Pegasus WMS: Enabling Large Scale Workflows on National Cyberinfrastructure

Karan Vahi, Ewa Deelman, Gideon Juve, Mats Rynge, Rajiv Mayani, Rafael Ferreira da Silva
University of Southern California / Information Sciences Institute

USC Viterbi
School of Engineering
Information Sciences Institute

Overview

- Pegasus is a system for mapping and executing abstract application workflows over a range of execution environments.
- The same abstract workflow can, at different times, be mapped different execution environments such as XSEDE, OSG, commercial and academic clouds, campus grids, and clusters.
- 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.
- Pegasus Workflow Management System (WMS) consists of three main components: the Pegasus Mapper, HTCondor DAGMan, and the HTCondor Schedd.
- XSEDE Tutorial https://sites.google.com/site/xsedeworkflows/pegasus-tutorial

HTCondor
High Throughput Computing

Workflow Design and Mapping

```
#!/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)
```

DAX Generator API

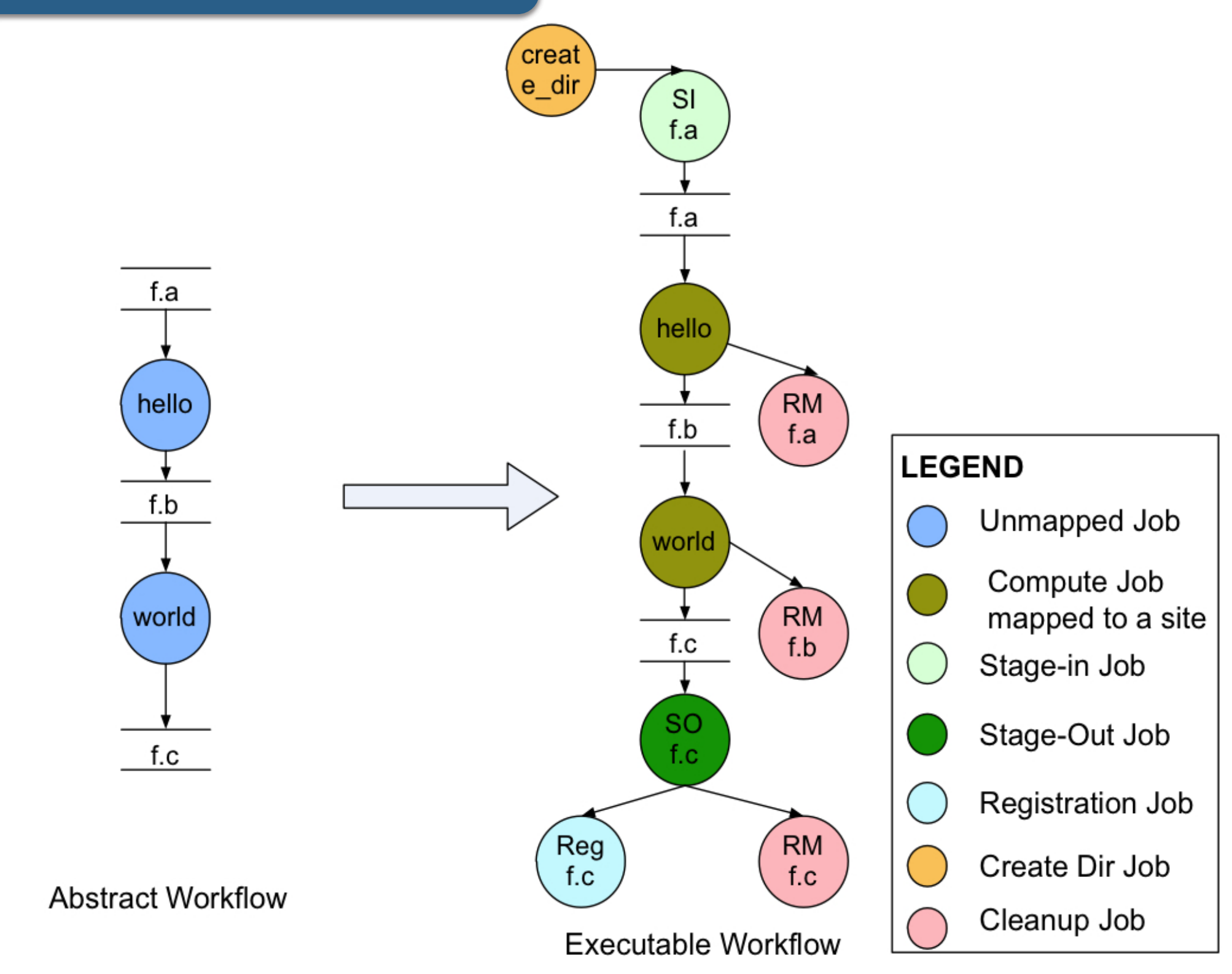
Easy to use APIs in Python, Java and Perl to generate an abstract workflow describing the users computation.

