Science Automation with the Pegasus Workflow Management System

Ewa Deelman

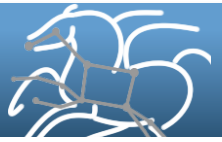
USC Information Sciences Institute

http://www.isi.edu/~deelman

Funding from DOE, NSF, and NIH

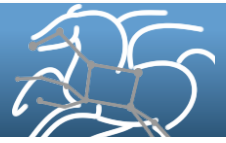
The Problem

- Scientific data is being collected at an ever increasing rate
  - The “old days” -- big, focused experiments– LHC
  - Today also “cheap” DNA sequencers – and an increasing number of them
- The complexity of the computational problems is ever increasing
- Local compute resources are often not enough
  - Too small, limited availability
  - Data sets are distributed
- The computing infrastructure keeps changing
  - Hardware, software, but also computational models



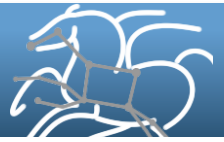
Our approach

- Provide a way to structure applications in such a way that enables them to be automatically managed
  - In a portable way: same description that works on different resources
  - In a way that scientists can interpret the results
- Develop a system that
  - Maps the application description onto the available resources
  - Manages its execution on heterogeneous resources
  - Sends results back to the user or archive
  - Provides good performance, reliability, scalability



Outline

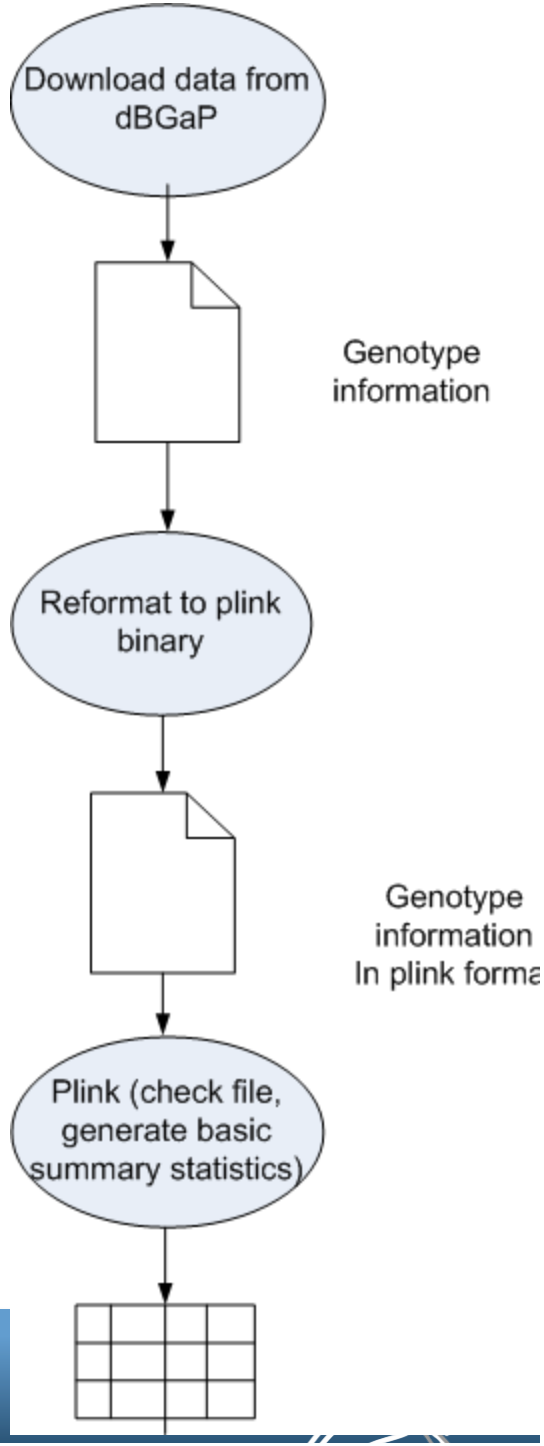
- Scientific Workflows and Application Examples
- Managing scientific workflows
- Pegasus and its features
- Conclusions

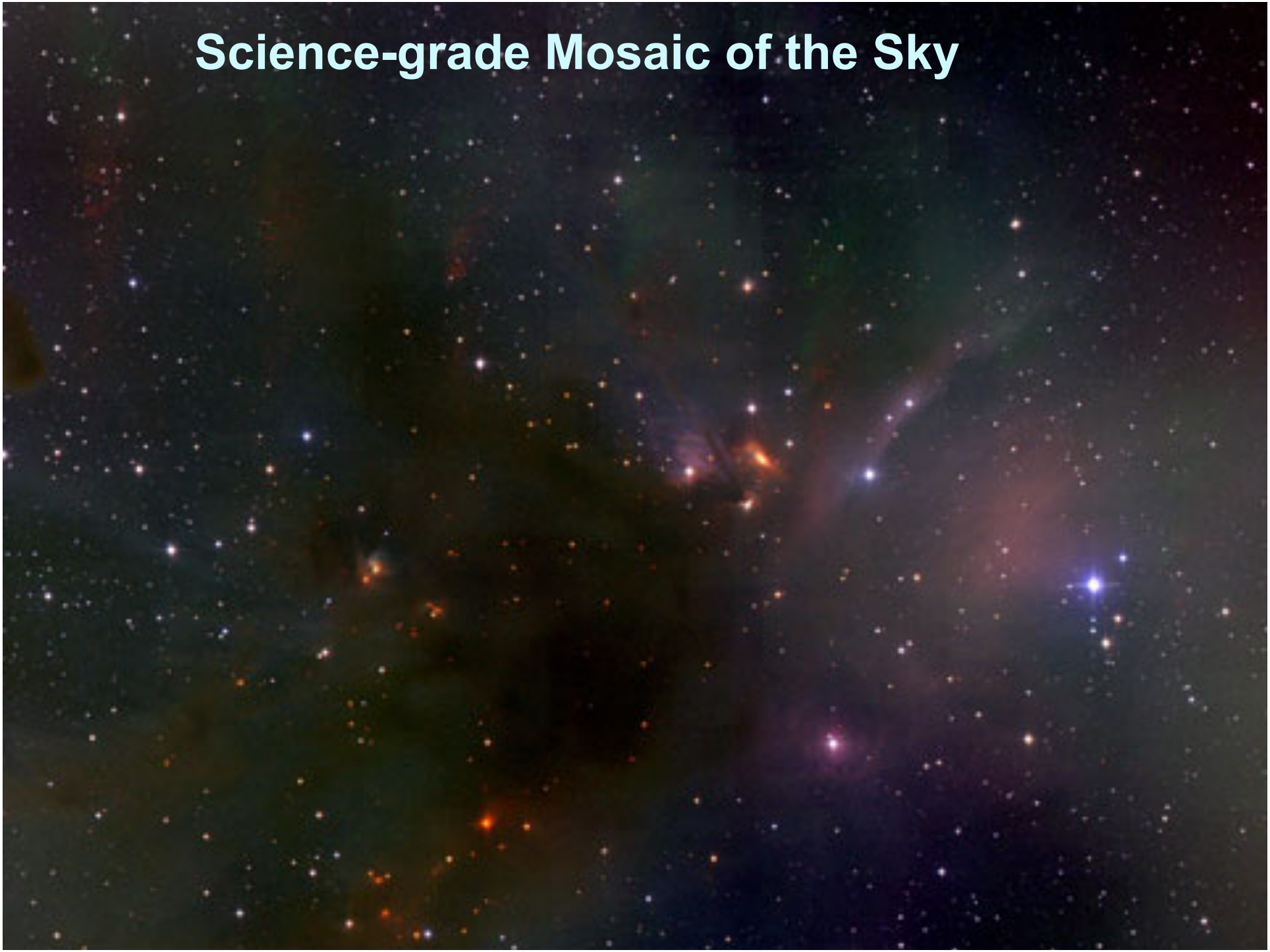


Scientific Workflows

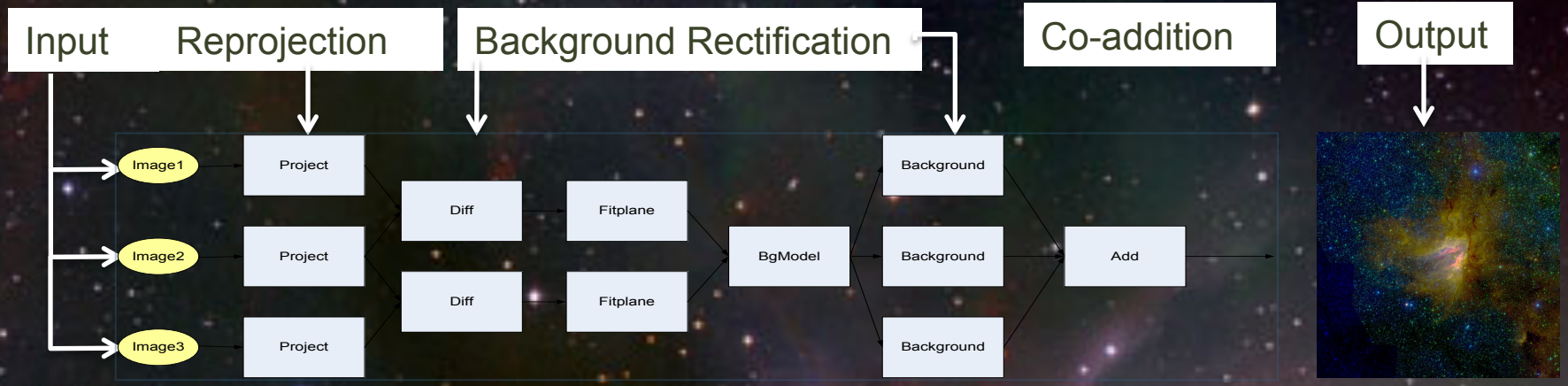
- Structure an overall computation
- Define the computation steps and their parameters
- Define the input/output data, parameters

- Invoke the overall computation
- Reuse with other data/parameters/ algorithms and share
- Workflows can be hidden behind a nice user interface (e.g. portal)




Science-grade Mosaic of the Sky

Science-grade Mosaic of the Sky

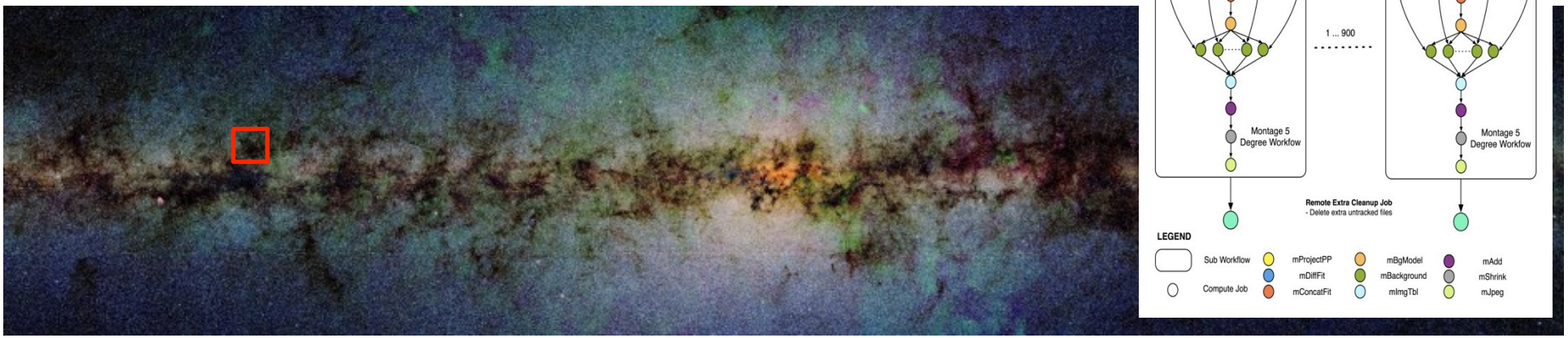


Montage Workflow

Amazon M1 large with 2 cores

Size of mosaic in degrees square	Number of input data files	Number of tasks	Number of intermediate files	Total data footprint	Cummulative wall time
1	84	387	770	1.8 GB	11 mins
2	300	1442	2880	6.4 GB	43 mins
4	685	3738	7466	17 GB	1 hour, 56 mins
6	1461	7462	14904	35 GB	3 hours, 42 mins
8	2565	12757	25480	59 GB	6 hours, 45 mins

Some workflows are large-scale and data-intensive

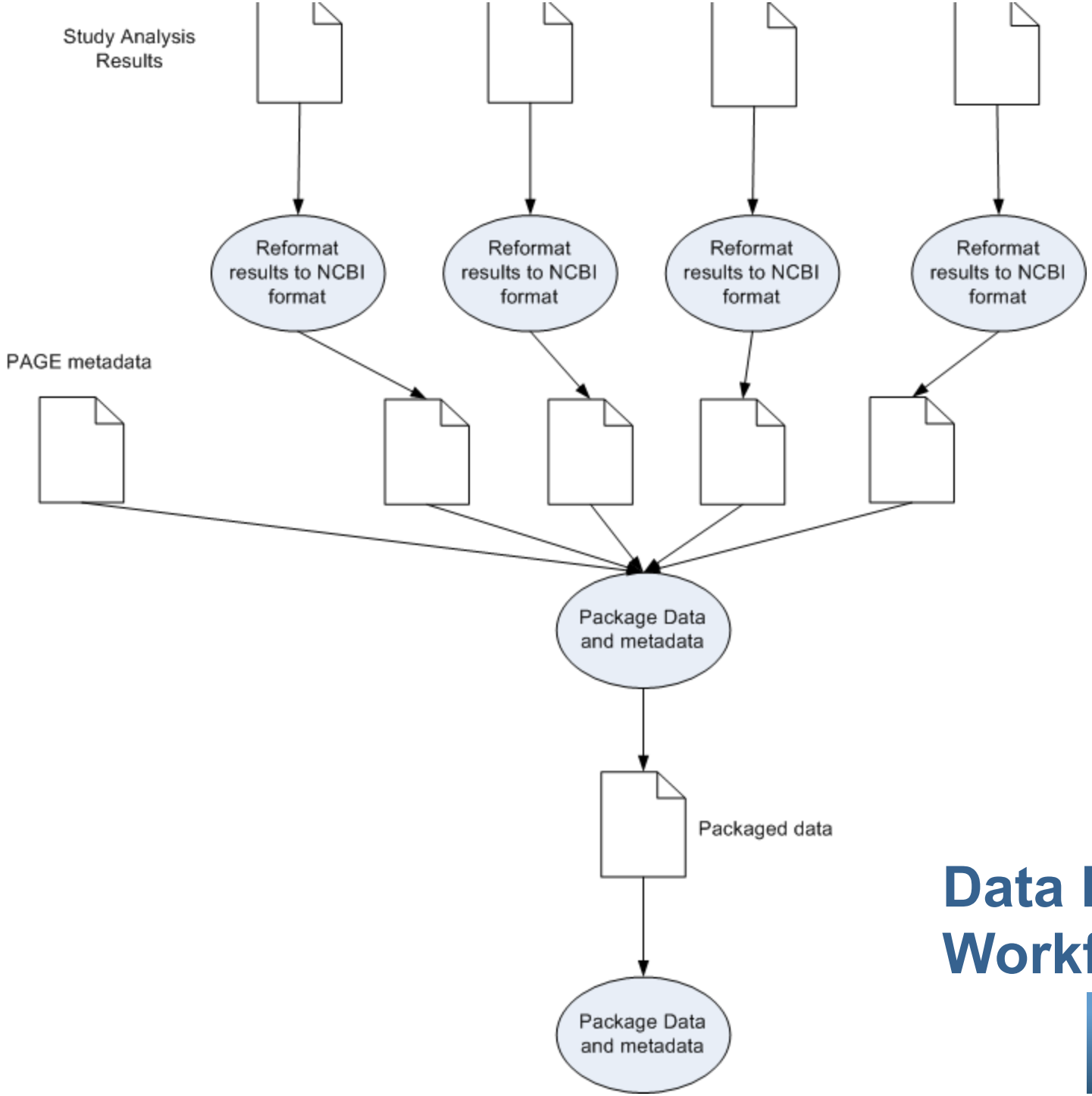


John Good (Caltech)

