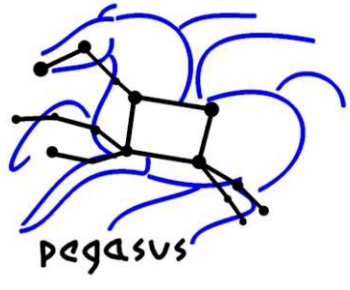


Pegasus
Workflow Management System

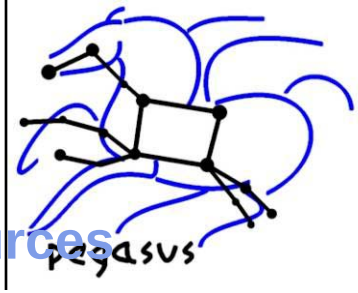
Ewa Deelman

http://pegasus.isi.edu

USC
VITERBI
SCHOOL OF
ENGINEERING

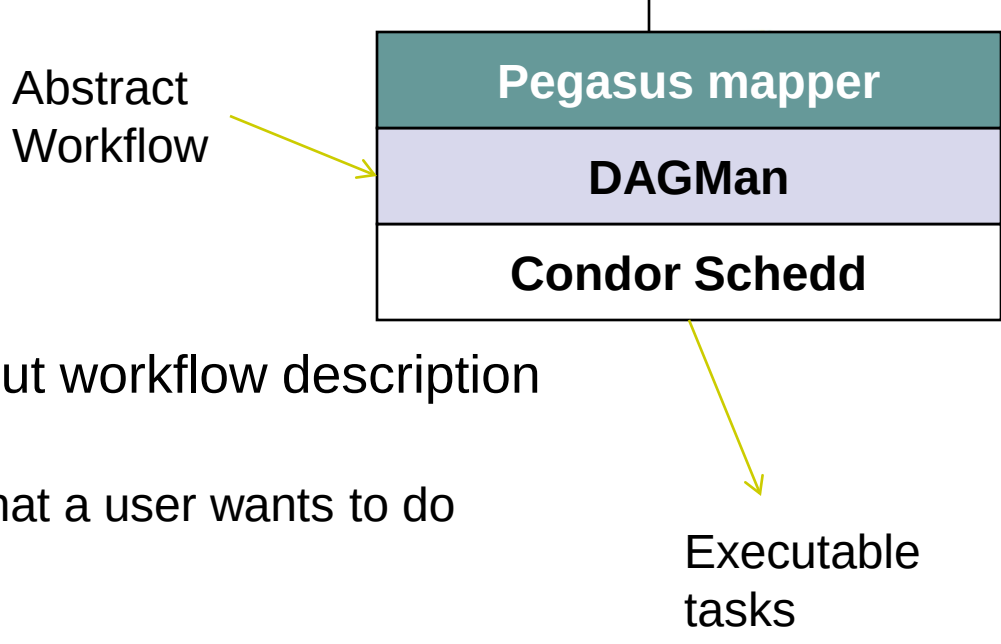
A collaboration with the Condor project , UW Madison
Funded by the NSF OCI SDCI project

Types of Workflow Applications



- **Processing large amounts of shared data on shared resources (LIGO project)**
  - Data captured by various instruments and cataloged in community data registries.
  - Amounts of data necessitate reaching out beyond local clusters
  - Automation, scalability and reliability
- **Providing a service to a community (Montage project)**
  - Data and derived data products available to a broad range of users
  - A limited number of small computational requests can be handled locally
  - For large numbers of requests or large requests need to rely on shared cyberinfrastructure resources
  - On-the fly workflow generation, portable workflow definition
- **Supporting community-based analysis (SCEC project)**
  - Codes are collaboratively developed
  - Codes are “strung” together to model complex systems
  - Ability to correctly connect components, scalability
- **Automating the work of one scientist (Epigenomic project, USC)**
  - Data collected in a lab needs to be analyzed in several steps
  - Automation, efficiency, and flexibility (scripts age and are difficult to change)
  - Need to have a record of how data was produced

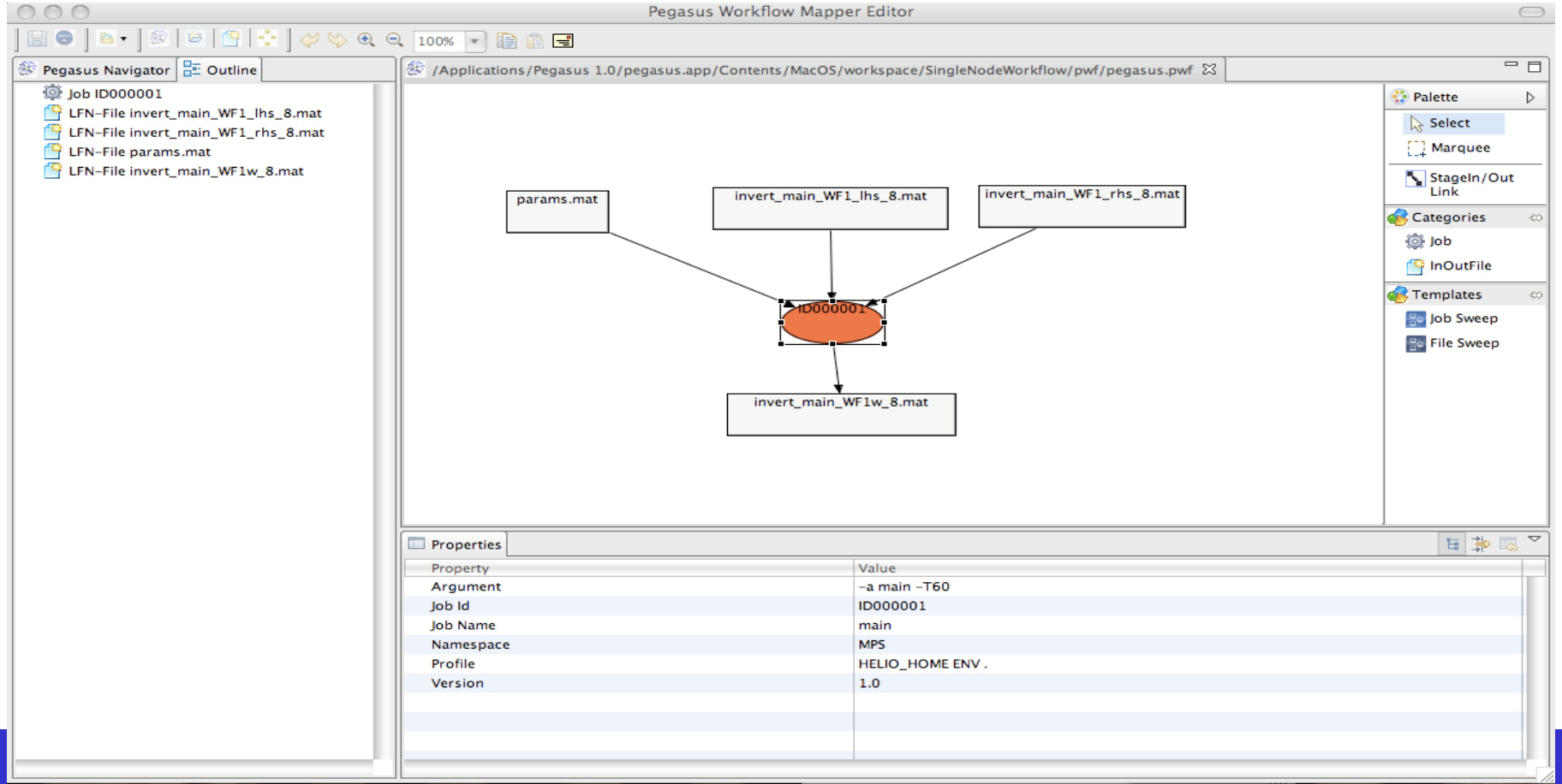
Pegasus WMS



- Abstract Workflows - Pegasus input workflow description
  - workflow “high-level language”
  - only identifies the computations that a user wants to do
  - devoid of resource descriptions
  - devoid of data locations
- Pegasus Mapper
  - a workflow “compiler”
  - target language - DAGMan’s DAG and Condor submit files
  - automatically locates physical locations for both workflow components and data
  - finds appropriate resources to execute the components
  - transforms the workflow for performance and reliability
  - provides runtime provenance
- DAGMan
  - A workflow executor
  - Scalable and reliable execution of an executable workflow

How to generate a DAX

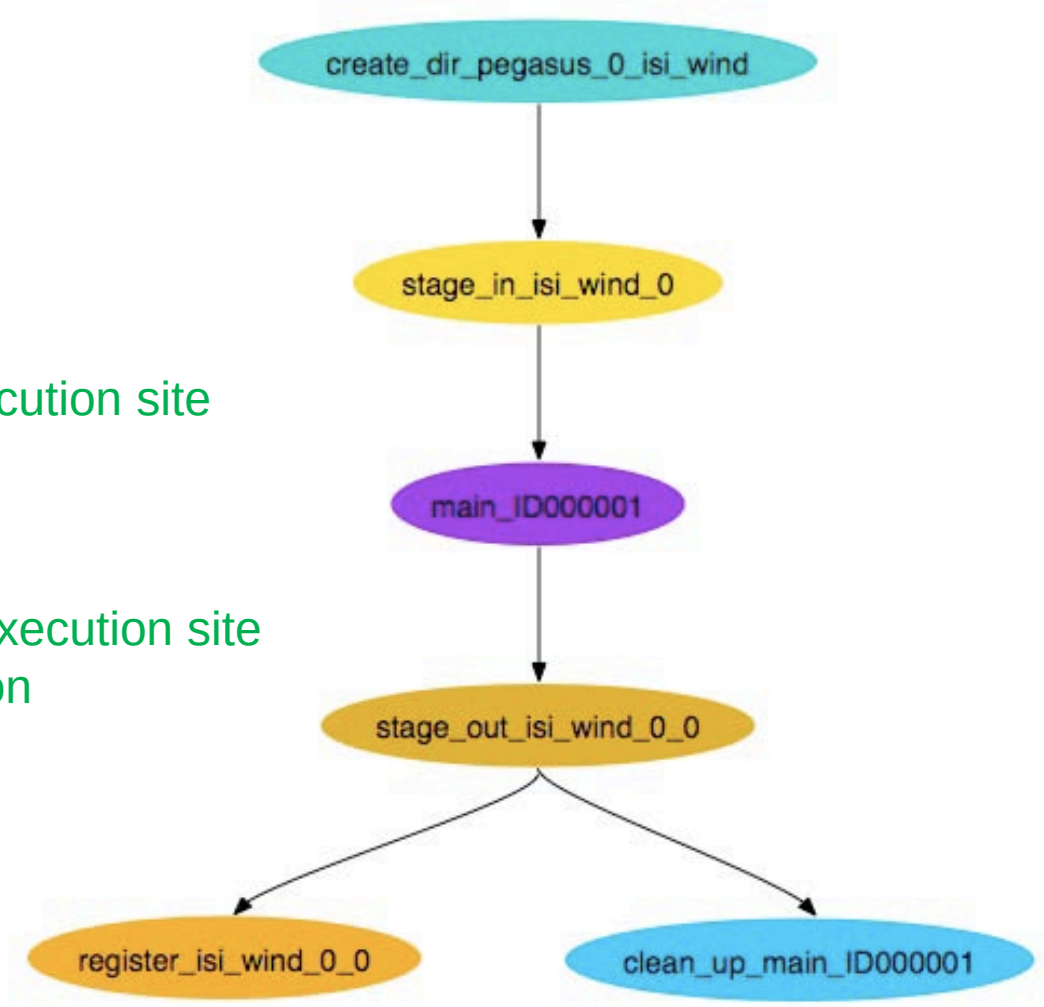
- Write the XML directly
- Use the Pegasus Java API (python, perl in the works)
- Simple GUI interface (eclipse plug-in)
- Use Wings for semantically rich workflow composition (http://www.isi.edu/ikcap/wings/)