Above is a simple two node hello world example.

```
<?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 Workflow (DAX)

The abstract workflow rendered as XML. It only captures the computations the user wants to do and is devoid of any physical paths. Input and output files are identified by logical identifiers. This representation is portable between different execution environments.



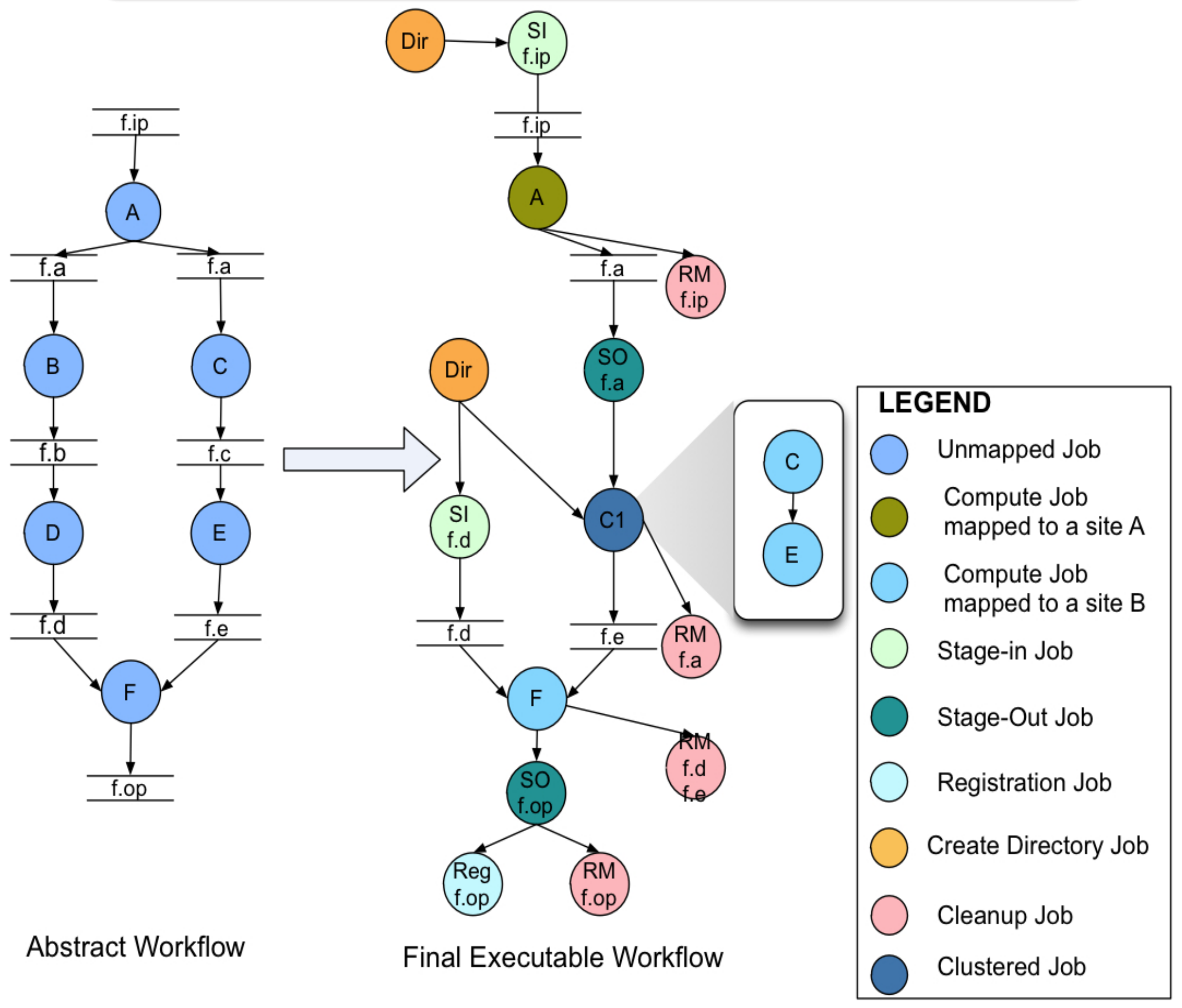
LEGEND	
●	Unmapped Job
●	Compute Job mapped to a site
○	Stage-in Job
●	Stage-Out Job
○	Registration Job
●	Create Dir Job
●	Cleanup Job

Abstract to Executable Workflow (Condor DAG) Mapping

The DAX is passed to the Pegasus Mapper and it generates a **HTCondor DAGMan** workflow that can be run on actual resource.

