



Pegasus WMS – Automated Data Management in Shared and Nonshared Environments

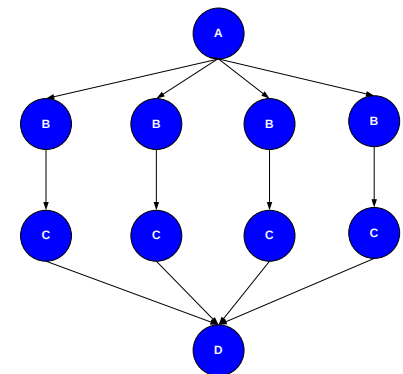
Mats Rynge

<rynge@isi.edu>

USC Information Sciences Institute

Pegasus Workflow Management System

- NSF funded project and developed since 2001 as a collaboration between USC Information Sciences Institute and the HTCondor Team at UW Madison
- Builds on top of HTCondor 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 HTCondor submit files
 - Transforms the workflow for performance and reliability
 - Automatically locates physical locations for both workflow components and data
 - Collects runtime provenance



Abstract Workflow

```
#!/usr/bin/env python

from Pegasus.DAX3 import *
import sys
import os

# Create a abstract dag
dax = ADAG("hello_world")

# Add the hello job
hello = Job(namespace="hello_world",
            name="hello", version="1.0")
b = File("f.b")
hello.uses(a, link=Link.INPUT)
hello.uses(b, link=Link.OUTPUT)
dax.addJob(hello)

# Add the world job (depends on the hello job)
world = Job(namespace="hello_world",
            name="world", version="1.0")
c = File("f.c")
world.uses(b, link=Link.INPUT)
world.uses(c, link=Link.OUTPUT)
dax.addJob(world)

# Add control-flow dependencies
dax.addDependency(Dependency(parent=hello,
                             child=world))

# Write the DAX to stdout
dax.writeXML(sys.stdout)
```



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

<!-- generator: python -->
<adag xmlns="http://pegasus.isi.edu/schema/DAX"
      version="3.4" name="hello_world">

  <!-- describe the jobs making
        up the hello world pipeline -->
  <job id="ID0000001" namespace="hello_world"
      name="hello" version="1.0">