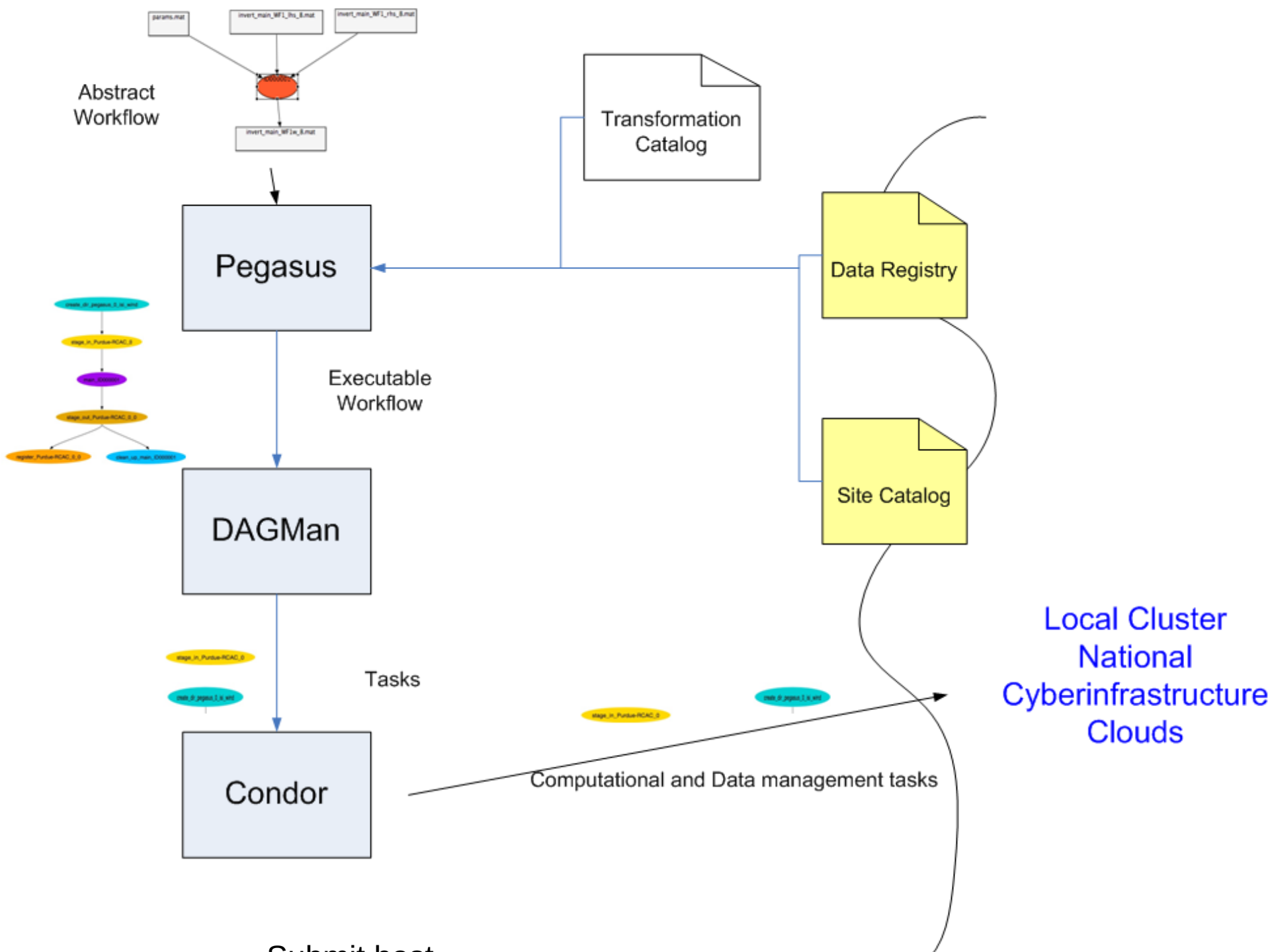


Executable Workflow Generated by Pegasus

Pegasus:
Selects an execution site
Selects a data archive
Creates a workflow that
- Creates a “sandbox” on the execution site
- Stages data
- Invokes the computation
- Stages out data
- Registers data and Cleans up execution site
- Captures provenance information

Performs other optimizations




Abstract
Workflow
Transformation
Catalog
Pegasus
Data Registry
Executable
Workflow
Site Catalog
DAGMan
Tasks
Local Cluster
National
Cyberinfrastructure
Clouds
Condor
Computational and Data management tasks

Populated by user or community

Transformation Catalog

```
#
# Transformation Catalog for MPS codes
#

#
# static binaries at submit host
#
local  MPS::main:1.0       gsiftp://submit.isi.edu/mps/main    STATIC_BINARY       INTEL32::LINUX NULL
local  MPS::inversion:1.0  gsiftp://submit.isi.edu/mps/inversion    STATIC_BINARY       INTEL32::LINUX NULL
local   MPS::combine:1.0   gsiftp://submiot.isi.edu/mps/combine    STATIC_BINARY       INTEL32::LINUX NULL

#
# binaries already installed
#
Purdue-RCAC  MPS::main:1.0   /nfs/home/inversion/COMPILE/main INSTALLED INTEL64::LINUX  NULL
Purdue-RCAC  MPS::inversion:1.0   /nfs/home/inversion/COMPILE/inversion INSTALLED INTEL64::LINUX  pegasus::bundle=10
Purdue-RCAC  MPS::combine:1.0   /nfs/home/inversion/COMPILE/combine INSTALLED INTEL64::LINUX  NULL
```

Populated automatically through pegasus-get-sites* or by the user

```
<site  handle="Purdue-RCAC" arch="x86" os="LINUX">
    <grid  type="gt2" contact="osg.rcac.purdue.edu:2119/jobmanager-fork" scheduler="Fork" jobtype="auxillary"/>
    <grid  type="gt2" contact="osg.rcac.purdue.edu:2119/jobmanager-condor" scheduler="Condor" jobtype="compute"/>
    <head-fs>
        <scratch>
            <shared>
                <file-server protocol="gsiftp" url="gsiftp://osg.rcac.purdue.edu" mount-point="/scratch/osg">
                </file-server>
                <internal-mount-point mount-point="/scratch/osg"/>
            </shared>
        </scratch>
        <storage>
            <shared>
                <file-server protocol="gsiftp" url="gsiftp://osg.rcac.purdue.edu" mount-point="/scratch/osg">
                </file-server>
                <internal-mount-point mount-point="/scratch/osg"/>
            </shared>
        </storage>
    </head-fs>
    <replica-catalog  type="LRC" url="rlsn://dummyValue.url.edu">
    </replica-catalog>
    <profile namespace="env" key="app" >/apps/osg</profile>
    <profile namespace="env" key="data" >/scratch/osg</profile>
    <profile namespace="env" key="tmp" >/scratch/osg</profile>
    <profile namespace="env" key="wntmp" >/tmp</profile>
</site>
```

DAX snippet

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- generated: 2009-02-27T10:49:29-08:00 -->
<!-- generated by: vahi [??] -->
<adag xmlns="http://pegasus.isi.edu/schema/DAX" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://pegasus.isi.edu/schema/DAX http://pegasus.isi.edu/schema/dax-2.1.xsd"
    version="2.1" count="1" index="0" name="pegasus" jobCount="1" fileCount="0" childCount="0">
<!-- part 1: list of all referenced files (may be empty) -->
<!-- part 2: definition of all jobs (at least one) -->
  <job id="ID0000001" namespace="MPS" name="main" version="1.0">
    <argument>-a main -T60-i <filename file="params.mat"/> <filename file="invert_main_WF1_lhs_8.mat"/>
      <filename file="invert_main_WF1_rhs_8.mat"/>
        -o <filename file="invert_main_WF1w_8.mat"/>  </argument>
    <profile namespace="ENV" key="HELIO_HOME">.</profile>
    <uses file="params.mat" link="input" register="true" transfer="true" type="data"/>
    <uses file="invert_main_WF1_lhs_8.mat" link="input" register="true" transfer="true
    <uses file="invert_main_WF1_rhs_8.mat" link="input" register="true" transfer="true
    <uses file="invert_main_WF1w_8.mat" link="output" register="true" transfer="true"
  </job>
<!-- part 3: list of control-flow dependencies (may be empty) -->
</adag>
```

Pegasus

CondorDAG

```
#########################################################################
# PEGASUS WMS GENERATED DAG FILE
# DAG invert
# Index = 0, Count = 1
#########################################################################

JOB inversion_ID0000002 inversion_ID0000001.sub
SCRIPT POST inversion_ID00000081 /lfs1/bin/exitpost  inversion_ID0000001.out
RETRY inversion_ID00000081 3
...

PARENT main_ID0000001 CHILD inversion_ID0000002
PARENT stage_out_Purdue-RCAC_0_0 CHILD register_Purdue-RCAC_0_0
PARENT stage_in_Purdue-RCAC_0 CHILD main_ID0000001

#########################################################################
# End of DAG
#########################################################################
```

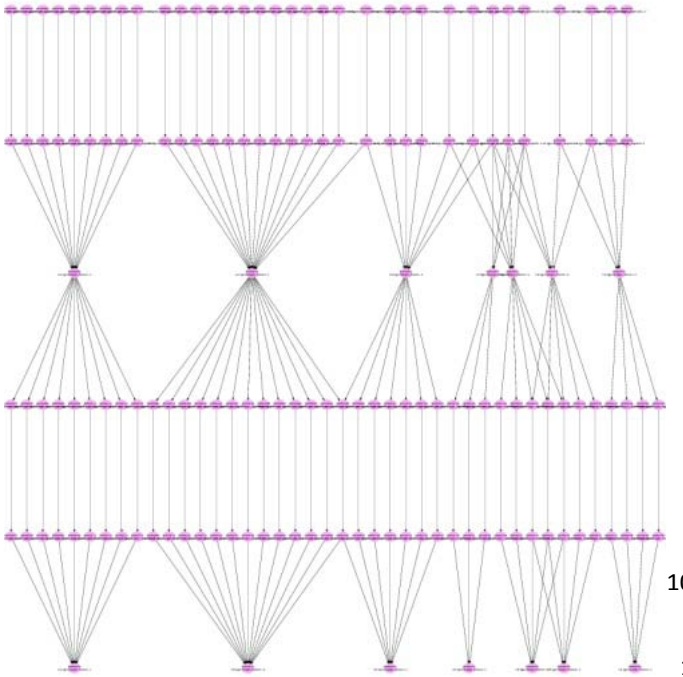
/ Condor Submit files

```
##################################################################
# PEGASUS WMS GENERATED SUBMIT FILE
# DAG : invert, Index = 0, Count = 1
# SUBMIT FILE NAME : main_ID0000001.sub
##################################################################
stream_error  = false
stream_output = false
environment = GLOBUS_LOCATION=/nfs/software/globus/default;
arguments = "-n MPS::main:1.0 -N null -R isi_wind /Codes/main parameters"
copy_to_spool = false
error = /lfs1/work/helio/dags/main_ID0000001.err
executable = /nfs/software/pegasus/default/bin/kickstart
globusrsl = (jobtype=single)
globusscheduler = purdue.isi.edu/jobmanager-condor
log = /tmp/invert-020787.log
notification = NEVER
output = /lfs1/work/helio/dags/main_ID0000001.out
periodic_release = (NumSystemHolds <= 3)
periodic_remove = (NumSystemHolds > 3)
remote_initialdir = /nfs/shared-scratch/helio/run0019
submit_event_user_notes = pool:isi_purdue
transfer_error = true
transfer_executable = false
transfer_output = true
universe = globus

