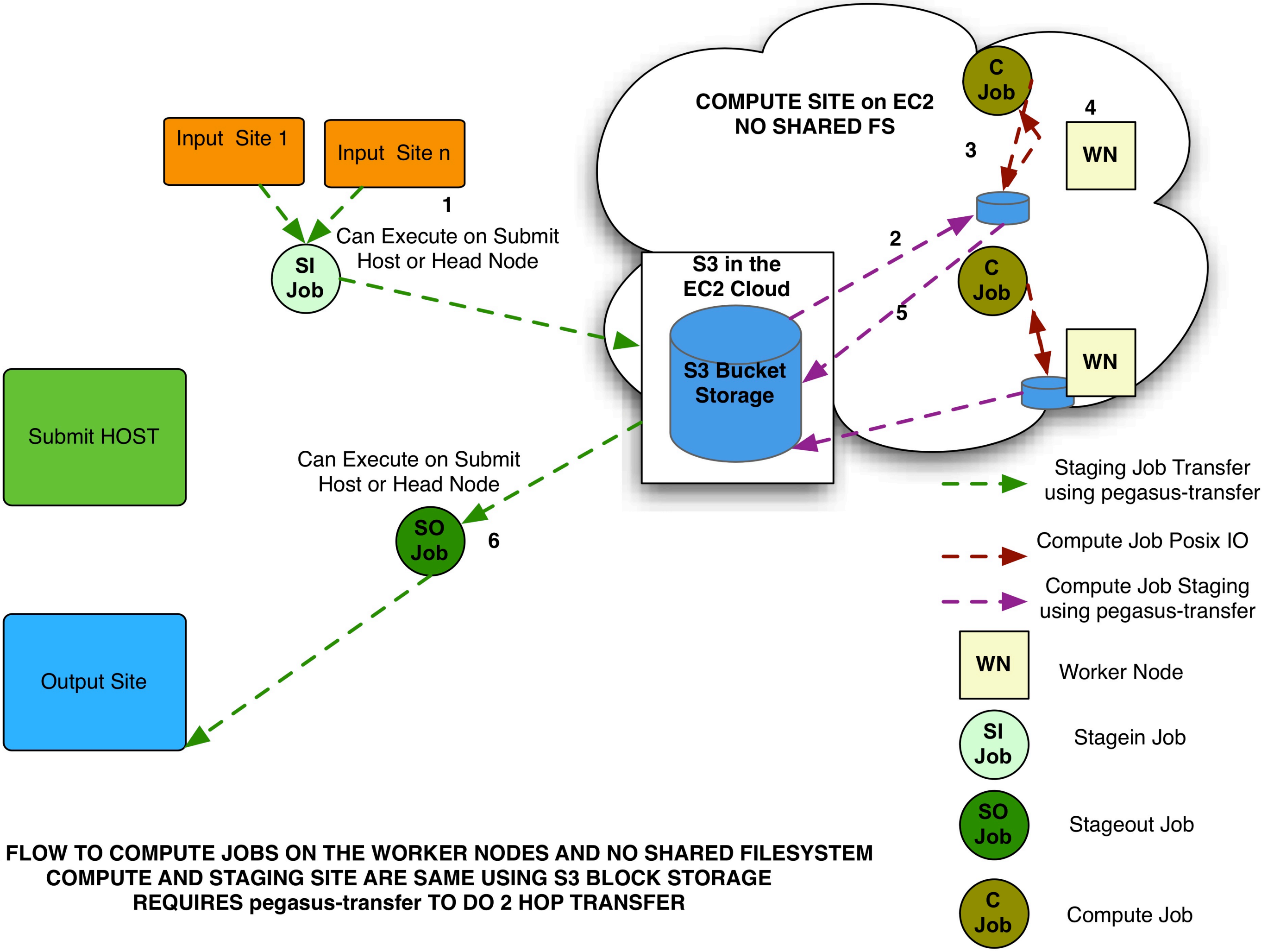


DATA FLOW TO COMPUTE JOBS ON A CONDOR POOL WITH NO SHARED FILESYSTEM AND USING CONDOR IO

SUBMIT HOST AND STAGING SITE ARE SAME

Tip: Set pegasus.data.configuration = condorio

NonShared Filesystem Setup in Cloud with S3 Storage



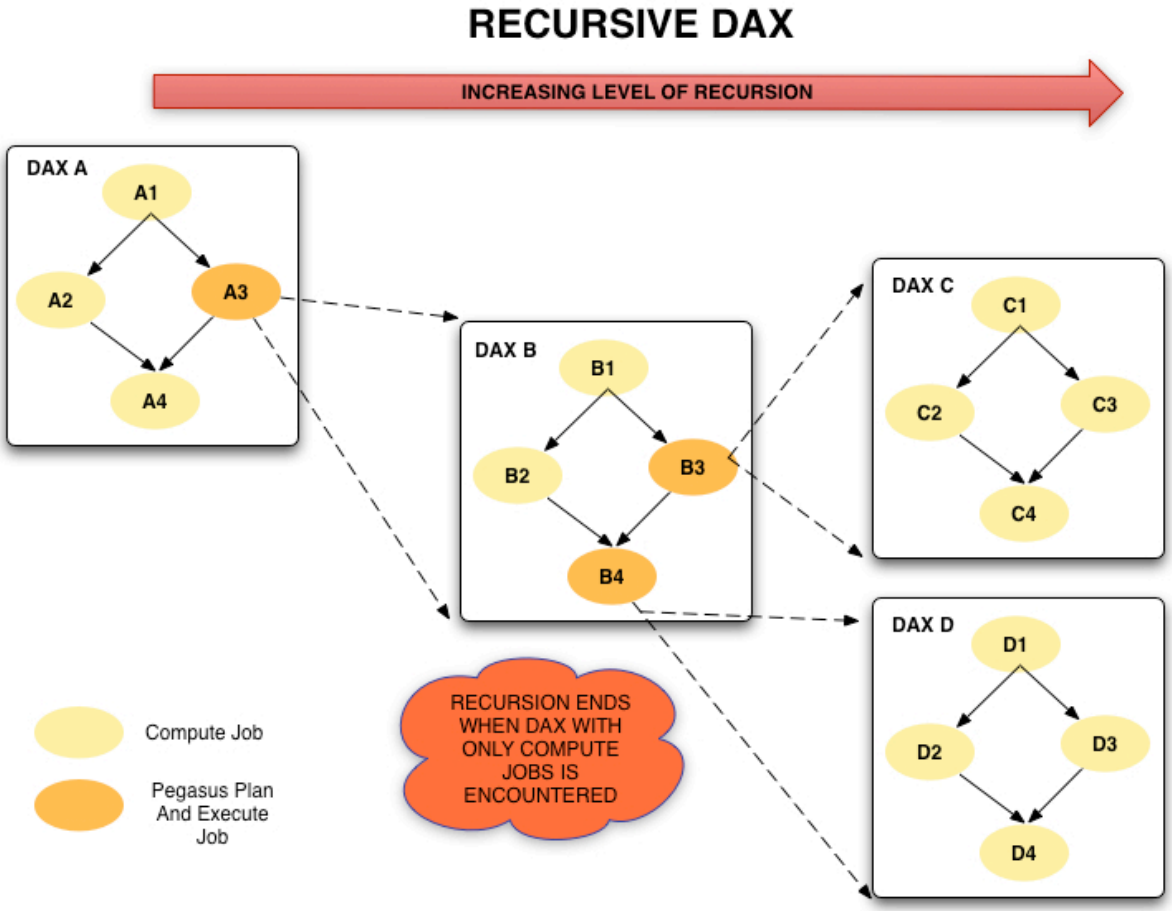
DATA FLOW TO COMPUTE JOBS ON THE WORKER NODES AND NO SHARED FILESYSTEM
COMPUTE AND STAGING SITE ARE SAME USING S3 BLOCK STORAGE
REQUIRES pegasus-transfer TO DO 2 HOP TRANSFER

Tip: Set pegasus.data.configuration = nonsharedfs with S3 as the staging site

