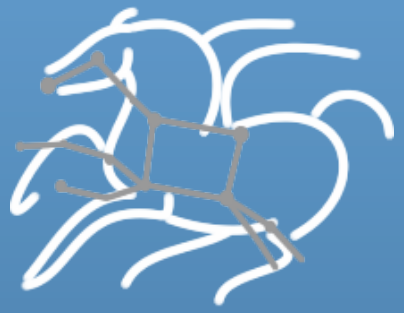


# Executing Computing Pipelines using Pegasus WMS

Karan Vahi

Science Automation Technologies Group

USC Information Sciences Institute

# Before Tutorial Starts

- **Tutorial Page**
  - https://pegasus.isi.edu/tutorial/isi/

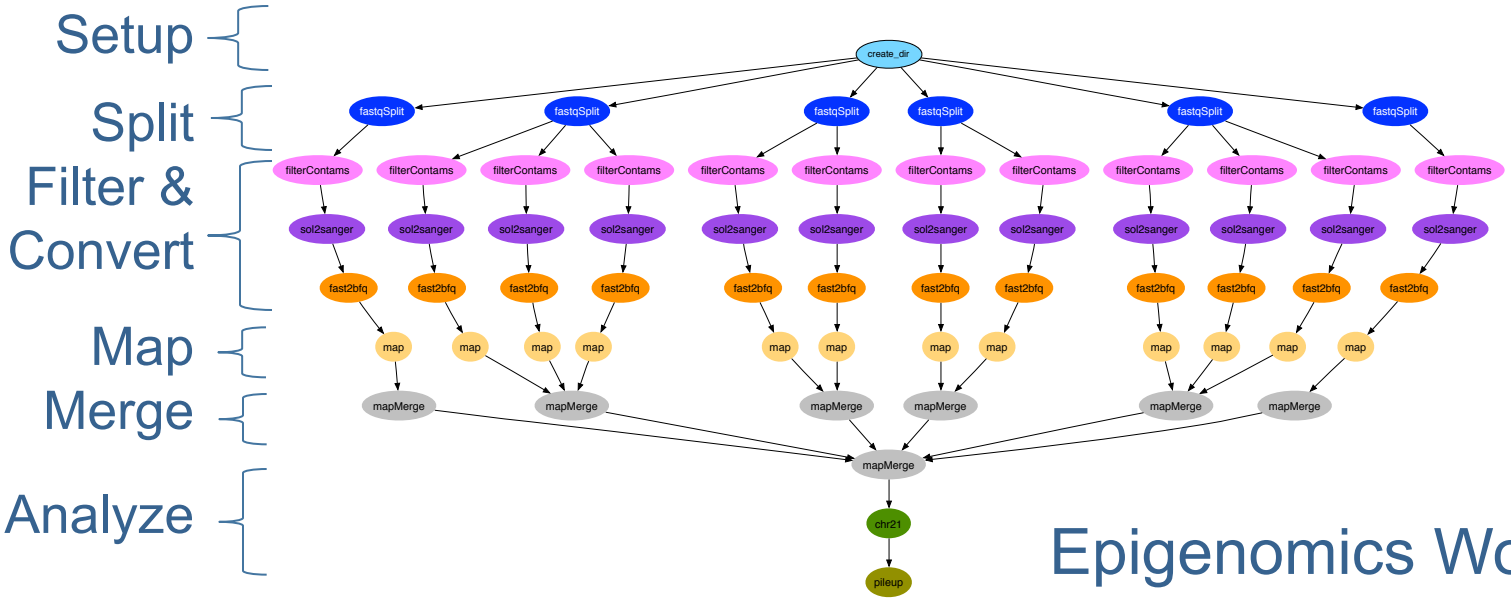
- **Logon to workflow.isi.edu using the training account assigned to you**
  - ssh pegtrainXX@workflow.isi.edu
  - It will prompt for a password

# Agenda

- 1:00 – 1:15 Introduction on Workflows and Pegasus
- 1:15 – 2:30 Hands on Exercises
- 2:30 – 3:00 Features addressing user problems

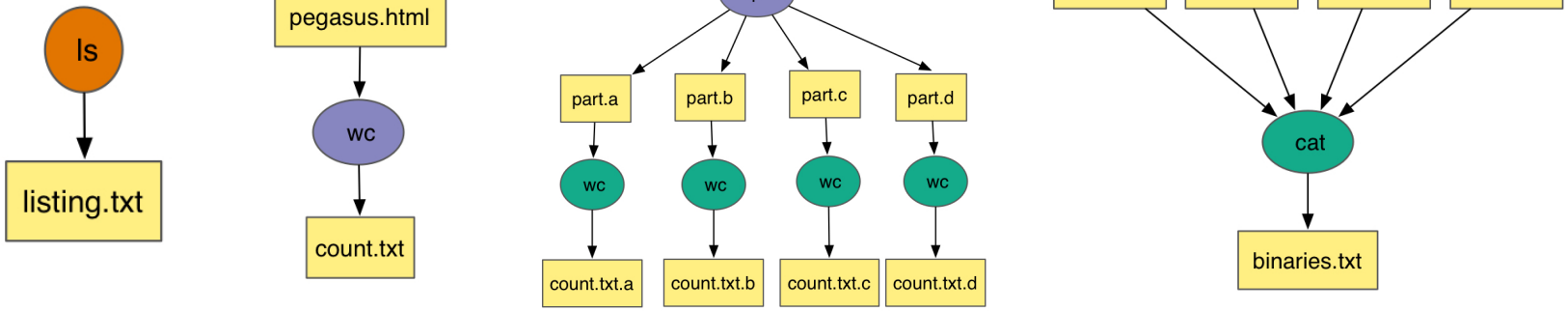
# Scientific Workflows

- **Orchestrate complex, multi-stage scientific computations**
- **Often expressed as directed acyclic graphs (DAGs)**
- **Capture analysis pipelines for sharing and reuse**
- **Can execute in parallel on distributed resources**




Epigenomics Workflow

# Simple Workflows – Building Blocks

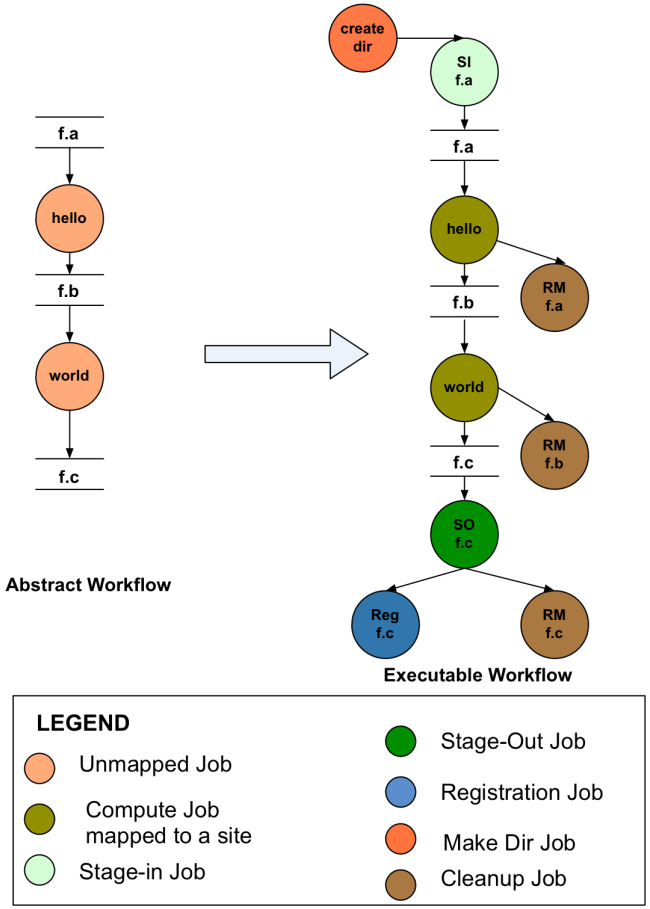


# Challenges while Executing Compute Pipelines

- **Portability**
  - How can you run a pipeline on Amazon EC2 one day, and a PBS cluster the next?
- **Data Management**
  - How do you ship in the small/large amounts data required by your pipeline?
  - Different protocols for different sites: Can I use SRM? How about GridFTP? HTTP and Squid proxies?
  - Can I use Cloud based storage like S3 on EC2?
- **Debug and Monitor Computations.**
  - 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 Pipelines for Improved Performance**
  - Short running tasks?
  - Data placement?