##################################################################
# END OF SUBMIT FILE
##################################################################
```

*OSG interface provided by Vikas Patel and Sebastian Goasguen

Reliability Features of Pegasus and DAGMan

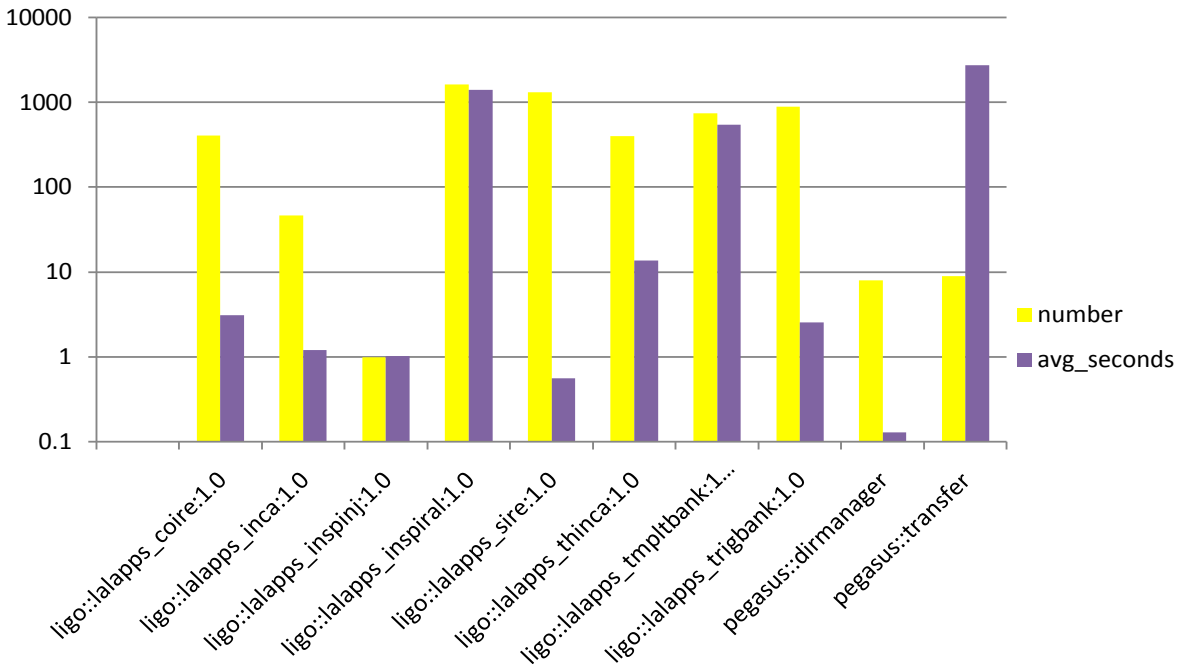
- Provides workflow-level checkpointing through data re-use
- Allows for automatic re-tries of
  - task execution
  - overall workflow execution
  - workflow mapping
- Tries alternative data sources for staging data
- Provides a rescue-DAG when all else fails
- Clustering techniques can reduce some of failures
  - Reduces load on CI services



Small LIGO Inspiral Workflow on OSG

~5,400 tasks

Britta Daudert and Kent Blackburn

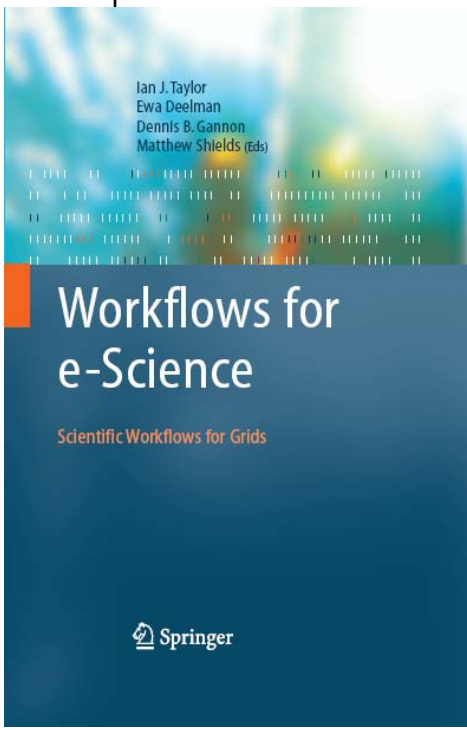


You can execute workflows across multiple sites

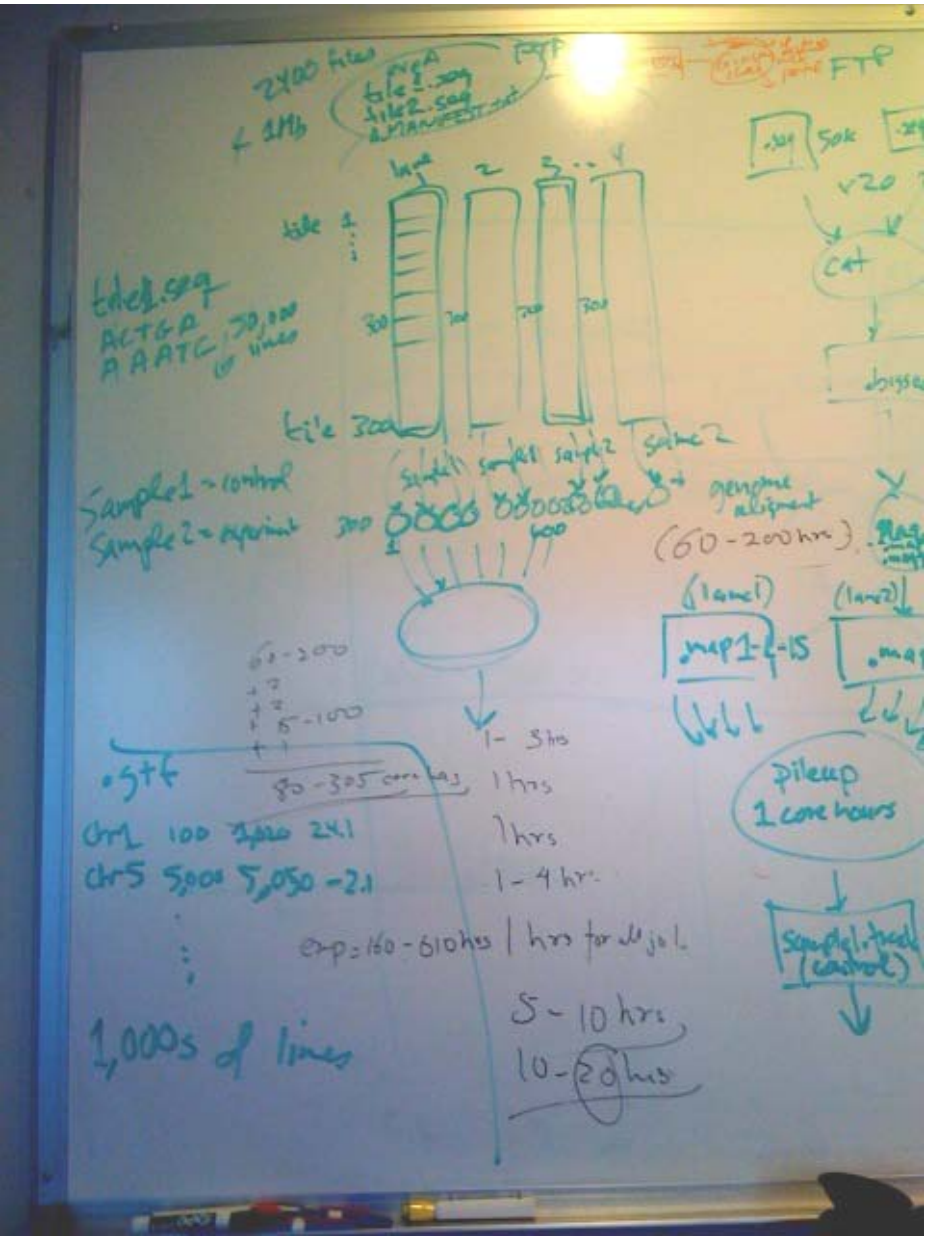
Execution analysis—a collaboration with Dan Gunter and the Netlogger group

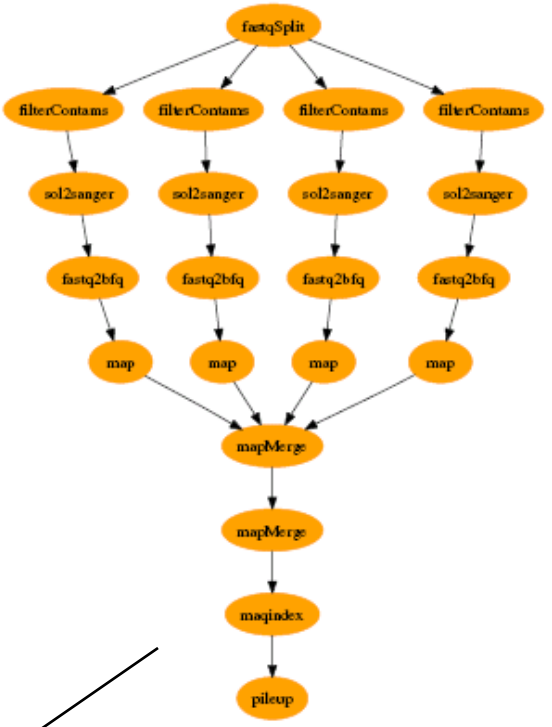
Relevant Links

- Pegasus: http://pegasus.isi.edu
  - Distributed as part of VDT
  - Can be downloaded directly from
    - http://pegasus.isi.edu/code.php
- Interested in trying out Pegasus
  - Do the tutorial
    - http://pegasus.isi.edu/tutorials.php
    - Send email to pegasus@isi.edu, to do tutorial on ISI cluster.
  - Documentations
    - Available at http://pegasus.isi.edu/mapper/doc.php .
- Contact us—we are happy to help!
  - pegasus-support@mailman.isi.edu

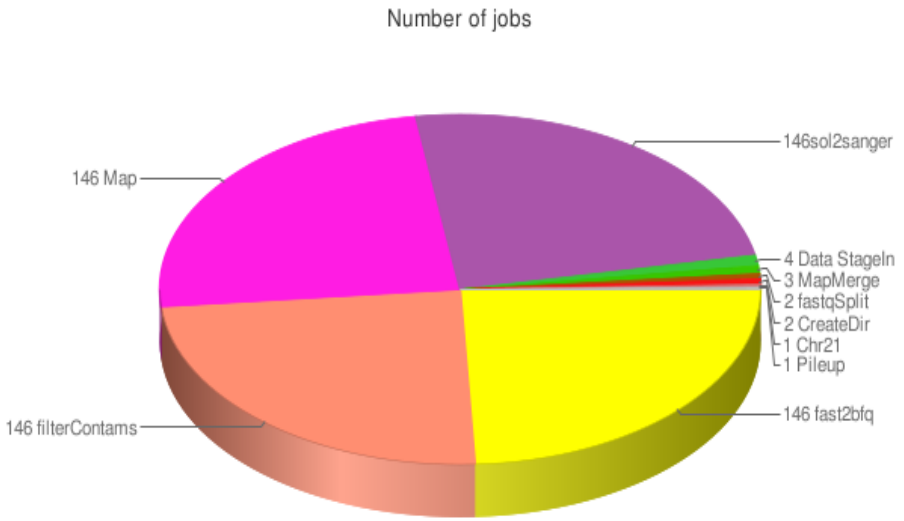


Workflow development process

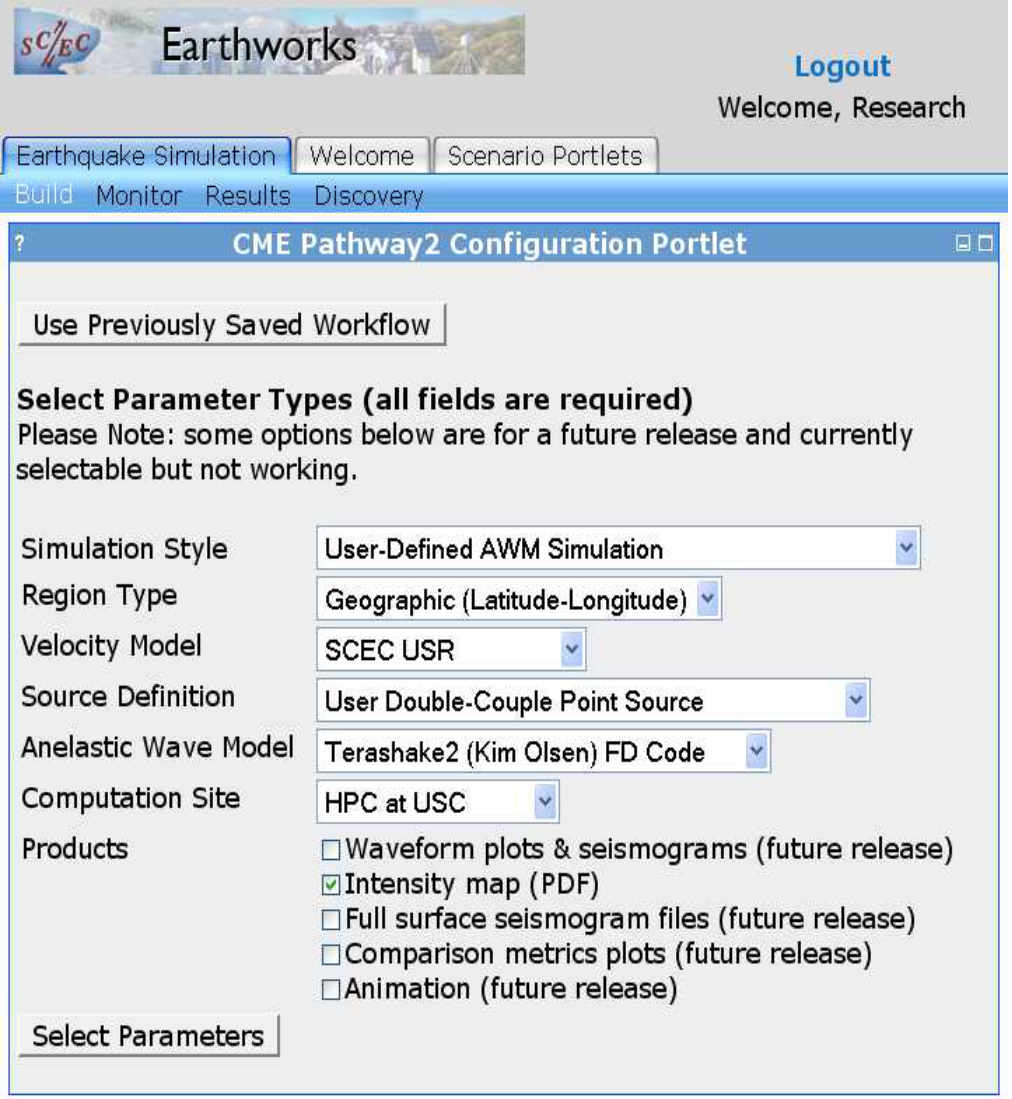


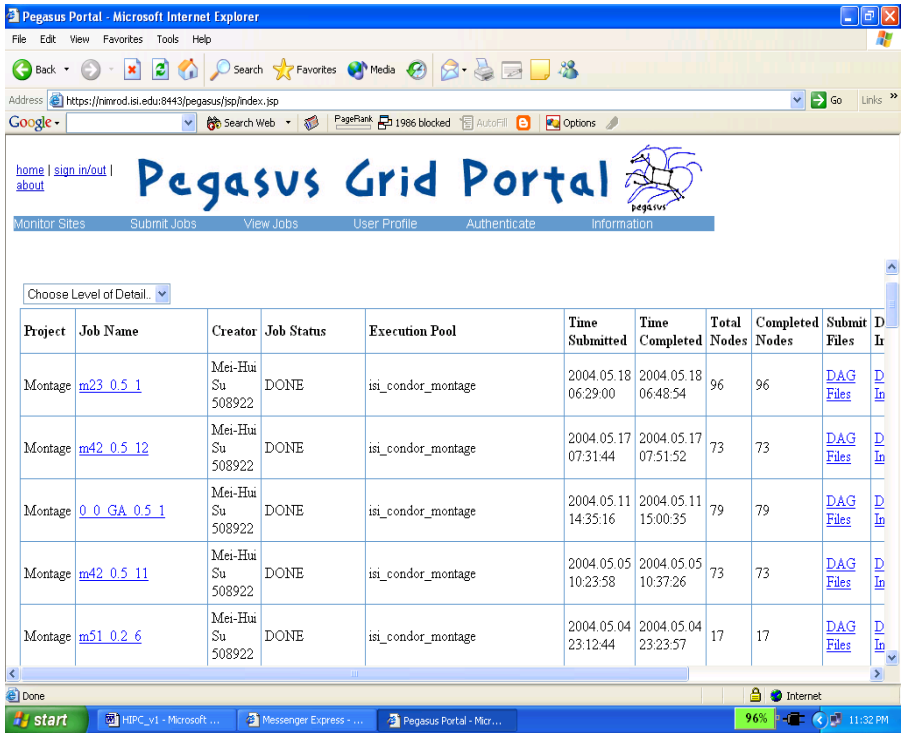