Transfer Throttling

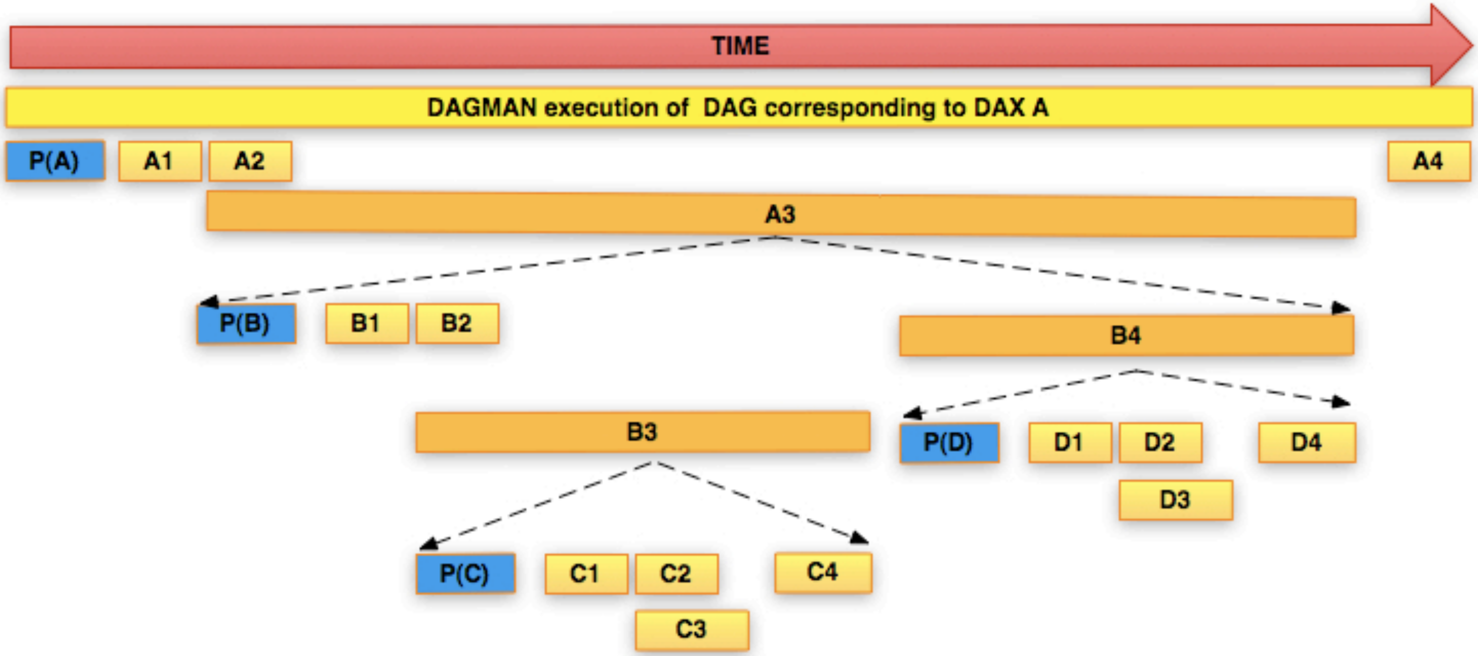
- Large Sized Workflows result in large number of transfer jobs being executed at once. Results in
  - Grid FTP server overload (connection refused errors etc)
  - May result in a high load on the head node if transfers are not configured for being executed as third party transfers
- Need to throttle transfers
  - Set pegasus.transfer.refiner property
  - Allows you to create chained transfer jobs or bundles of transfer jobs
  - Looks in your site catalog for pegasus profile "stagein.clusters"

