Pegasus 4.2 on the Open Science Grid

Mats Rynge

USC Information Sciences Institute

Workflows on OSG: What do users care about?

- **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?

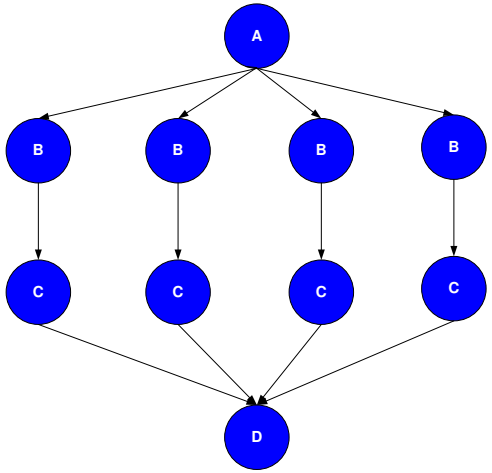
- **Debug and Monitor Workflows**
  - Users need automated tools to go through the log files
  - Need to correlate data across lots of log files
  - Need to know what host a job ran on and how it was invoked

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

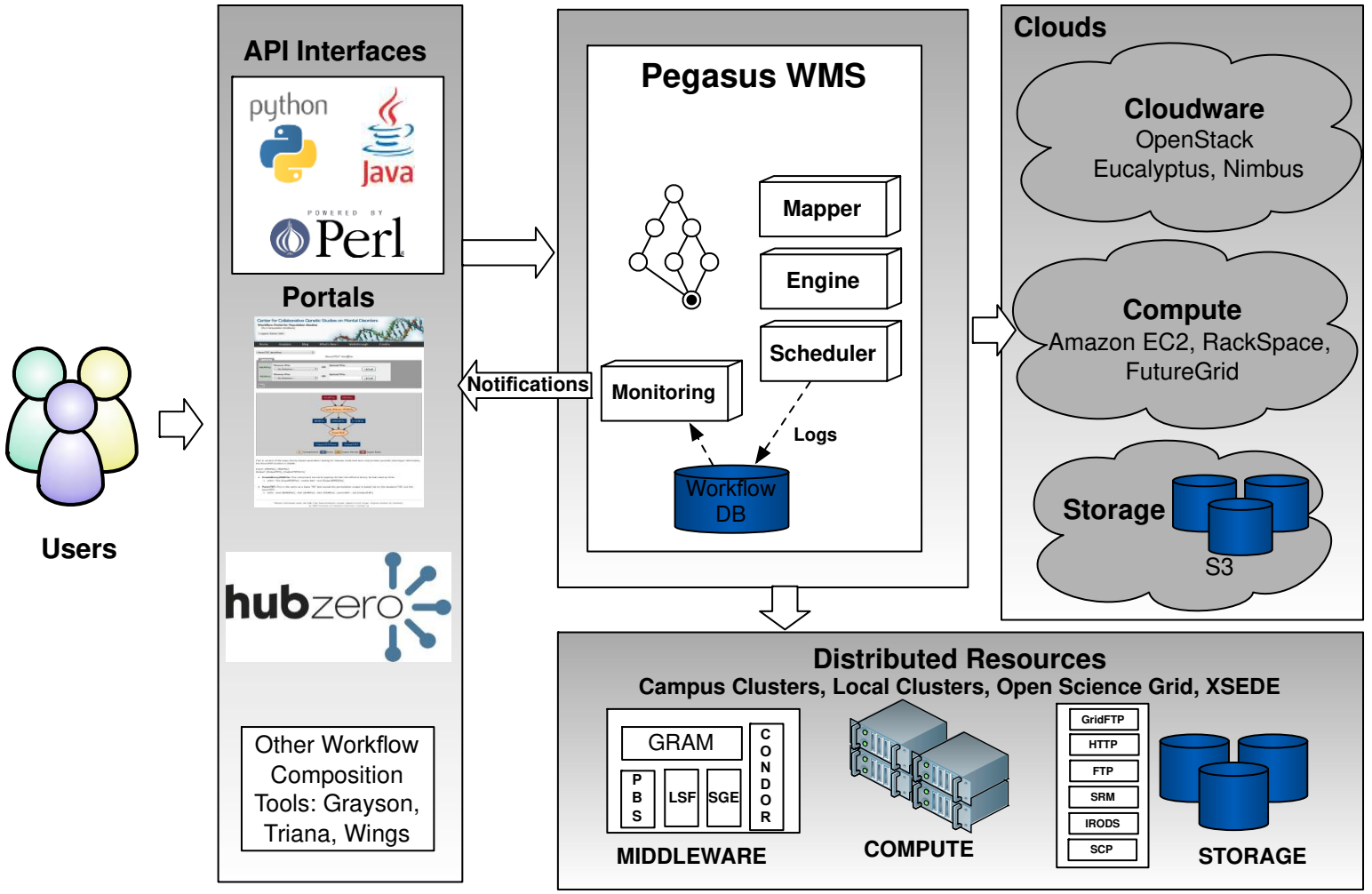
- **Integrate with existing OSG infrastructure for provisioning resources such as GlideinWMS, BOSCO, and higher level tools such as HubZero**