# Pegasus Workflow Management System

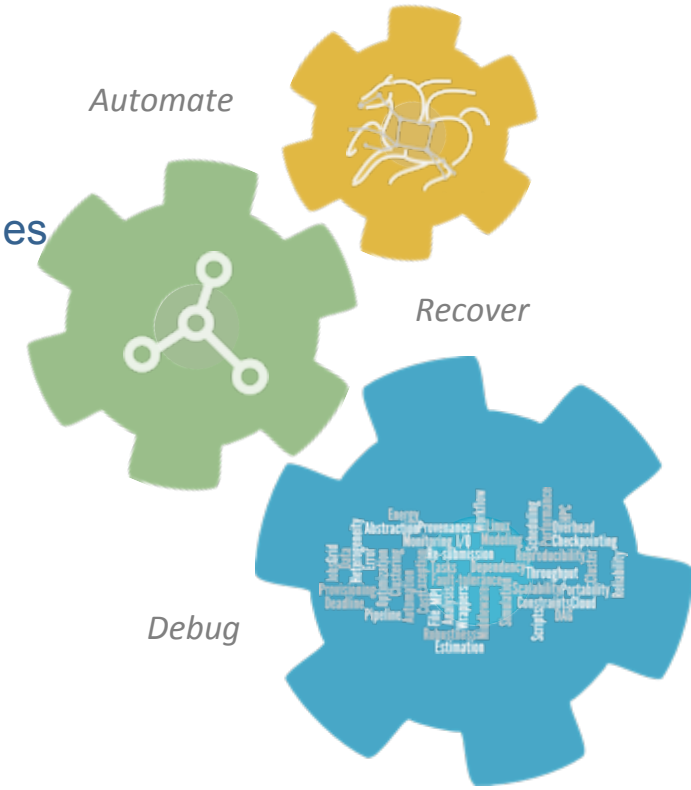
- **NSF funded project since 2001**
  - Developed as a collaboration between USC Information Sciences Institute and the HTCondor Team at UW Madison
- **Builds on top of HTCondor DAGMan.**
- **Abstract Workflows - Pegasus input workflow description**
  - Workflow "high-level language"
  - Only identifies the computation, devoid of resource descriptions, devoid of data locations
  - File Aware – For each task you specify the input and output files
- **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
# Workflow Management System (est. 2001)

- **Maps a resource-independent "abstract" workflow onto resources and executes the "executable" workflow**

## Benefits




- **Automation**
  - Automates complex, multi-stage processing pipelines
  - Enables parallel, distributed computations
  - Automatically executes data transfers
  - Reusable portable description of workflows that aids reproducibility
- **Recovery**
  - Automatic job retries
  - Ability to restart pipelines without repeating steps
- **Debug**
  - Records how data was produced (provenance)
  - Simple easy to use command line tools for monitoring and debugging
  - Web Based Dashboard Available

# Agenda

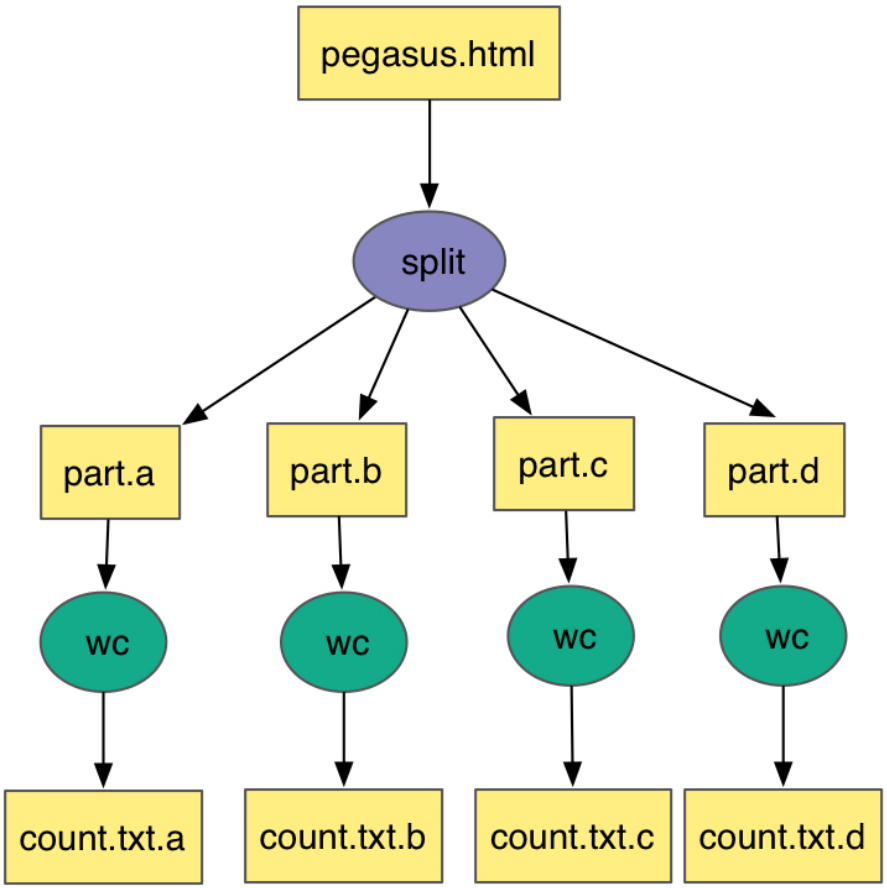
- 1:00 – 1:15 Introduction on Workflows and Pegasus
- 1:15 – 2:30 Hands on Exercises
  - https://pegasus.isi.edu/tutorial/isi/tutorial.php
- 2:30 – 3:00 Features addressing user problems


# Topics Covered

https://pegasus.isi.edu/tutorial/isi/tutorial.php

- What are Scientific Workflows
- Submission of an already generated example workflow with Pegasus.
- How to use the Pegasus Workflow Dashboard for monitoring workflows.
- Command line tools for monitoring, debugging and generating statistics.
- Recovery from failures
- Creation of workflow using system provided API
- Information catalogs configuration.
- Job Clustering


# Tutorial Workflow




- Simple workflow that breaks a file into 4 chunks
- Counts the number of words in each chunk as a separate job.
- The inputs (pegasus.html) are in a directory called input
- The outputs will be generated and placed in directory called outputs
- A workflow specific scratch directory in /scratch is used for intermediate files, and for staging purposes.

