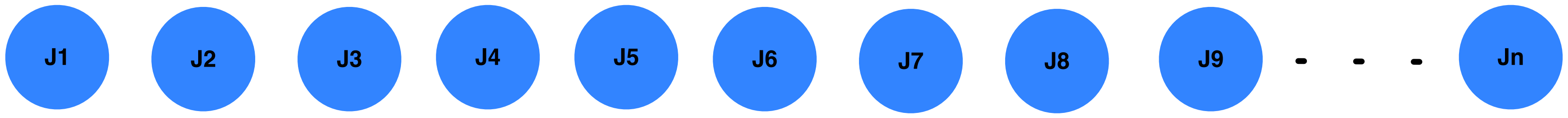
Managing Workloads with Pegasus and DAGMan

Karan Vahi

Science Automation Technologies Group

USC Information Sciences Institute

Workloads – Simple Workflows.



Workloads or Workflows: Users have same concerns!

- **Data Management**
  - 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?

- **Debug and Monitor Workflows**
  - Users need automated tools to go through the log files
  - Need to correlate data across lots of log files
  - Need to know what host a job ran on and how it was invoked

- **Restructure Workflows for Improved Performance**
  - Short running tasks?
  - Data placement?

- **Integrate with higher level tools such as HubZero and provisioning infrastructure**
  - such as GlideinWMS, BOSCO

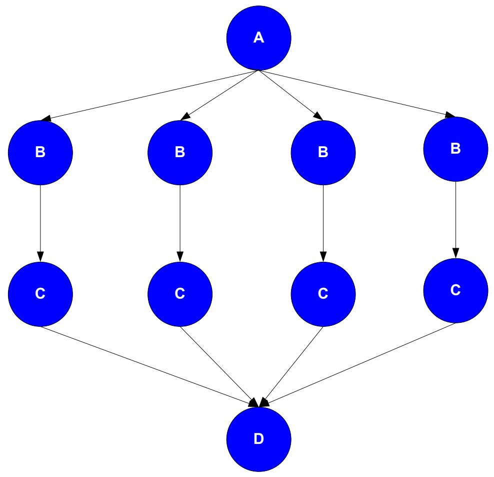
Pegasus Workflow Management System

- **NSF funded project since 2001**
  - Developed as a collaboration between USC Information Sciences Institute and the Condor Team at UW Madison