Pegasus Workflow Management System

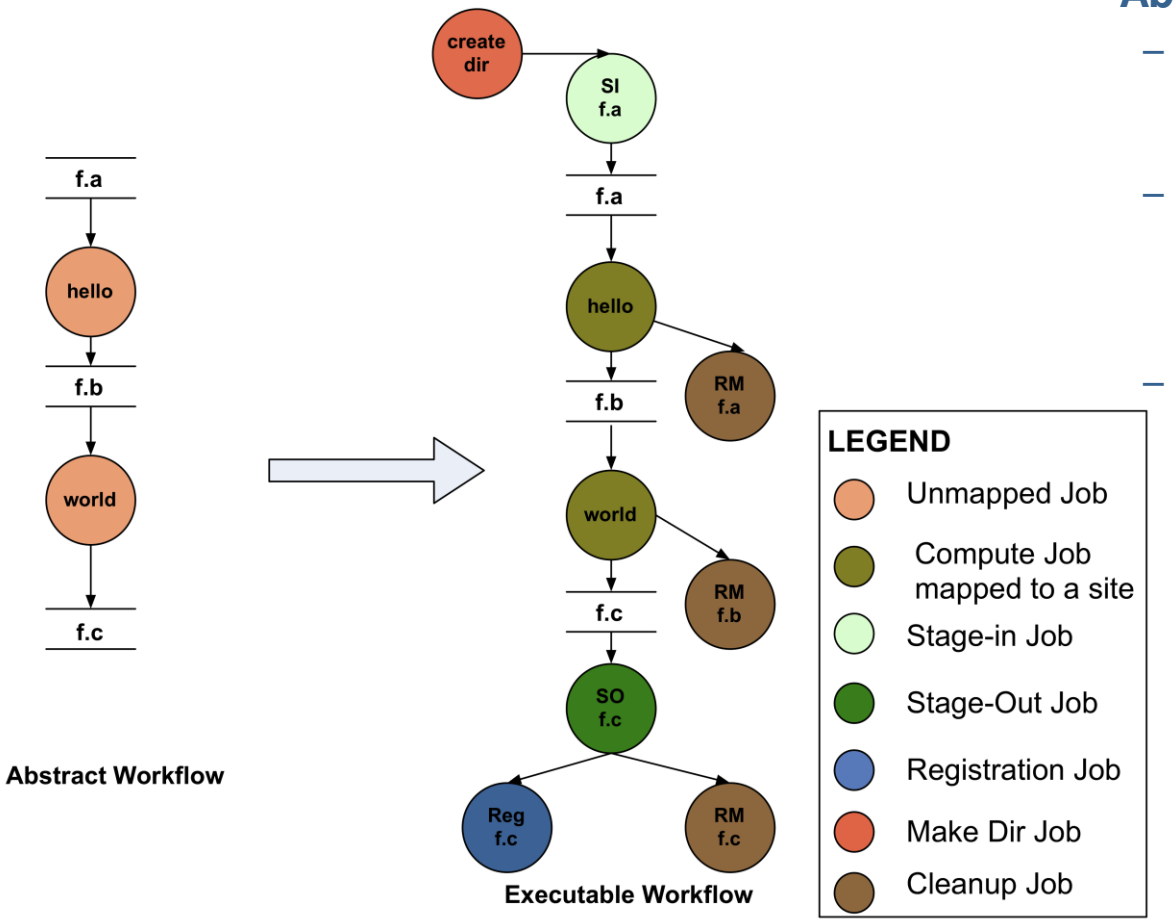
- NSF funded project and developed since 2001 as a collaboration between USC Information Sciences Institute and the Condor Team at UW Madison
- Builds on top of Condor DAGMan.
- Abstract Workflows - Pegasus input workflow description
  - Workflow "high-level language"
  - Only identifies the computation, devoid of resource descriptions, devoid of data locations
- 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