**Jobs will be executed on Open Science Grid**

# Simple Steps to Run Pegasus

1. **Specify your computation in terms of DAX**
   - Write a simple DAX generator
   - Python, Java , Perl based API provided with Pegasus
2. **Set up your catalogs**
   - Replica catalog, transformation catalog and site catalog.
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 and Analyze your workflow**
   - Use *pegasus-status* | *pegasus-analyzer* to monitor the execution of your workflow
5. **Workflow Statistics**
   - Run pegasus-statistics to generate statistics about your workflow run.

# Agenda

- 1:00 – 1:15 Introduction on Workflows and Pegasus
- 1:15 – 2:30 Hands on Exercises
- 2:30 – 3:00 Features addressing user problems


# Failure Recovery

*Job Failure Detection*

*Job Retry*

helps with transient failures
set number of retries per job and run

*Checkpoint Files*

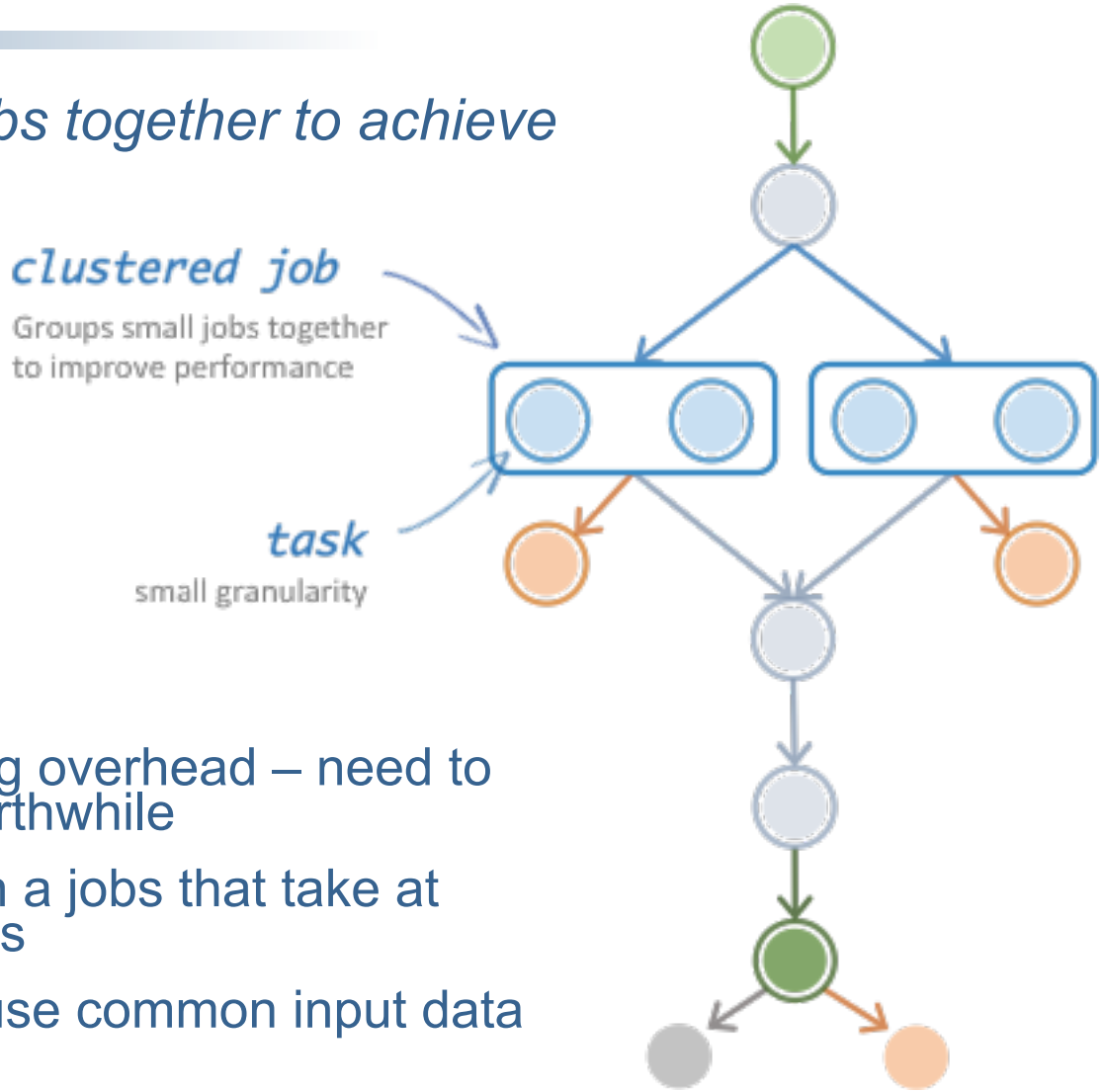
job generates checkpoint files
staging of checkpoint files is automatic on restarts

*Rescue DAGs*

workflow can be restarted from checkpoint file
recover from failures with minimal loss

# Task Clustering

*Cluster small running jobs together to achieve better performance*




## Why?

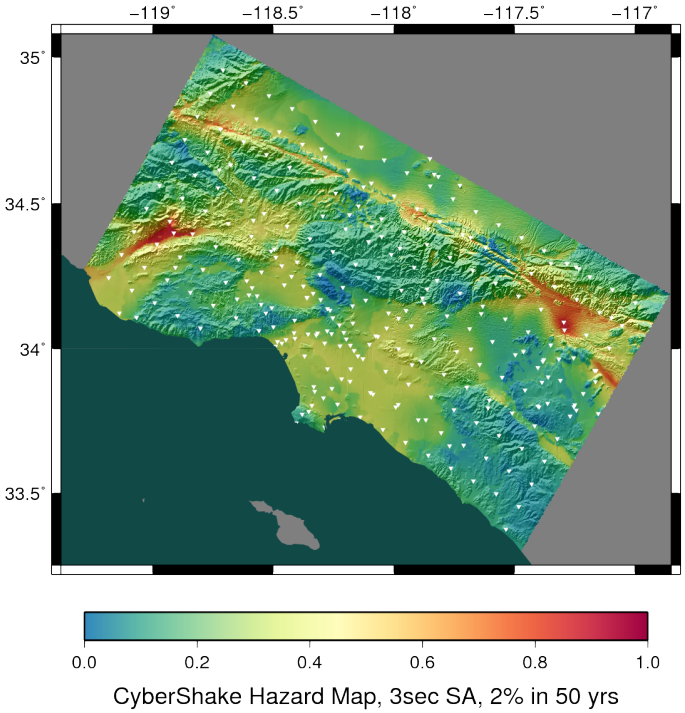
- Each job has scheduling overhead – need to make this overhead worthwhile
- Ideally users should run a jobs that take at least 10/30/60/? minutes
- Clustered tasks can reuse common input data – less data transfers

# Fine-Grained Workflows

- **Problem: Many scientific workflows are fine-grained**
  - Thousands of tasks
  - Short duration
  - Serial
- **Collectively, these tasks require distributed resources to finish in a reasonable time, but individually they are relatively small**
  - Touch many GB or TB of data
  - Consume thousands of CPU hours
- **Many large-scale compute resources are optimized for a few, large, parallel jobs, not many small, serial jobs**
  - Serial tasks face long queue times due to low priority
  - Batch schedulers have low throughput