    <uses name="f.b" link="output"/>
    <uses name="f.a" link="input"/>
  </job>

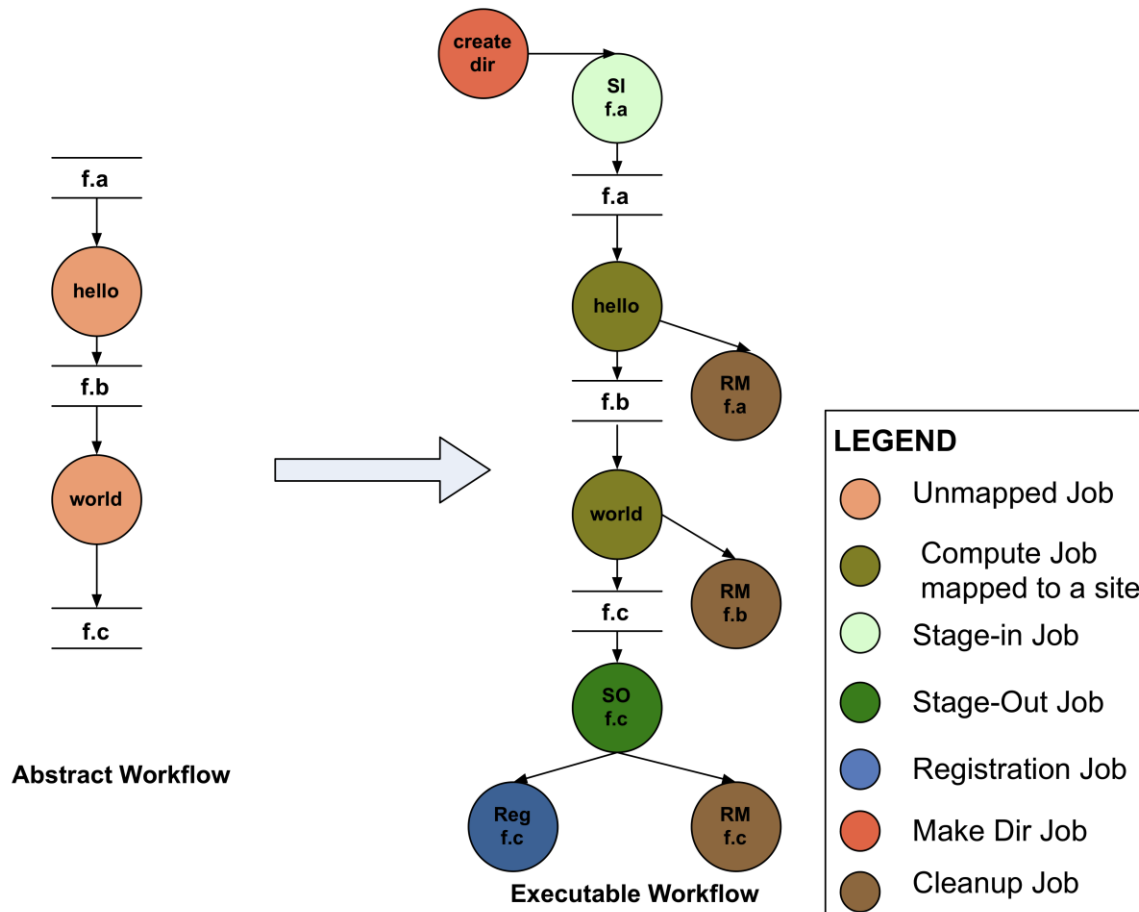
  <job id="ID0000002" namespace="hello_world"
      name="world" version="1.0">