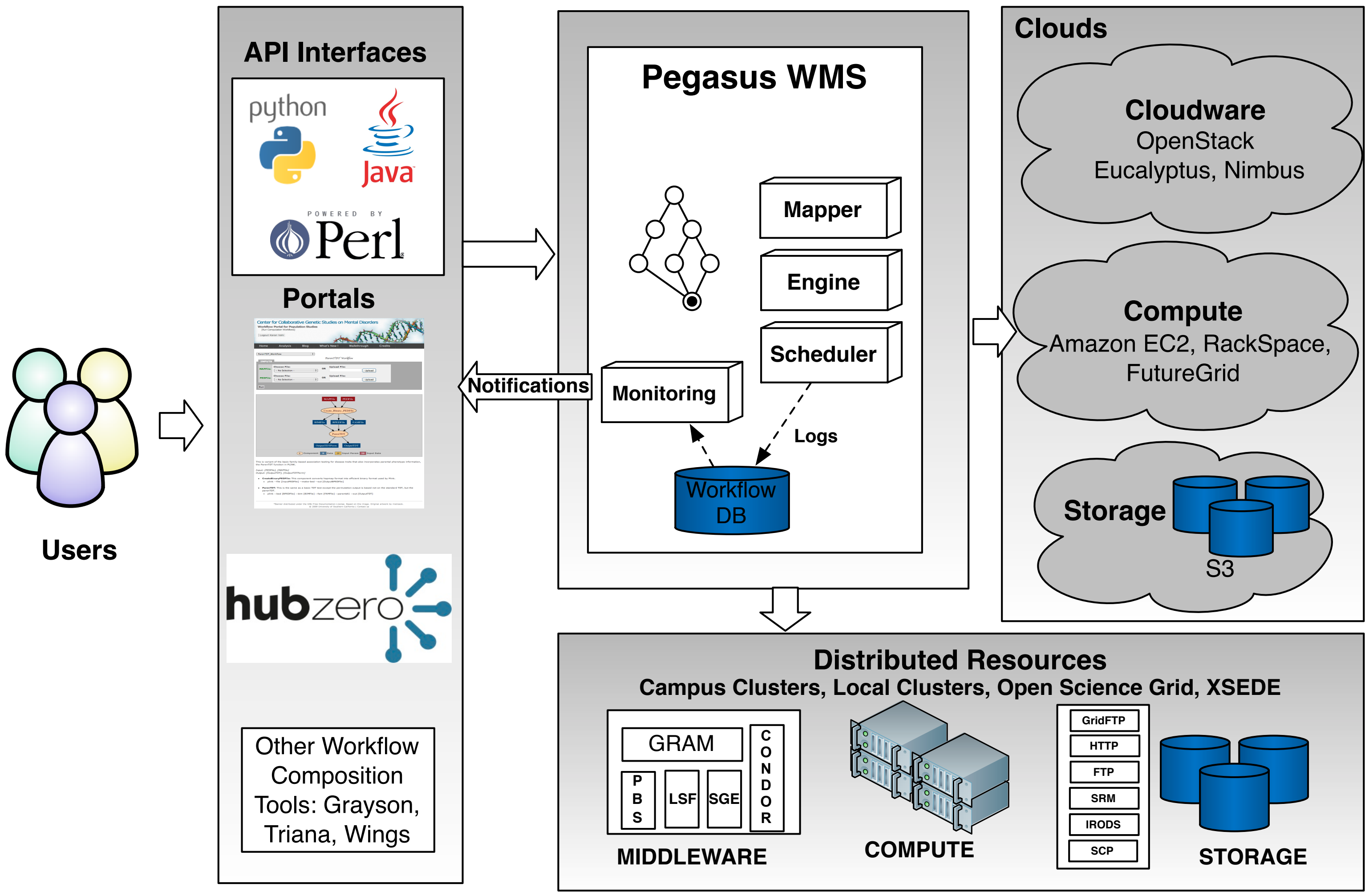
- **Builds on top of Condor DAGMan.**

- **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

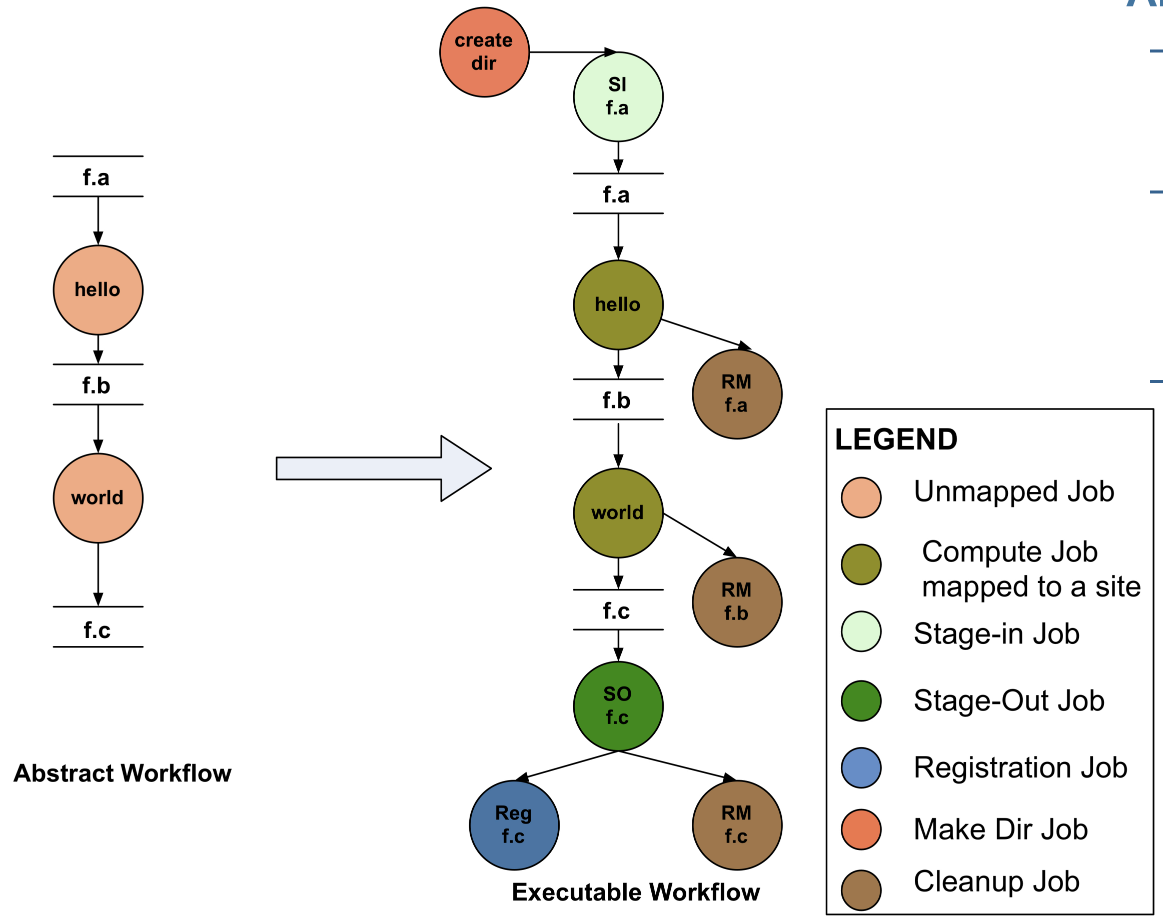


Pegasus WMS



Users
API Interfaces
python
Java
POWERED BY Perl
Portals
hubzero
Other Workflow Composition Tools: Grayson, Triana, Wings
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