Hierarchal Workflows


RECURSIVE DAX
INCREASING LEVEL OF RECURSION
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

Hierarchal Workflows

RECURSIVE DAX EXECUTION TIMELINE



TIME

DAGMAN execution of DAG corresponding to DAX A

P(A) A1 A2 A4

A3

P(B) B1 B2

B4

B3

P(D) D1 D2 D4

D3

P(C) C1 C2 C4

C3

🟦 Pegasus Planning

🟨 Compute Job

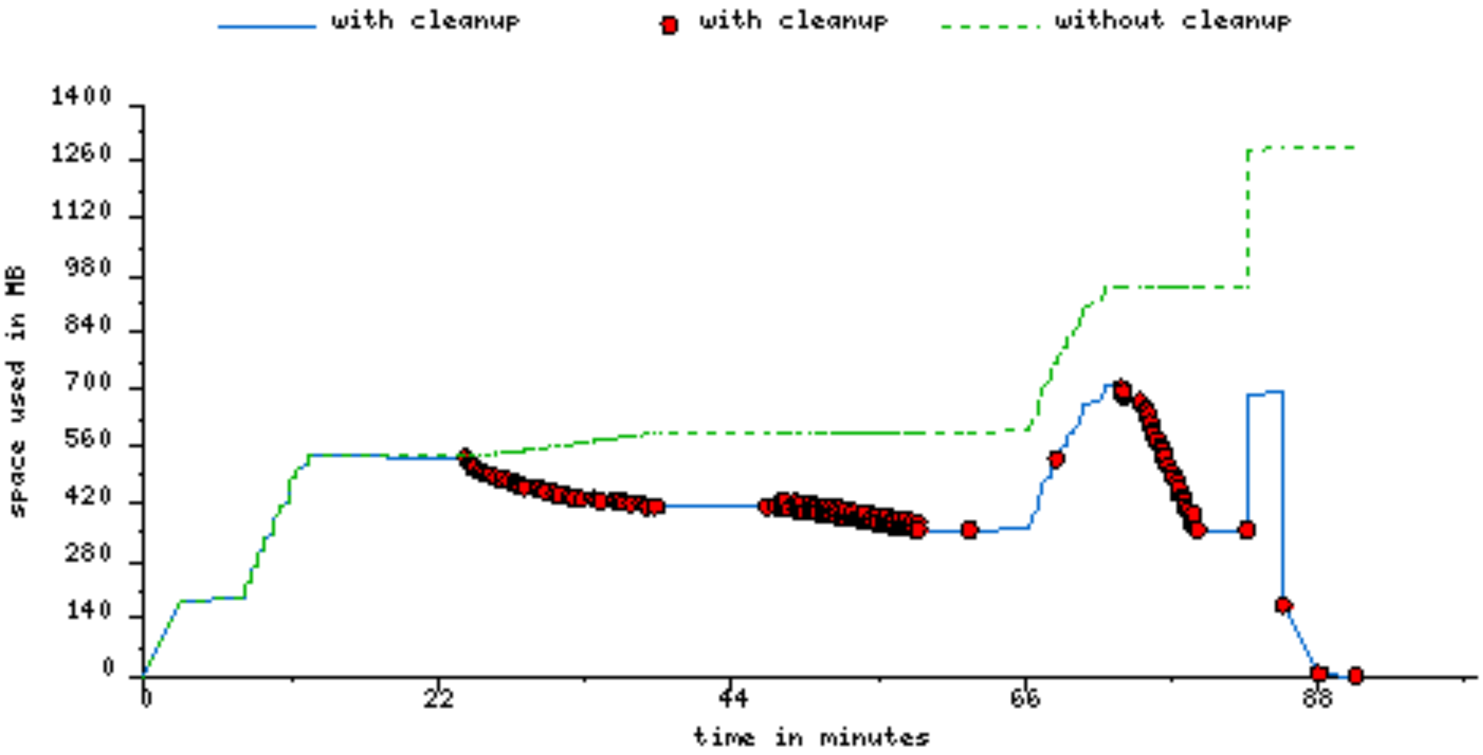
🟧 Pegasus Plan And Execute DAGMAN Job

File cleanup

- Problem: Running out of space on shared scratch
  - In OSG scratch space is limited to 30Gb for all users
- Why does it occur
  - Workflows 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

How to: *remove the –nocleanup option to the pegasus-plan*

Storage Improvement for Montage Workflows



Montage 1 degree workflow run with cleanup on OSG-PSU

Summary – What Does Pegasus provide an Application - I

- **Portability / Reuse**
  - User created workflows can easily be run in different environments without alteration.
- **Performance**
  - The Pegasus mapper can reorder, group, and prioritize tasks in order to increase the overall workflow performance.
- **Scalability**
  - Pegasus can easily scale both the size of the workflow, and the resources that the workflow is distributed over. Pegasus runs workflows ranging from just a few computational tasks up to 1 million.