The above example highlights addition of ***data movement nodes*** to staging in the input data and stage out the output data; addition of ***data cleanup nodes*** to remove data that is no longer required; and ***registration nodes*** to catalog output data locations for future discovery.

Data Reuse Example



LEGEND	
●	Unmapped Job
●	Compute Job mapped to a site A
●	Compute Job mapped to a site B
○	Stage-in Job
●	Stage-Out Job
○	Registration Job
●	Create Directory Job
●	Cleanup Job
●	Clustered Job

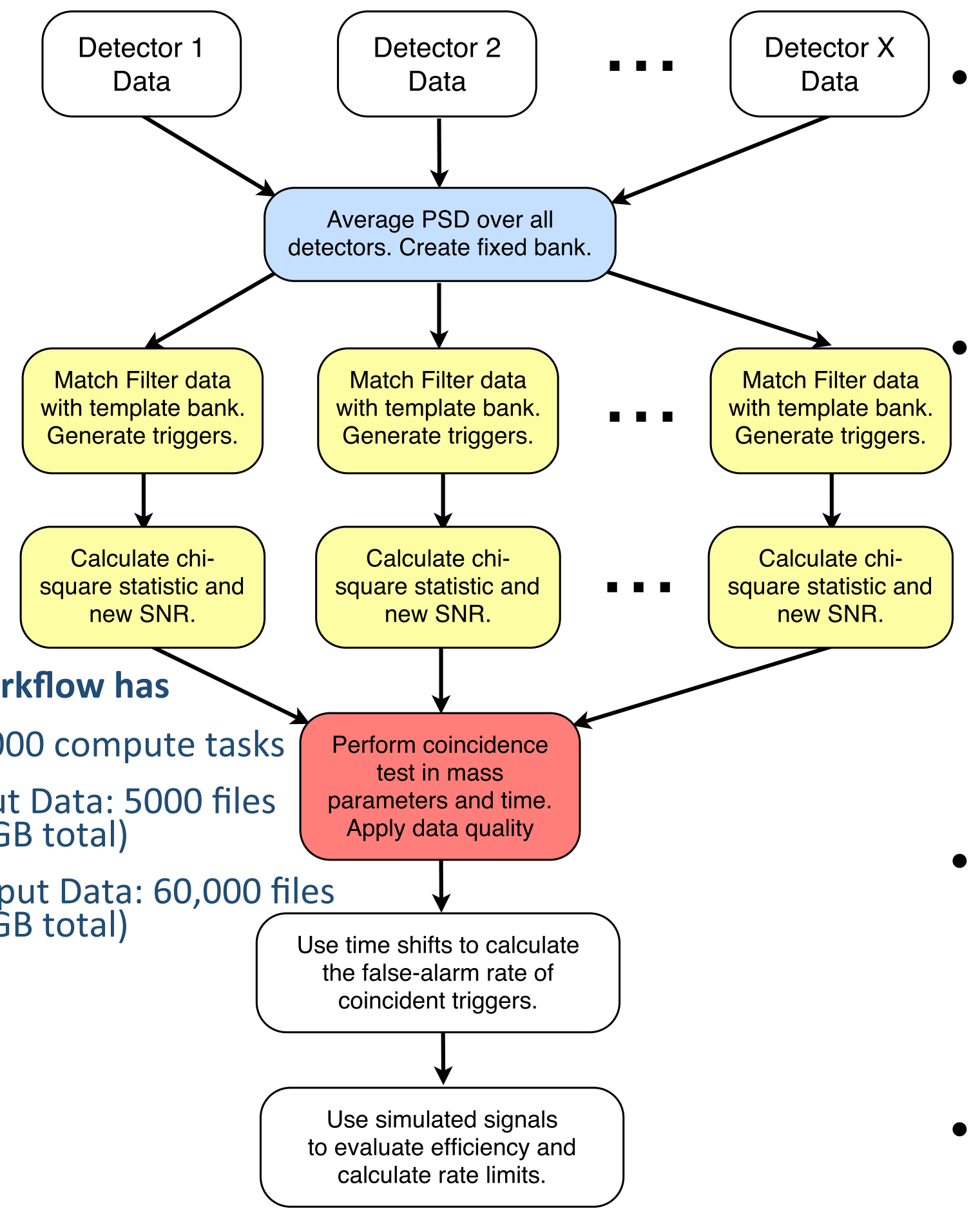
Additional Capabilities Highlighted

Data Reuse: Jobs B and D are removed from the workflow as file f.d already exists. The f.d is staged in , instead of regenerating it by executing jobs B and D.