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)

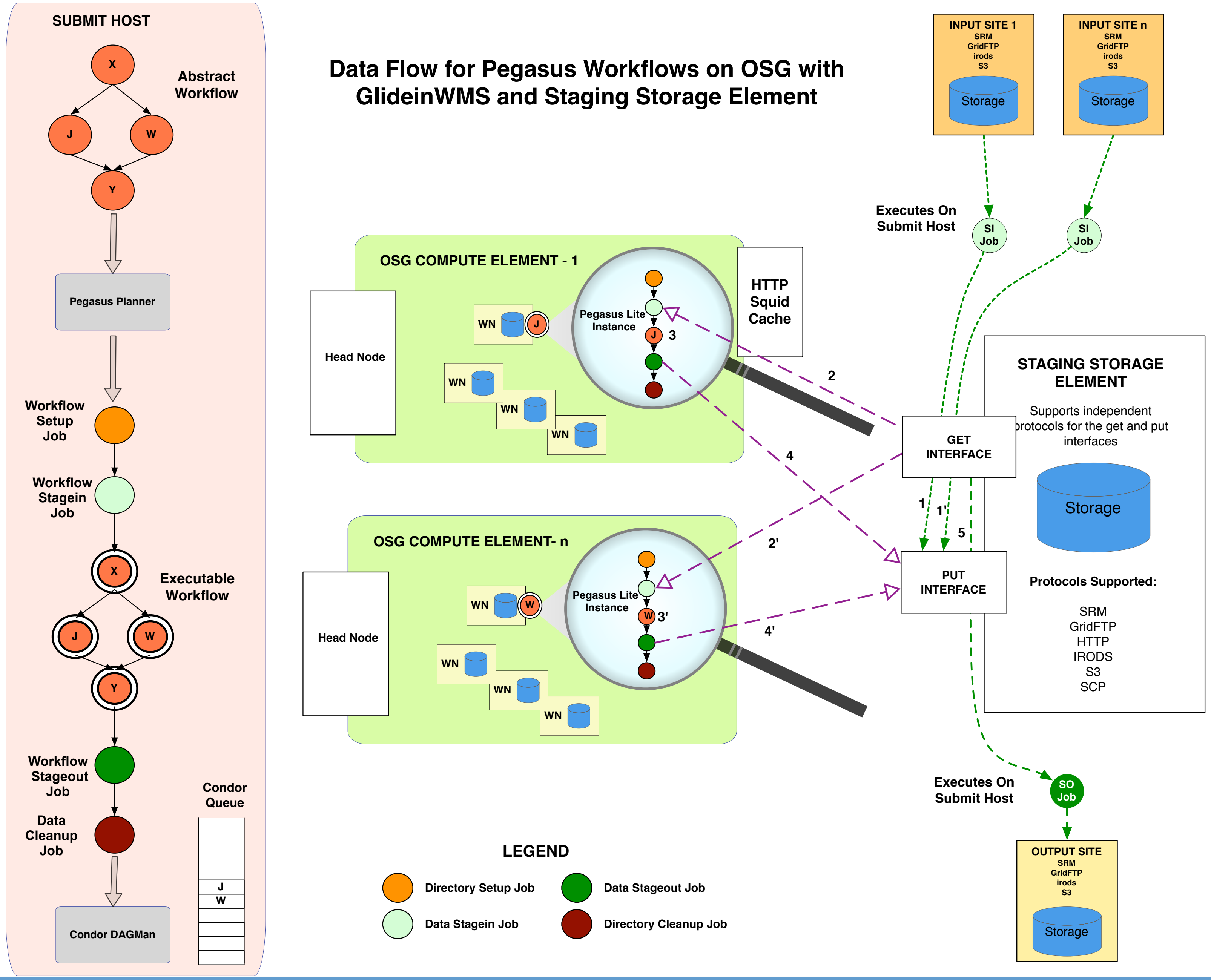
Supported Data Staging Approaches

Three Main Configurations

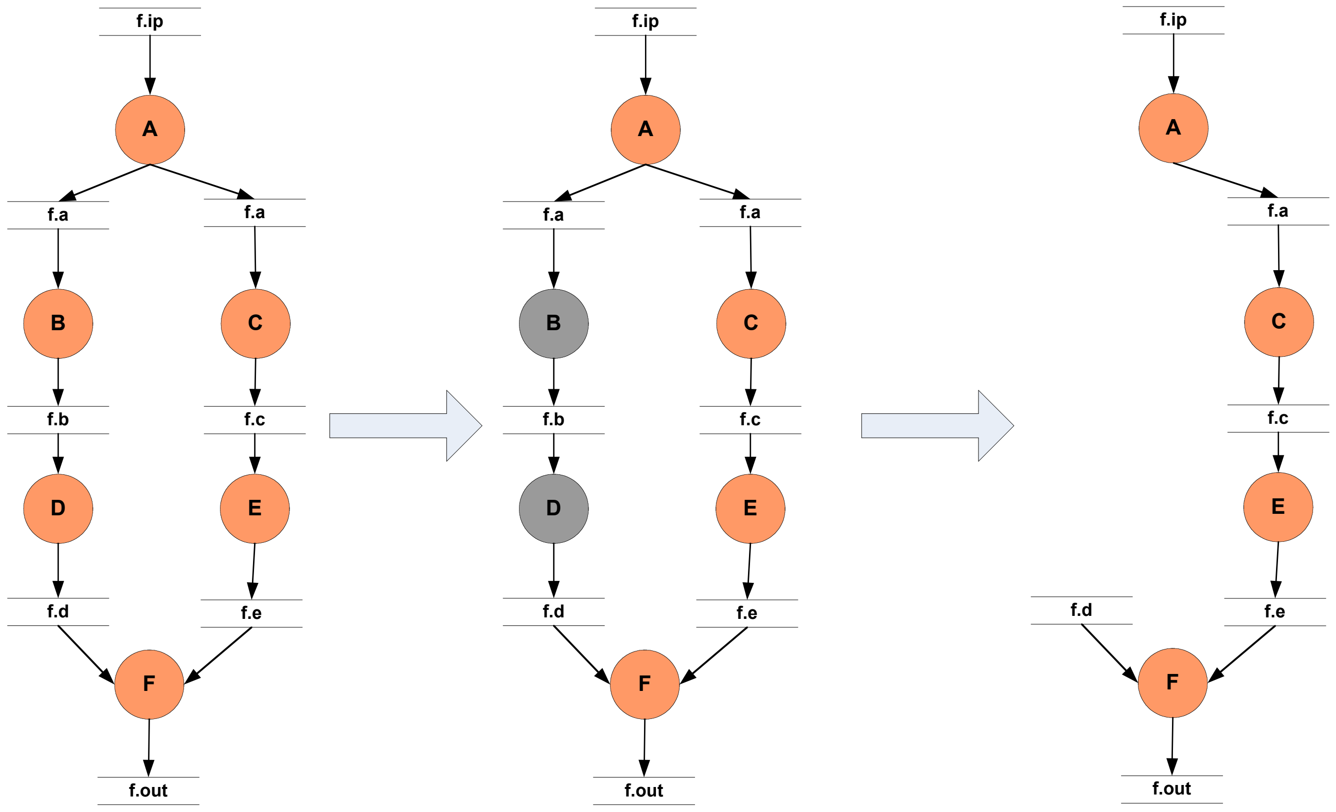
- **Condor IO (Typical of large Condor Pools like CHTC)**
  - Worker nodes don't share a filesystem
  - Symlink against datasets available locally
  - Data is pulled from / pushed to the submit host via Condor file transfers
- **NonShared filesystem setup using an existing storage element for staging (typical of OSG and campus Condor pools)**
  - Worker nodes don't share a filesystem.