API Interfaces
python
Java
POWERED BY
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

Abstract to Executable Workflow Mapping

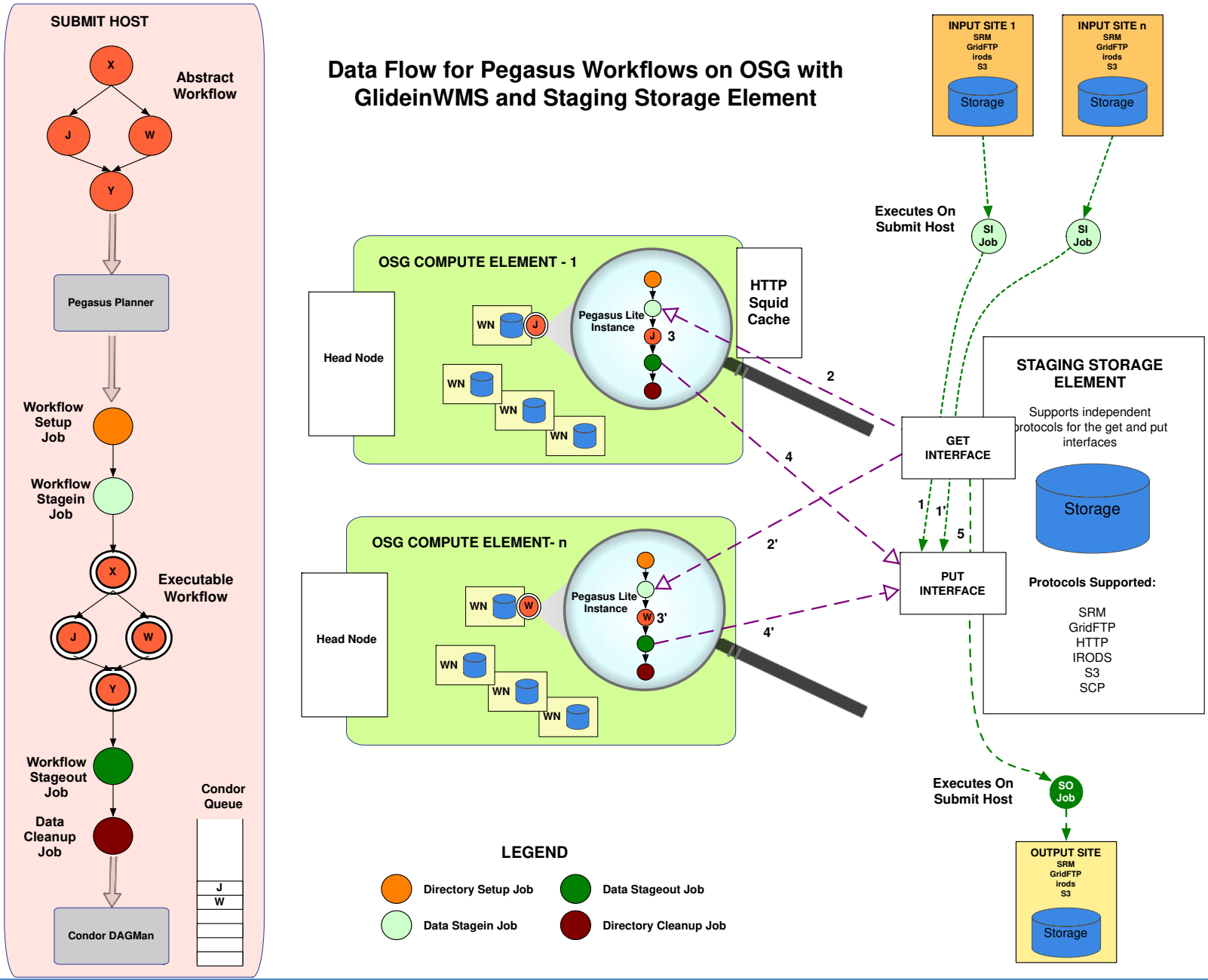


- Abstraction provides
  - Ease of Use (do not need to worry about low-level execution details)
  - Portability (can use the same workflow description to run on a number of resources and/or across them)
  - Gives opportunities for optimization and fault tolerance
    - automatically restructure the workflow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