- **Montage Galactic Plane Workflow**
  - 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

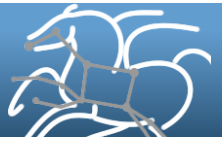
- **Need to support hierarchical workflows and scale**



Data Management Workflow

Outline

- Scientific Workflows and Application Examples
- Managing scientific workflows
- Pegasus and its features
- Conclusions



Specification: Place Y = F(x) at L

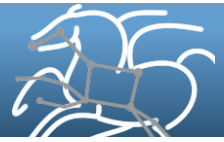
- Find where x is--- {S1,S2, …}
- Find where F can be computed--- {C1,C2, …}
- Choose *c* and *s* subject to constraints (performance, space availability,….)
- Move x from *s* to *c*
  - *Move F to c*
- Compute F(x) at *c*
- Move Y from *c* to L
- Register Y in data registry
- Record provenance of Y, performance of F(x) at c

Error! *x* was not at *s!*

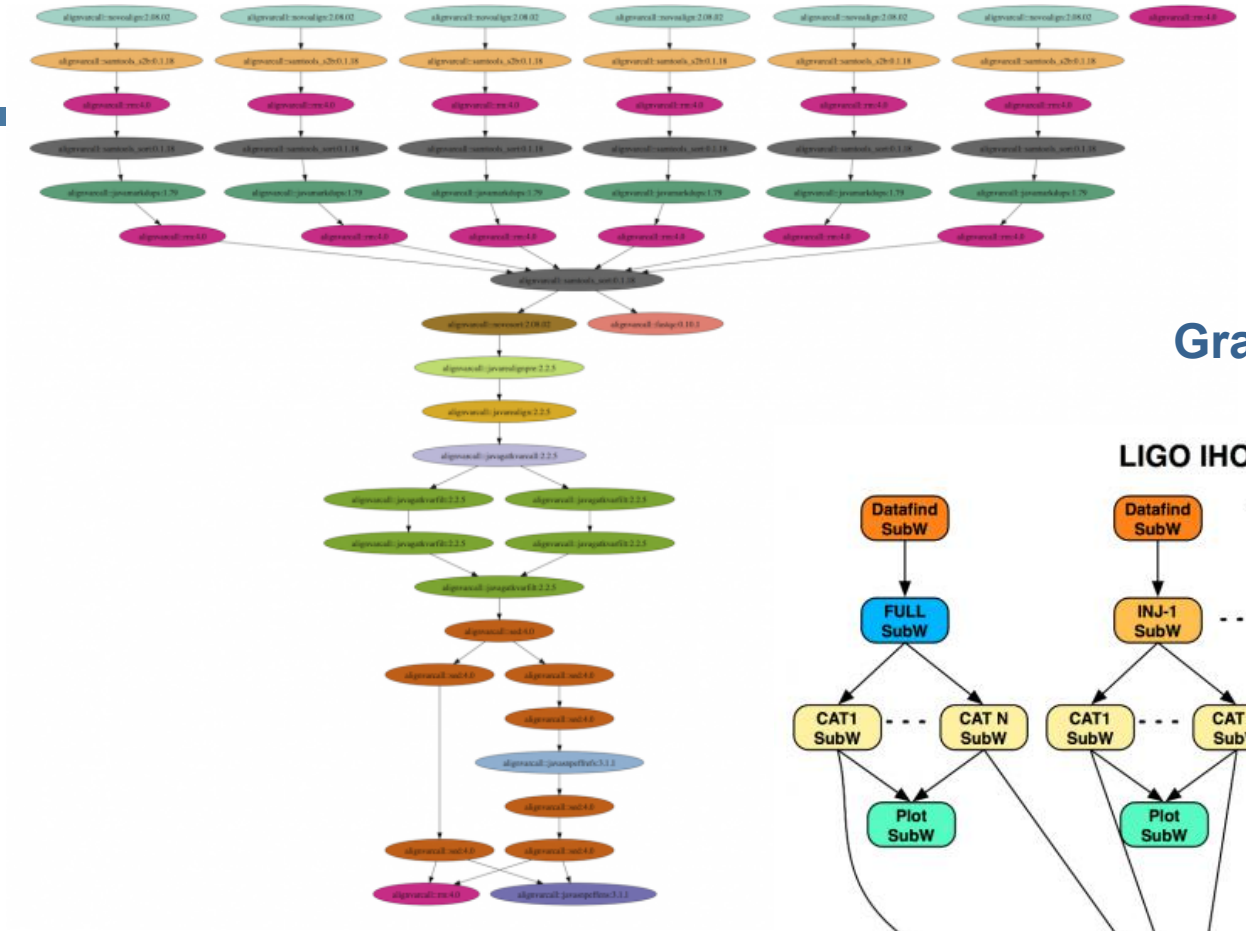
Error! *F(x)* failed!

Error! *c* crashed!

Error! *there is not enough* space at L!

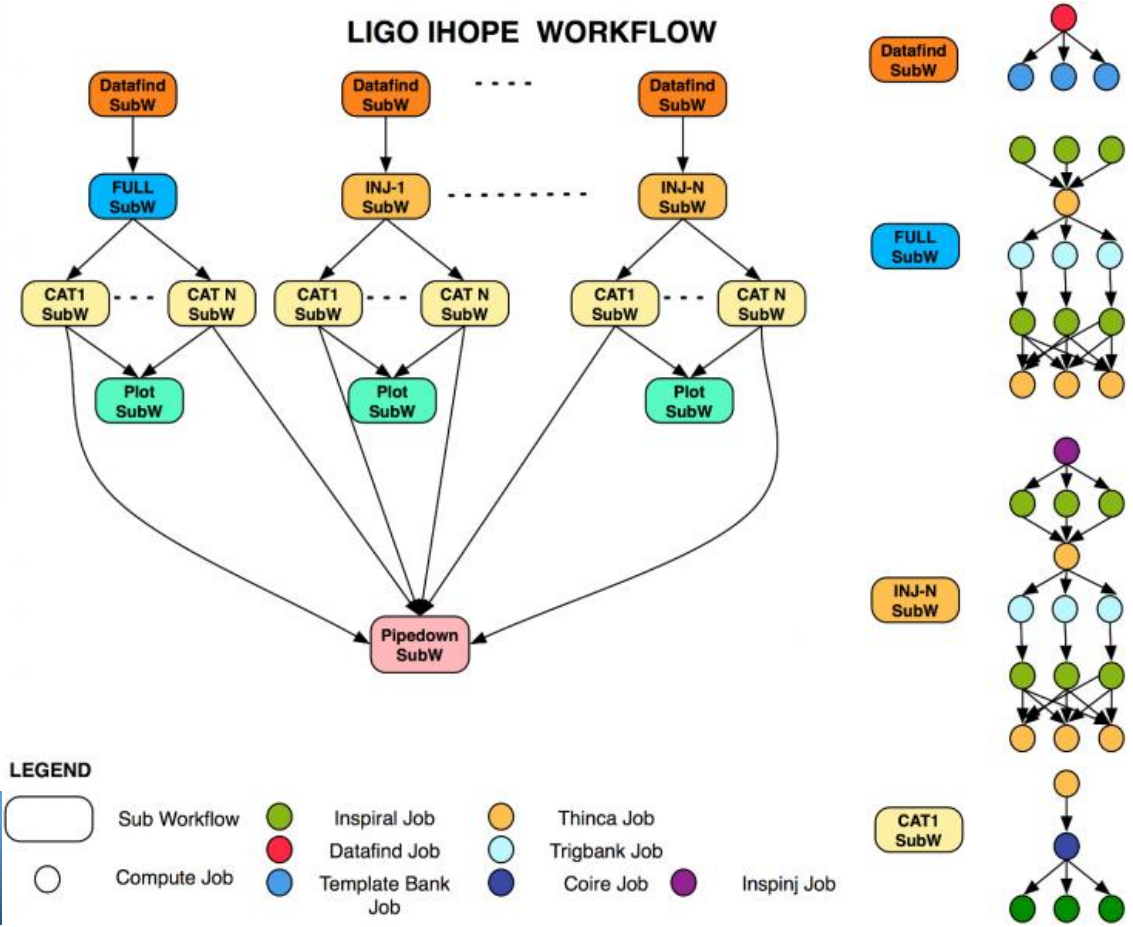


Some workflows are structurally complex



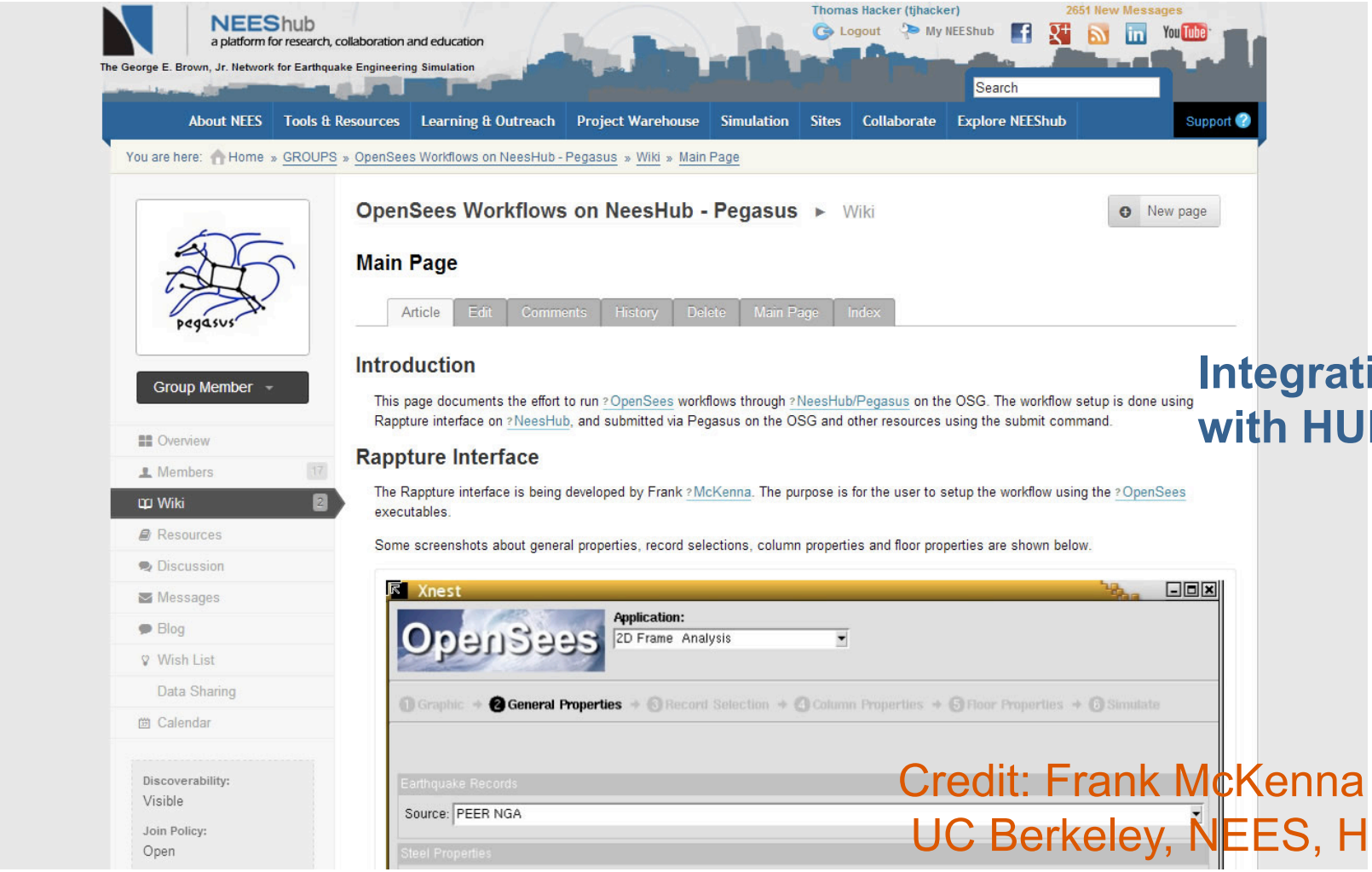
Genomics Workflow

Gravitational-wave physics



Workflows can be simple!

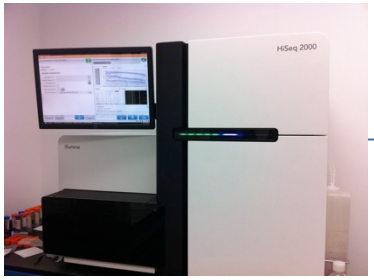
Sometimes you want to “hide” the workflow