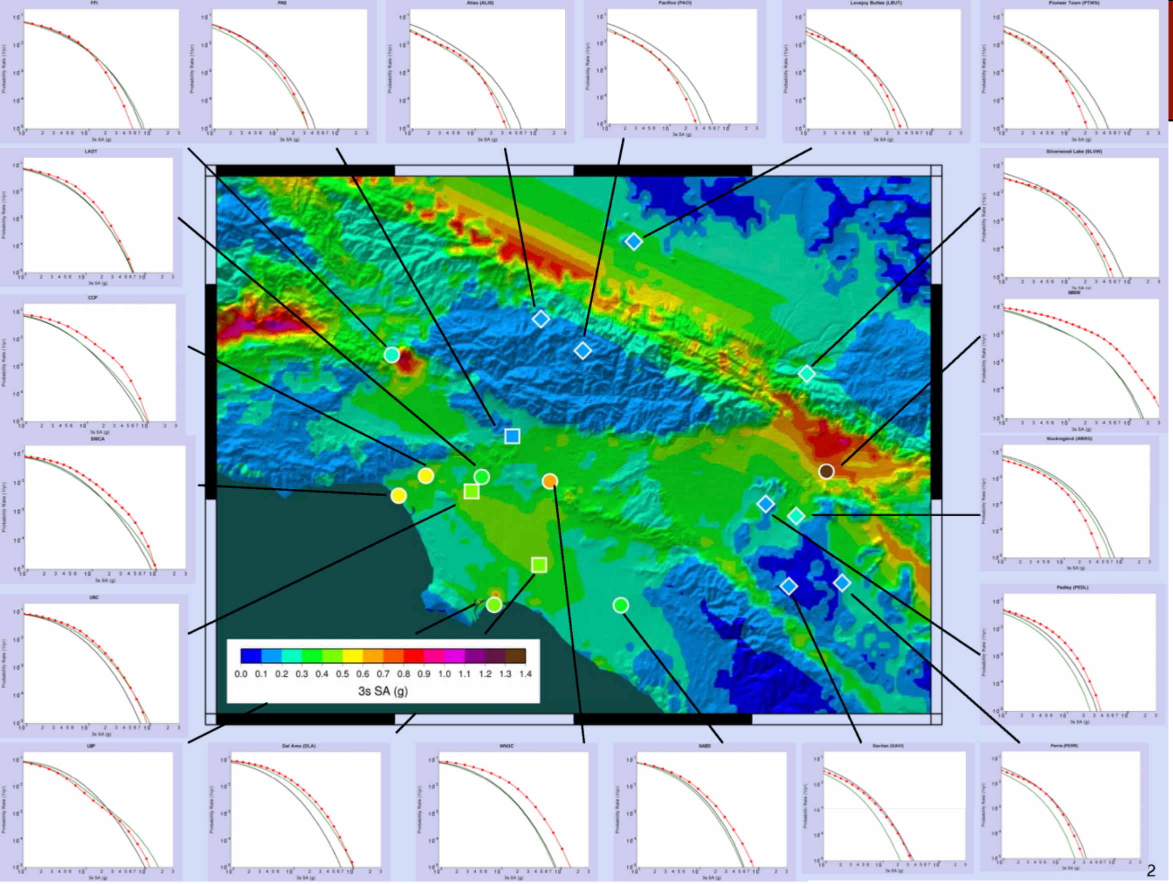
Summary –
What Does Pegasus provide an Application - II

- **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.
- **Data Management**
  - Pegasus handles replica selection, data transfers and output registrations in data catalogs. These tasks are added to a workflow as auxilliary jobs by the Pegasus planner.
- **Reliability**
  - 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.
- **Error Recovery**
  - Retries tasks in case of failures

Some Applications using Pegasus

- **Astronomy**
  - Montage , Galactic Plane, Periodograms
- **Bio Informatics**
  - Brain Span, RNA Seq, SIPHT, Epigenomics, Seqware
- **Earthquake Science**
  - Cybershake, Broadband from Southern California Earthquake Center
- **Physics**
  - LIGO