Execution on USC resources



Portal Interfaces for Pegasus workflows

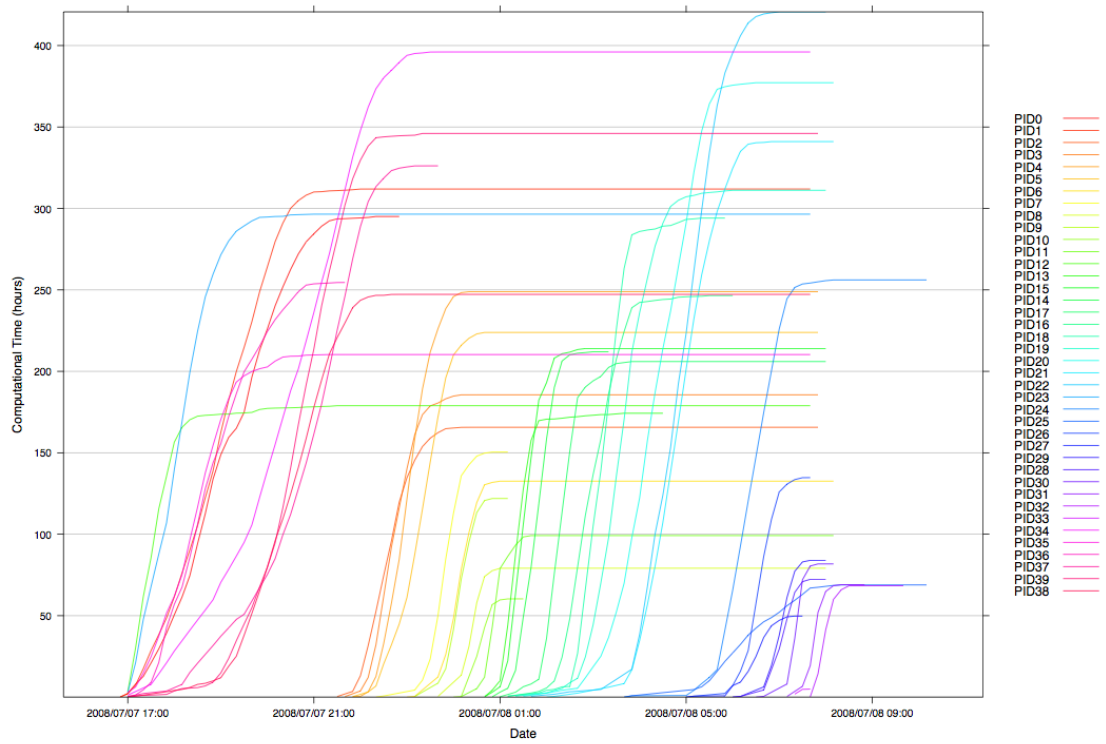




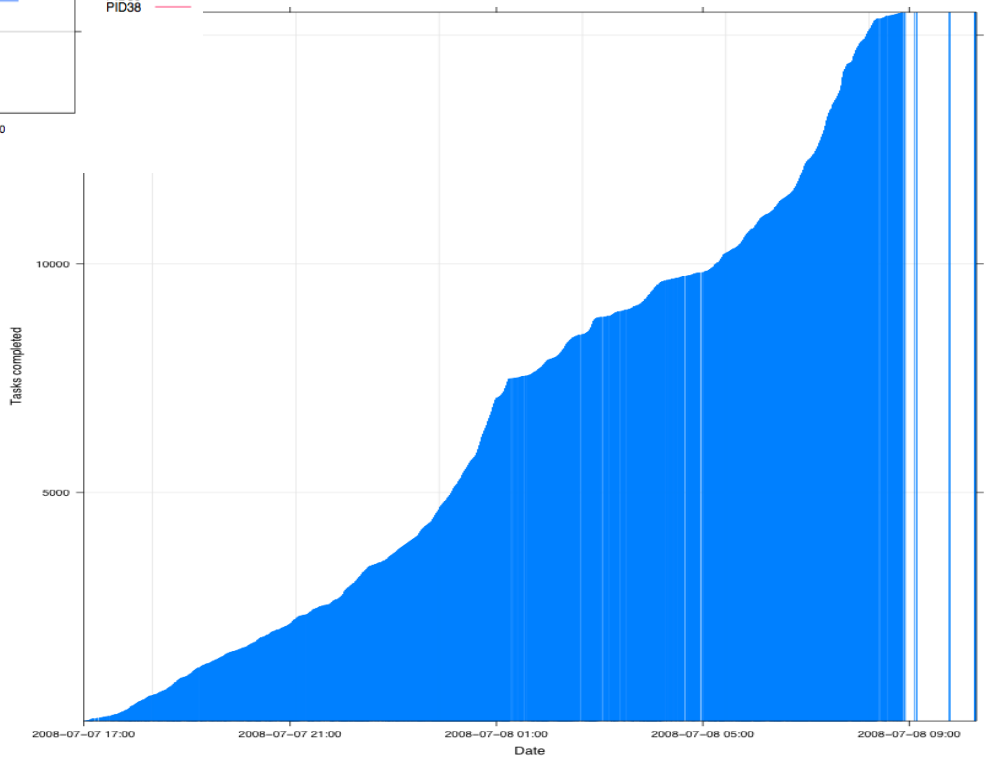
SCEC

Gridsphere-based portal for workflow monitoring

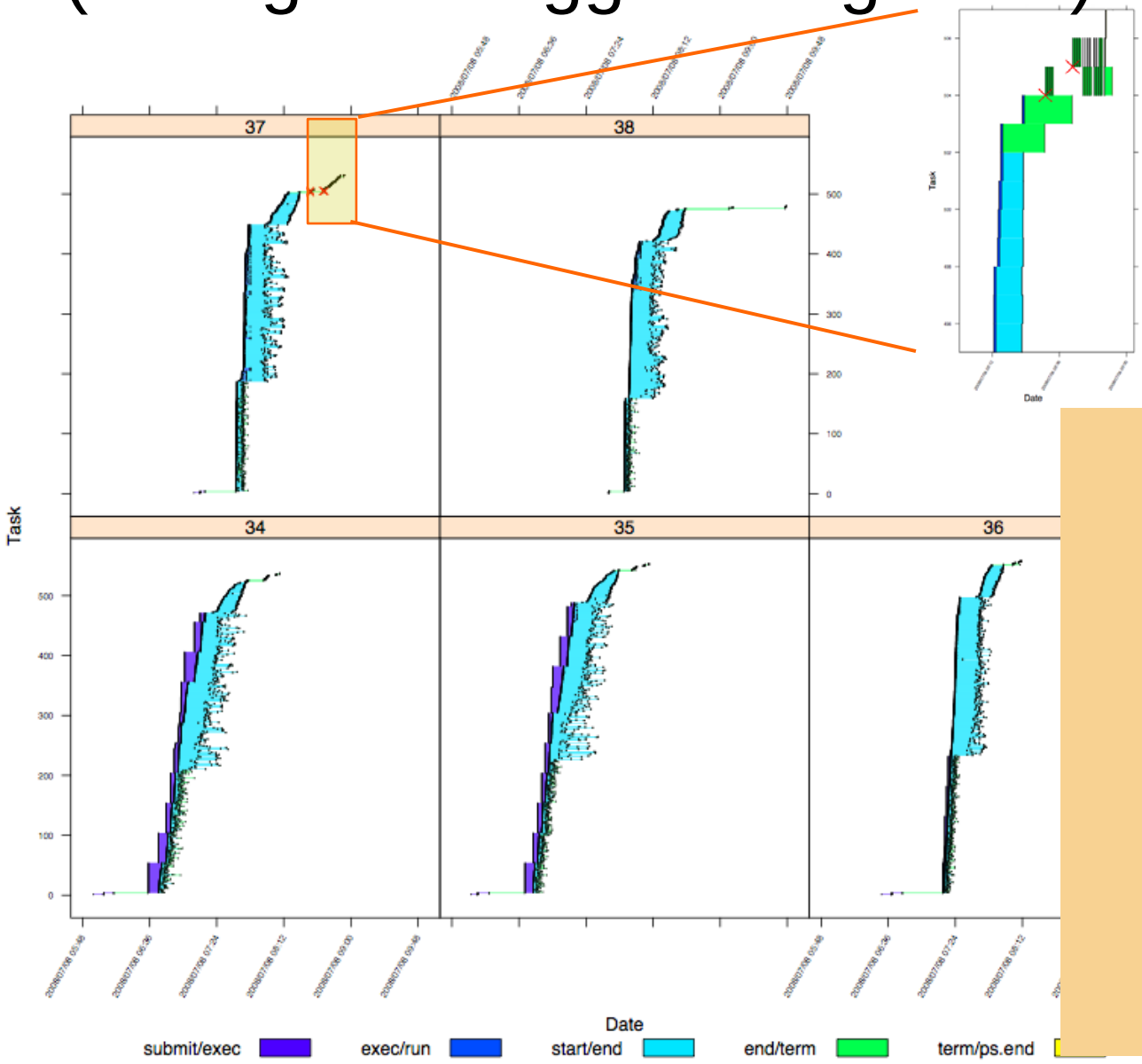
Computational time by workflow partition



Total number of tasks completed



Error Analysis (through Netlogger Integration)



Zoom in on errors

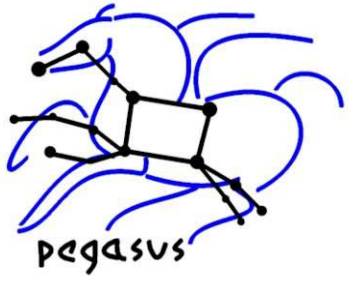
Detailed analysis of time spent in each stage of the hundreds of workflow tasks in each partition of the workflow.

Augmented “Gantt charts” can correlate errors with performance patterns.

Additional Mapping Elements

- Add data cleanup nodes to remove data from remote sites when no longer needed
  - reduces workflow data footprint
- Cluster compute nodes in small computational granularity applications
- Add nodes that register the newly-created data products
- Provide provenance capture steps
  - Information about source of data, executables invoked, environment variables, parameters, machines used, performance
- Supports recursive workflow definition
- Scale matters--today we can handle:
  - 1 million tasks in the workflow instance (SCEC)
  - 10TB input data (LIGO)

Simple Steps to run Pegasus

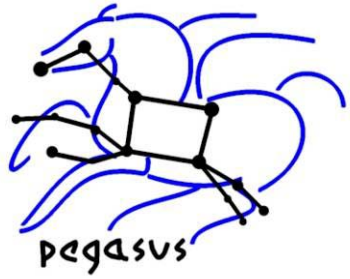


1. Specify your computation in terms of DAX
   - Write a simple DAX generator
   - Java based API provided with Pegasus or GUI
2. Set up your catalogs
   - Use *pegasus-get-sites* to generate site catalog and transformation catalog for your environment