  - Data is pulled from / pushed to the existing storage element.
  - (Pictured on the next slide)
- **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

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

Data Flow for Pegasus Workflows on OSG with GlideinWMS and Staging Storage Element



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!

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

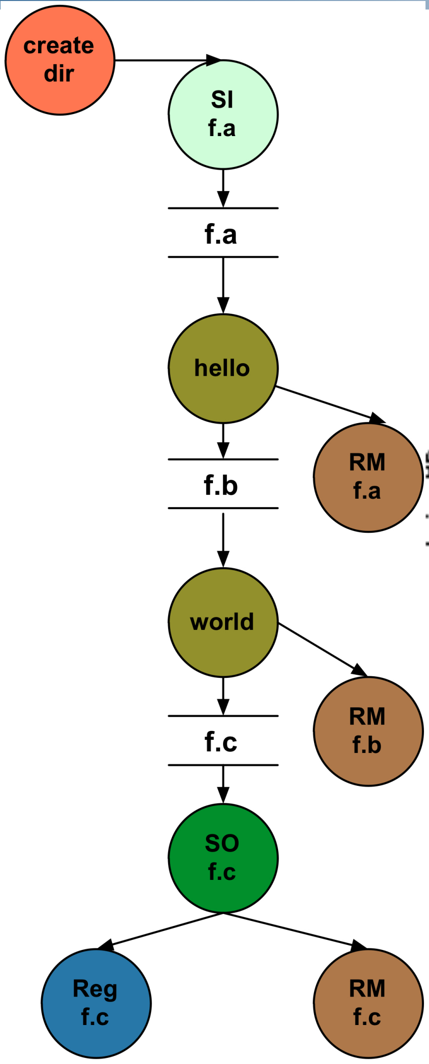
- **Solution**
  - **Do cleanup after workflows finish**
    - Does not work as the scratch may get filled much before during execution