Complete Listing: *http://pegasus.isi.edu/applications*

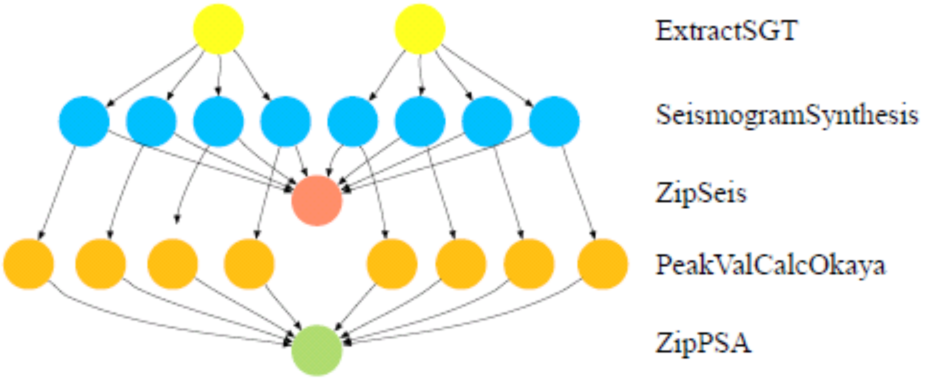


USC Southern California Earthquake Center

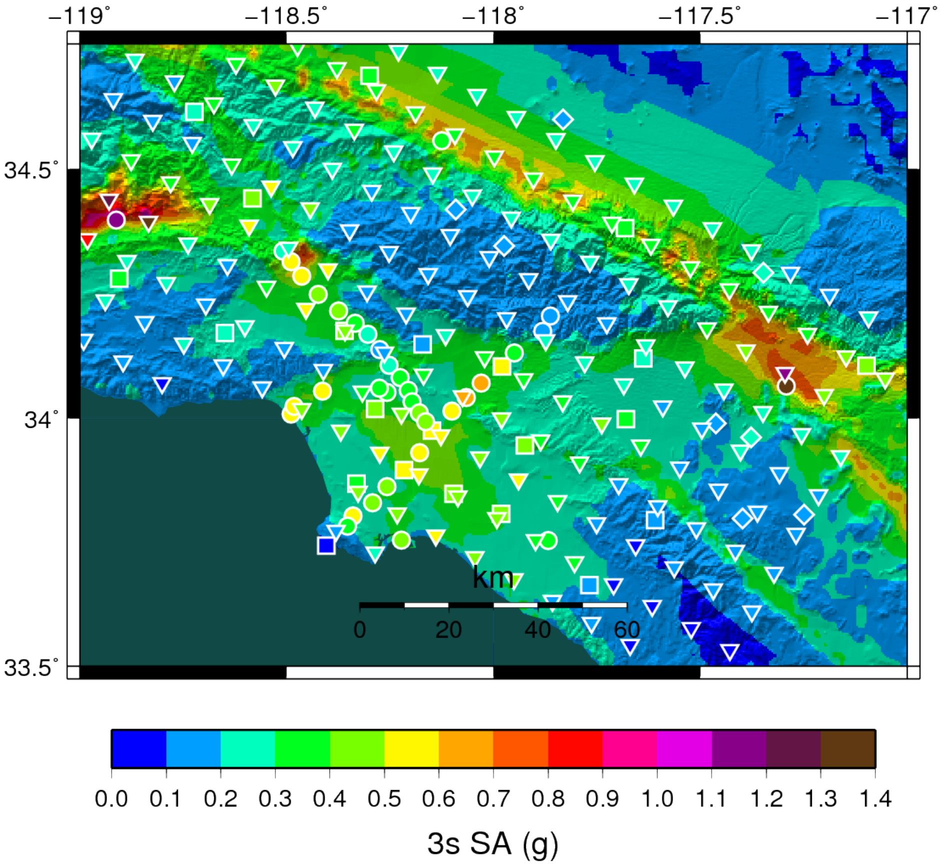
CyberShake Application

Breaks up large simulation data into smaller chunks for parallel processing

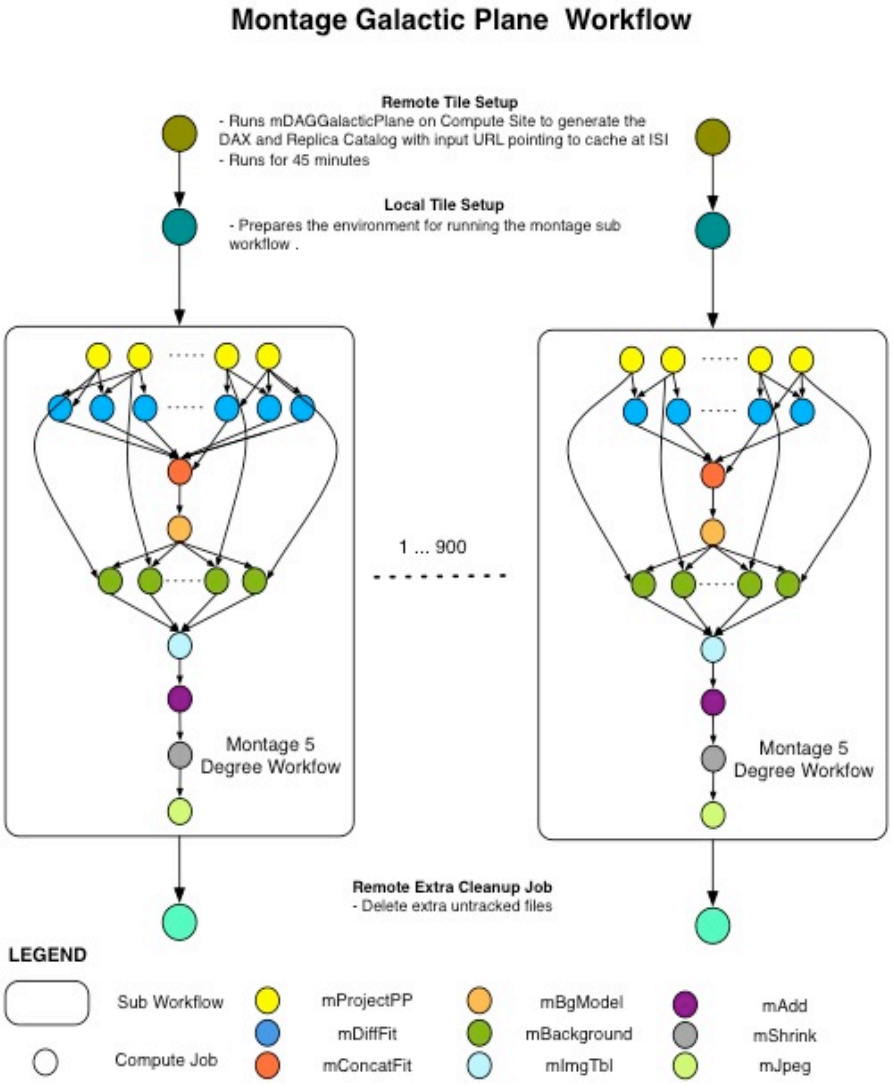
Each Curve/workflow
- ~800,000 jobs
- 18.3 ± 3.9 hours on 400 processors (December 2008)
- 90 GB of output
- 7.5 million data files