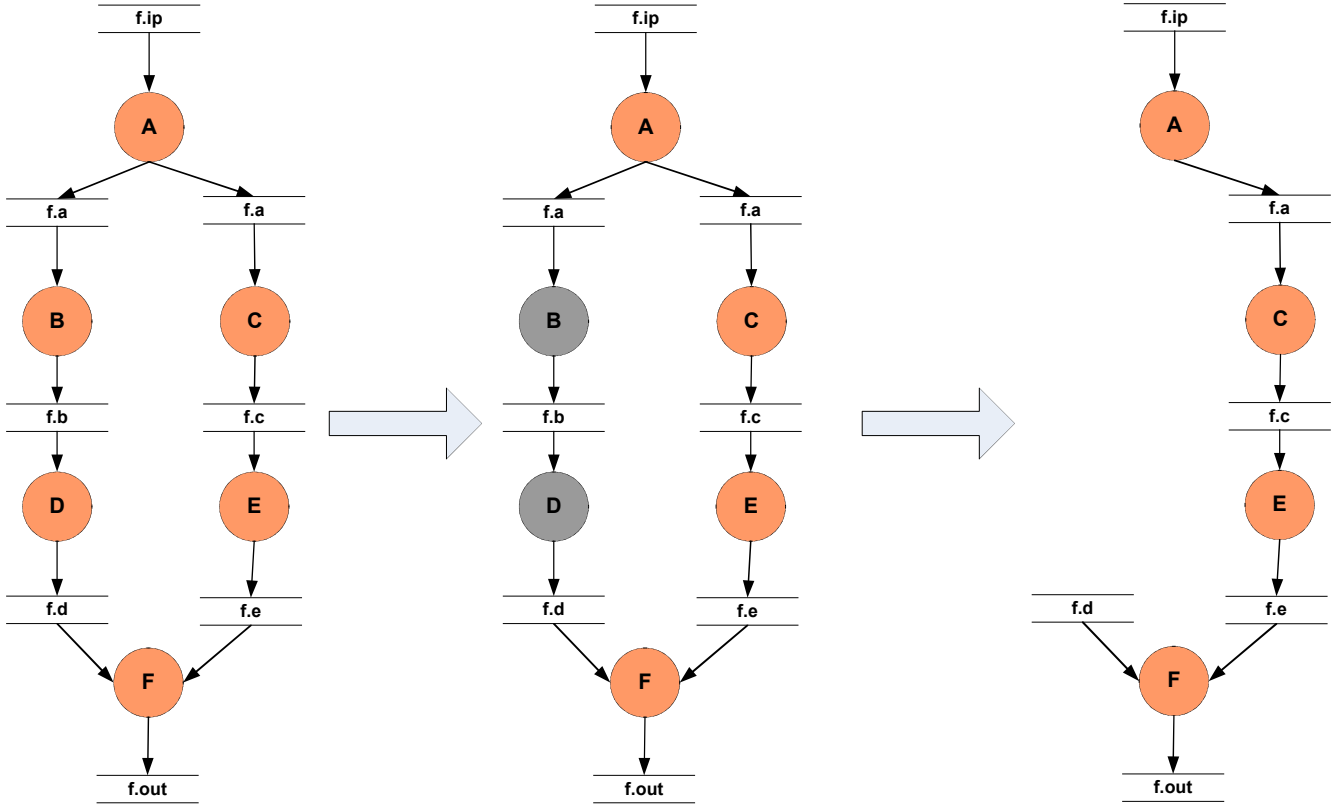
Workflows can be simple

Supported Data Staging Approaches

- 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)
- Condor IO
  - Worker nodes don't share a filesystem
  - Data is pulled from / pushed to the submit host via Condor file transfers
- 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


SUBMIT HOST
X
J
W
Y
Abstract Workflow
Pegasus Planner
Workflow Setup Job
Workflow Stagein Job
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