Job Clustering: Jobs C and E are clustered together into a single clustered job.

Cross Site Run: Single Workflow can be executed on multiple sites, with Pegasus taking care of the data movement between the sites.

Advanced LIGO pyCBC Workflows


Detector 1 Data
Detector 2 Data
. . .
Detector X Data
Average PSD over all detectors. Create fixed bank.
Match Filter data with template bank. Generate triggers.
Match Filter data with template bank. Generate triggers.
. . .
Match Filter data with template bank. Generate triggers.
Calculate chi-square statistic and new SNR.
Calculate chi-square statistic and new SNR.
. . .
Calculate chi-square statistic and new SNR.
Perform coincidence test in mass parameters and time. Apply data quality
Use time shifts to calculate the false-alarm rate of coincident triggers.
Use simulated signals to evaluate efficiency and calculate rate limits.

Each workflow has

- 60,000 compute tasks
- Input Data: 5000 files (10GB total)
- Output Data: 60,000 files (60GB total)

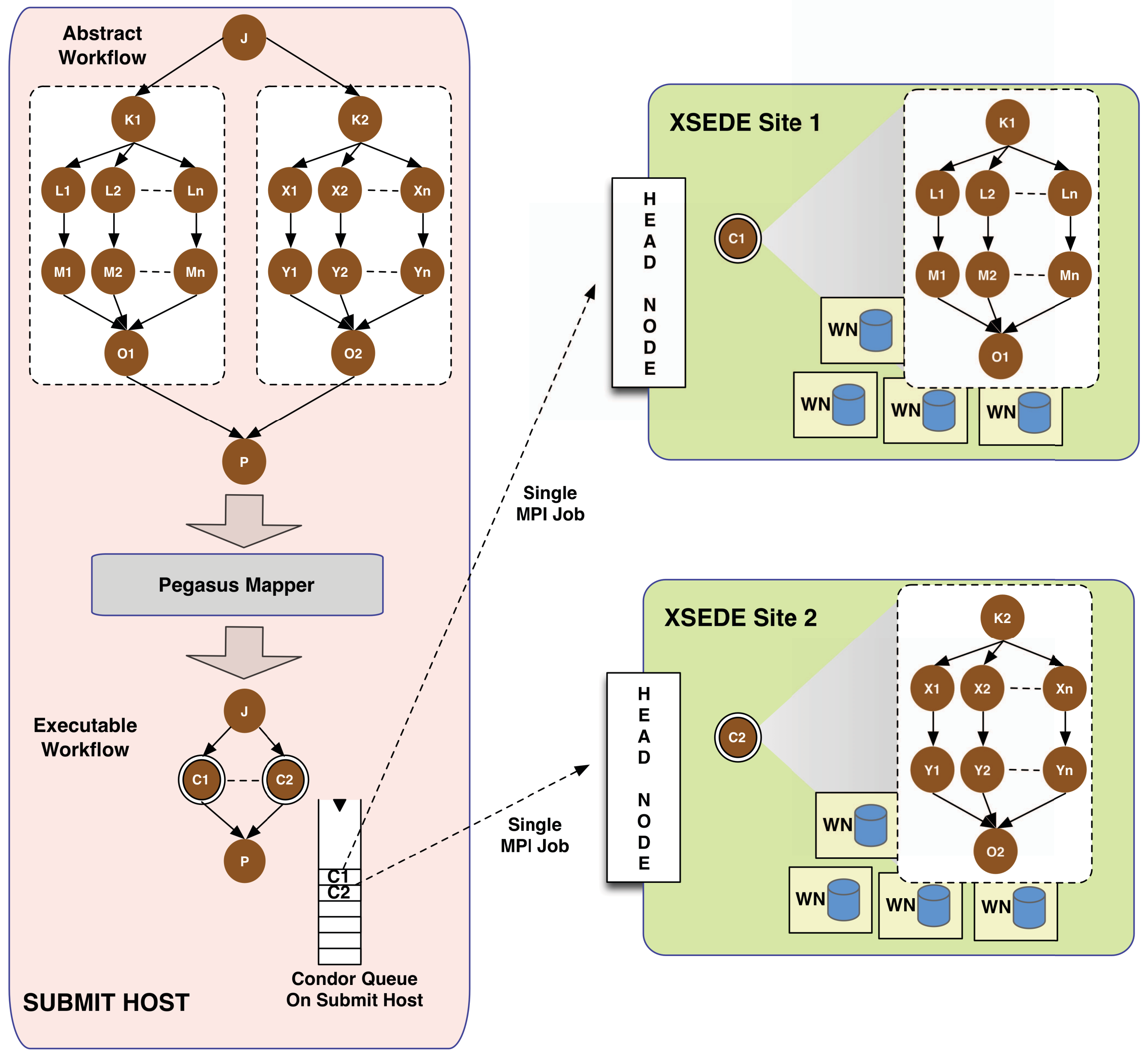
Advanced LIGO pyCBC pipeline

- A single stage ihope pipeline for analyzing data from various LIGO and VIRGO detectors.
- Designed to search for gravitational waves from compact object binaries containing neutron stars and stellar-mass black holes have been performed.
- Actual runs on real data expected to start in September 2015.