**Results in poor workflow performance**

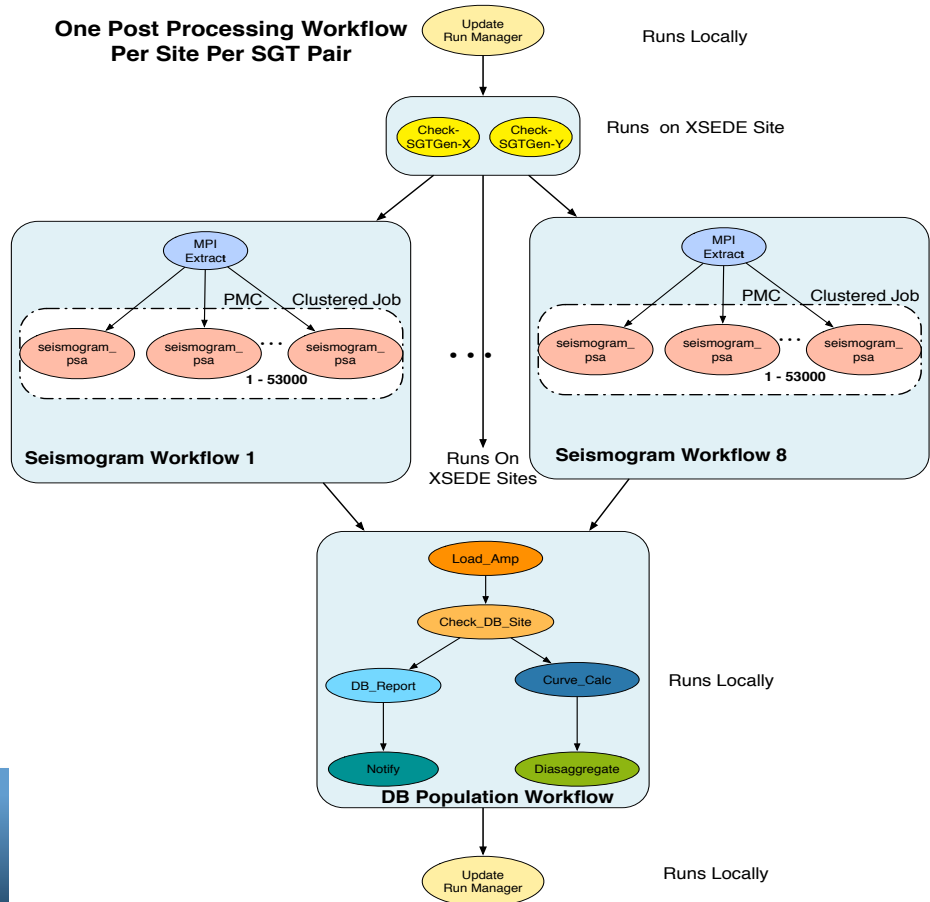
# Fine Grained Workflows



CyberShake Hazard Map, 3sec SA, 2% in 50 yrs

## CyberShake PSHA Workflow

- Builders ask seismologists: "What will the peak ground motion be at my new building in the next 50 years?"
- Seismologists answer this question using Probabilistic Seismic Hazard Analysis (PSHA)



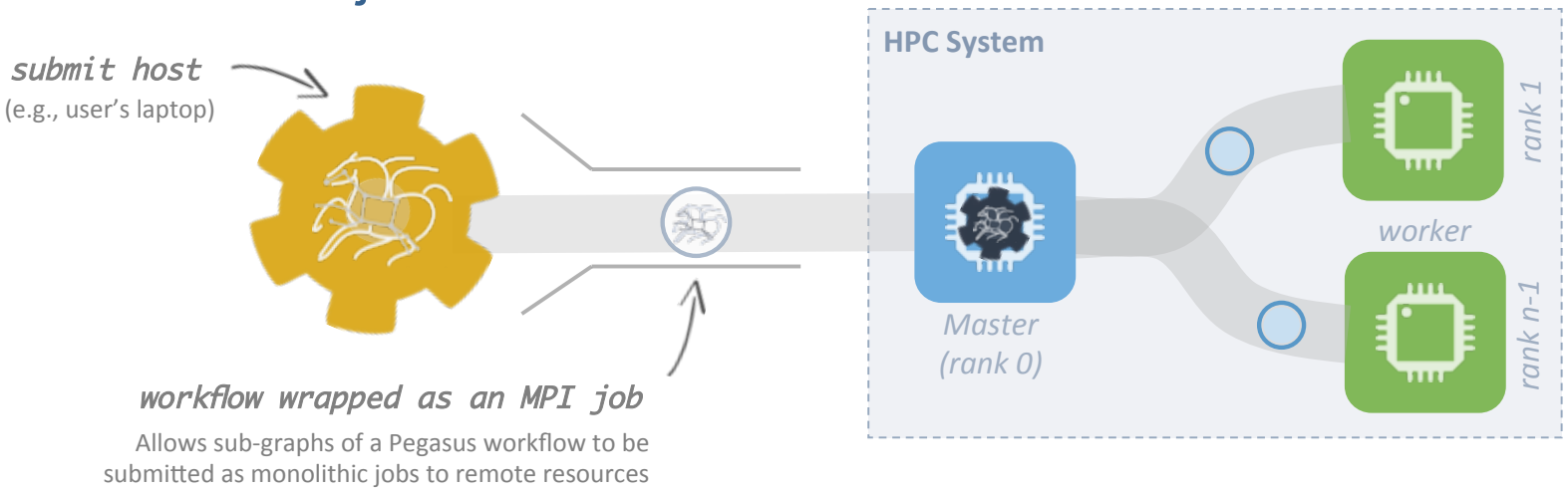

**2014: 286 Sites, 4 models**

- **Each site = one workflow**
- **Each workflow has 420,000 tasks in 21 jobs**

# Fine Grained Workflows

## Solution: Pegasus-MPI-Cluster

- A master/worker task scheduler for running fine-grained workflows on batch systems
- Runs as an MPI job
  - Uses MPI to implement master/worker protocol
- Allows sub-graphs of a Pegasus workflow to be submitted as monolithic jobs to remote resources

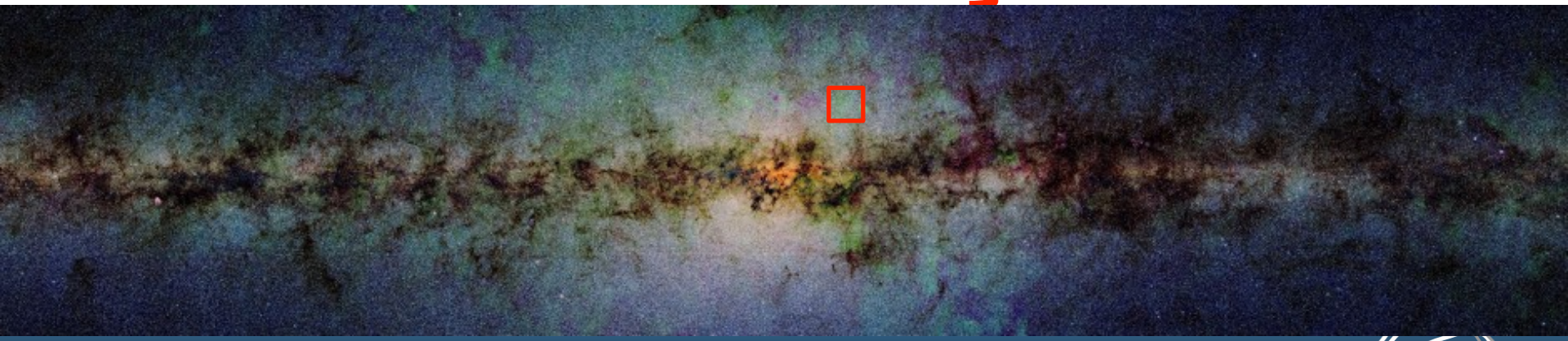


# Connecting Pipelines

- **Problem**
  - For large scale analysis, scientists often want to compose larger constructs from smaller working pipelines.
  - Groups from the same domain collaborate and want to connect pipelines as part of larger analysis
  - Also want to scale to workflows with millions of tasks in total.