Pegasus Lite Instance
3
HTTP Squid Cache
2
4
OSG COMPUTE ELEMENT- n
Head Node
WN
W
Pegasus Lite Instance
W
3'
2'
4'
STAGING STORAGE ELEMENT
Supports independent protocols for the get and put interfaces
GET INTERFACE
PUT INTERFACE
1
1'
5
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

Workflow Reduction (Data Reuse)



Abstract Workflow

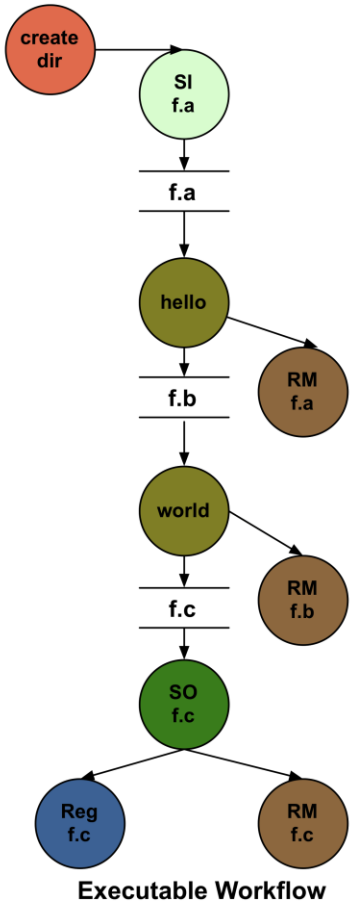
File f.d exists somewhere.
Reuse it.
Mark Jobs D and B to delete