   - Record the locations of your input files in a replica client using *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 your workflow
   - Use *pegasus-status* to monitor the execution of your workflow

Ensemble Manager

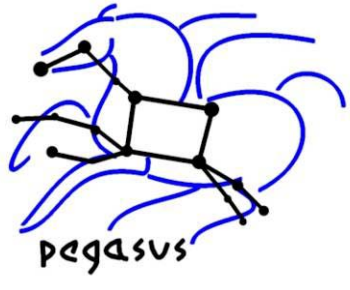
- Manages multiple workflows at any time
- Workflow collections—ensembles
- Supports prioritization
- Submit/start, kill
- Monitor progress of ensembles and individual workflows

Resources Provisioning with the Corral Glidein Service



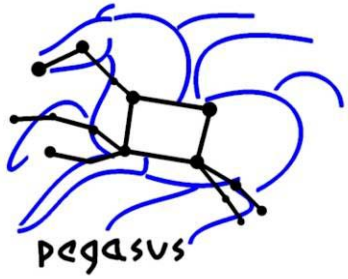
- GT4 grid service for running glideins
  - Automates the installation and configuration of Condor on grid site
  - Simplifies the complex setup and configuration required to run glideins
- Separate setup and provisioning steps
  1. Create “sites” for remote installation and setup of Condor executables
  2. Create “glideins” for resource allocation

Features



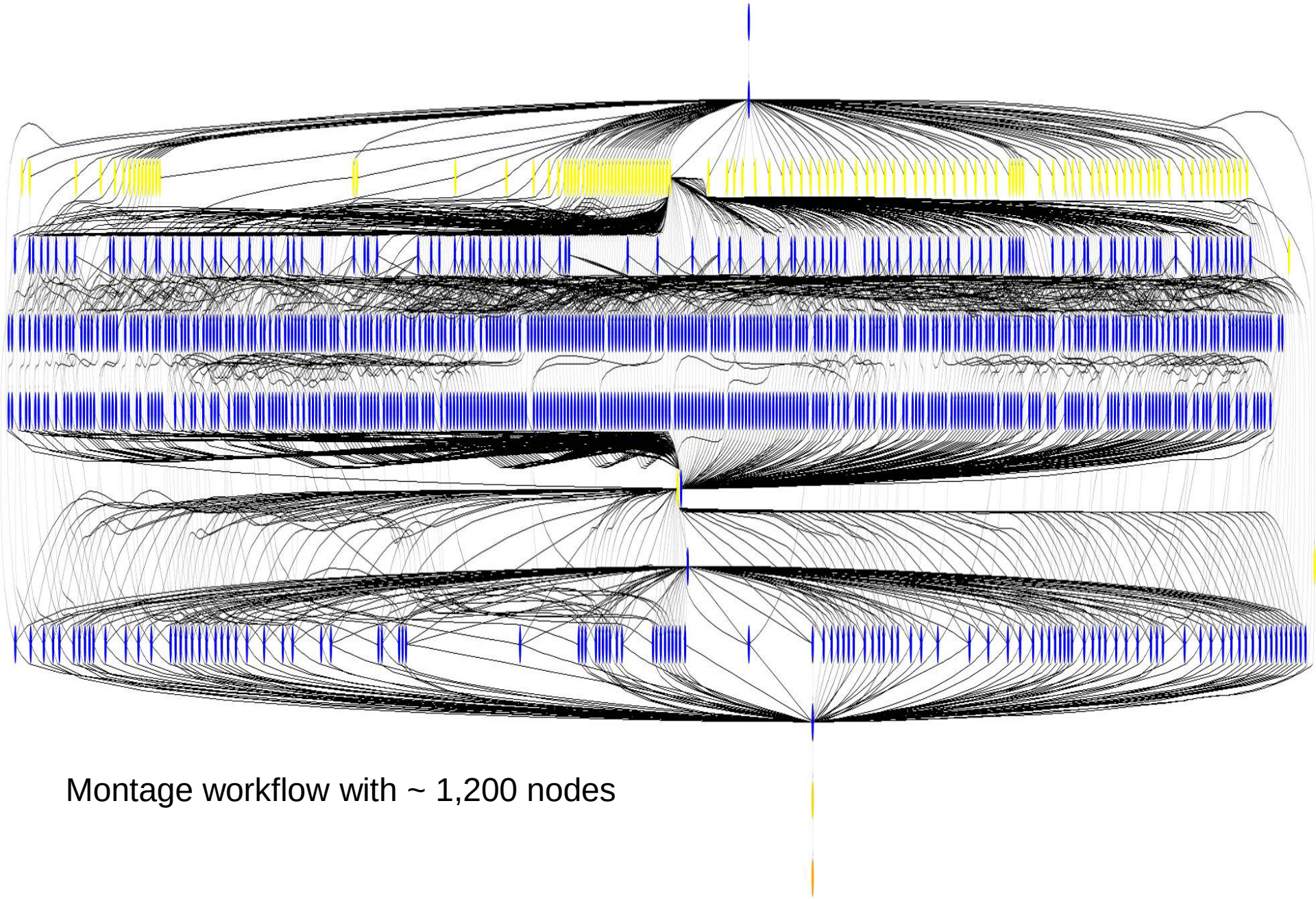
- Auto-configuration
  - Detect architecture, OS, glibc -> Condor package
  - Determine public IP
  - Generate Condor config file
- Multiple interfaces
  - Command-line
  - SOAP
  - Java API
- Automatic resubmission of glideins
  - Indefinitely, N times, until date/time
- Notifications
- History tracking

What does Pegasus do for an application?



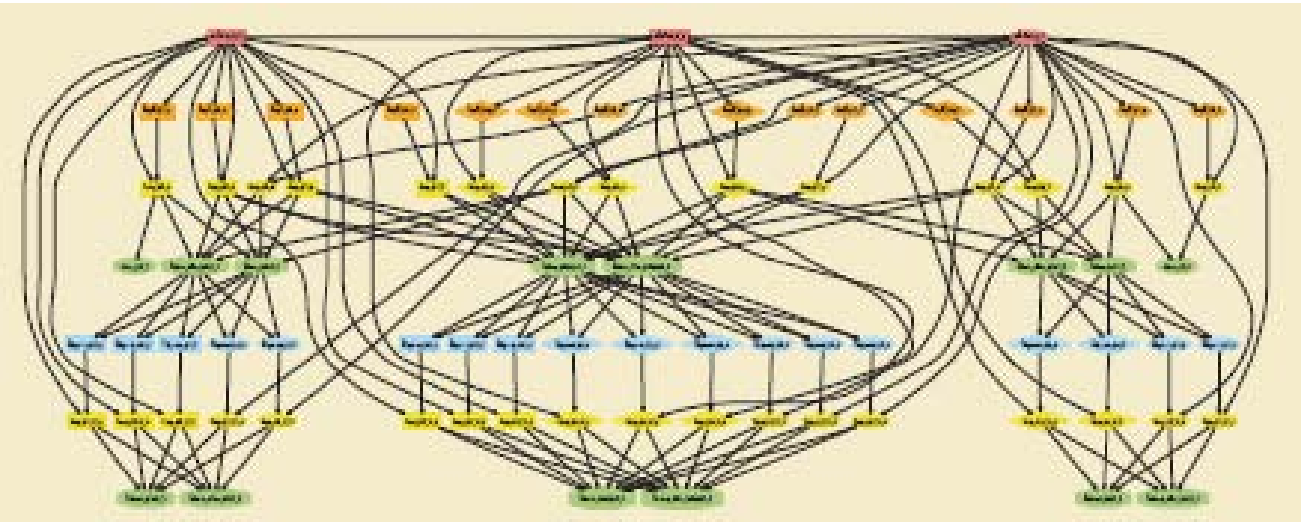
- Provides a Grid and Cloud-aware workflow management tool
  - Interfaces with data registries to discover data
  - Does replica selection to select data
  - Manages data transfer by interfacing to various transfer services like RFT, Stork and clients like globus-url-copy.