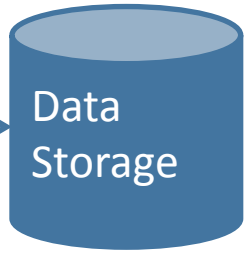


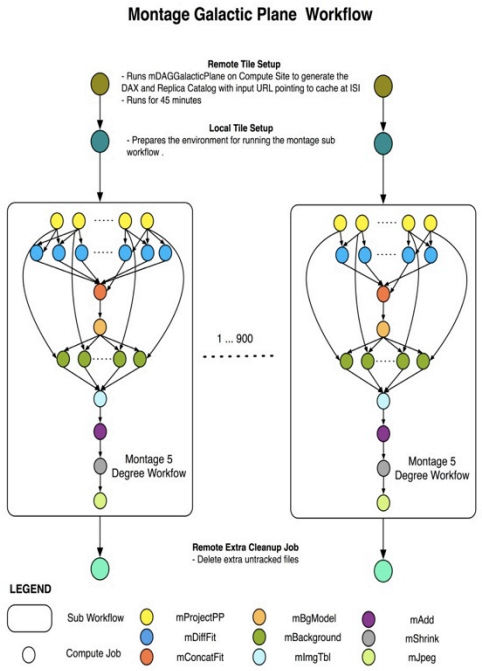
Integration with HUBzero

Credit: Frank McKenna
UC Berkeley, NEES, HUBzero

Sometimes the environment is complex







Work definition

Local Resource

Campus Cluster

XSEDE

NERSC

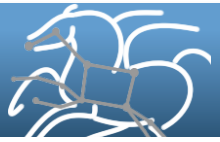
ALCF

OLCF

Open Science Grid

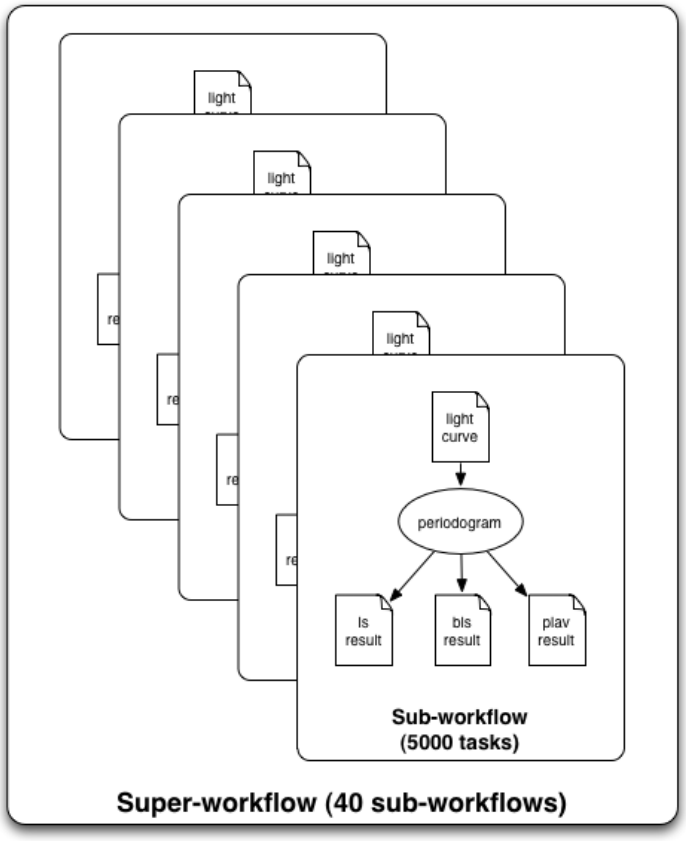
FutureGrid

Amazon Cloud



Sometimes the environment is just not exactly right

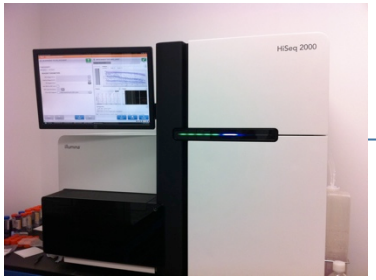
Single core workload



Cray XT System Environment / ALPS / aprun

- Designed for MPI codes

Sometime you want to change or combine resources

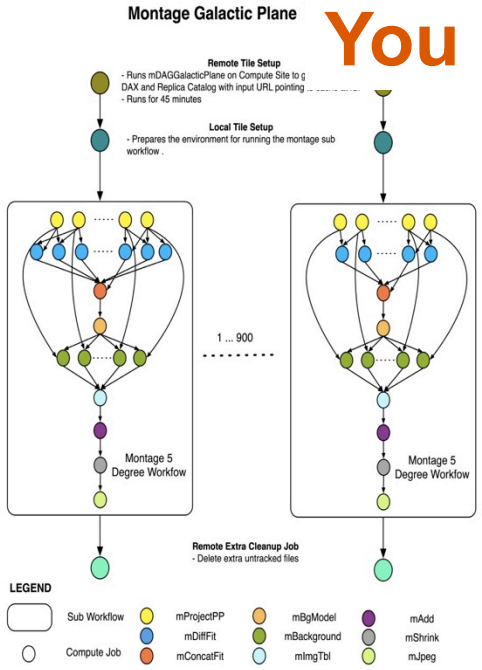


Data Storage

data

Campus Cluster

XSEDE



You don't want to recode your workflow

Local Resource

work

ALCF

OLCF

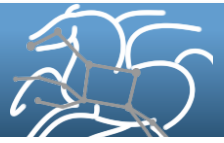
Open Science Grid

FutureGrid

Amazon Cloud

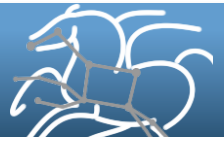
Workflow Management

- Assume a high-level workflow specification
- Assume the potential use of different resources within a workflow or over time
  - Need a planning capability to map from high-level to executable workflow
  - Need to manage the task dependencies
  - Need to manage the execution of tasks on the remote resources
- Need to provide provenance information
- Need to provide scalability, performance, reliability



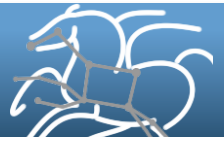
Outline

- Scientific Workflows and Application Examples
- Managing scientific workflows
- Pegasus and its features
- Conclusions

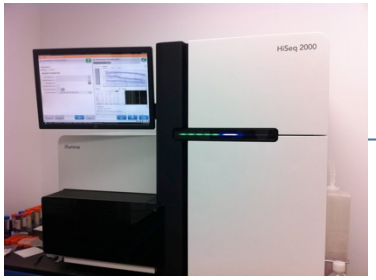


Our Approach

- **Analysis Representation**
  - Support a declarative representation for the workflow (dataflow)
  - Represent the workflow structure as a Directed Acyclic Graph (DAG)
  - Tasks operate on files
  - Use recursion to achieve scalability
- **System (Plan for the resources, Execute the Plan, Manage tasks)**
  - Layered architecture, each layer is responsible for a particular function
  - Mask errors at different levels of the system
  - Modular, composed of well-defined components, where different components can be swapped in
  - Use and adapt existing graph and other relevant algorithms



Submit locally, compute Globally



Data Storage

data

Campus Cluster

XSEDE

NERSC

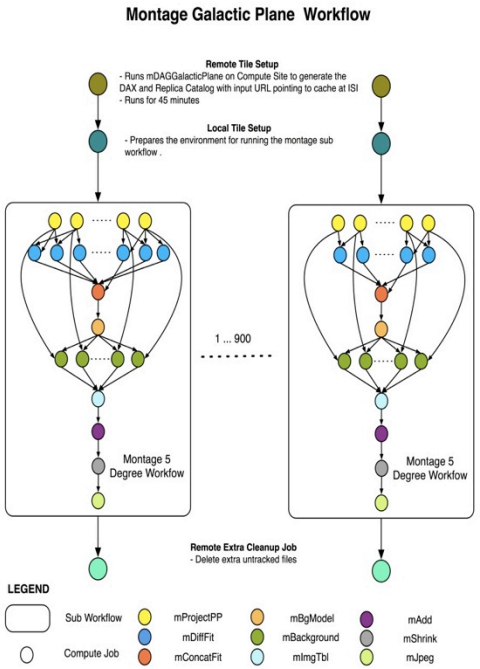
ALCF

OLCF

Open Science Grid

FutureGrid

Amazon Cloud



Work definition

Workflow Management System

work

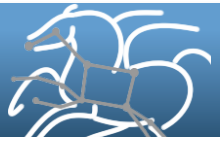
Local Resource

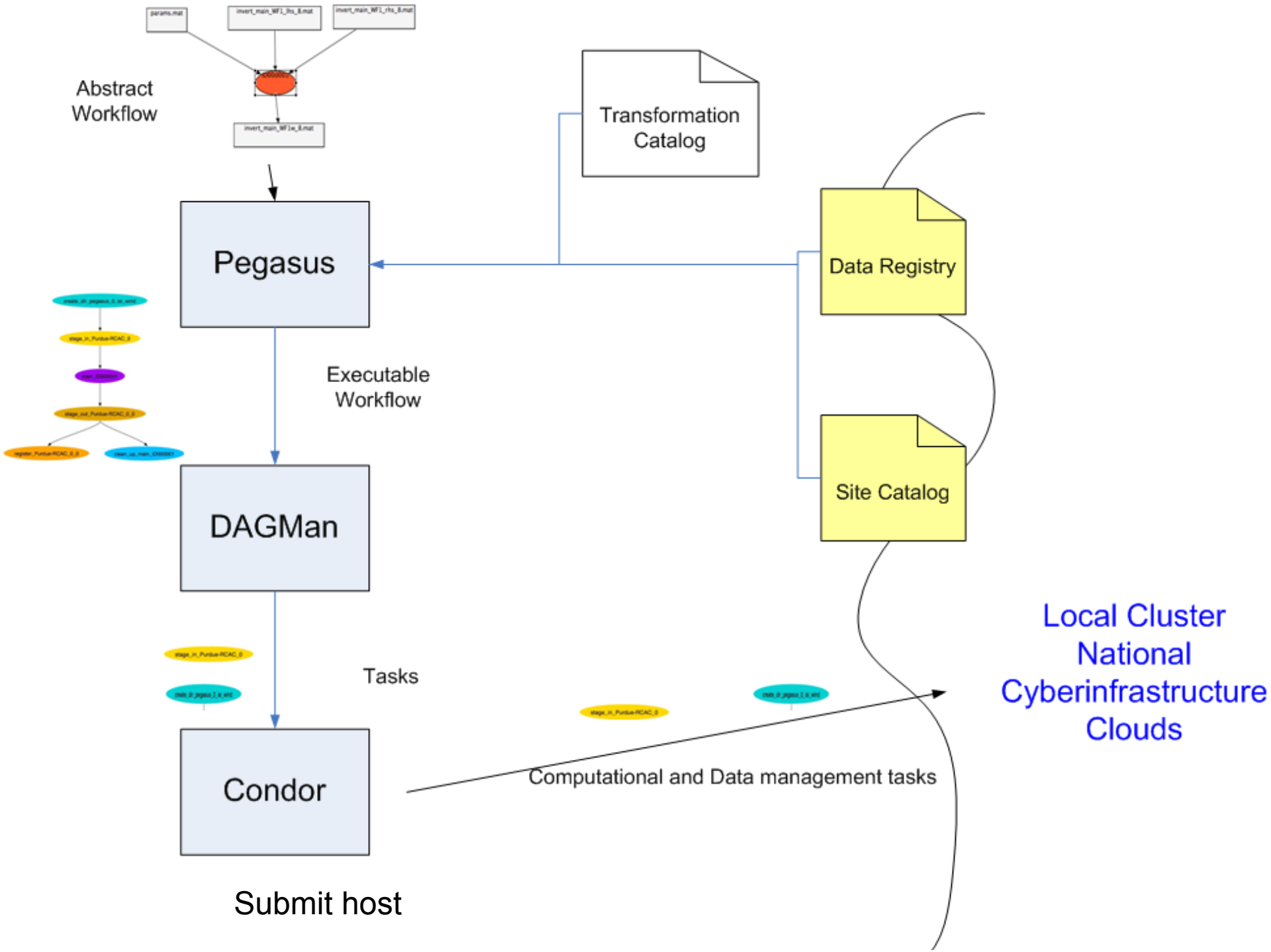
Pegasus
Workflow Management System (est. 2001)

- A collaboration between USC and the Condor Team at UW Madison
- Maps a resource-independent "abstract" workflow onto resources and executes the "concrete" workflow
- Used by a number of applications in a variety of domains
- Provides reliability—can retry computations from the point of failure
- Provides scalability—can handle large data and many computations (kbytes-TB of data, 1-10^6 tasks)
- Infers data transfers, restructures workflows for performance
- Automatically captures provenance information
- Can run on resources distributed among institutions, laptop, campus cluster, Grid (OSG, XSEDE), Cloud (Amazon, FutureGrid)

Pegasus Workflow Management System

- A workflow "compiler"
  - Input: abstract workflow description, resource-independent
  - Auxiliary Info (catalogs): available resources, data, codes
  - Output: executable workflow with concrete resources
  - Automatically locates physical locations for both workflow tasks and data
  - Transforms the workflow for performance and reliability
- A workflow engine (DAGMan)
  - Executes the workflow on local or distributed resources (HPC, clouds)
  - Task executables are wrapped with *pegasus-kickstart* and managed by Condor *schedd*
- Provenance and execution traces are collected and stored
- Traces and DB can be mined for performance and overhead information



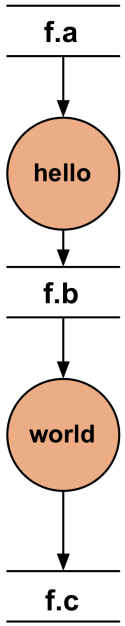

Abstract
Workflow
Transformation
Catalog
Pegasus
Data Registry
Executable
Workflow
Site Catalog
DAGMan
Tasks
Local Cluster
National
Cyberinfrastructure
Clouds
Condor
Computational and Data management tasks
Submit host

Generating executable workflows

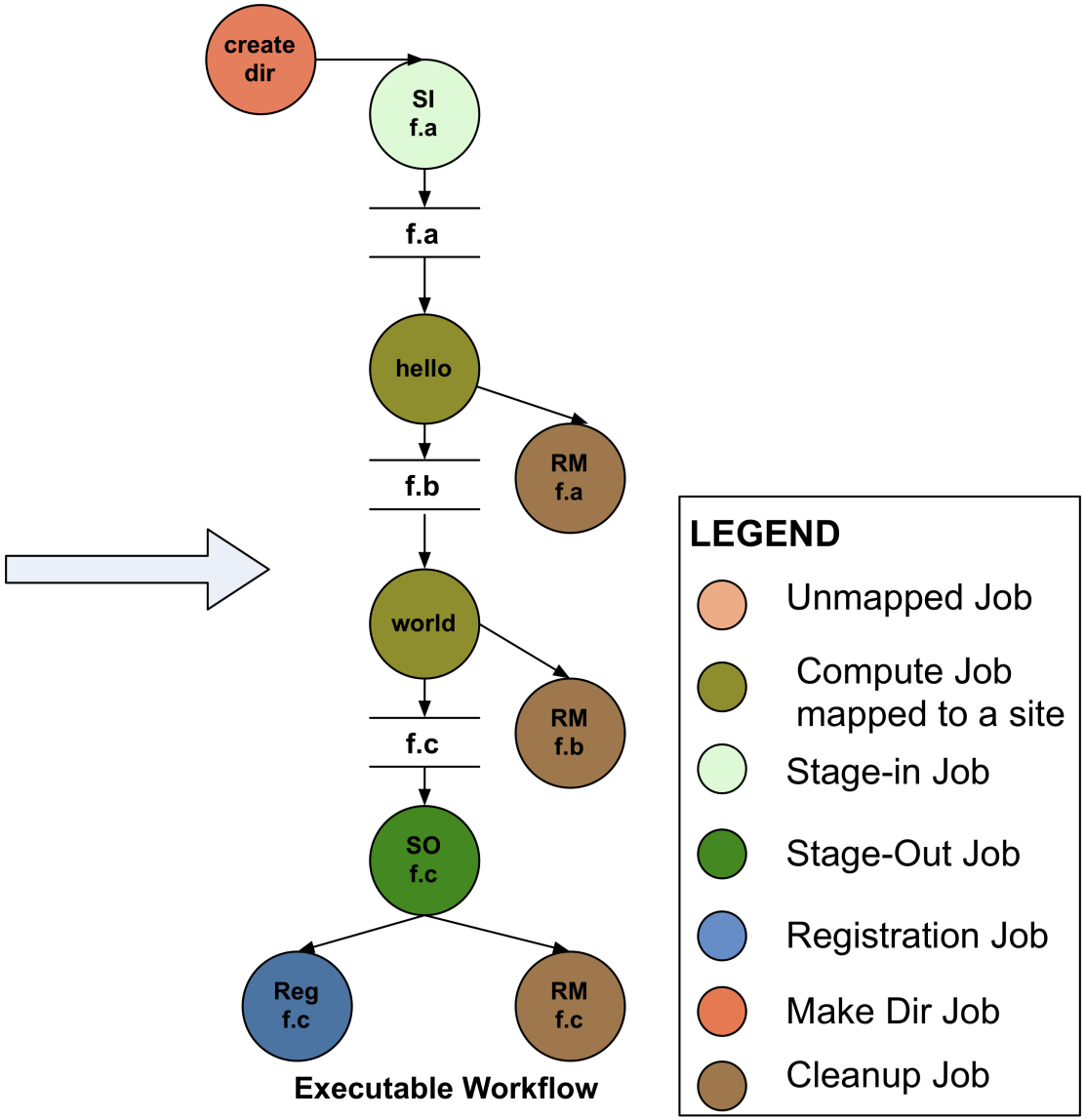
(DAX)