Delete Job D and Job B

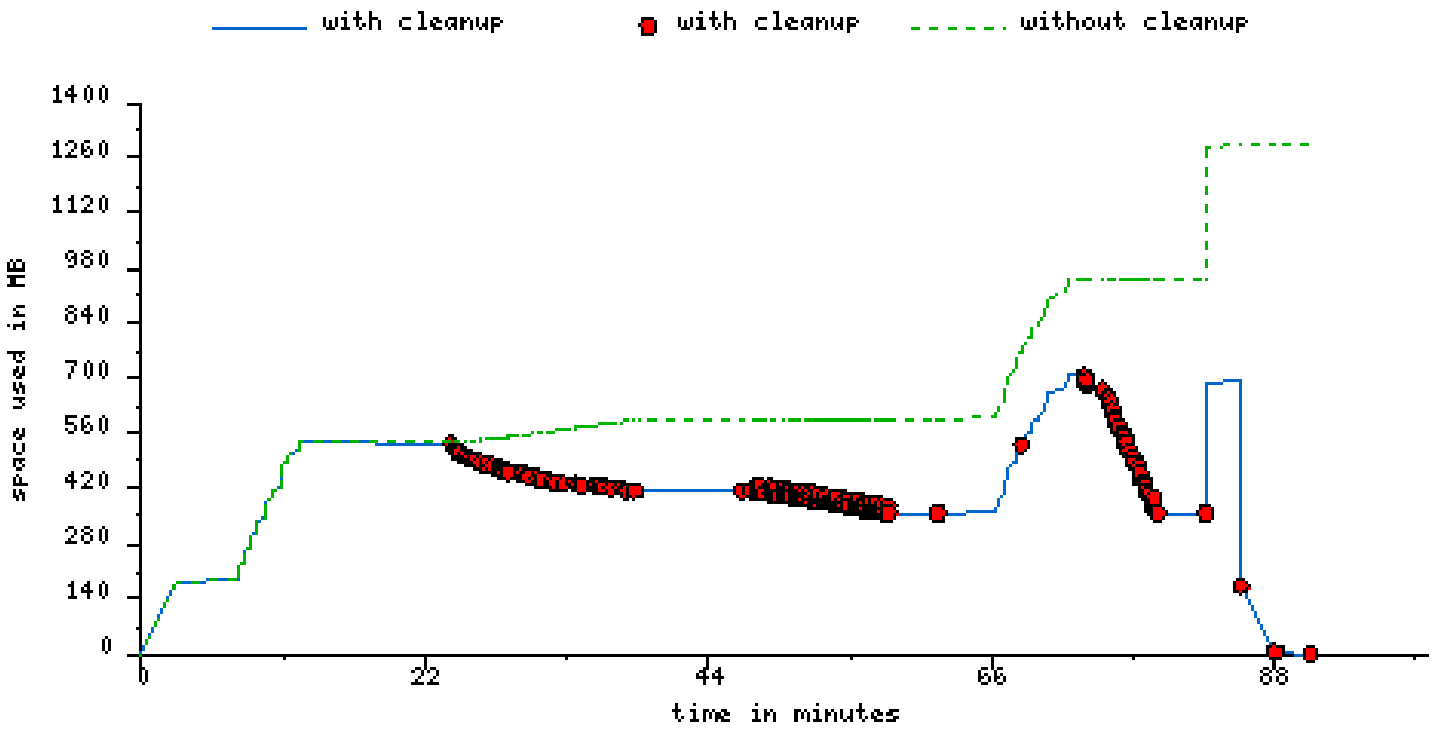
File cleanup

- 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

File cleanup (cont)