Large Scale Workflows through Pegasus - SCEC Cybershake (2009)



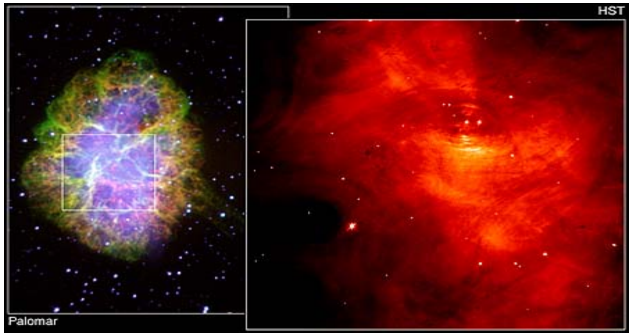
- TACC’s Ranger –Teragrid
- 223 sites
  - Curve produced every 5.4 hrs
- 1207 wallclock hrs
  - 4,420 cores on average
  - 14,540 peak (23% of Ranger)
- 189 million tasks
  - 43 tasks/sec
  - 3.8 million Condor jobs
    - 289 failures
  - 3952 Ranger queue jobs
- 189 million files
  - 11 TB output, 165 TB temp


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 Workflow
Montage 5 Degree Workflow
Remote Extra Cleanup Job
- Delete extra untracked files
LEGEND
Sub Workflow
Compute Job
mProjectPP
mDiffFit
mConcatFit
mBgModel
mBackground
mImgTbl
mAdd
mShrink
mJpeg

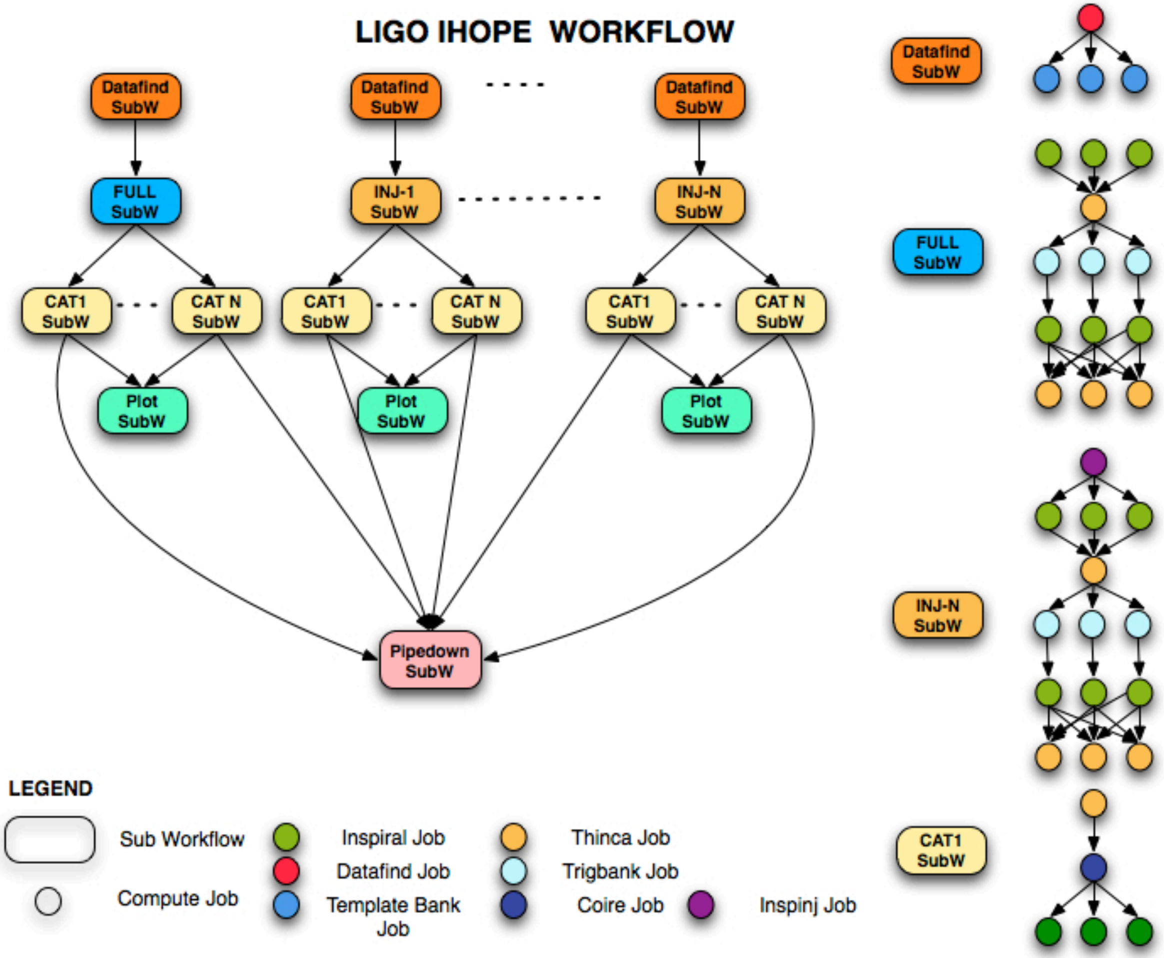
- **Description**
  - Galactic Plane for generating mosiacs from the Spitzer Telescope
  - Used to generate tiles 360 x 40 around the galactic equator
  - A tile 5 x 5 with 1 overlap with neighbors
  - Output datasets to be used in NASA Sky and Google Sky
  - One workflow run for each of 17 bands (wavelengths)
  - Each sub workflow uses **3.5TB** of input imagery (1.6 million files)
  - Each workflow consumes **30K CPU hours** and produces 900 tiles in FITS format
- **Proposed Runs on Xsede and OSG**
  - Run workflows corresponding to each of the 17 bands
  - Total Number of Data Files – **18 million**
  - Potential Size of Data Output – **86 TB**