- **Example Application Use Case - Montage Galactic Plane Workflow**
  - Existing montage workflow that can generate 5 degree mosiacs
  - 18 million input images (~2.5 TB)
  - 900 output images (2.5 GB each, 2.4 TB total)
  - 10.5 million tasks (34,000 CPU hours)

× 17

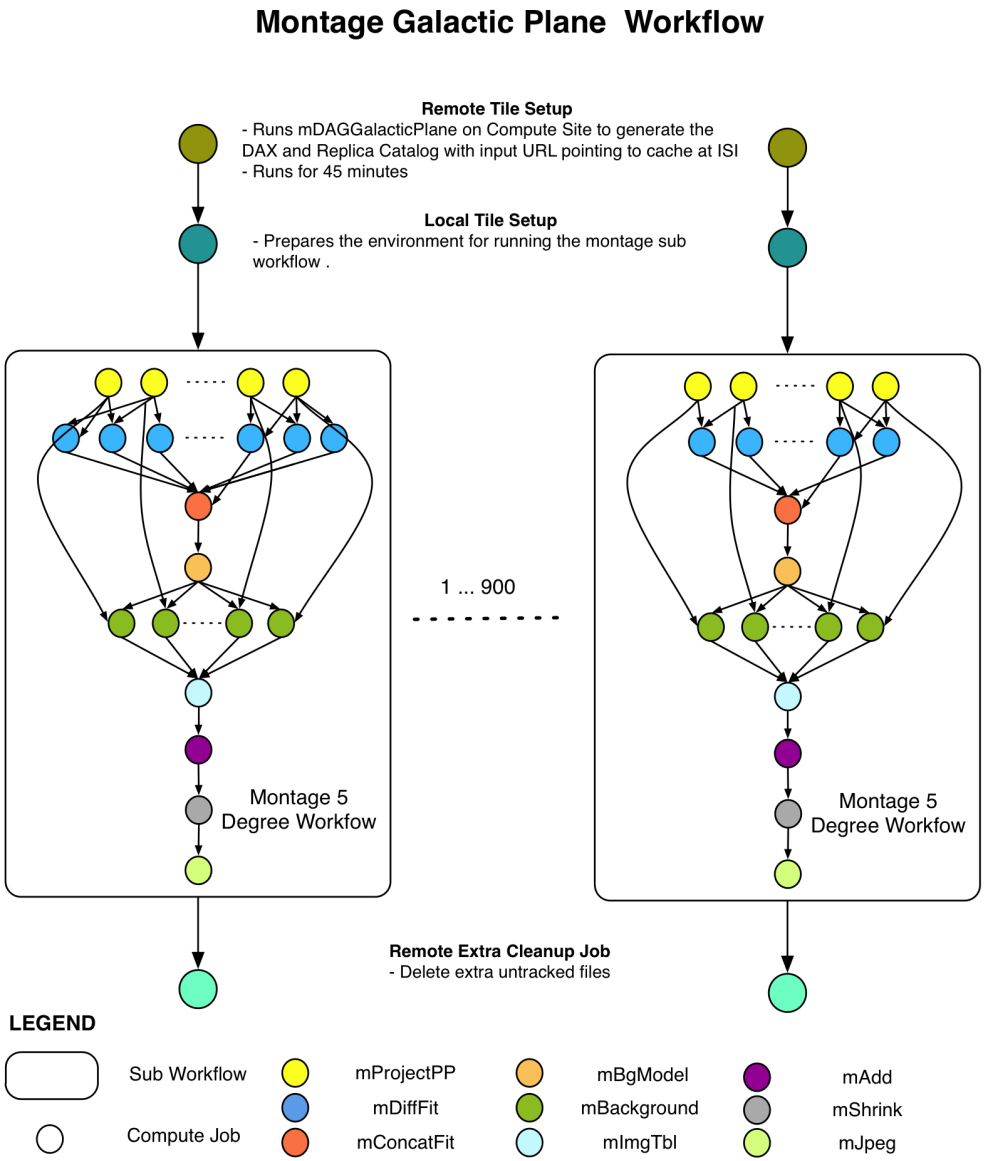


# Connecting Pipelines

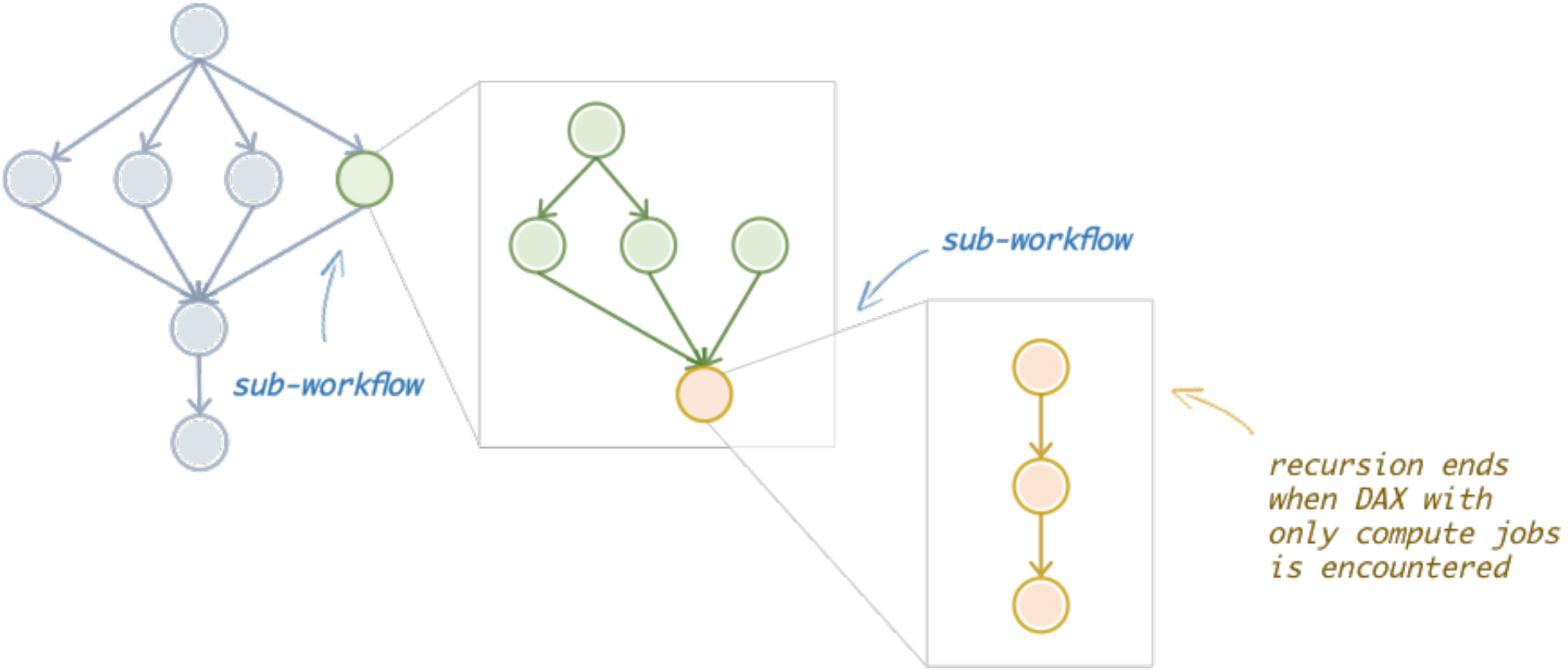
## Solution

- Pegasus allows jobs/nodes in a workflow to refer to other workflows, allowing users to connect distinct workflows for analysis.