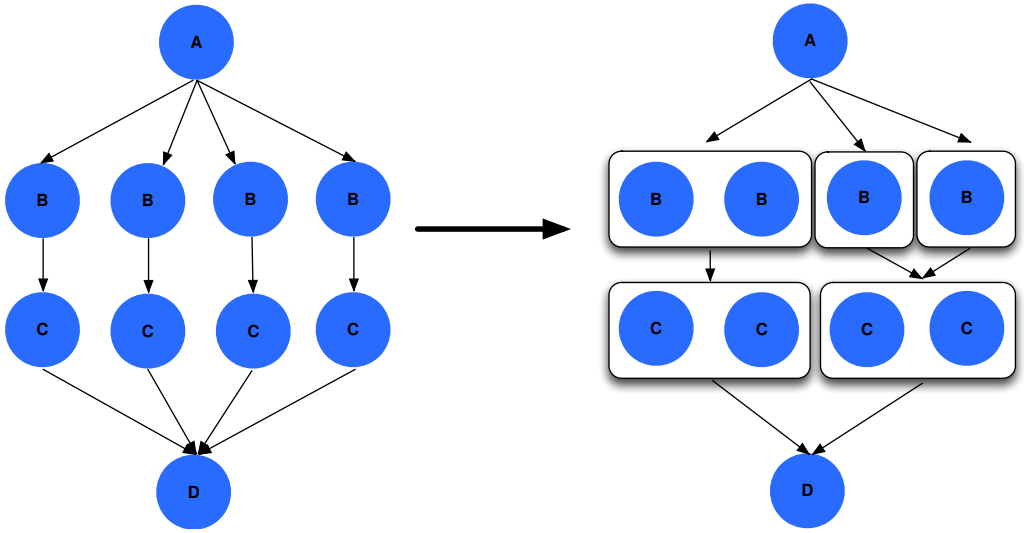
Executable Workflow



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

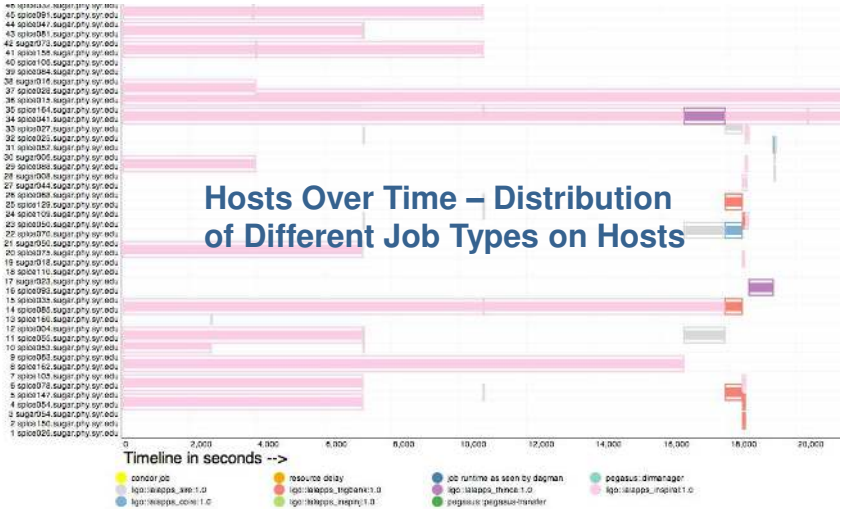
Workflow Monitoring - Stampede

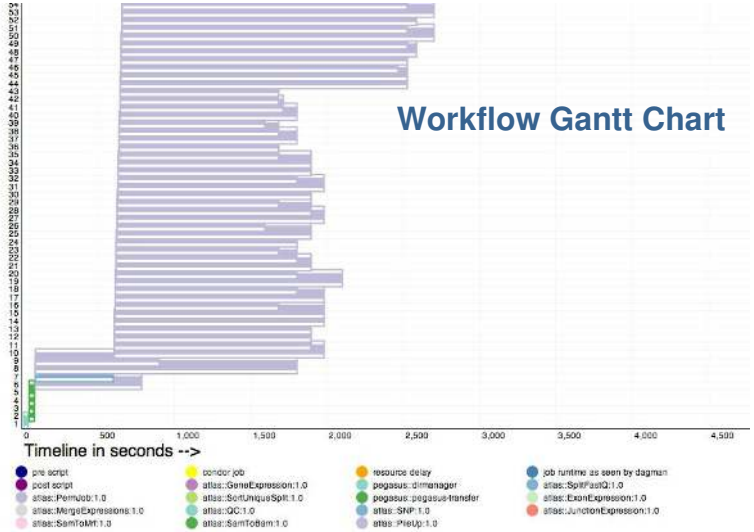
- **Leverage Stampede Monitoring framework with DB backend**
  - Populates data at runtime. A background daemon monitors the logs files and populates information about the workflow to a database
  - Stores workflow structure, and runtime stats for each task.