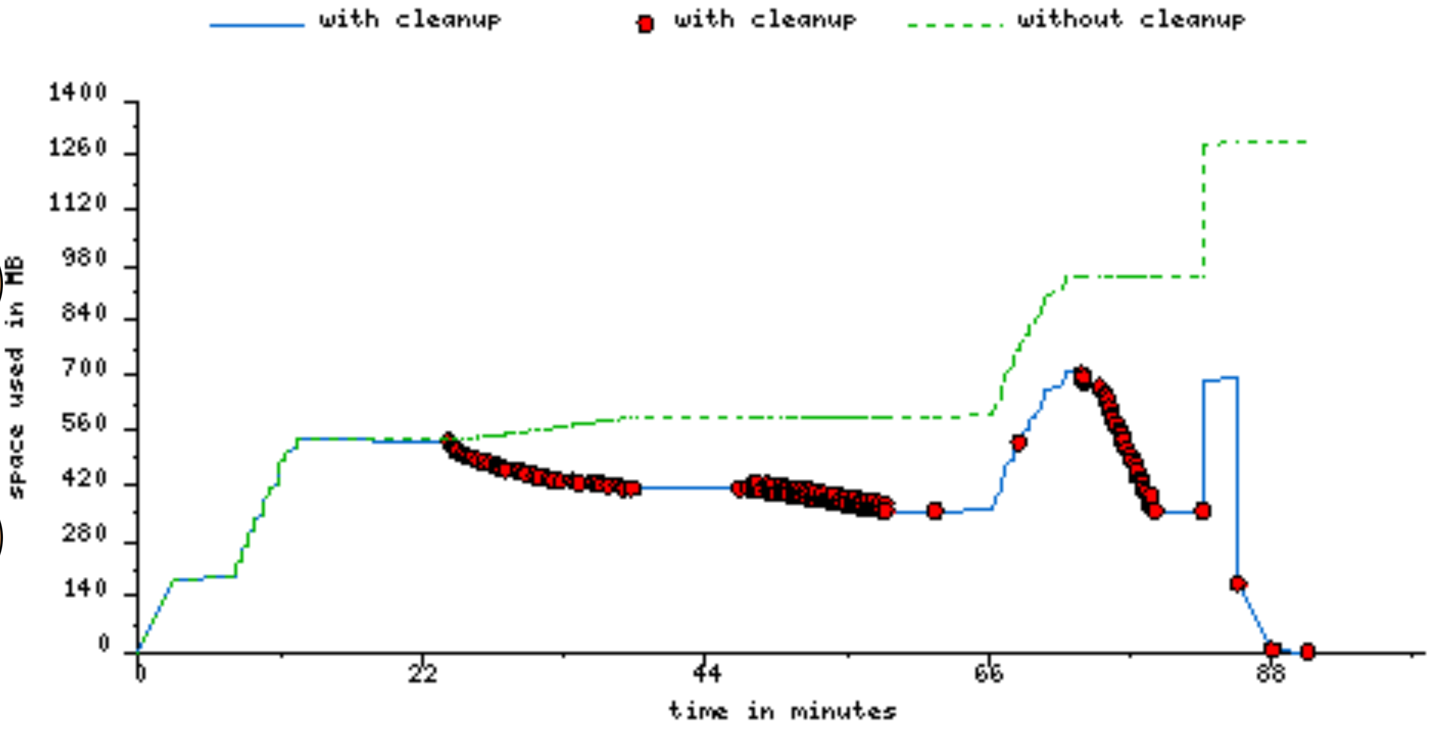
  - **Interleave cleanup automatically during workflow execution.**
    - Requires an analysis of the workflow to determine, when a file is no longer required

  - **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)