    <uses name="f.b" link="input"/>
    <uses name="f.c" link="output"/>
  </job>

  <!-- describe the edges in the DAG -->
  <child ref="ID0000002">
    <parent ref="ID0000001"/>
  </child>
</adag>
```



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)

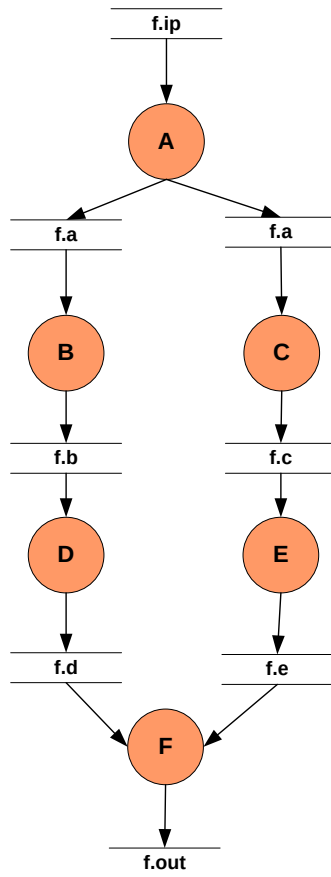


Supported Data Staging Approaches

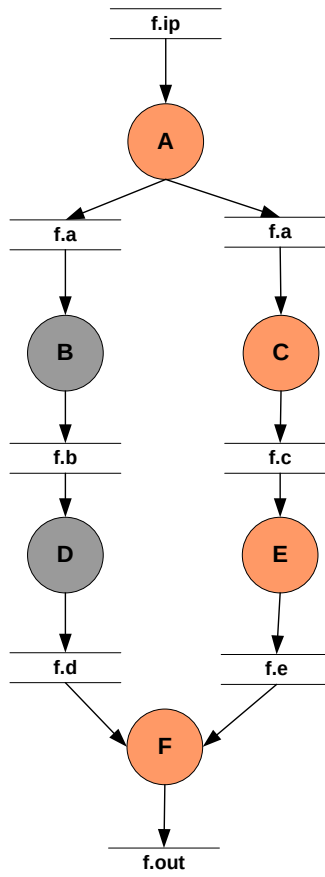
- **Shared Filesystem setup (typical of XSEDE and HPC sites)**
 - Worker nodes and the head node have a shared filesystem, usually a parallel filesystem with great I/O characteristics
- **Condor IO**
 - Worker nodes don't share a filesystem