  - No need to stage-in data before hand. We do it within the workflow as and when it is required.
  - Reduced Storage footprint. Data is also cleaned as the workflow progresses.
  - Improves successful application execution
  - Improves application performance
  - Data Reuse
    - Avoids duplicate computations
    - Can reuse data that has been generated earlier.

Montage Astronomy Workflow



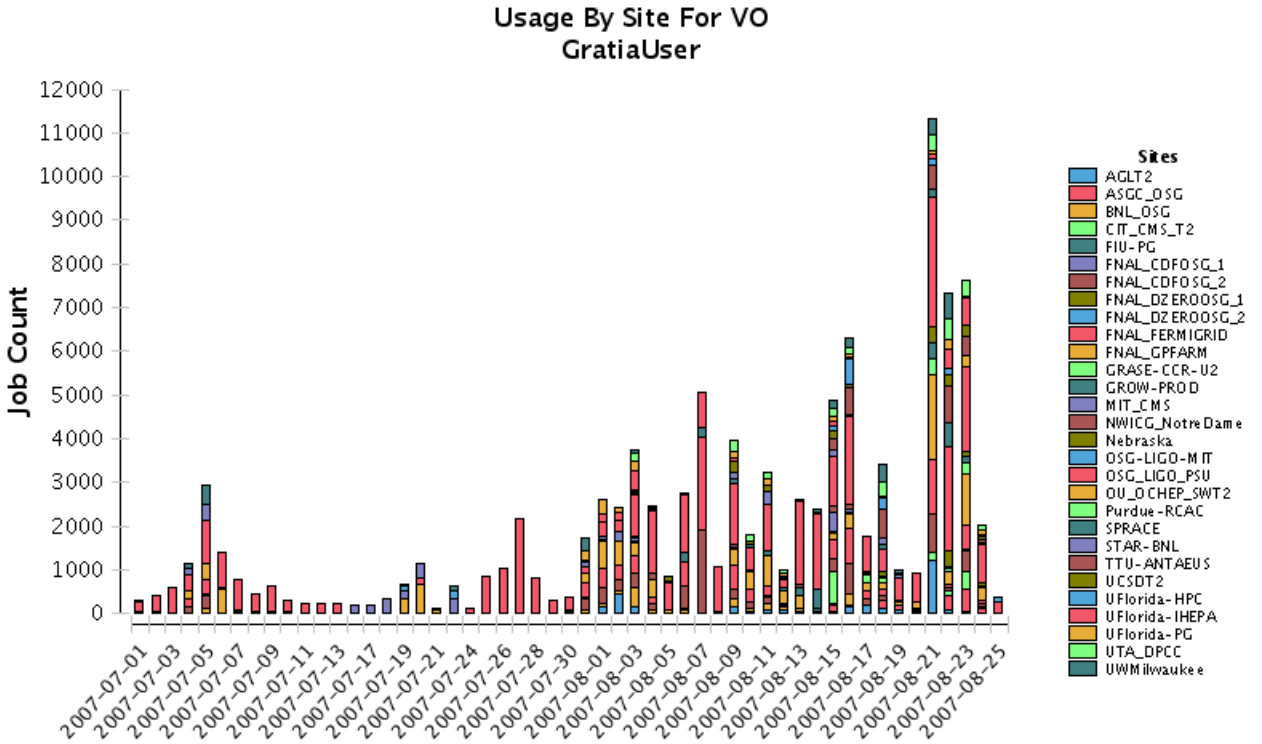
Montage workflow with ~ 1,200 nodes

Pegasus Applications-LIGO

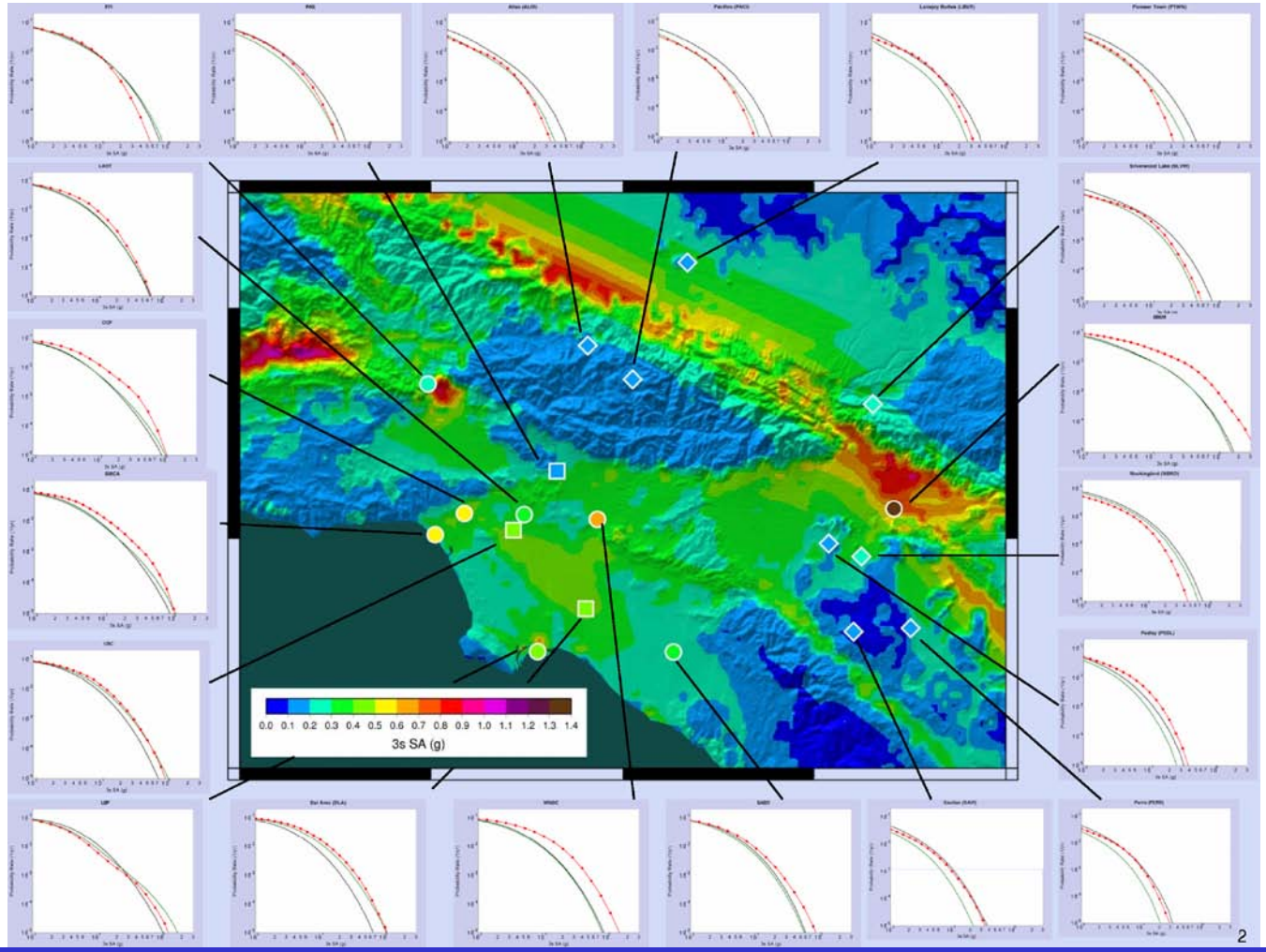


Support for LIGO on Open Science Grid
LIGO Workflows: 185,000 nodes, 466,000 edges 10 TB of input data, 1 TB of output data.

LIGO Collaborators:
Kent Blackburn, Duncan Brown, Britta Daubert, Scott Koranda, Stephen Fairhurst, and others

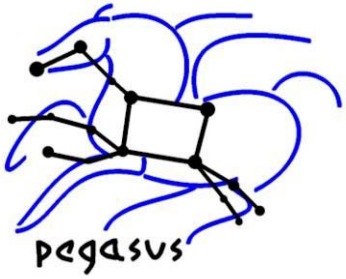


CyberShake, an Earthquake Science application (SCEC,USC)



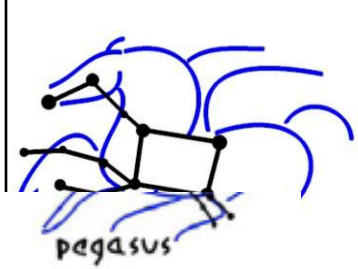
Current SCEC Analysis

June 26, 2008 – July 18, 2008



Site	Pegasus jobs	Extraction	Synthesis	PSA	Zip	Total jobs	Failed jobs	Walltime (min)
1	504	7000	417,886	417,886	80	843,356	264	1042
2	950	7093	418256	418256	154	844709	483	1066
3	1074	7026	417954	417954	156	844164	624	1287
4	896	6932	415416	415416	158	838818	422	988
5	736	6912	415778	415778	156	839360	268	1542
6	703	7045	426740	426740	156	861384	235	964
7	946	6823	416090	416090	158	840107	472	792
8	1552	6919	418946	418946	156	846519	1084	1287
9	923	6947	417772	417772	156	843570	455	890
All	8284	62697	3,764,838	3,764,838	1330	7,601,987	4307	1095

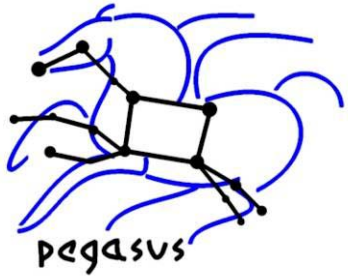
Pegasus DAX



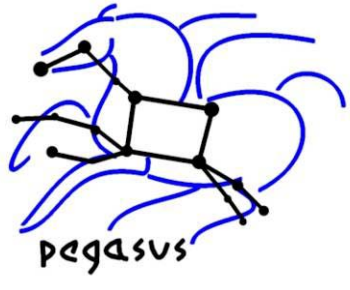