LIGO Scientific Collaboration

- 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 is performed for potential sources of continuous periodic waves near the Galactic Center and the galactic core
- Search for binary inspirals collapsing into black holes.
- The search is very compute and data intensive



LIGO



Support for LIGO on LIGO Data Grid

LIGO Workflows:

185,000 nodes,

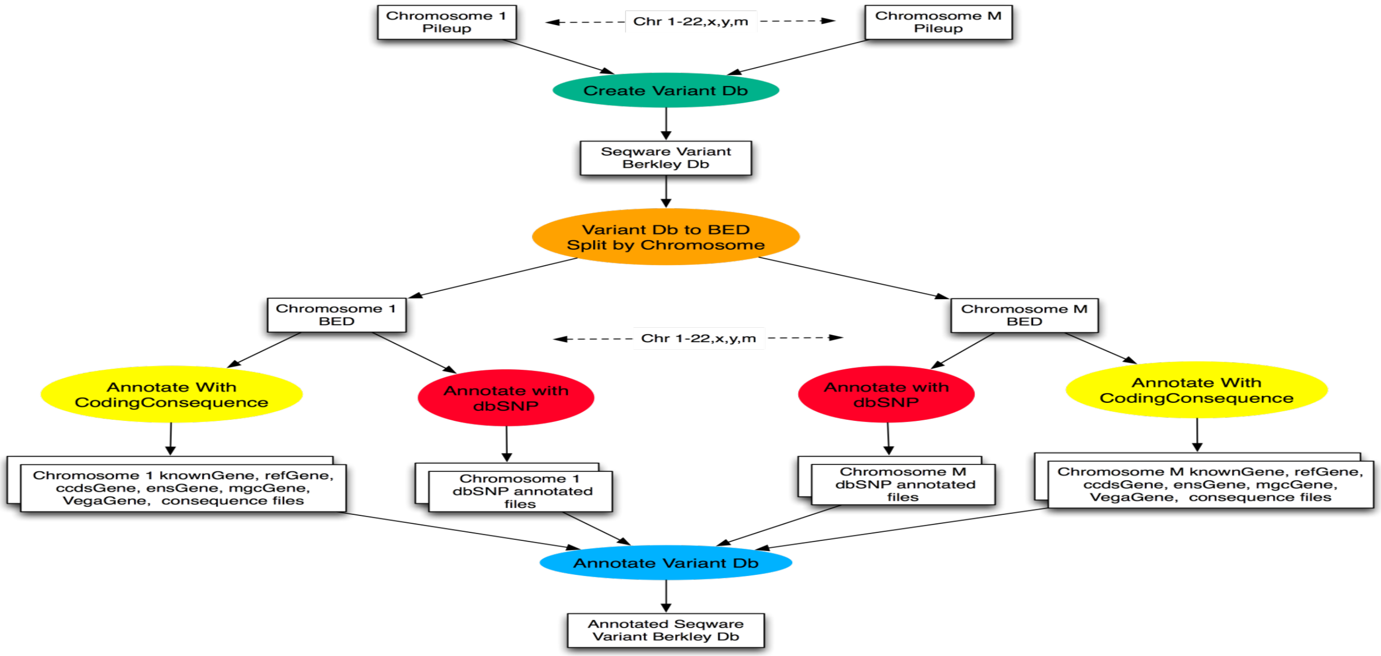
466,000 edges

10 TB of input data,

1 TB of output data.

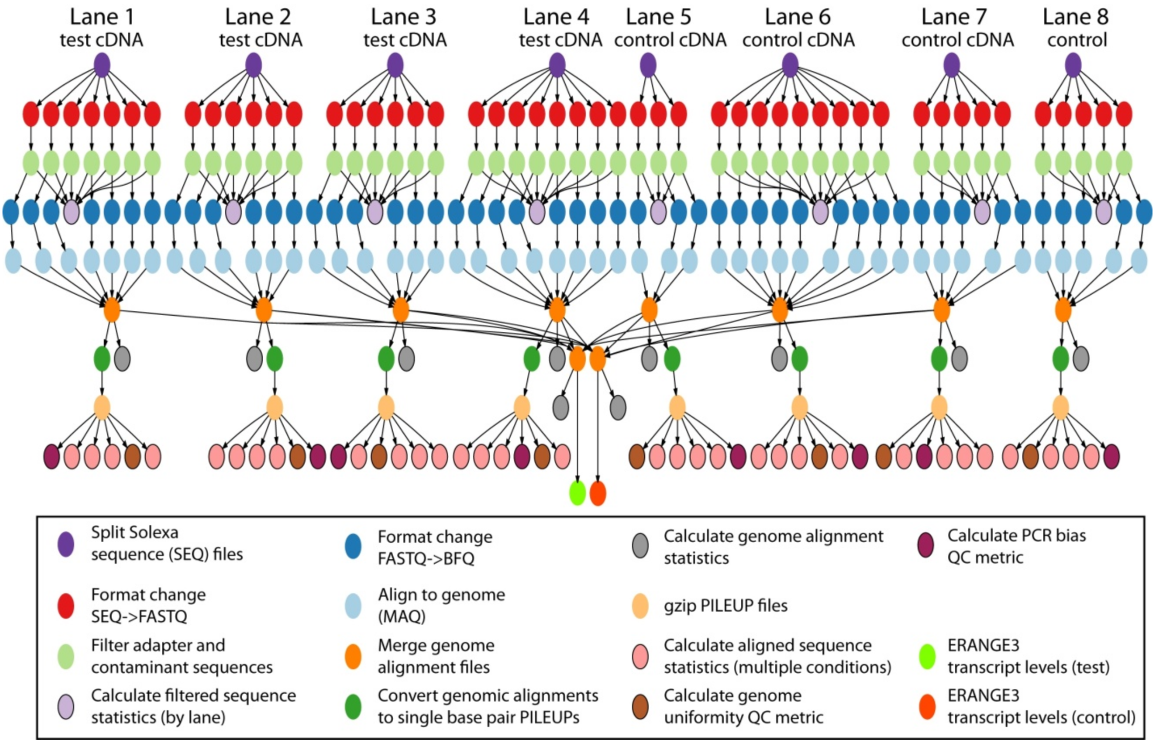
THE CANCER GENOME ATLAS
Seqware

- Computed over 800 sequences using SeqWare framework
- Built on Pegasus WMS.
- Provided as a VM technology


Chromosome 1 Pileup
Chr 1-22,x,y,m
Chromosome M Pileup
Create Variant Db
Seqware Variant Berkley Db
Variant Db to BED Split by Chromosome
Chromosome 1 BED
Chr 1-22,x,y,m
Chromosome M BED
Annotate With CodingConsequence
Annotate with dbSNP
Annotate with dbSNP
Annotate With CodingConsequence
Chromosome 1 knownGene, refGene, ccdsGene, ensGene, mgcGene, VegaGene, consequence files
Chromosome 1 dbSNP annotated files
Chromosome M dbSNP annotated files
Chromosome M knownGene, refGene, ccdsGene, ensGene, mgcGene, VegaGene, consequence files
Annotate Variant Db
Annotated Seqware Variant Berkley Db

Epigenomic Workflows

Mapping the epigenetic state of human cells on a genome-wide scale


Lane 1
test cDNA
Lane 2
test cDNA
Lane 3
test cDNA
Lane 4
test cDNA
Lane 5
control cDNA