- Updated our debugging tools to handle these workflow of workflows.



# Hierarchical Workflows



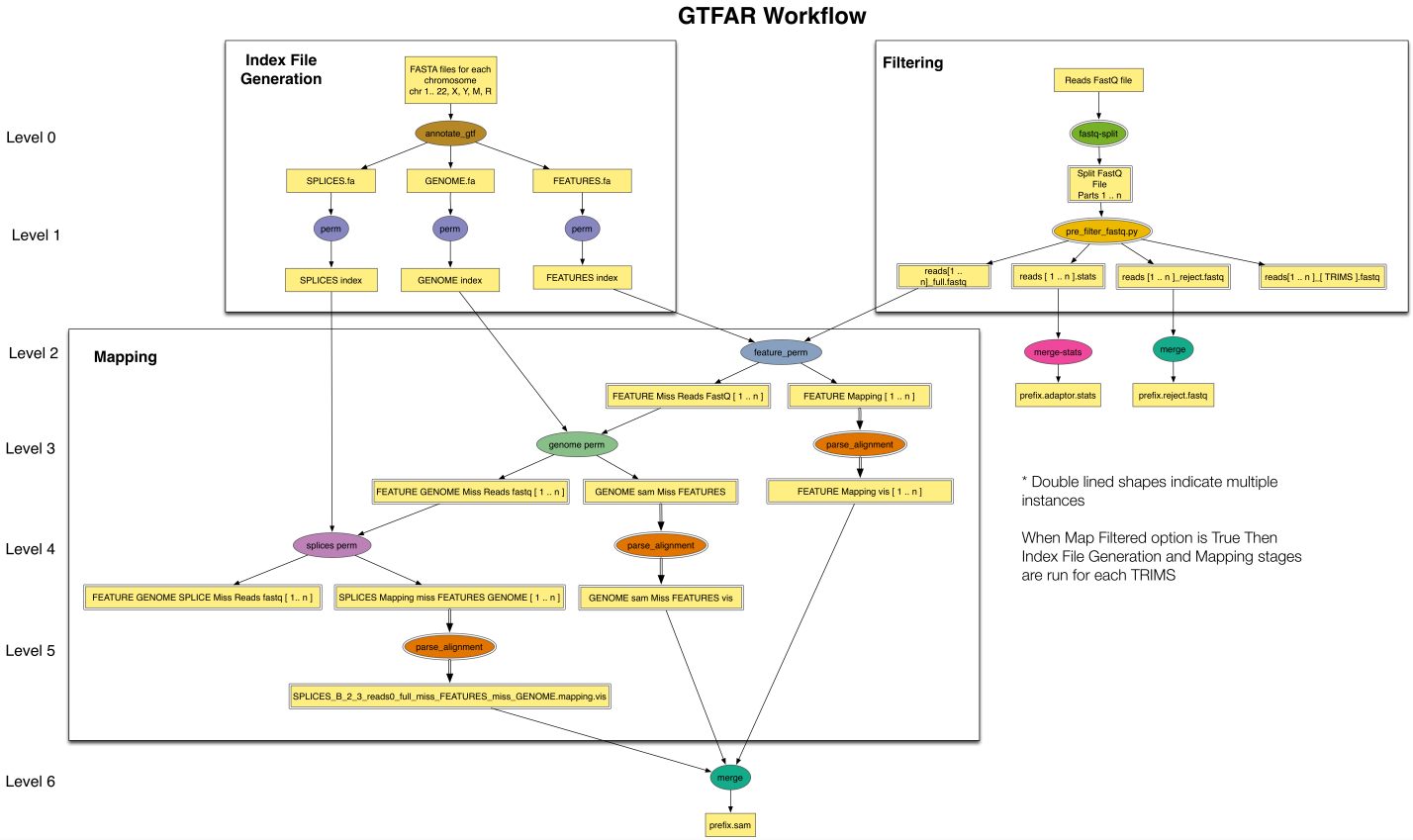
# Reusing Data Products

- **Problem**

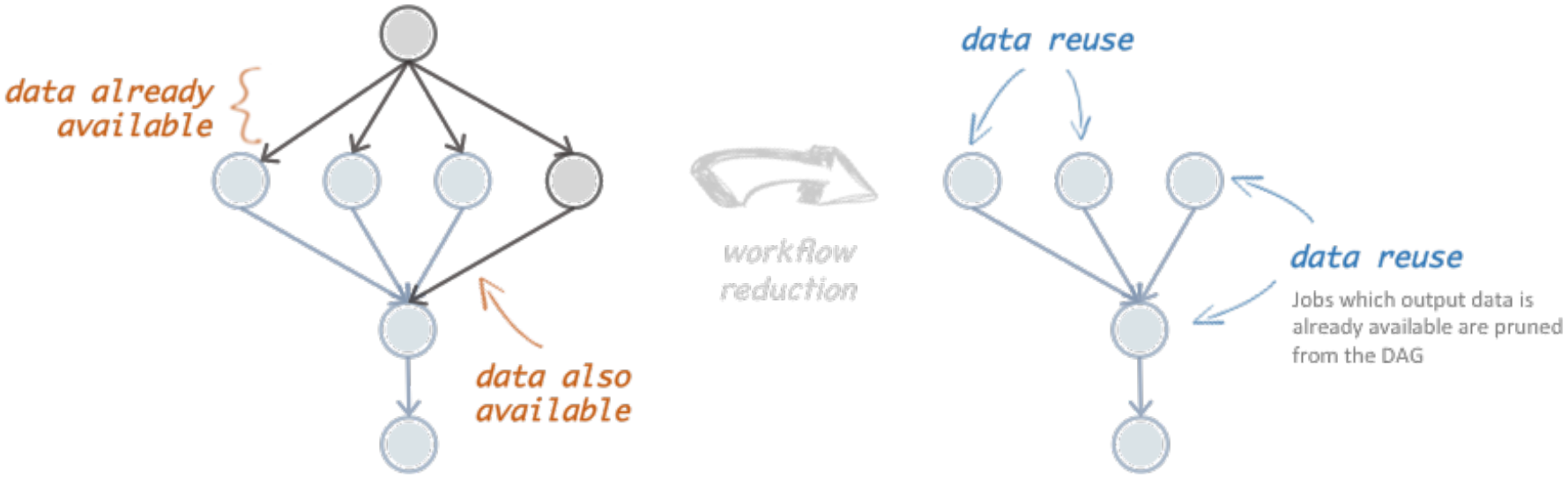
  Users want to be able to intelligently reuse previously generated outputs.