```
<!-- part 1: list of all files used (may be empty) -->
  <filename file="f.input" link="input"/>
  <filename file="f.intermediate" link="input"/>
  <filename file="f.output" link="output"/>

<!-- part 2: definition of all jobs (at least one) -->
  <job id="ID000001" namespace="pegasus" name="preprocess" version="1.0" >
            <argument>-a top -T 6  -i <filename file="f.input"/>  -o <filename file="f.intermediate"/>
            </argument>
            <uses file="f.input" link="input" dontRegister="false" dontTransfer="false"/>
            <uses file="f.intermediate" link="output" dontRegister="true" dontTransfer="true"/>
  </job>
  <job id="ID000002" namespace="pegasus" name="analyze" version="1.0" >
            <argument>-a top -T 6  -i <filename file="f.intermediate"/>  -o <filename file="f.output"/>
            </argument>
            <uses file="f.input" link="input" dontRegister="false" dontTransfer="false"/>
            <uses file="f.intermediate" link="output" dontRegister="false" dontTransfer="false"/>
  </job>
```

- Resource-independent
- Portable across platforms

Comparing a DAX and a Condor DAG

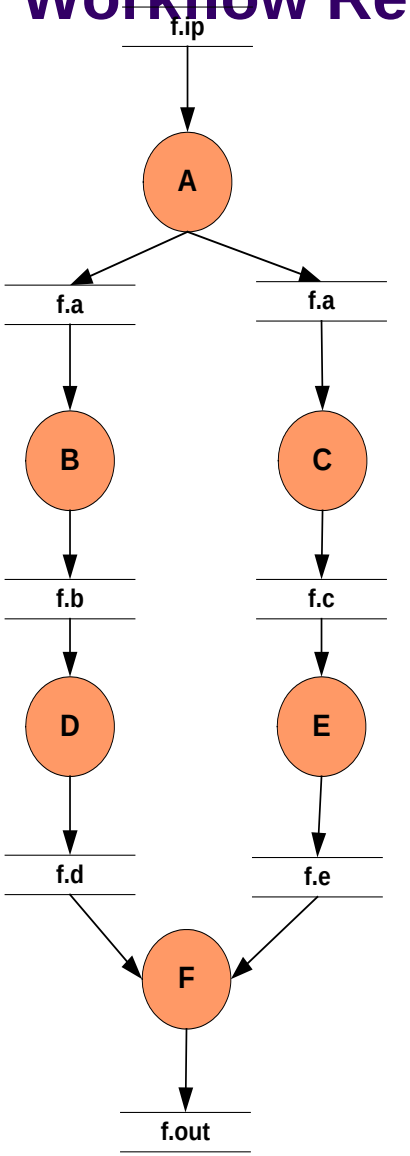


DAX	DAG
Single XML File	One dag file and multiple submit files
Describes your workflow at a logical level	Describes your workflow in terms of physical files and paths
Site Independent	Site Specific
Captures just the computation that the user (you) want to do	Has auxillary jobs like data movement etc.

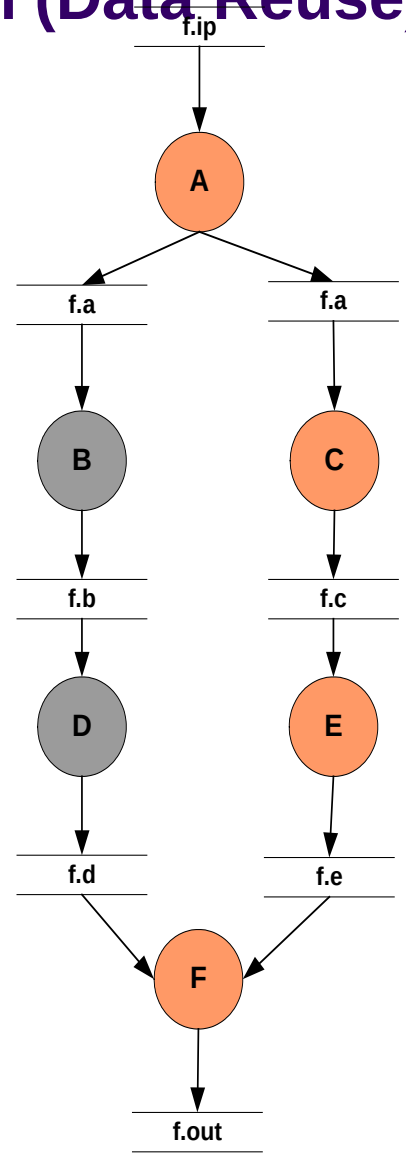


PEGASUS OPTIMIZATIONS

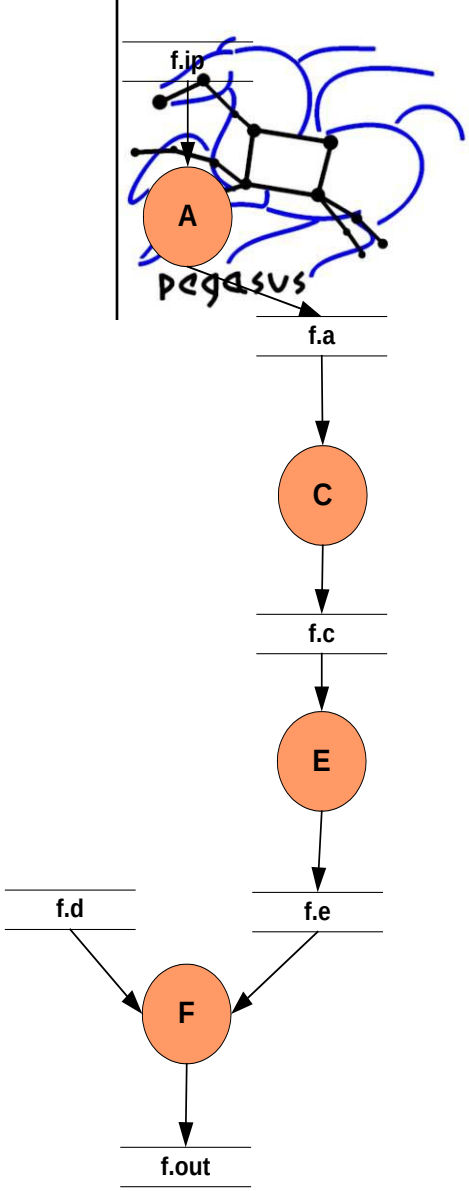
Workflow Reduction (Data Reuse)



Abstract Workflow



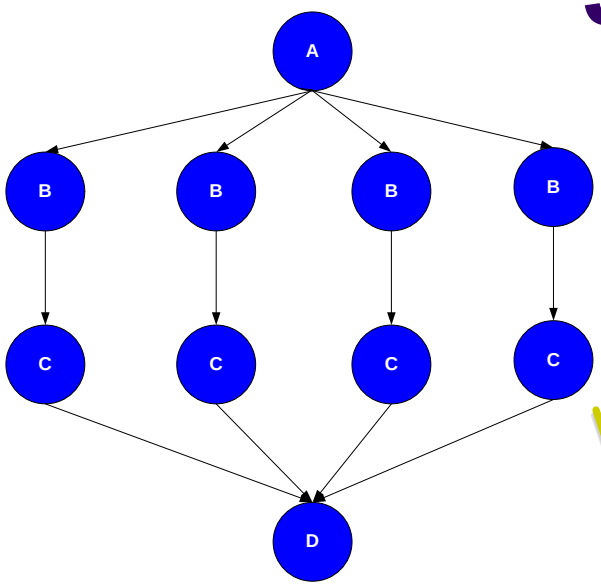
File f.d exists somewhere.
Reuse it.
Mark Jobs D and B to delete

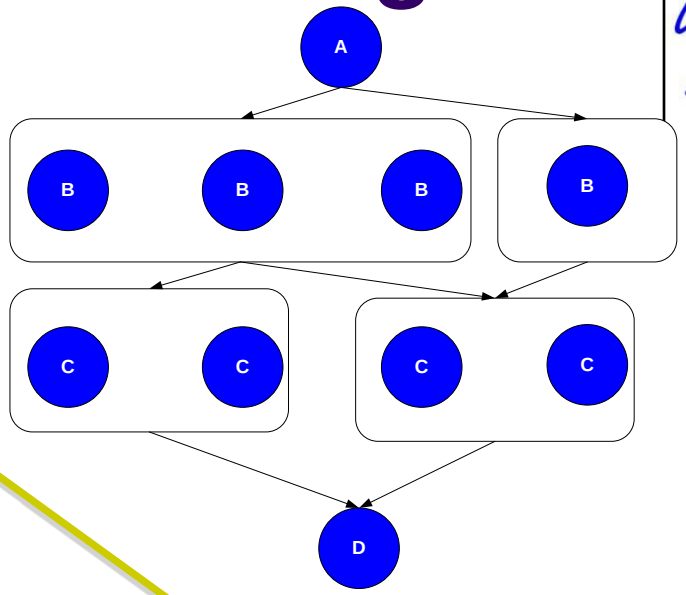


Delete Job D and Job B

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

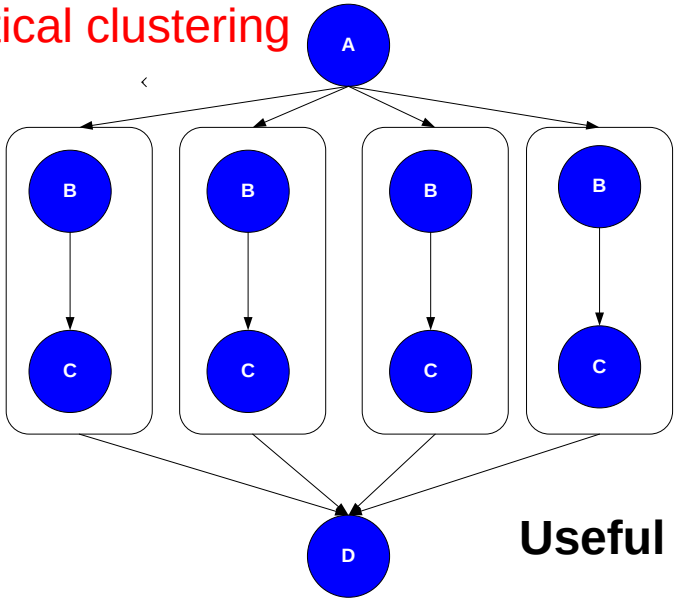
Job clustering



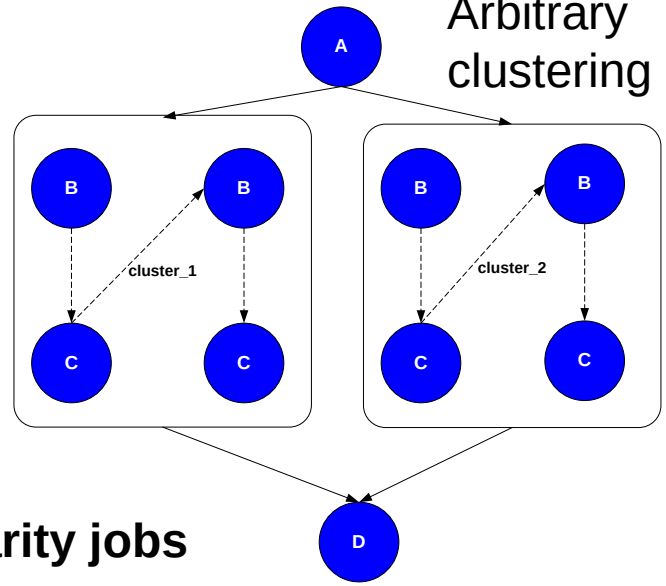