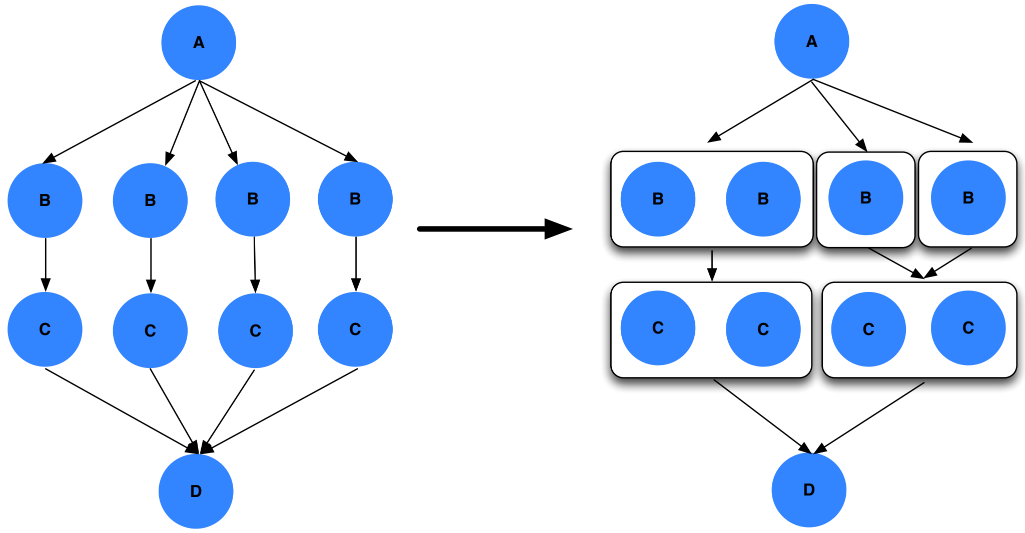




Montage 1 degree workflow run with cleanup

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

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!

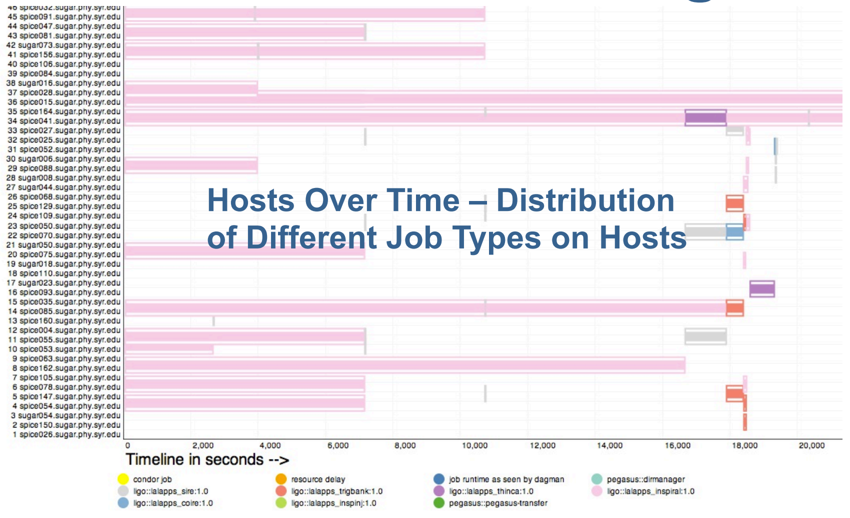
Workflow Monitoring Dashboard: pegasus-dashboard

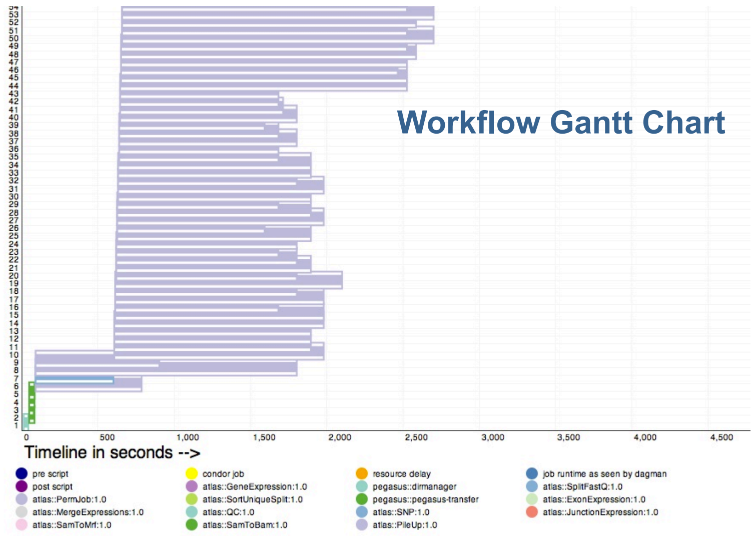
- **A python based online workflow dashboard**
  - Uses the FLASK framework
  - Beta version released in 4.2
  - Queries the STAMPEDE database
- **Lists all the user workflows on the home page and are color coded.**
  - Green indicates a successful workflow,
  - Red indicates a failed workflow
  - Blue 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.