  Genomic Workflows want to avoid index generation steps for raw input files

Long running analysis fail, and users realize they need to regenerate workflow description, as wrong arguments set or missing jobs.



# Reusing Data Products




**Solution: Workflow Reduction**

- Don’t execute jobs at runtime for which data products already exist.
- Similar to make style semantics for compiling code

# File cleanup

- **Problem**

  Users run out out of disk space during workflow execution

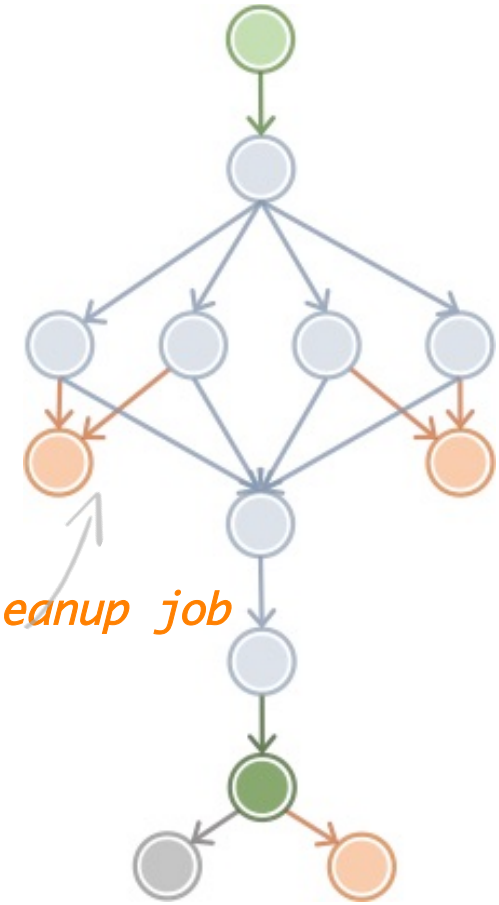
- **Why does it occur**
  - Workflows could bring in huge amounts of data
  - Data is generated during workflow execution
  - Users don't worry about cleaning up after they are done

- **Example Application**
  - SoyKB workflows from iPlant
  - Epigenomics Pipelines generating 60TB of data in a single execution

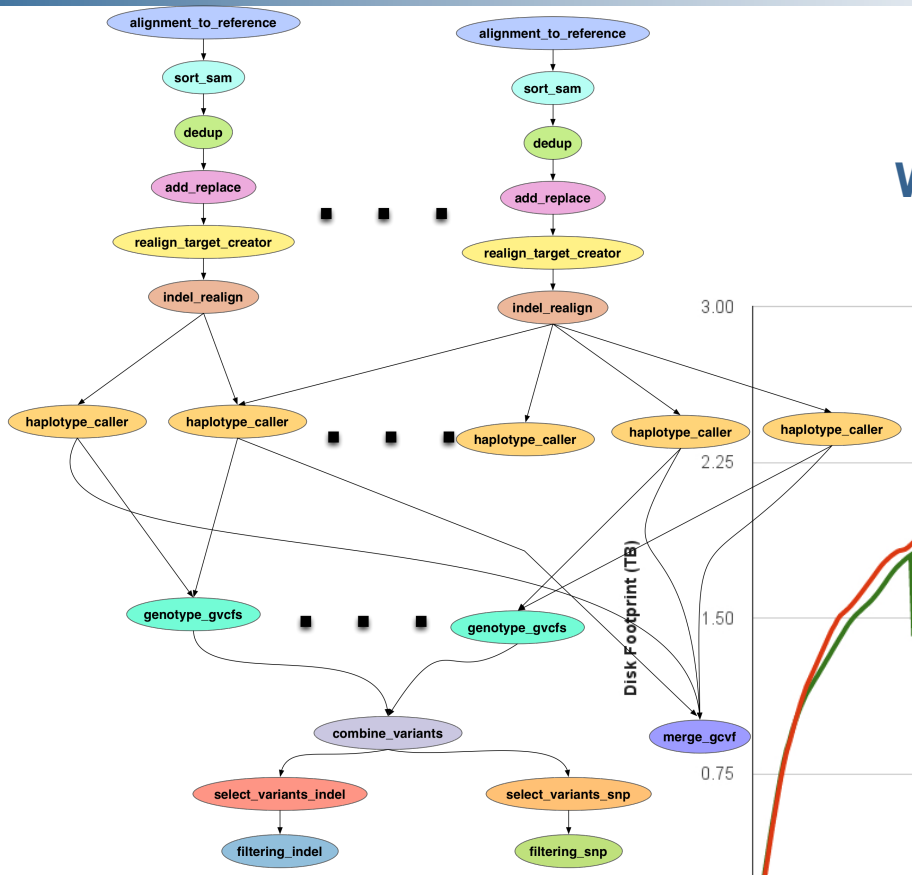


# File cleanup

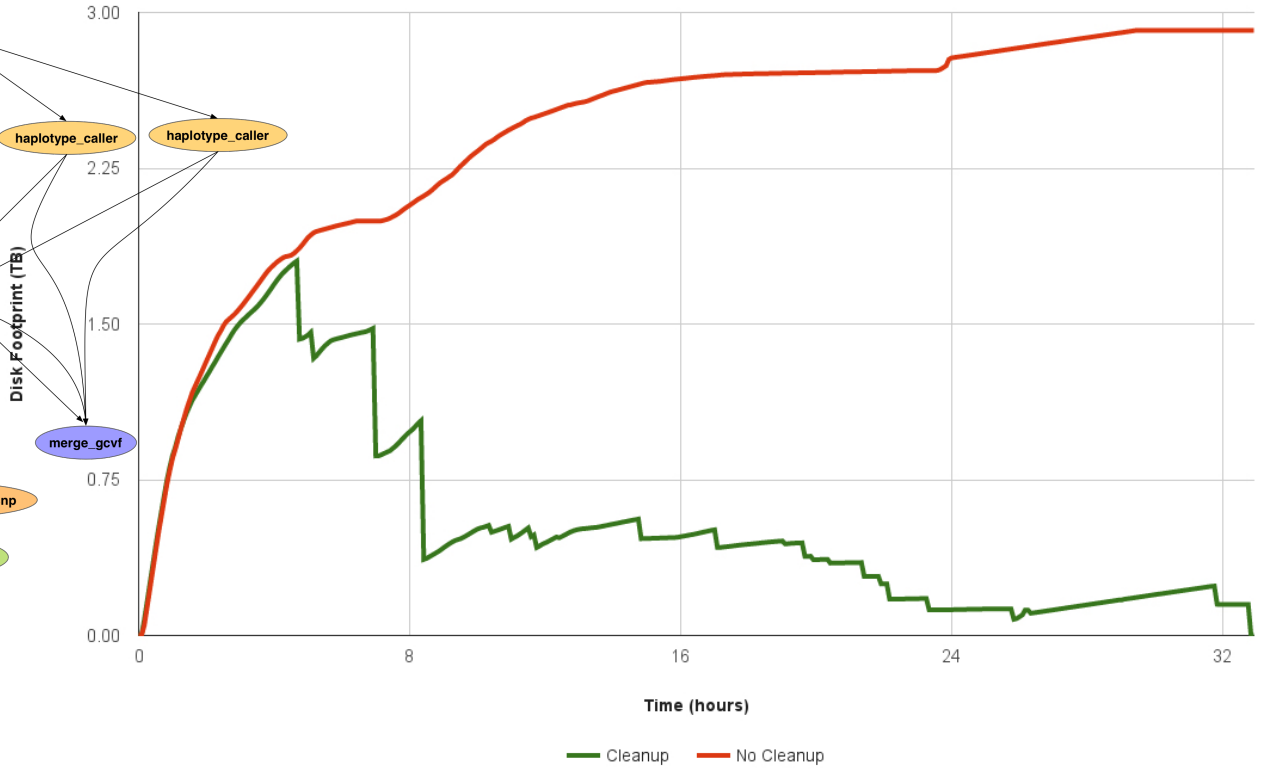
- **Solution**
  - **Do cleanup after workflows finish**
    - Does not work as the scratch may get filled much before during execution
  - **Interleave cleanup automatically during workflow execution.**
    - Requires an analysis of the workflow to determine, when a file is no longer required
  - **Cluster the cleanup jobs by level for large workflows**
    - Too many cleanup jobs adversaly affect the walltime of the workflow.