Lane 6
control cDNA
Lane 7
control cDNA
Lane 8
control
Split Solexa sequence (SEQ) files
Format change SEQ->FASTQ
Filter adapter and contaminant sequences
Calculate filtered sequence statistics (by lane)
Format change FASTQ->BFQ
Align to genome (MAQ)
Merge genome alignment files
Convert genomic alignments to single base pair PILEUPs
Calculate genome alignment statistics
gzip PILEUP files
Calculate aligned sequence statistics (multiple conditions)
Calculate genome uniformity QC metric
Calculate PCR bias QC metric
ERANGE3 transcript levels (test)
ERANGE3 transcript levels (control)

- split sequence files into multiple parts to be processed in parallel
- convert sequence files to the appropriate file format
- filter out noisy and contaminating sequences
- map sequences to their genomic locations
- merge output from individual mapping steps into a single global map
- use sequence maps to calculate the sequence density at each position in the genome

~7 hours on 8 procs, 6GB of data footprint

Ben Berman and others at USC

Relevant Links

- Pegasus WMS: http://pegasus.isi.edu/wms
- Tutorial and VM : http://pegasus.isi.edu/wms/docs/4.0/tutorial_vm.php
- Ask not what you can do for Pegasus, but what Pegasus can do for you : pegasus@isi.edu
- Support Lists
  - pegasus-users@isi.edu , **pegasus-support@isi.edu**

Acknowledgements

- Pegasus Team, Condor Team, all the Scientists that use Pegasus, Funding Agencies NSF, NIH..