APIs for workflow specification (DAX--- DAG in XML)

Java, Perl, Python

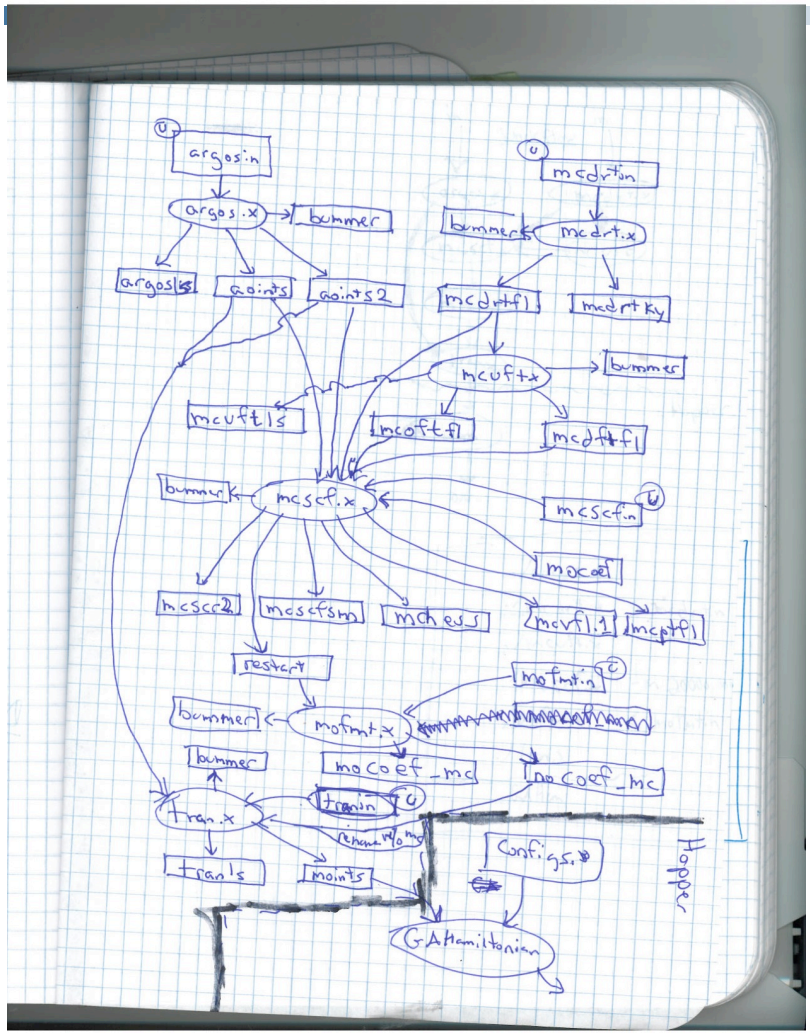


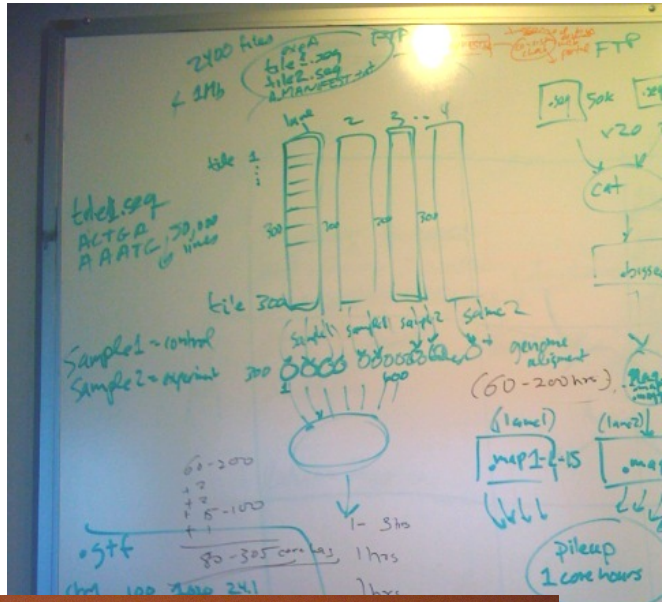
Abstract Workflow

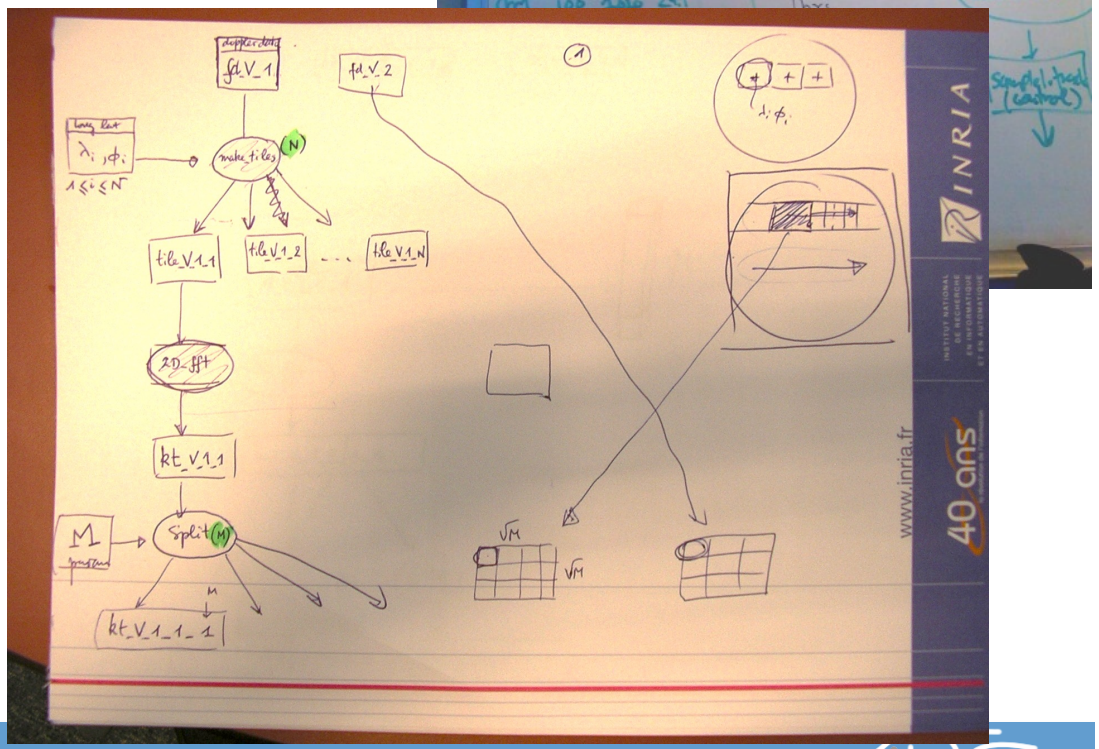


Executable Workflow

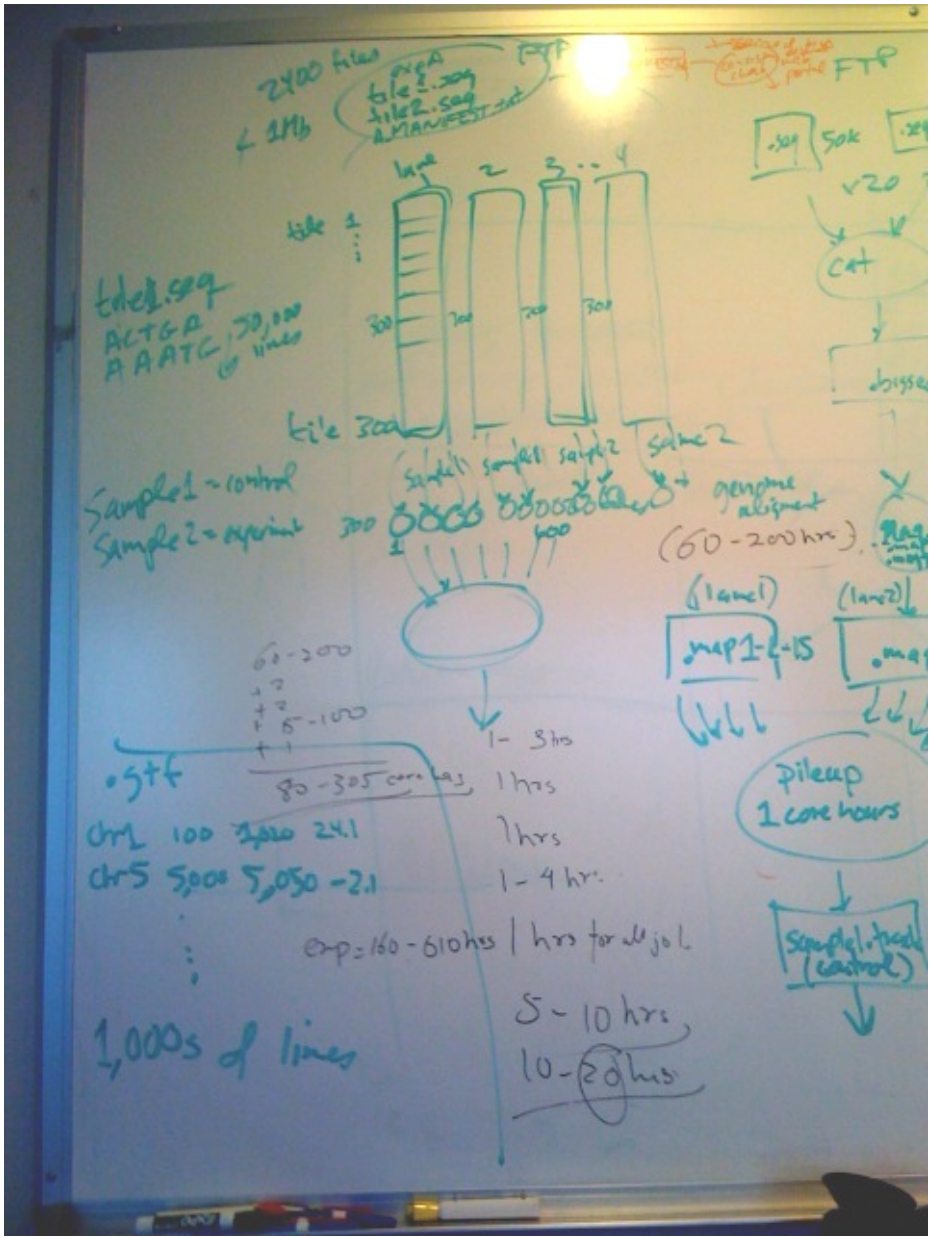
How do workflows start?

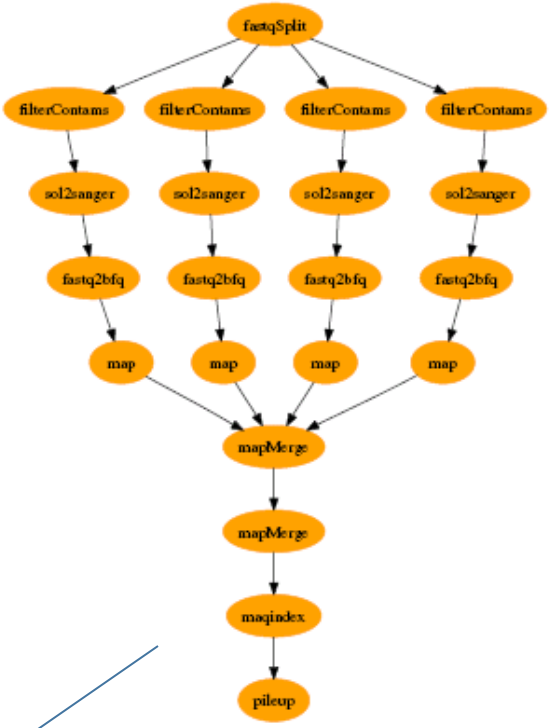




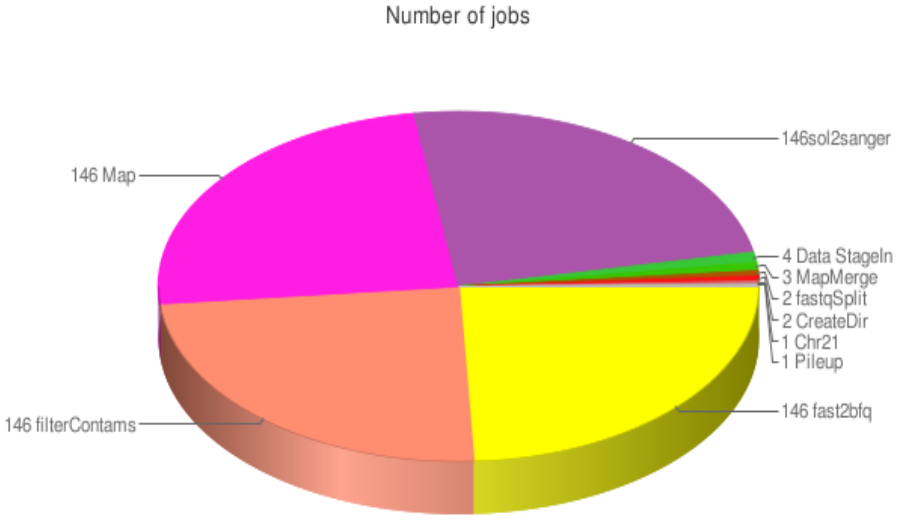


Time to solution ~ 2 weeks-3 months



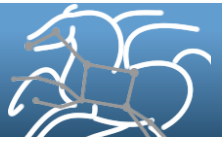


Execution on USC resources



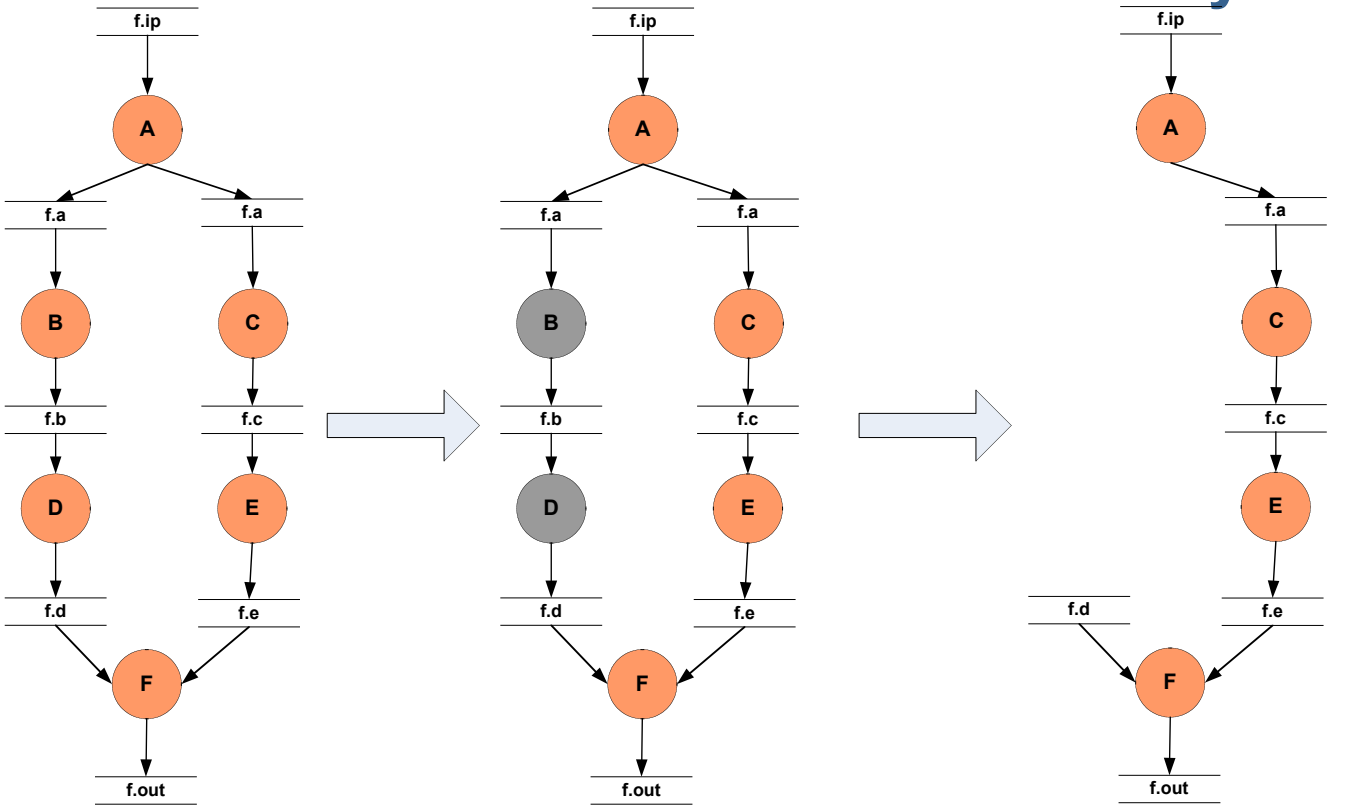
Pegasus optimizations address issues of:

- Failures in the execution environment or application
- Data storage limitations on execution sites
- Performance
  - Small workflow tasks
- Heterogeneous execution architectures
  - Different file systems (shared/non-shared)
  - Different system architectures (Cray XT, Blue Gene, …)



Sometimes fatal errors occur during workflow execution

Want to restart the workflow from where it left off
Sometimes intermediate data is already available



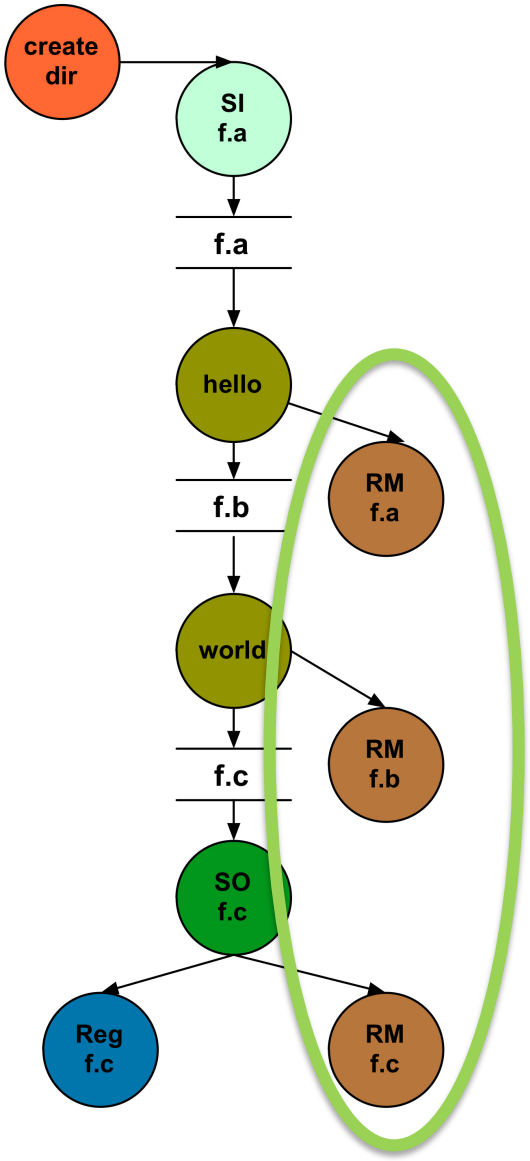
Workflow Reduction

Data Reuse

Workflow-level checkpointing

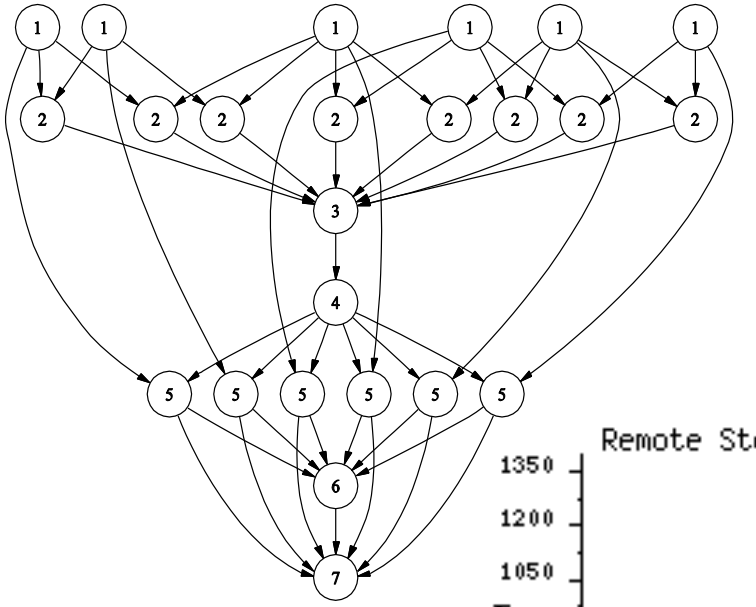
Storage limitations

"Small" amount of space

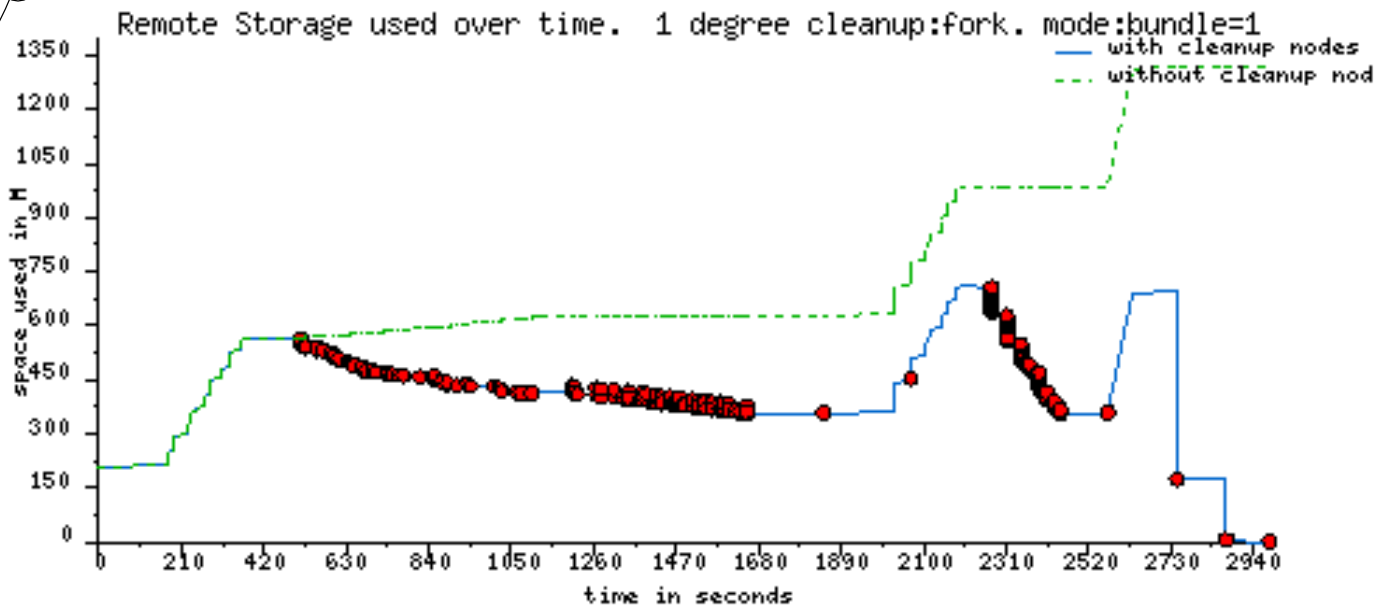


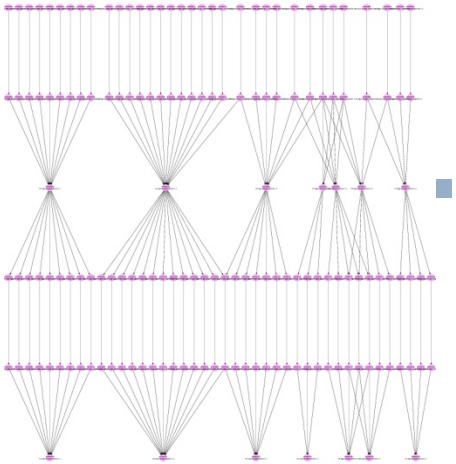
Automatically add tasks to "clean up" data no longer needed

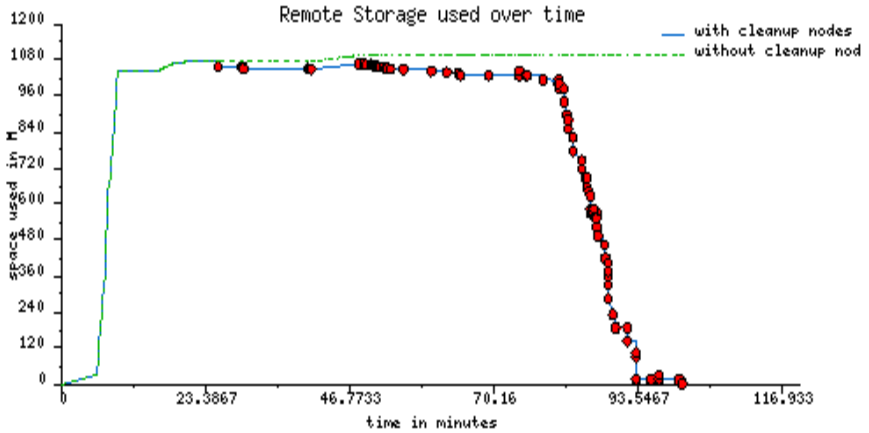
LIGO and Montage



1.25GB versus 4.5 GB

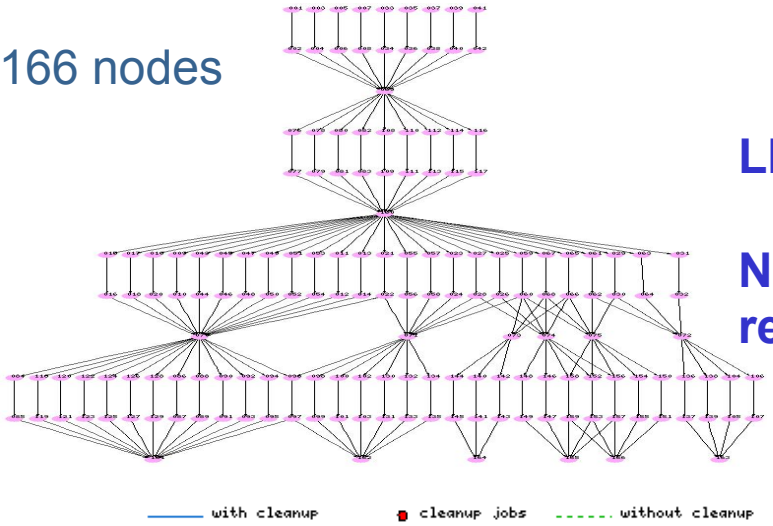






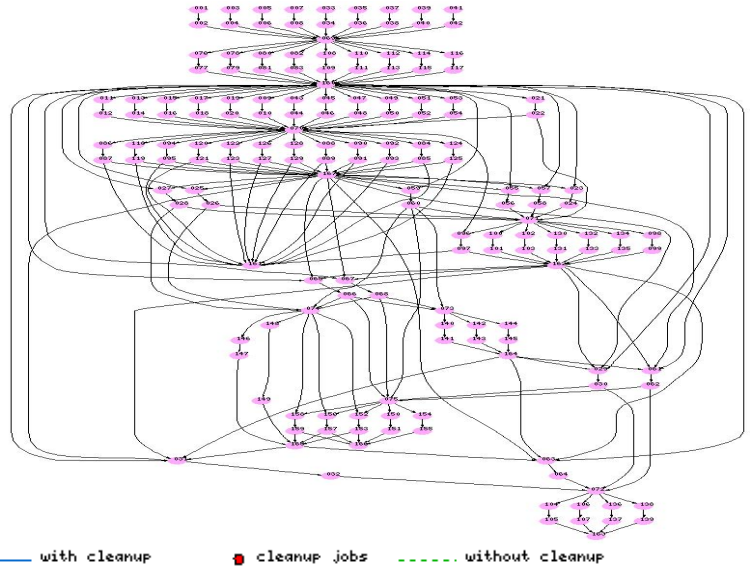
Full workflow:
185,000 nodes
466,000 edges
10 TB of input data
1 TB of output data.