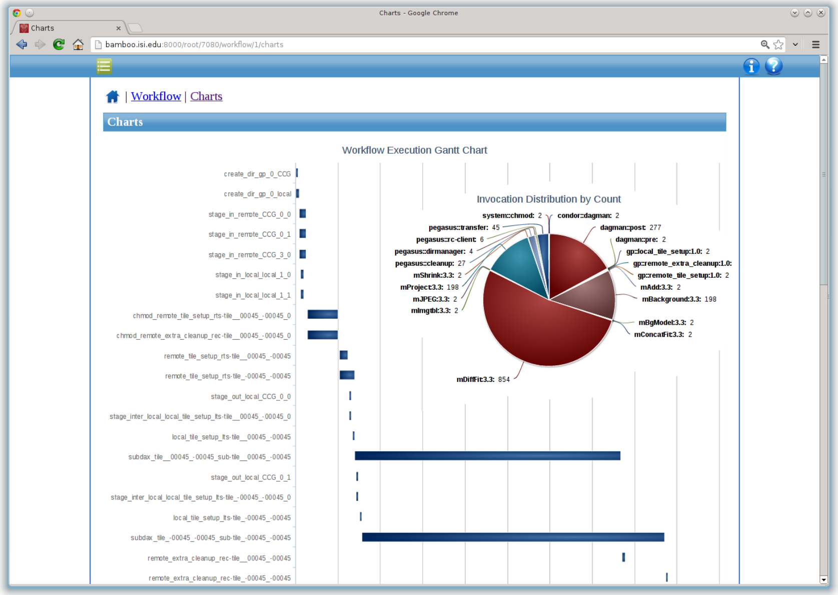
Workflow Monitoring Dashboard: pegasus-dashboard

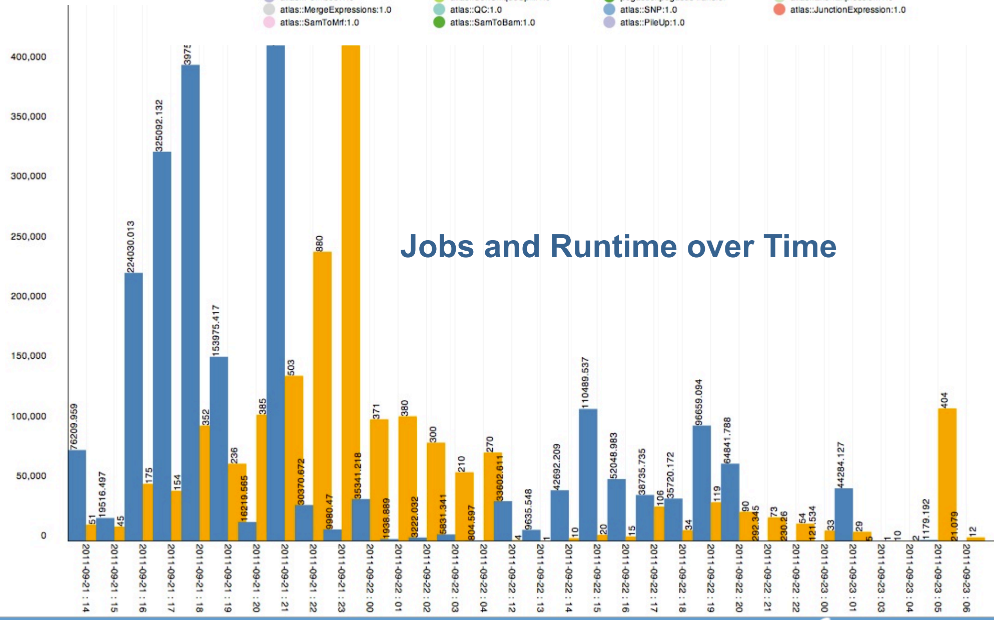
- **Job Page**
  - Lists information captured in kickstart record for the job.
  - Will show the various retries of the job
- **Statistics Page for the Workflow**
  - Generates Statistics for the workflow, similar to pegasus-statistics command line tool
- **Charts Page For the Workflow**
  - Workflow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

Workflow Monitoring Dashboard – pegasus-dashboard


Hosts Over Time – Distribution of Different Job Types on Hosts


Workflow Gantt Chart


Charts - Google Chrome
bamboo.isi.edu:8000/root/7080/workflow/1/charts
Workflow | Charts
Charts
Workflow Execution Gantt Chart
Invocation Distribution by Count