 - Data is pulled from / pushed to the submit host via Condor 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.



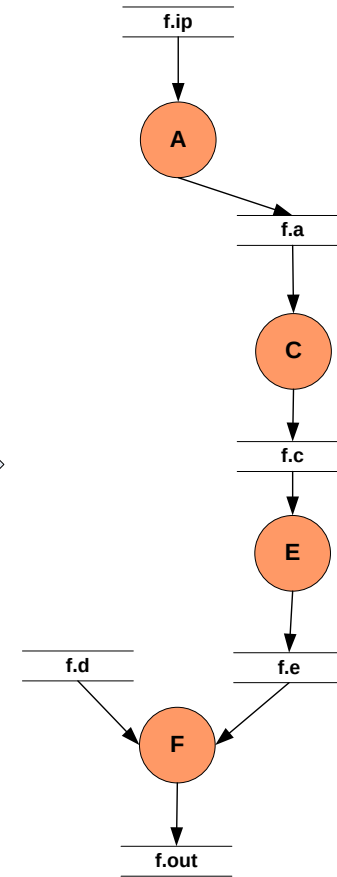
Workflow Reduction (Data Reuse)



Abstract Workflow



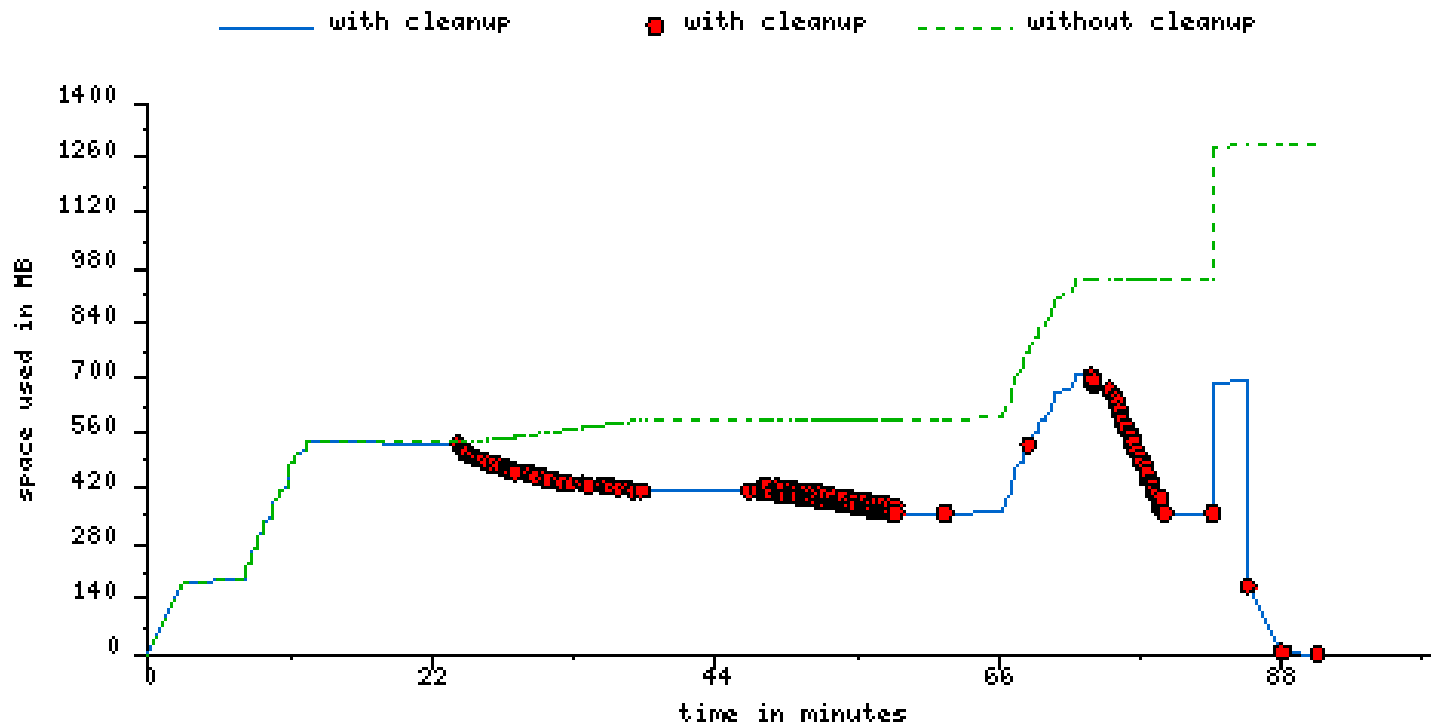
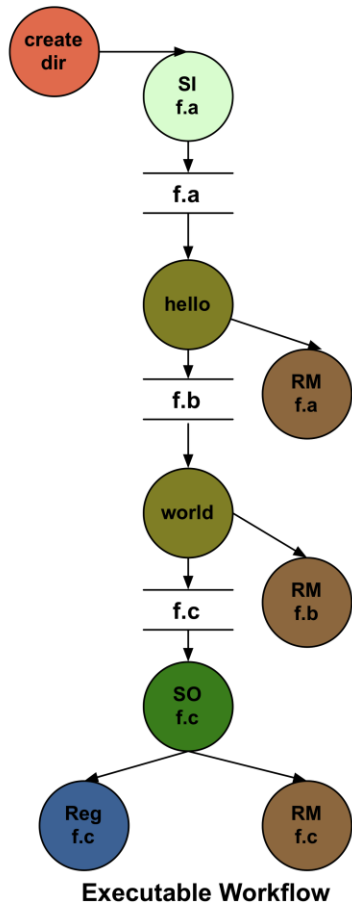
File **f.d** exists somewhere.
Reuse it.
Mark Jobs **D** and **B** to delete



Delete Job **D** and Job **B**

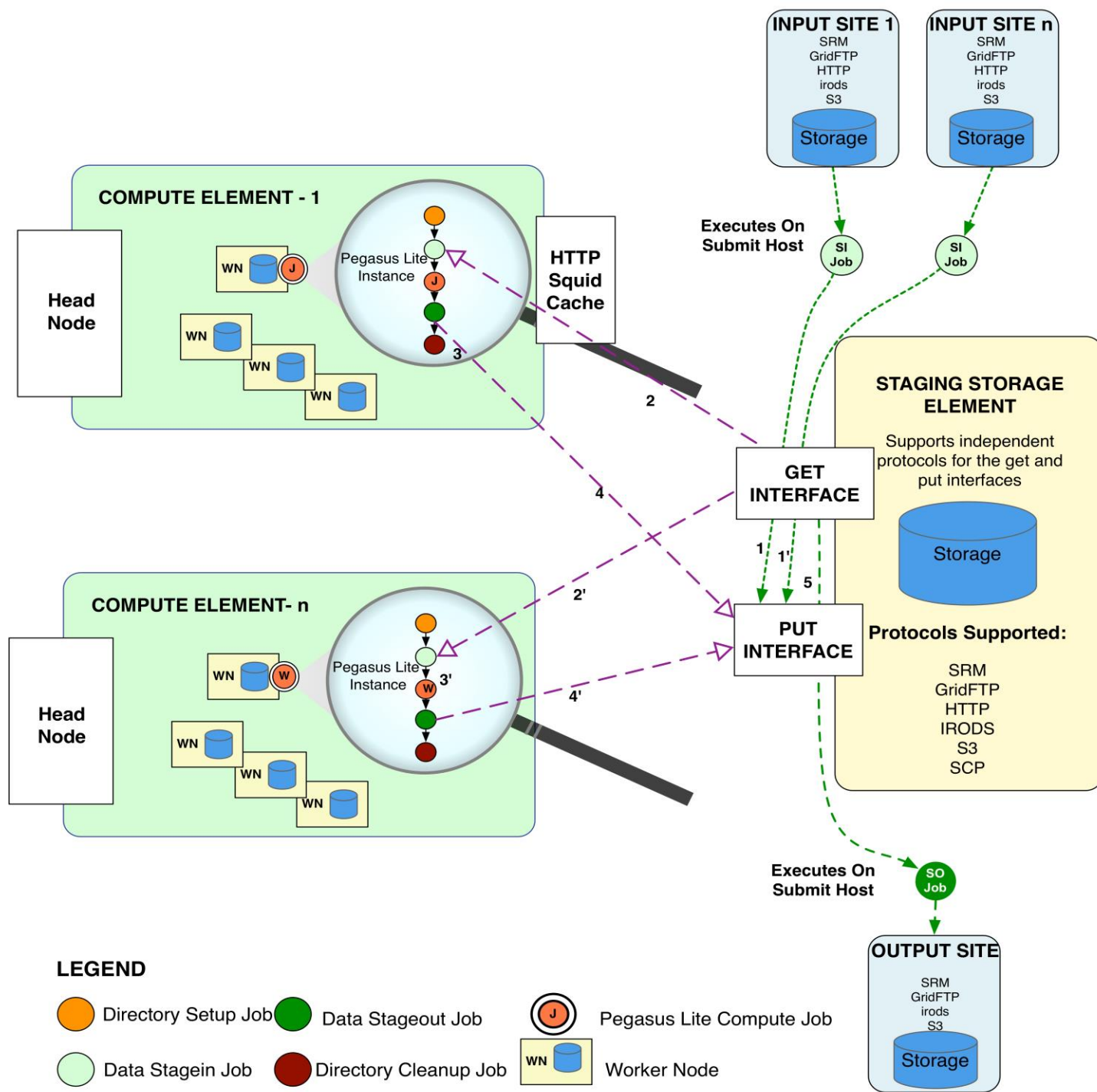
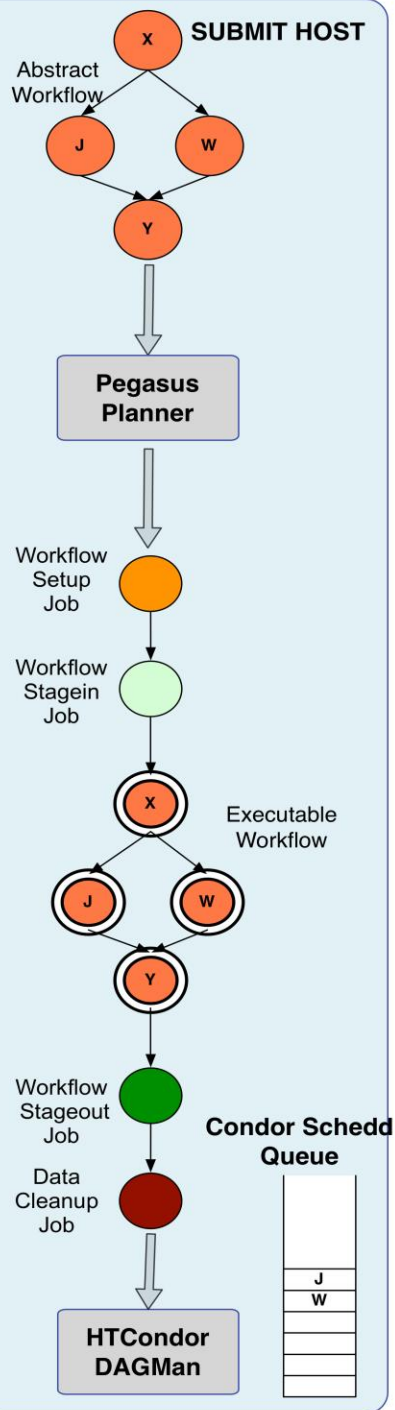


File cleanup (cont)

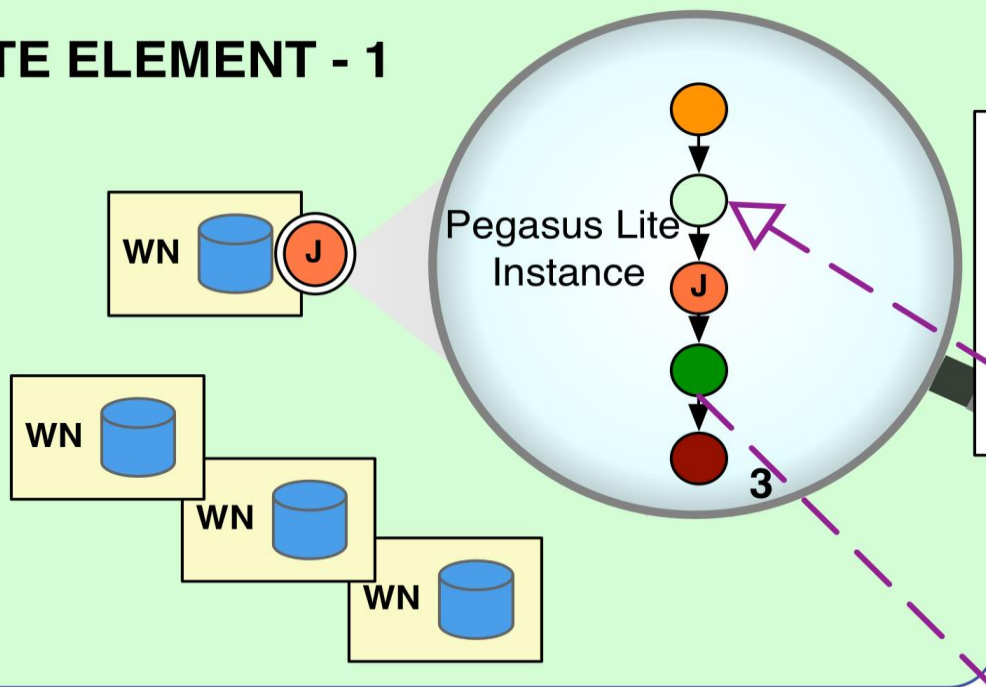


Montage 1 degree workflow run with cleanup





COMPUTE ELEMENT - 1



HTTP
Squid
Cache

2

4

2'

COMPUTE ELEMENT- n