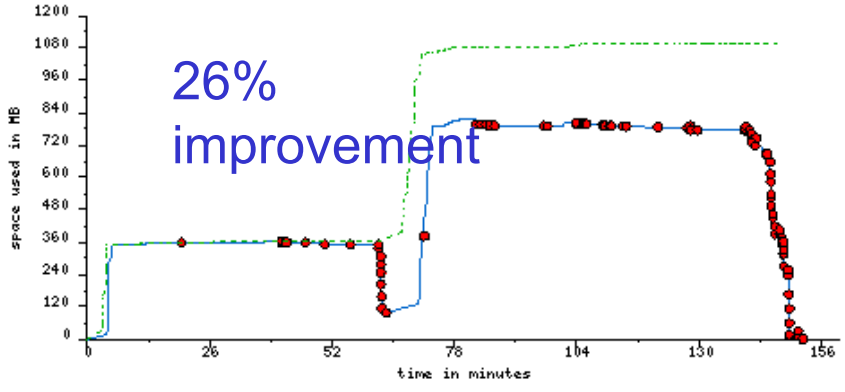
166 nodes



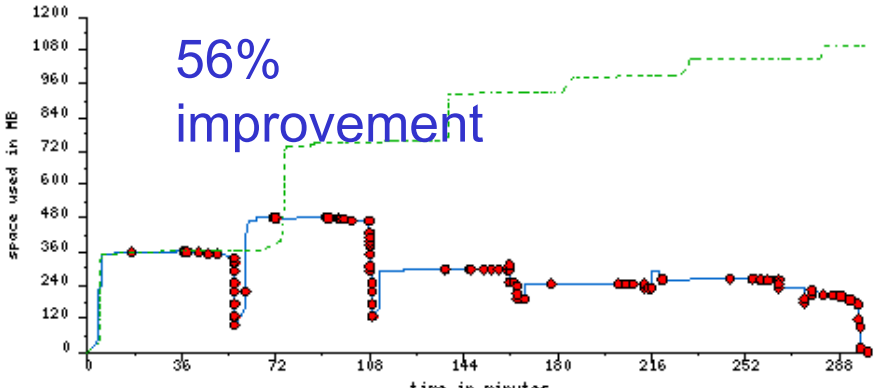
LIGO Workflows

Need additional restructuring





26% improvement

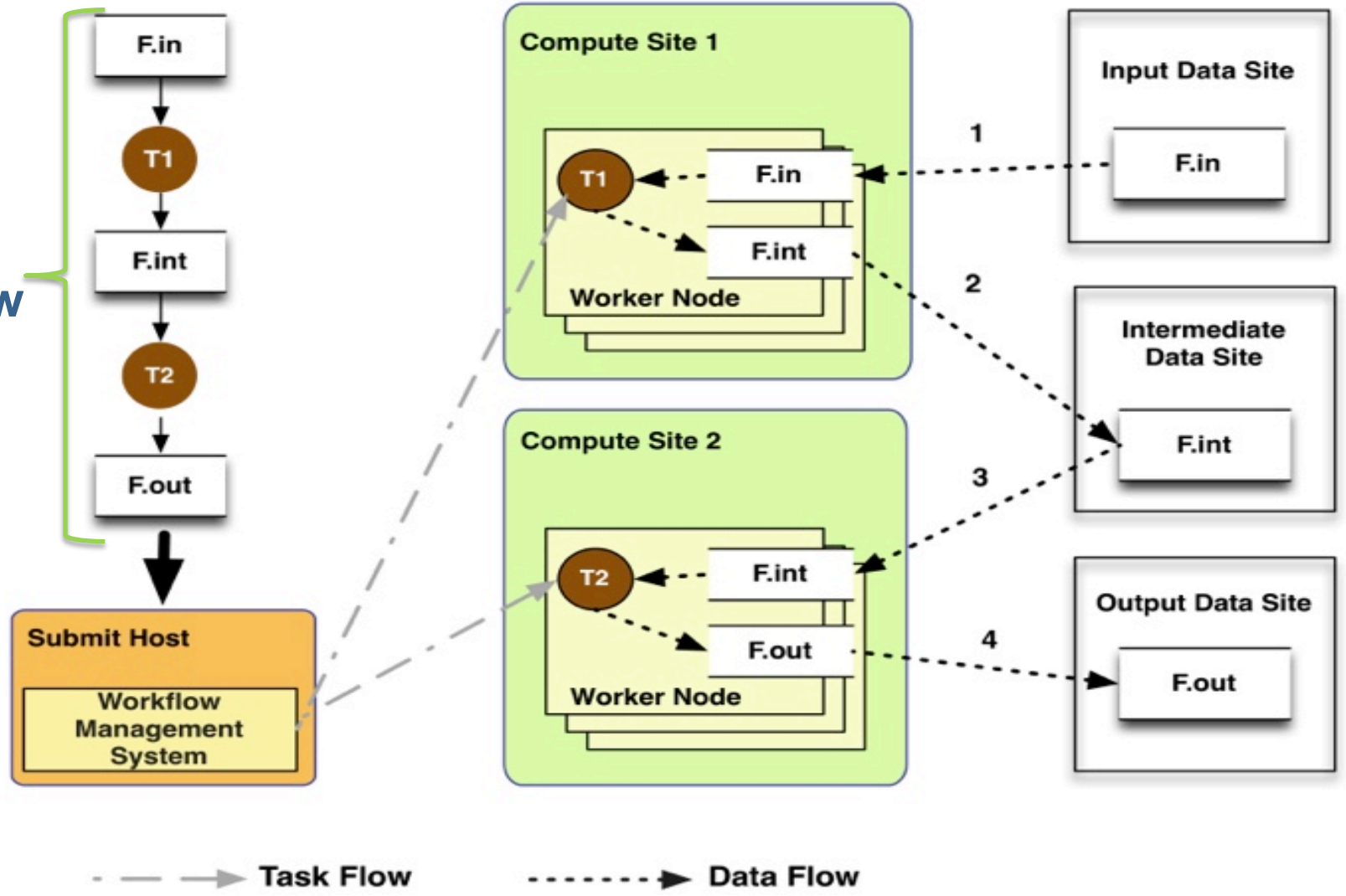


56% improvement

Storage limitations

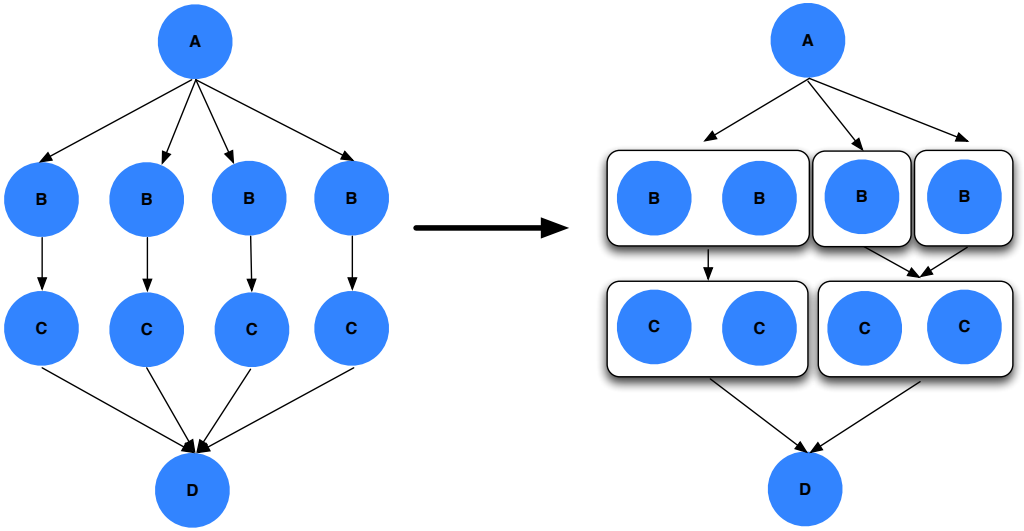
Variety of file system deployments: shared vs non-shared

User workflow



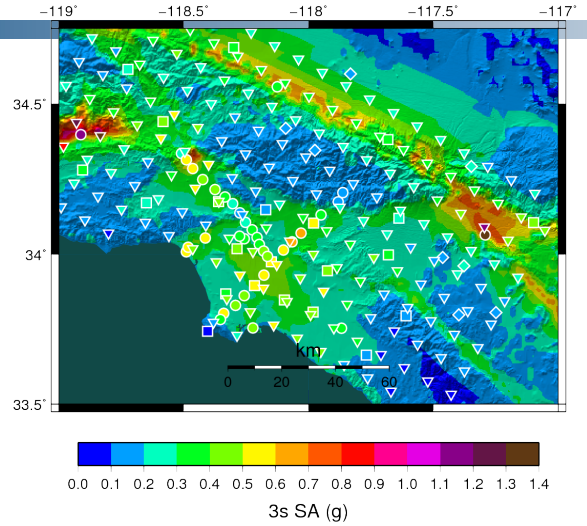
Workflow Restructuring to improve application performance

- **Cluster small running jobs together to achieve better performance**
- **Why?**
  - Each job has scheduling overhead – submit host
  - Ideally users should run a job on the grid/cloud that takes at least 10/30/60/? minutes to execute
  - Clustered tasks can reuse common input data – less data transfers



Level-based clustering
Label-based clustering
Time-based clustering

Southern California Earthquake Center

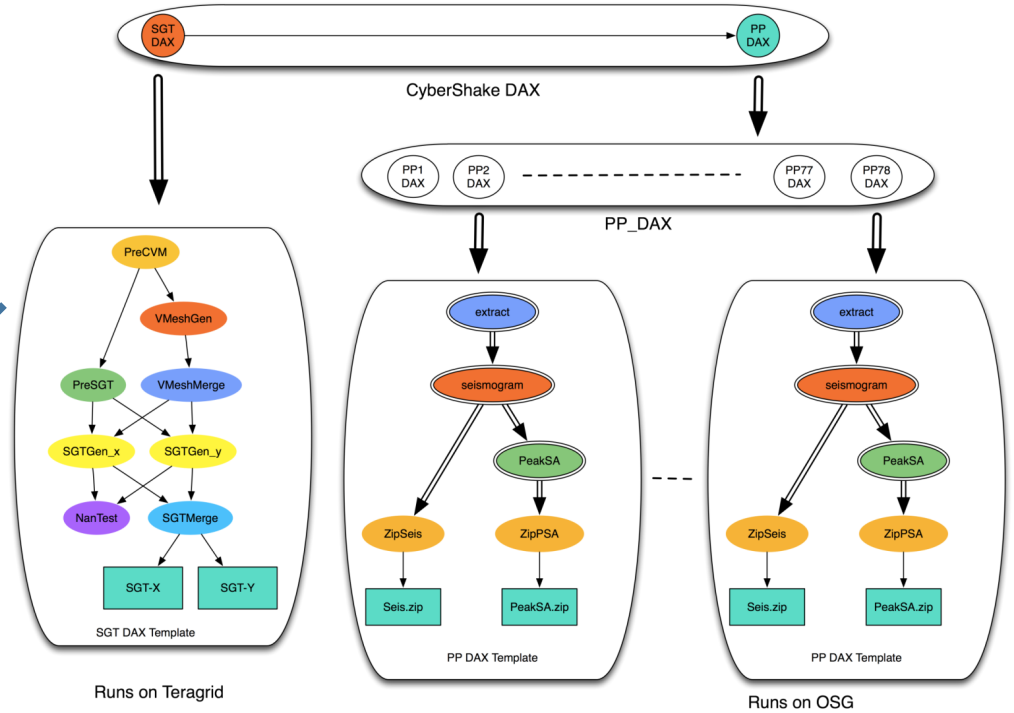


CyberShake PSHA Workflow

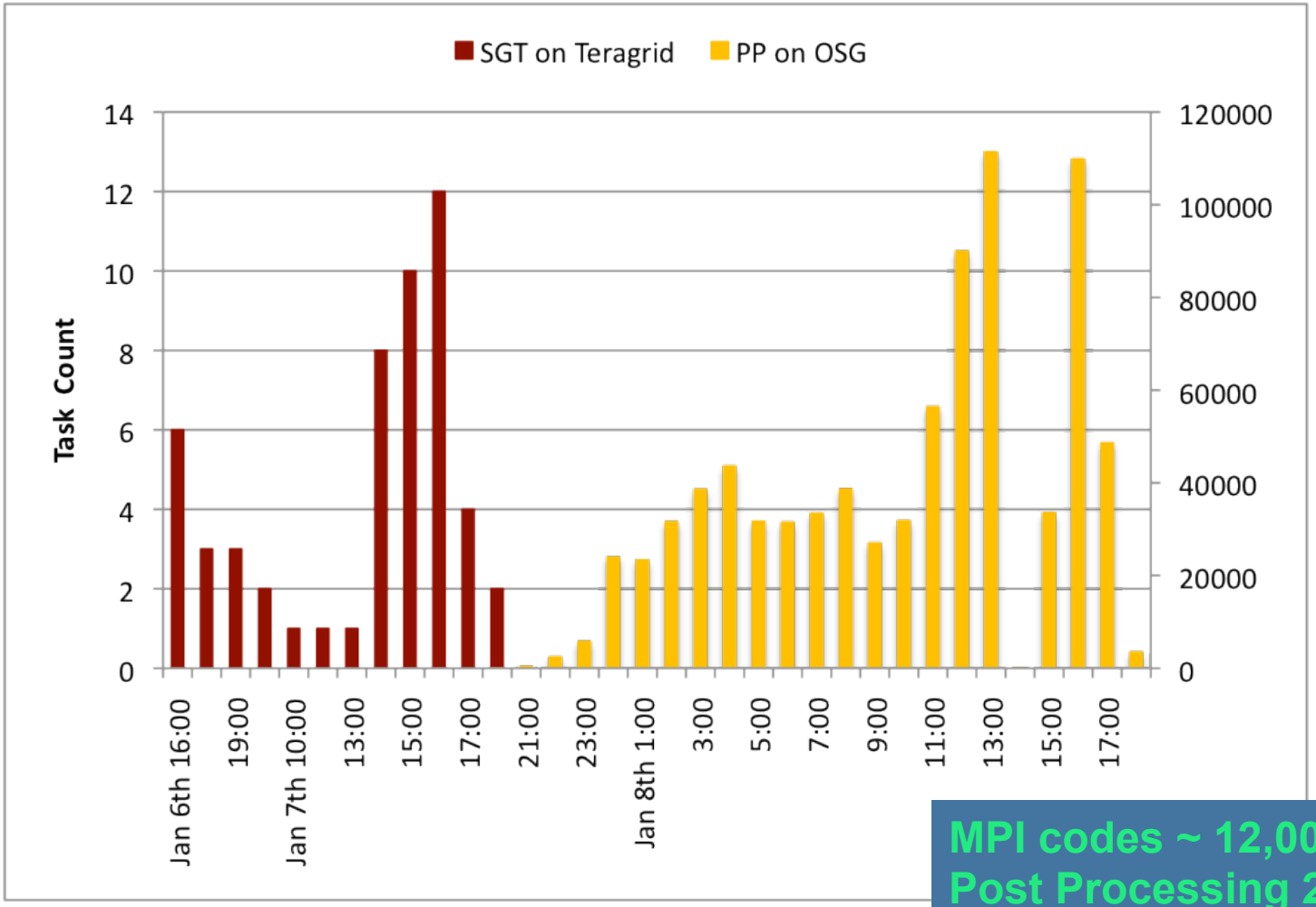
- **Description**
  - Builders ask seismologists: "What will the peak ground motion be at my new building in the next 50 years?"
  - Seismologists answer this question using Probabilistic Seismic Hazard Analysis (PSHA)

239 Workflows

- **Each site in the input map corresponds to one workflow**
- **Each workflow has:**
  - **820,000 tasks**

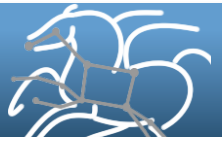


Workflows have different computational needs



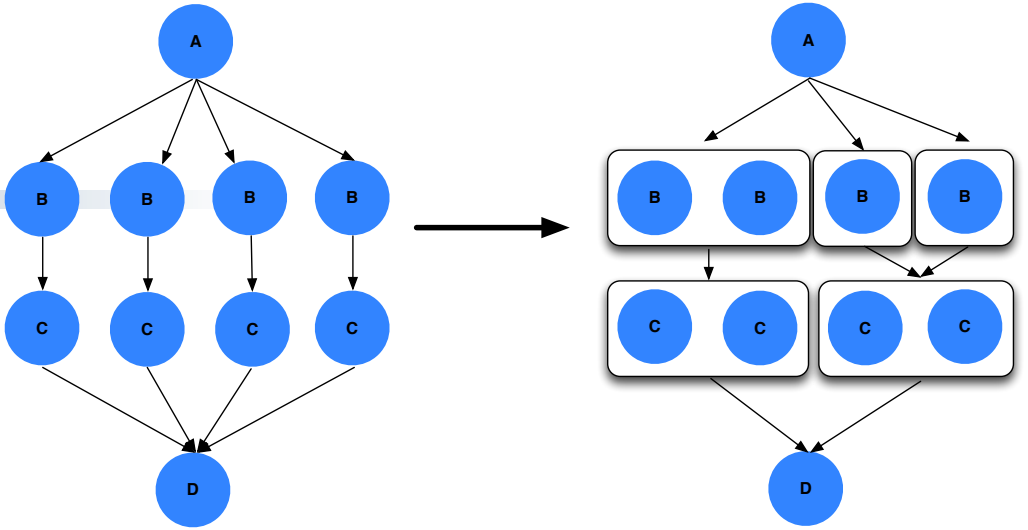
SoCal Map needs 239 of those

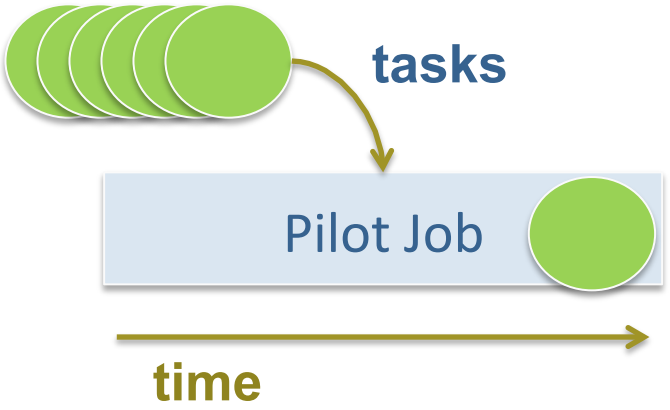
MPI codes ~ 12,000 CPU hours,
Post Processing 2,000 CPU hours
Data footprint ~ 800GB