- Uses Pegasus WMS to run on XSEDE, LIGO Data Grid and OSG resources.

Test Runs on TACC Stampede with Pegasus:

- Uses Pegasus MPI Cluster to manage parallel FFT jobs into large clusters (using 256, 512 and 1024 cores) submitted to the SLURM batch queue via Globus GRAM.
- Task affinity is set in pegasus-mpi-cluster so that the threads for the FFTs for a single job stay pinned to a single processor to obtain optimal user of the CPU's L3 cache during execution of the FFT.
- Pegasus stages the outputs back to LIGO Data Grid for post processing.

Pegasus Workflows with PMC on XSEDE



XSEDE Challenges

- HPC resources with remote job submission via GRAM, with strict queue limits per user (=~50)
- Workflows can have a large number of tasks that cannot be all submitted through remote job submission interface

Solution:

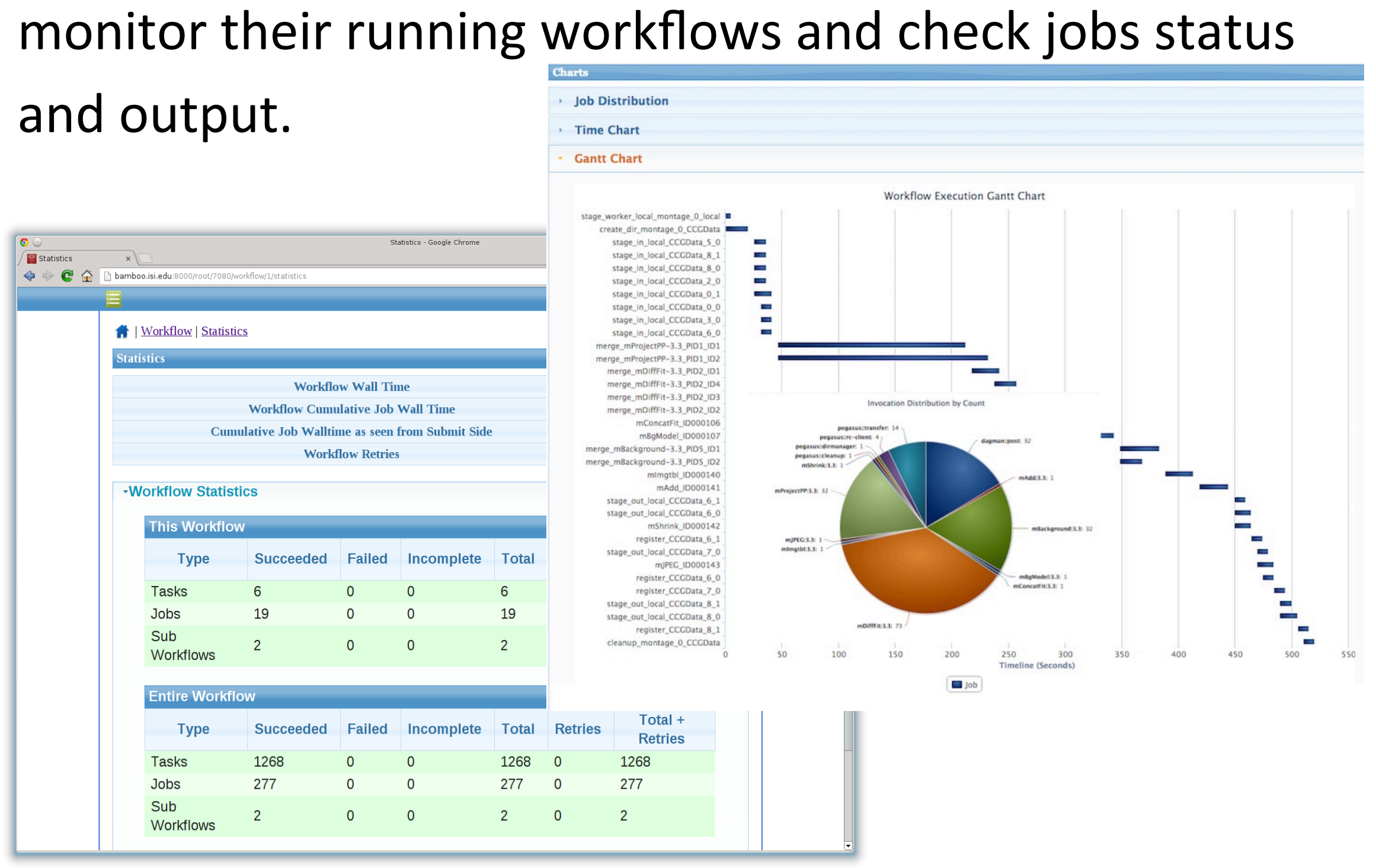
- The workflow is partitioned into independent sub graphs, which are submitted as self-contained Pegasus MPI Cluster (PMC) jobs to the remote sites.
- PMC relies on standard MPI constructs.

A PMC job is expressed as a DAG and PMC uses the MPI master-worker paradigm to farm out individual tasks to worker nodes. PMC acts a scheduler and considers core and memory requirements of the tasks when making scheduling decisions. PMC can be easier to setup than pilot jobs / HTCondor glideins as no special networking is required.

Monitoring and Debugging