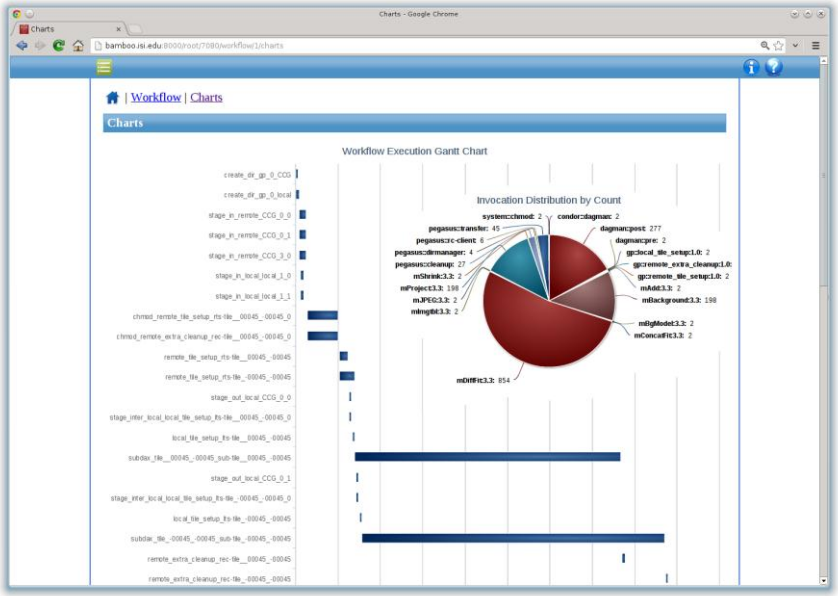
- **Tools for querying the monitoring framework**
  - **pegasus-status**
    - Status of the workflow
  - **pegasus-statistics**
    - Detailed statistics about your finished workflow
  - **pegasus-plots**
    - Visualization of your workflow execution

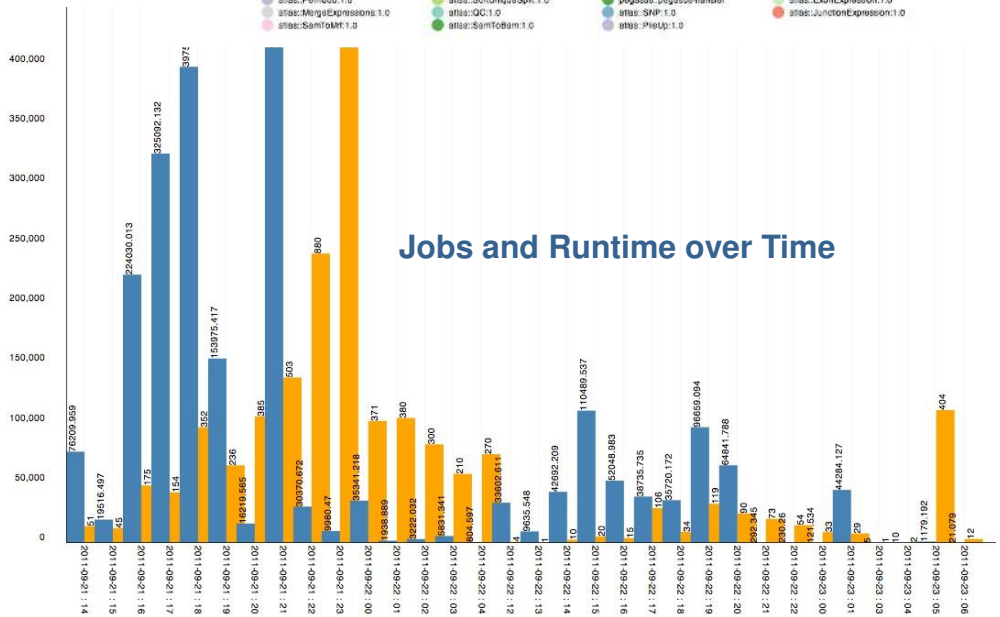
```
------------------------------------------------------------------------------
Type           Succeeded Failed  Incomplete  Total     Retries   Total+Retries
Tasks          135002    0       0           135002    0         135002
Jobs           4529      0       0           4529      0         4529
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 Monitoring - Stampede









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

- 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/**

Acknowledgements

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