Solutions

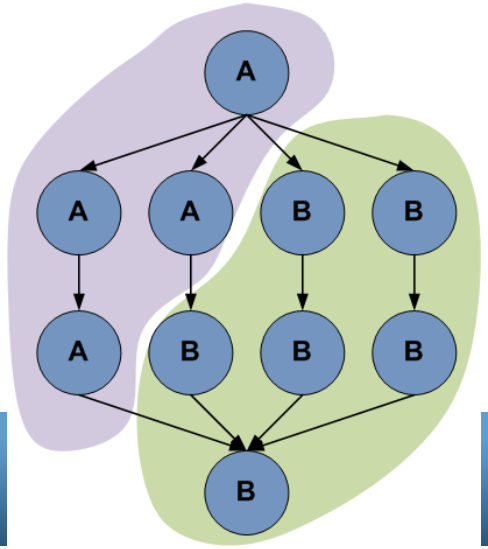
Cluster tasks





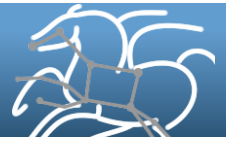
Use "pilot" jobs to dynamically provision a number of resources at a time

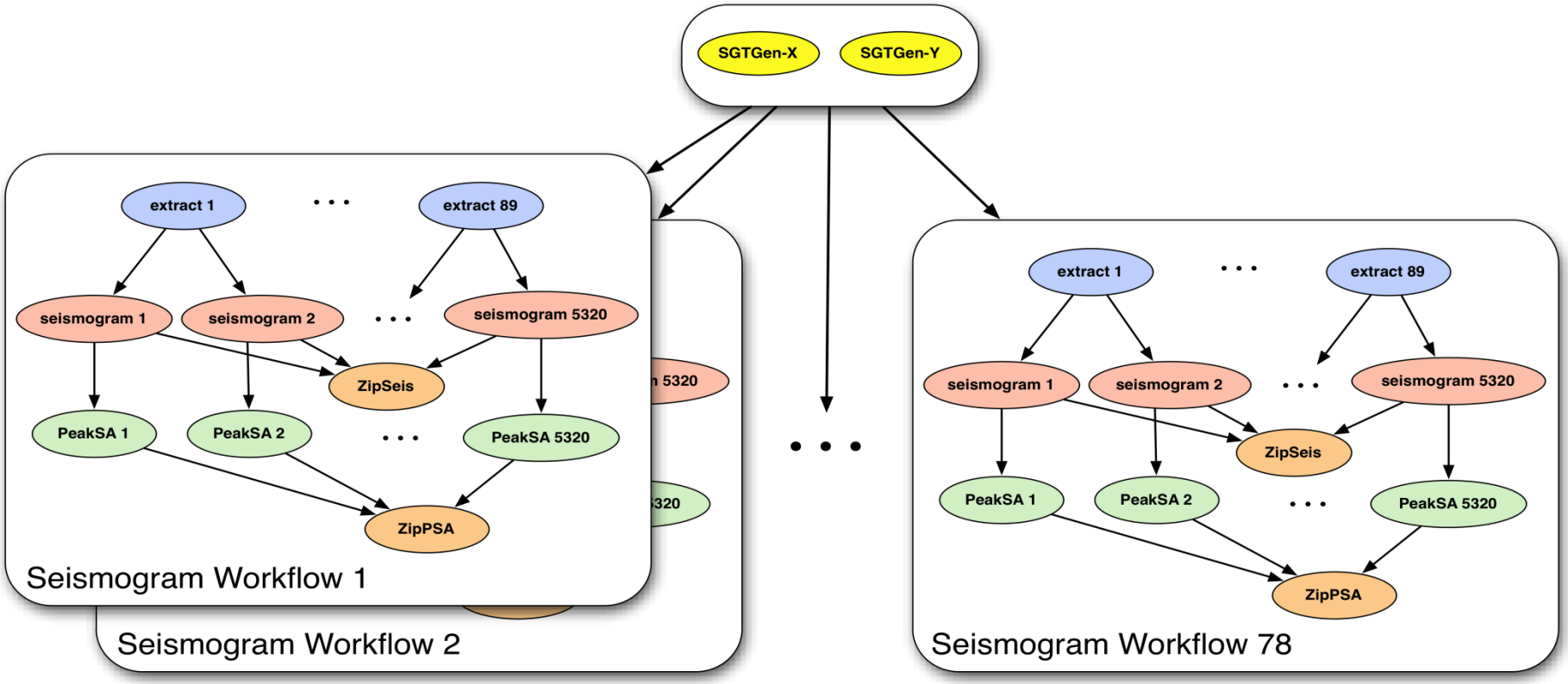
Develop an MPI-based workflow management engine to manage sub-workflows



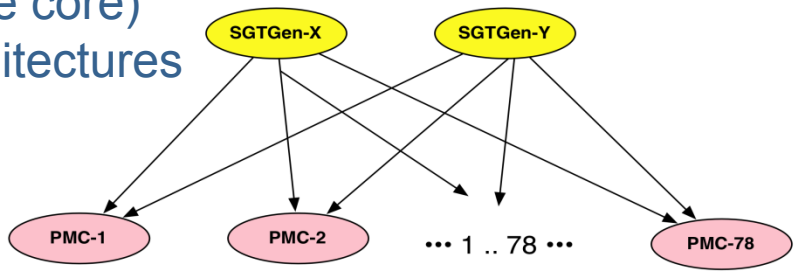
Pegasus-MPI-Cluster

- 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
- Works on most HPC systems
  - Requires: MPI, a shared file system, and fork()
- Allows sub-graphs of a Pegasus workflow to be submitted as monolithic grid jobs to remote resources





Enables earthquake scientists (SCEC) to run post-processing (single core) computations on new architectures (Titan)



Workflow Monitoring - Stampede

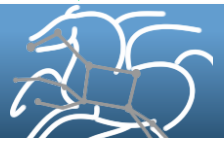
- **Leverage Stampede Monitoring framework with DB backend**
  - Populates data at runtime. A background daemon monitors the logs files and populates information about the workflow to a database
  - Stores workflow structure, and runtime stats for each task.