At runtime, a database is populated with workflow and task runtime provenance, including which software was used and with what parameters, execution environment, runtime statistics and exit status.

Pegasus comes with command line monitoring and debugging tools. A web dashboard now allows users to monitor their running workflows and check jobs status and output.


Charts
Job Distribution
Time Chart
Gantt Chart
Workflow Execution Gantt Chart
Invocation Distribution by Count
Timeline (Seconds)
Job
Statistics - Google Chrome
Statistics
bamboo.isi.edu:8000/root/7080/workflow/1/statistics
Workflow | Statistics

Statistics

Workflow Wall Time
Workflow Cumulative Job Wall Time
Cumulative Job Walltime as seen from Submit Side
Workflow Retries

Workflow Statistics

This Workflow

Type	Succeeded	Failed	Incomplete	Total
Tasks	6	0	0	6
Jobs	19	0	0	19
Sub Workflows	2	0	0	2

Entire Workflow

Type	Succeeded	Failed	Incomplete	Total	Retries	Total + Retries
Tasks	1268	0	0	1268	0	1268
Jobs	277	0	0	277	0	277
Sub Workflows	2	0	0	2	0	2

Acknowledgments:

- Pegasus WMS is funded by the National Science Foundation OCI SDCI program grant #1148515.
- HTCondor : Miron Livny, Kent Wenger, University of Wisconsin Madison
- LIGO: Larne Pekowsky , Duncan Brown, Alexander H. Nitz and others from LIGO CBC group

http://pegasus.isi.edu

USC Viterbi School of Engineering

The University of Wisconsin Madison