# File cleanup (cont)




**Single SoyKB NGS Pegasus Workflow with 10 input reads.**



# What Does Pegasus provide an Application - I

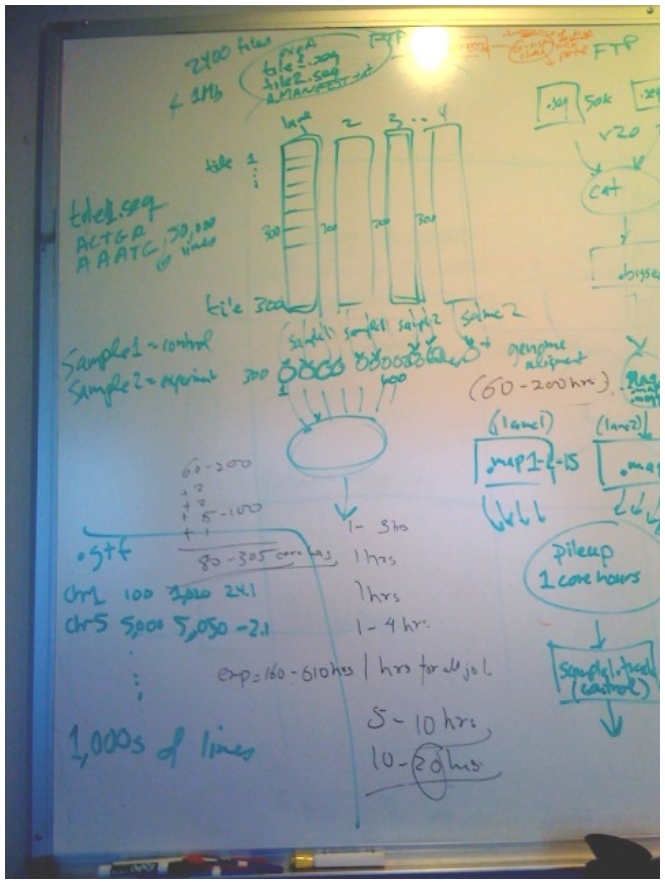
- **Portability / Reuse**
  - User created workflows can easily be mapped to and run in different environments without alteration.
- **Data Management**
  - Pegasus handles replica selection, data transfers and output registrations in data catalogs. These tasks are added to a workflow as auxiliary jobs by the Pegasus planner.
- **Performance**
  - The Pegasus mapper can reorder, group, and prioritize tasks in order to increase the overall workflow performance.

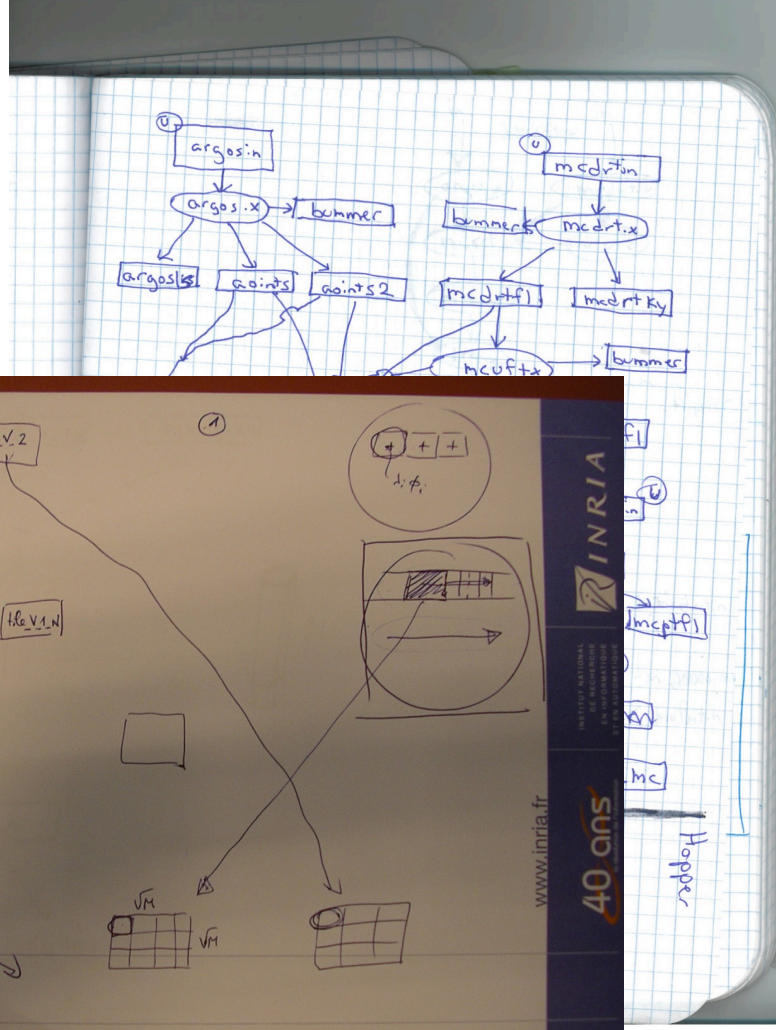
# What Does Pegasus provide an Application - II

- **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.
- **Scalability**
  - Hierarchal workflows
  - Scale to hundreds of thousands of nodes in a workflow.

# If you get stuck…

**And you can draw….**







**We can help you!**

# Pegasus

est. 2001

Automate, recover, and debug scientific computations.

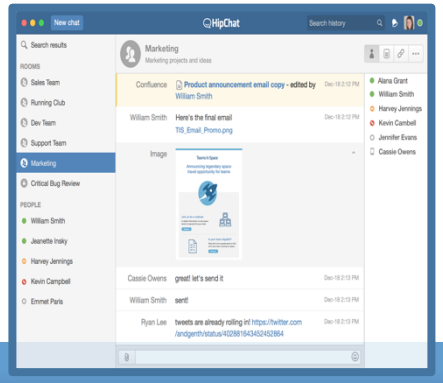
## Get Started

**Pegasus Website**
http://pegasus.isi.edu

**Users Mailing List**
pegasus-users@isi.edu

**Support**
pegasus-support@isi.edu

**HipChat**



# Pegasus

est. 2001

Automate, recover, and debug scientific computations.

## Thank You

## Questions?

## Meet our team


Ewa Deelman


Karan Vahi


Gideon Juve


Mats Rynge


Rajiv Mayani


Rafael Ferreira da Silva