Pegasus Lite

Executes On
Submit Host

SI
Job

GET
INTERFACE

1

1'

5

PUT
INTERFACE

pegasus-transfer subsystem

- Command line tool used internally by Pegasus workflows
- Input is a list of source and destination URLs
- Transfers the data by calling out to tools – provided by the system (cp, wget, ...) Pegasus (pegasus-gridftp, pegasus-s3) or third party (gsutil)
- Transfers are parallelized
- Transfers between non-compatible protocols are split up into two transfers using the local filesystem as a staging point
 - for example: GridFTP->GS becomes GridFTP->File and File->GS

Supported URLs

GridFTP
SRM
iRods
S3
GS
SCP
HTTP
File
Symlink



Relevant Links

<http://pegasus.isi.edu>

Tutorial and documentation:

<http://pegasus.isi.edu/wms/docs/latest/>

Mats Rynge rynge@isi.edu



Catalogs

- Pegasus uses 3 catalogs to fill in the blanks of the abstract workflow
- Site catalog
 - Defines the execution environment and potential data staging resources
 - Simple in the case of Condor pool, but can be more complex when running on grid resources
- Transformation catalog
 - Defines executables used by the workflow
 - Executables can be installed in different locations at different sites
- Replica catalog
 - Locations of existing data products – input files and intermediate files from previous runs