Level-based clustering

Vertical clustering



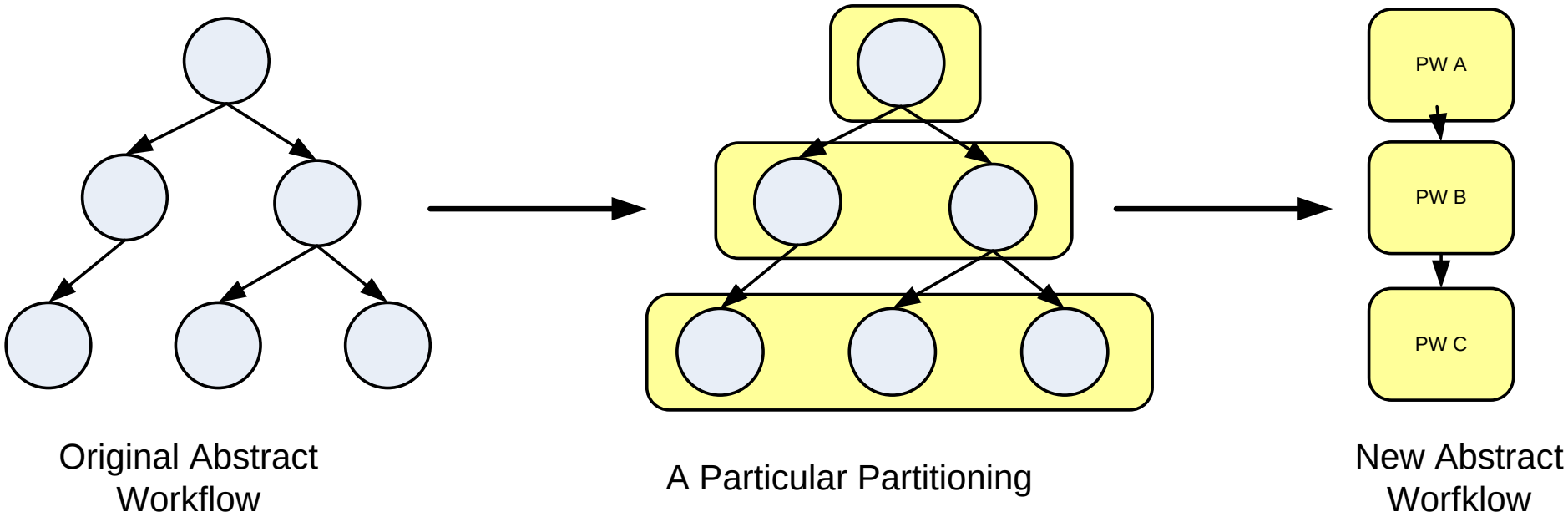
Arbitrary clustering



Useful for small granularity jobs

How to: *To turn job clustering on, pass* ***--cluster*** *to* ***pegasus-plan***

Managing execution environment changes through partitioning



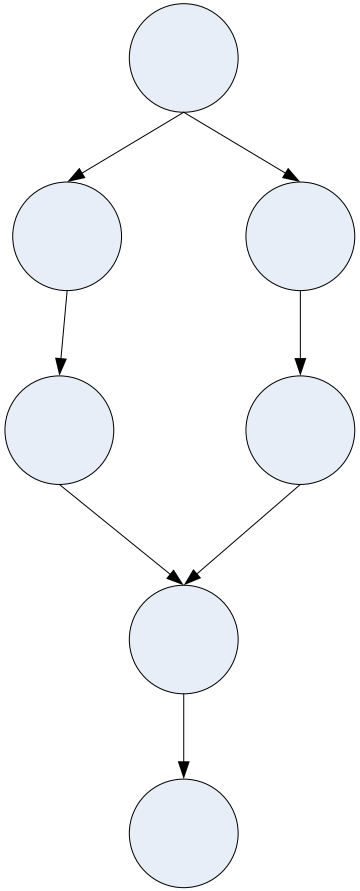
Provides reliability—can replan at partition-level
Provides scalability—can handle portions of the workflow at a time

How to: *1) Partition the workflow into smaller partitions at runtime using* ***partitiondax*** *tool.*

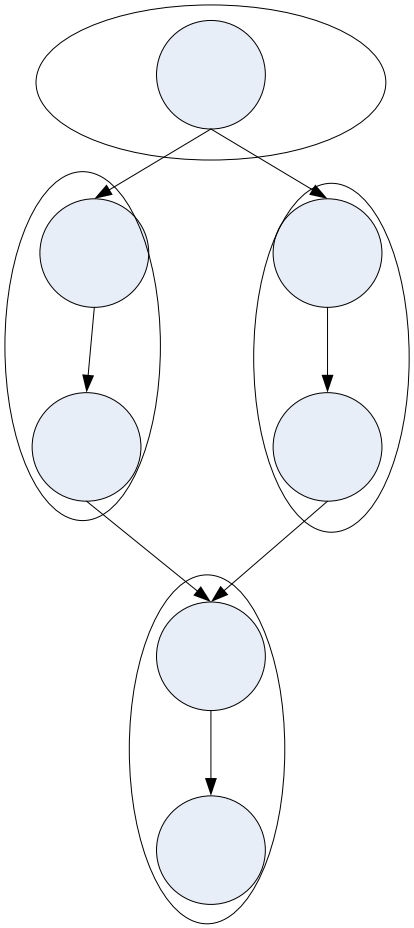
2) Pass the partitioned dax to ***pegasus-plan*** *using the* ***--pdax*** *option.*

Paper: ***"Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems",*** *E.*

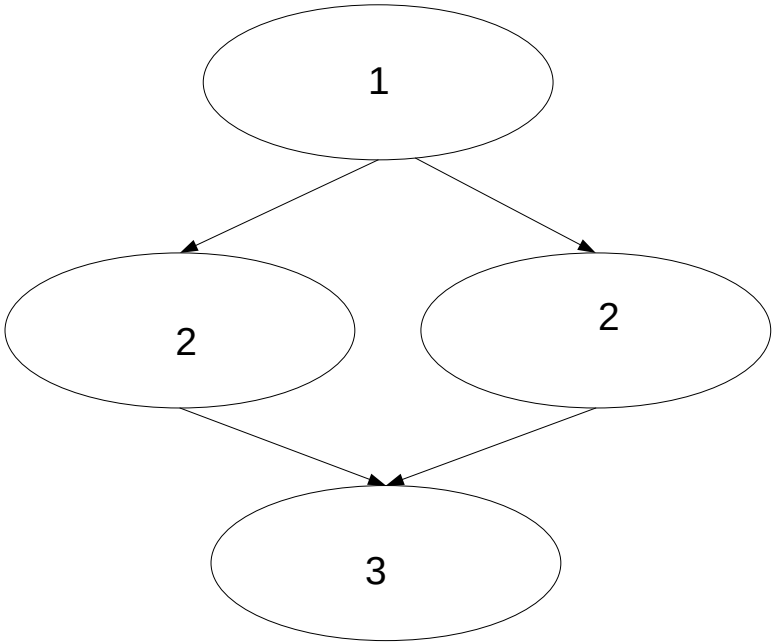
Recursive Workflows



Original Workflow

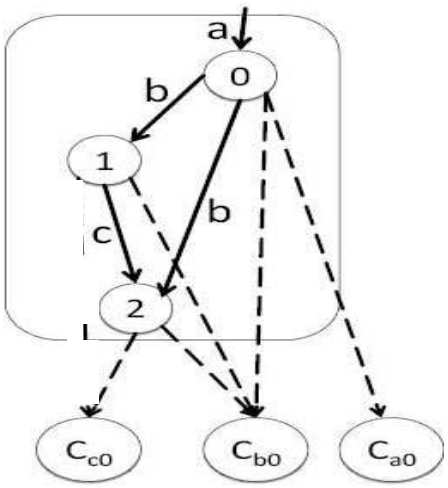


Workflow Described In a Recursive Way

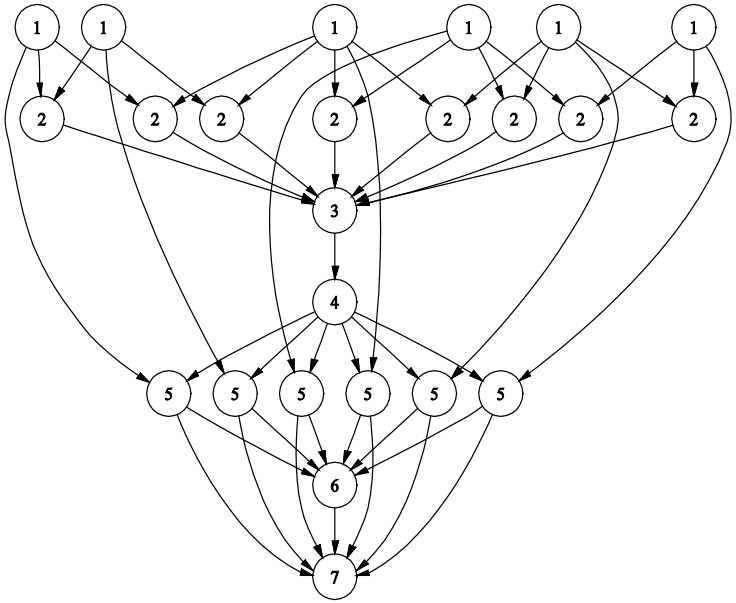


Order of Planning and Execution of the workflow

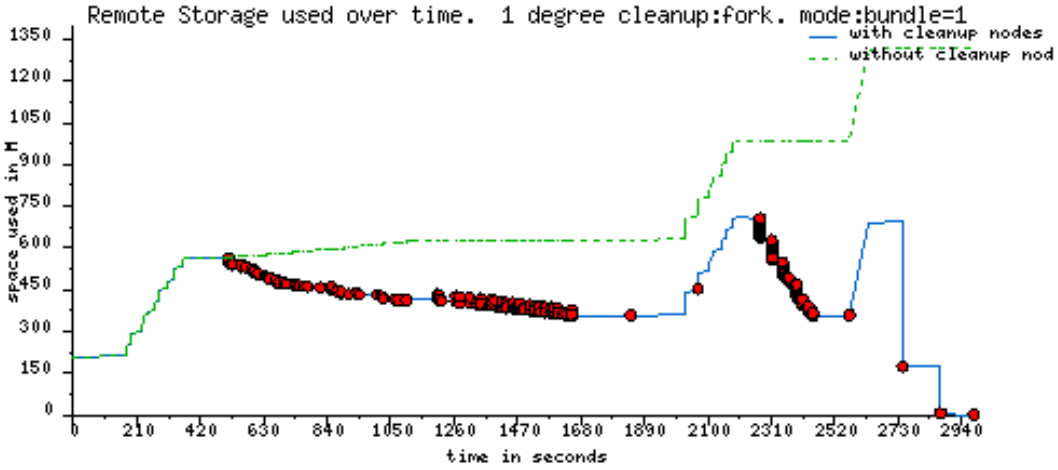
LIGO and Montage

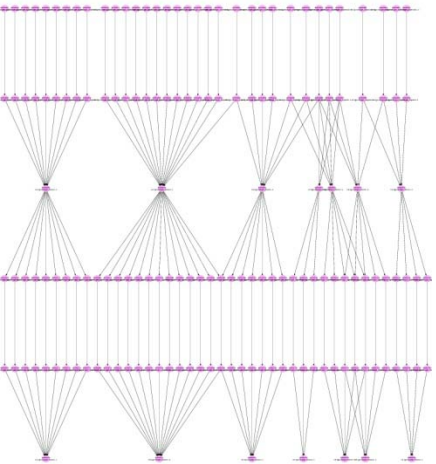


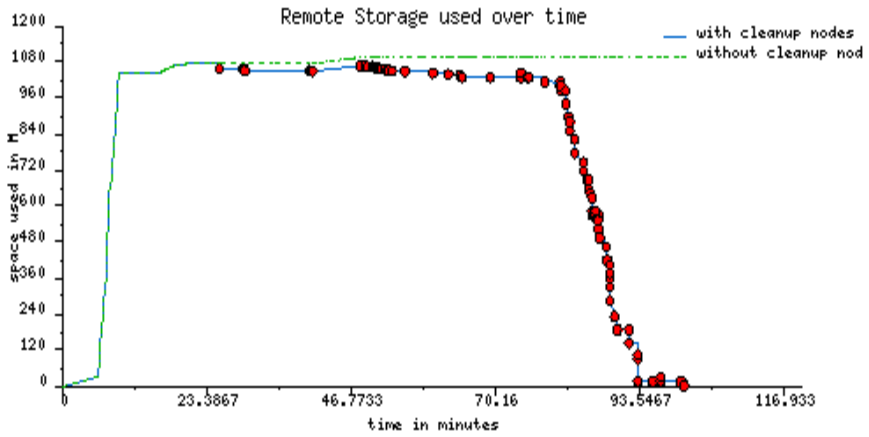
Adding cleanup nodes to the workflow



1.25GB versus 4.5 GB

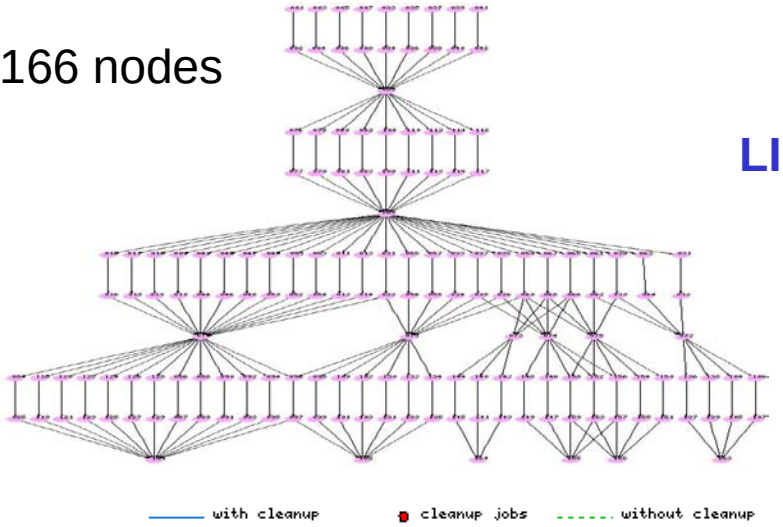




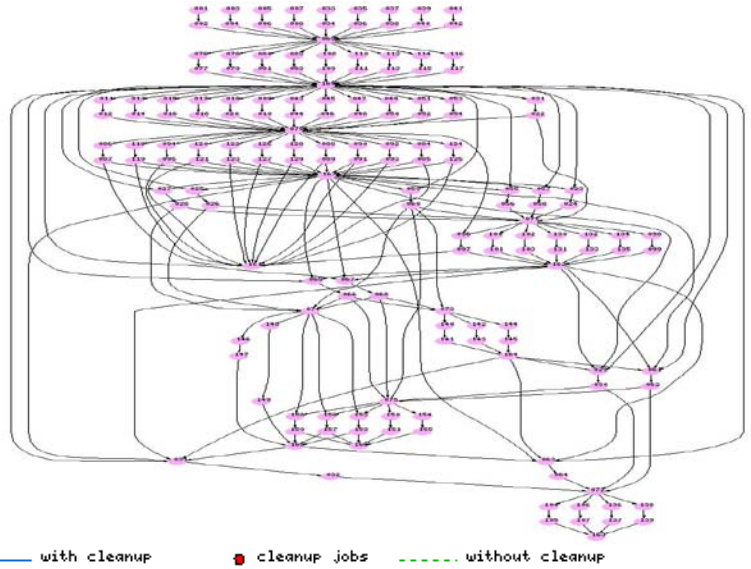


Full workflow:
185,000 nodes
466,000 edges
10 TB of input data
1 TB of output data.

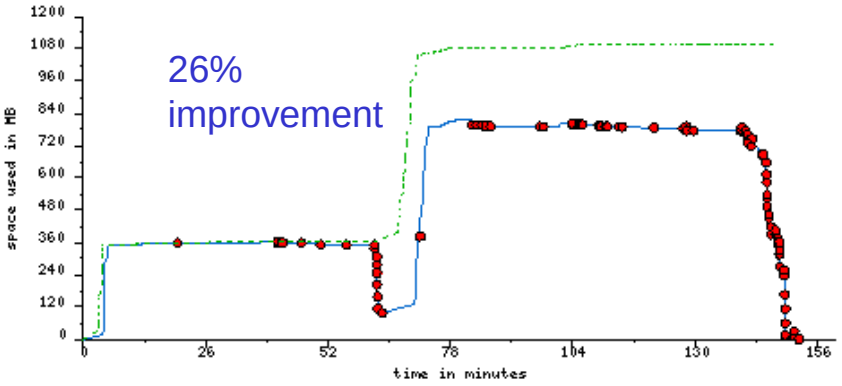
166 nodes

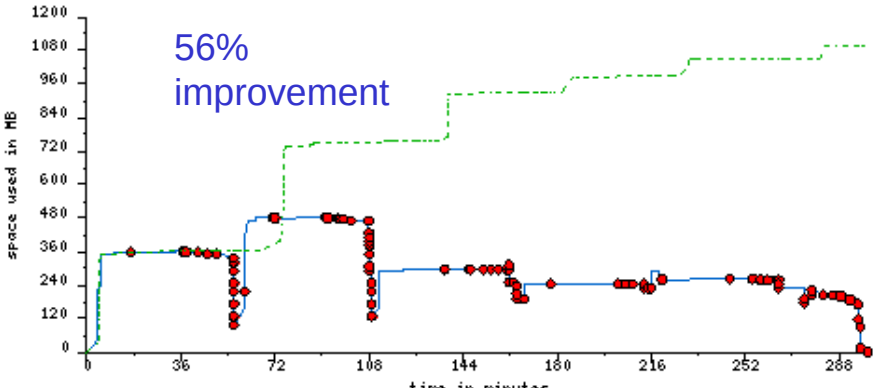


LIGO Workflows

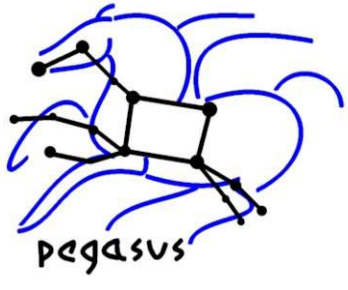


with cleanup cleanup jobs without cleanup





NetLogger Toolkit



- Software tools and a methodology for troubleshooting and performance analysis of complex applications and middleware
  - Normalizes and correlates the flood of log information
  - Uses a relational database back-end to collect distribute information
  - Uses R to produce high-quality plots and analyses

Integration with Netlogger