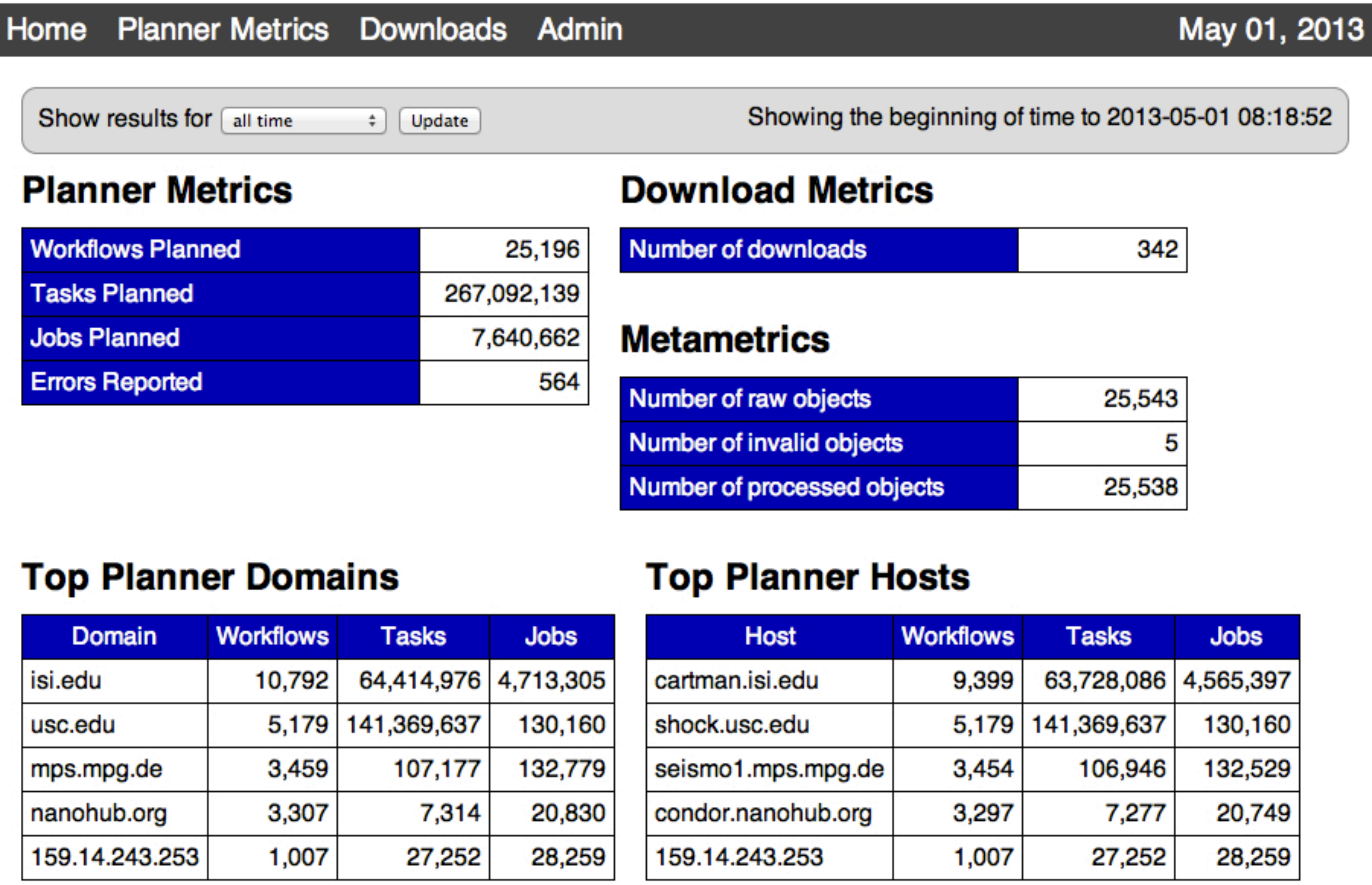

Jobs and Runtime over Time

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

Metrics Collection

- **Why?**
  - A requirement of being funded as part of the NSF SI2 Program
  - Reporting ON by default. Can be turned off.
- **What do we collect?**
  - Anonymous planner metrics
    - Duration of the planner
    - Start and end time
    - Exitcode
    - Breakdown of tasks and jobs in the workflow
  - We leave a copy of the metrics file in the submit directory for the users
- **Capturing Errors**
  - In addition to capturing usage data, the planner also reports back **fatal errors**
  - Using it to drive usability improvements for Pegasus
- http://pegasus.isi.edu/wms/docs/latest/funding_citing_usage.php#usage_statistics


Home Planner Metrics Downloads Admin May 01, 2013

Show results for [all time] Update Showing the beginning of time to 2013-05-01 08:18:52

Planner Metrics

Workflows Planned	25,196
Tasks Planned	267,092,139
Jobs Planned	7,640,662
Errors Reported	564

Download Metrics

Number of downloads	342

Metametrics

Number of raw objects	25,543
Number of invalid objects	5
Number of processed objects	25,538

Top Planner Domains

Domain	Workflows	Tasks	Jobs
isi.edu	10,792	64,414,976	4,713,305
usc.edu	5,179	141,369,637	130,160
mps.mpg.de	3,459	107,177	132,779
nanohub.org	3,307	7,314	20,830
159.14.243.253	1,007	27,252	28,259

Top Planner Hosts

Host	Workflows	Tasks	Jobs
cartman.isi.edu	9,399	63,728,086	4,565,397
shock.usc.edu	5,179	141,369,637	130,160
seismo1.mps.mpg.de	3,454	106,946	132,529
condor.nanohub.org	3,297	7,277	20,749
159.14.243.253	1,007	27,252	28,259

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, Condor Team, funding agencies, NSF, NIH, and everybody who uses Pegasus.