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

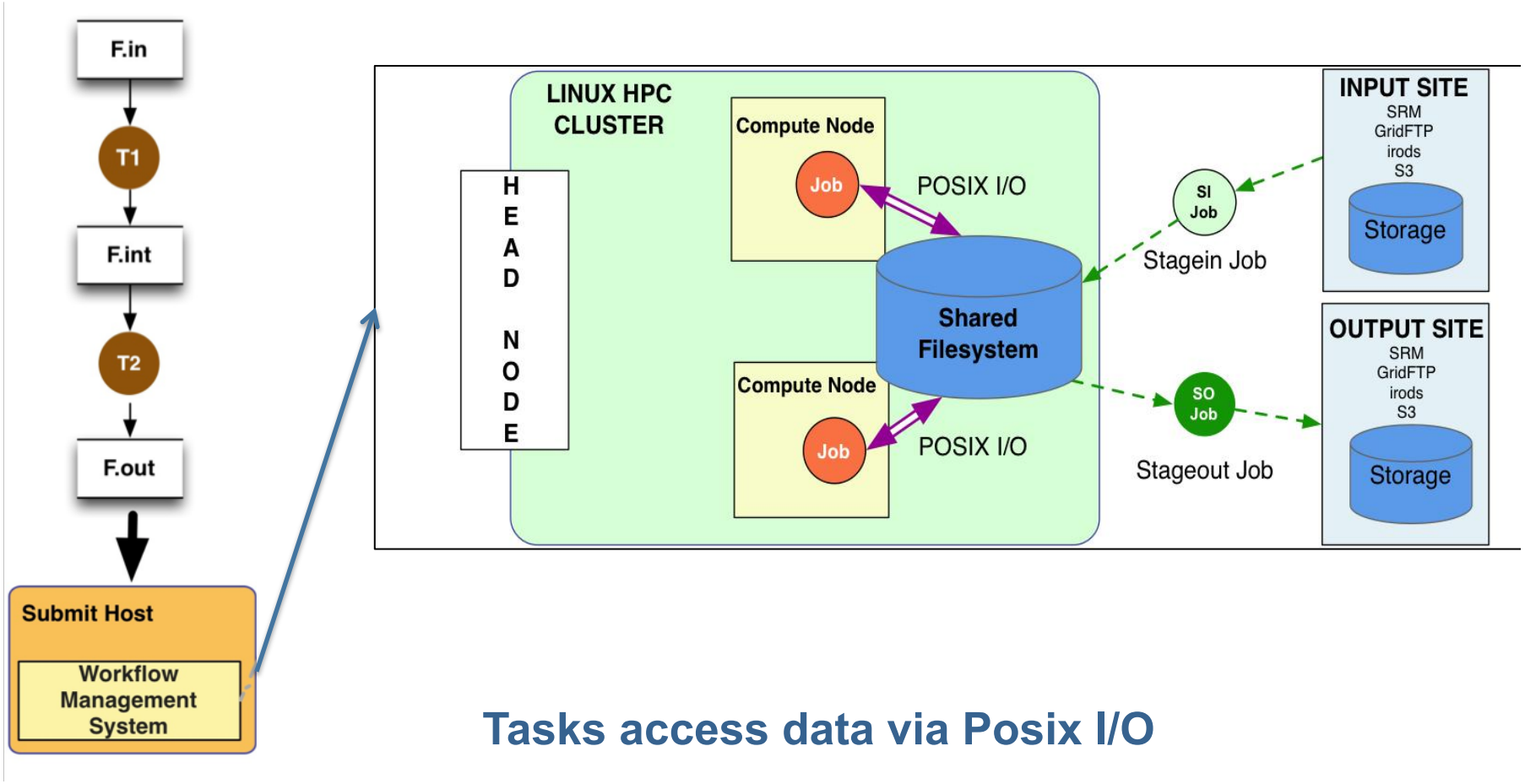
```
------------------------------------------------------------------------------
Type           Succeeded Failed  Incomplete  Total     Retries   Total+Retries
Tasks          135002    0       0           135002    0         135002
Jobs           4529      0       0           4529      0         4529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

Collaboration with Dan Gunter and Taghrid Samak, LBNL



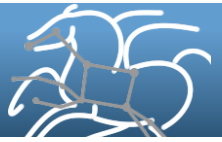
Typical Deployment



Tasks access data via Posix I/O

Posix Access for Tasks in the workflow

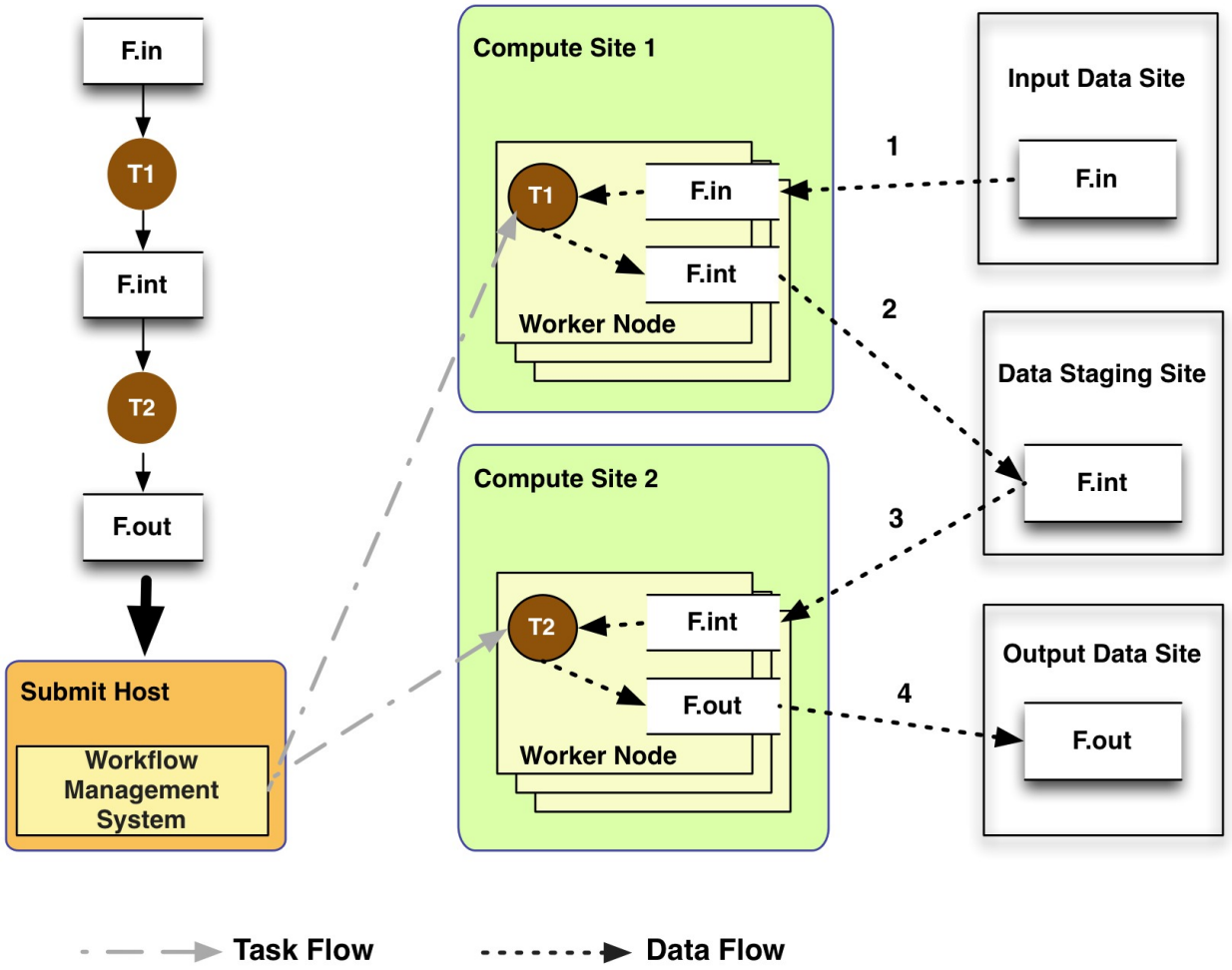
- How do you ensure Posix access for the tasks?
  - Place it on a shared filesystem shared across nodes.
  - Place it directly on local filesystem of the worker node from the input site.
- Direct Transfers to local filesystem
  - Job starts and retrieves input data from input site.
  - Not efficient for large datasets that are shared across jobs.
- Shared Filesystem sounds appealing but problems for Big Data workflows
  - Shared storage at a computational site maybe limited. Cannot accommodate all files required for a large workflow.
  - In some cases, shared filesystem may have limited scalability---NFS
  - Harder to setup a shared filesystem in a dynamic environment like computational clouds
  - Some systems just don't support it



Object Storage for Workflows

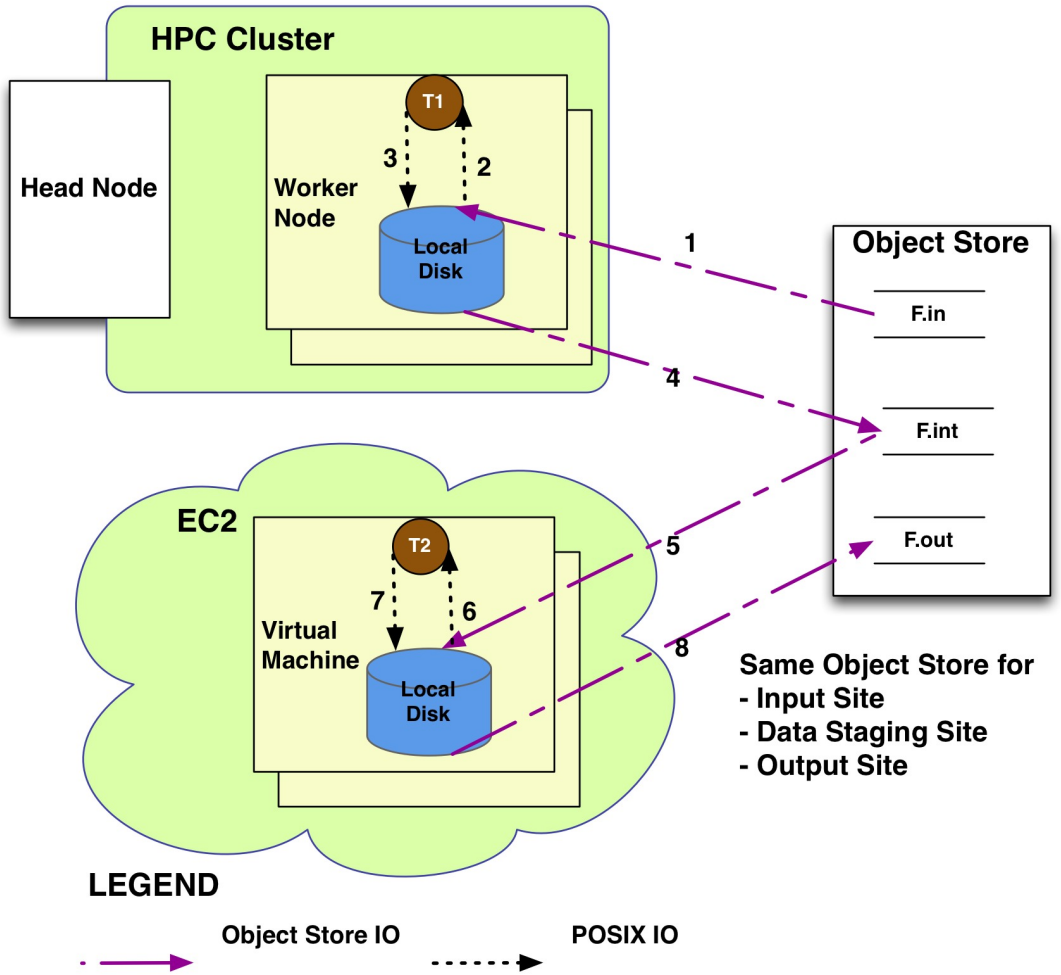
- Clouds such as Amazon provide object stores--- S3
- Object Store: high level storage service with limited operations
  - Store, retrieve and delete data objects (files)
  - Doesn't provide byte level access
    - Cannot open a file in an object store, read and update it and then close it.
    - Instead a client needs to download the file, update it and then store as a new object.
- View traditional Grid services like GridFTP, SRM, IRODS as object stores
  - Store, retrieve and delete data (files)
  - Don't support random read or writes like object stores.
  - This generalization is important to lay out the different data management models.

General Workflow Execution Model, Cannot Assume Shared FS



- Input Data Site, Compute Site and Output Data Sites can be co-located
  - Example: Input data is already present on the compute site.

Exclusive Use of Object Stores



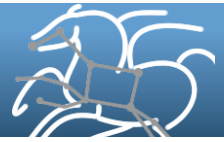
Advantages

- Can leverage scalable stores
- Distribute computations across resources, such as supporting spillover from local resources to cloud resources.
- Great bandwidth

Disadvantages

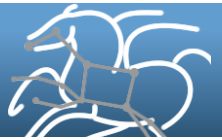
- Duplicate Transfers
- Latencies in transferring large number of files
- Added costs for duplicate transfers.

The Workflow System retrieves files from Object Store and makes it available to the workflow task on the local disk on a worker node.

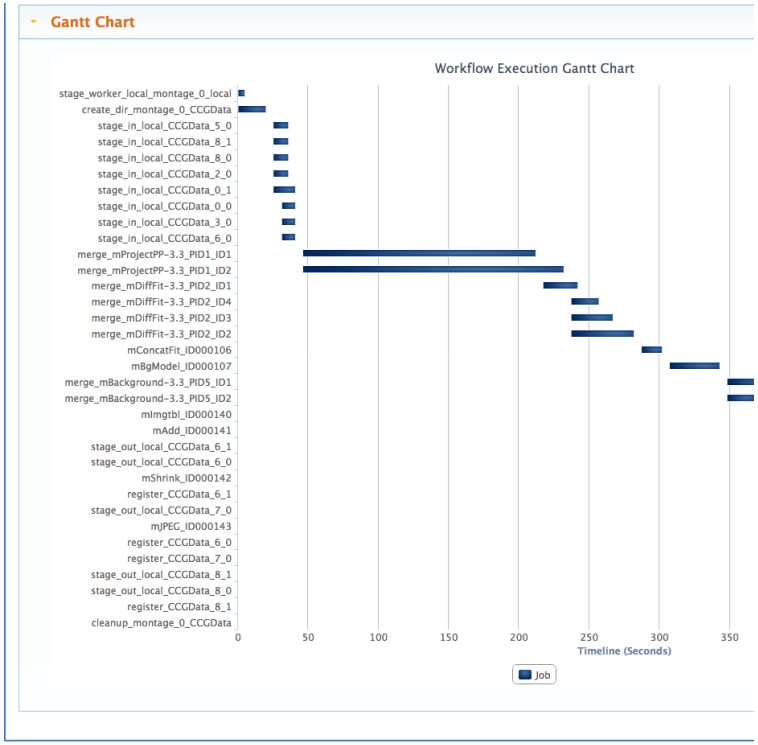


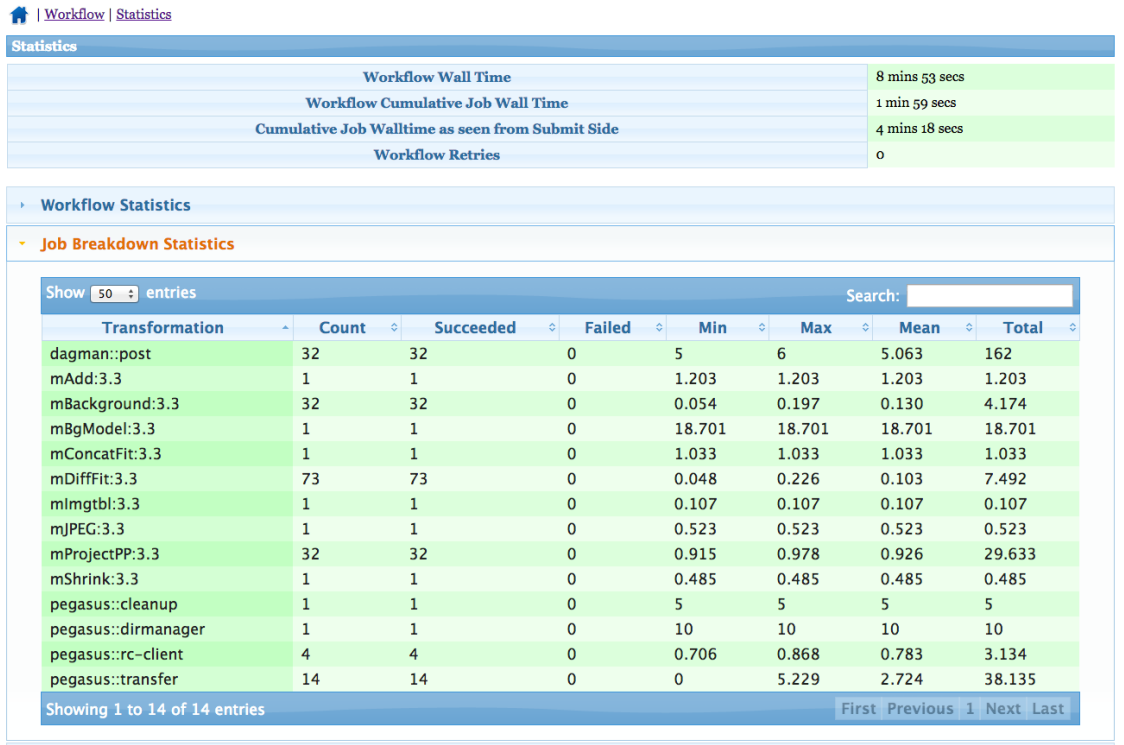
Pegasus-kickstart

- Lightweight C based executable to launch jobs
- Captures job runtime provenance and logs it as a XML record
- Following information is captured about each job on all supported platforms
  - exit code with which the job it launched exited
  - start time and duration of the job
  - hostname and IP address of the host the job ran on
  - stdout and stderr of the job
  - arguments with which it launched the job
  - directory in which the job was launched
  - environment that was set for the job before it was launched
- Additional profiling
  - peak memory usage (resident set size, and vm size)
  - total I/O read and write,
  - Pid
  - all files accessed (total read and write per file)



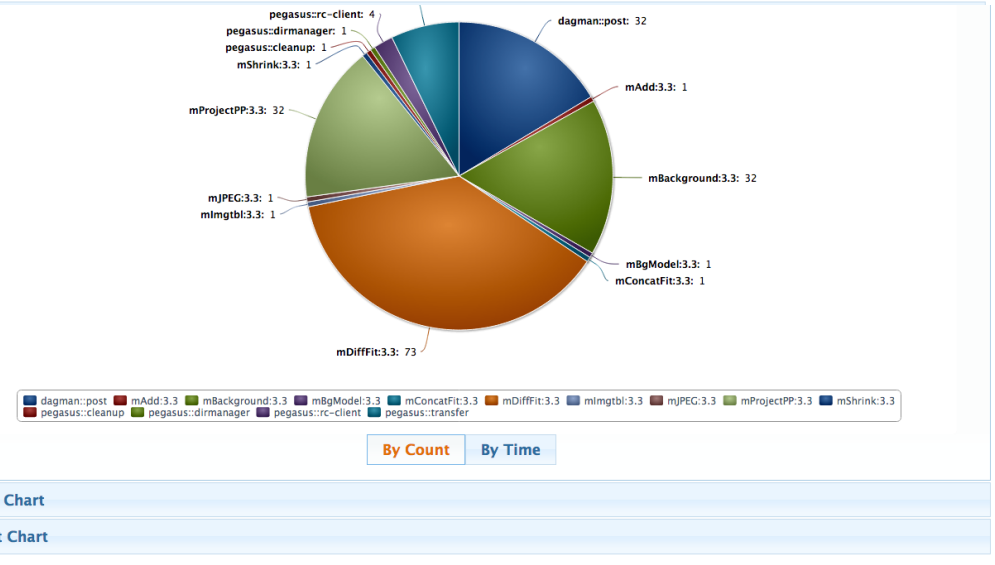
Workflow Monitoring Dashboard – *pegasus-dashboard*





Status, statistics, timeline of jobs

Helps pinpoint errors

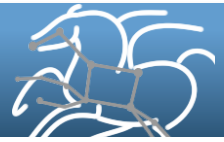


USC Viterbi School of Engineering

Tools to calculate job statistics

Task Type	Count	Runtime(s)	IO Read (MB)	IO Write (MB)	Memory Peak(MB)	CPU Utilization(%)
mProjectPP	2102	1.73	2.05	8.09	11.81	86.96
mDiffFit	6172	0.66	16.56	0.64	5.76	28.39
mConcatFit	1	143.26	1.95	1.22	8.13	53.17
mBgModel	1	384.49	1.56	0.10	13.64	99.89
mBackground	2102	1.72	8.36	8.09	16.19	8.46
mImgtbl	17	2.78	1.55	0.12	8.06	3.48
mAdd	17	282.37	1102	775.45	16.04	8.48
mShrink	16	66.10	412	0.49	4.62	2.30
mJPEG	1	0.64	25.33	0.39	3.96	77.14

Table 1. Execution profile of the Montage workflow, averages calculated

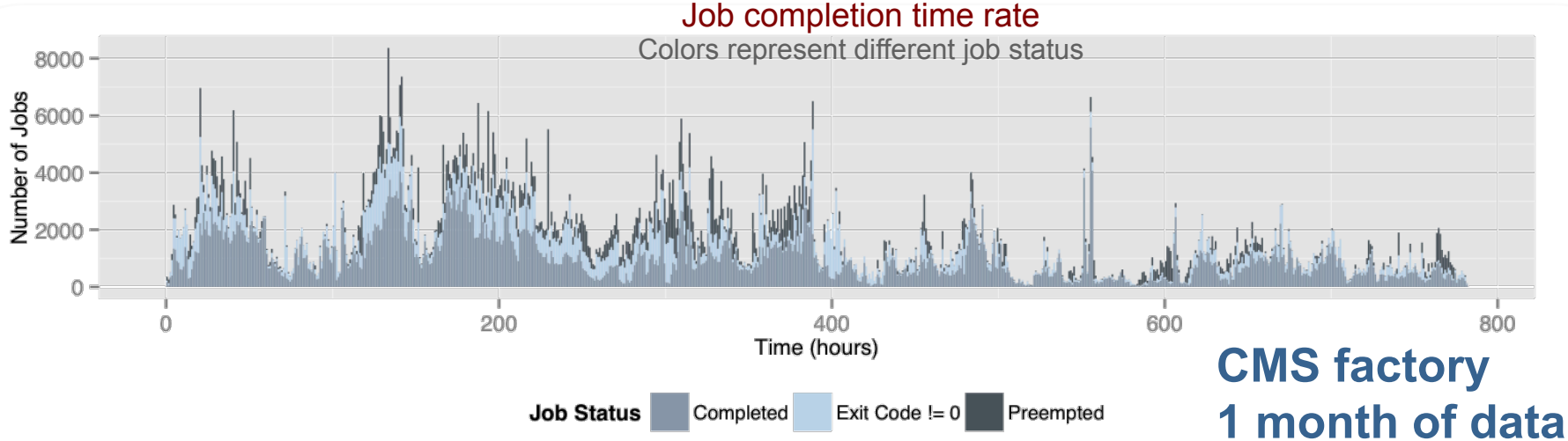


dv/dt – Accelerating the rate of progress towards extreme scale collaborative science, (9/12-8/15, DOE)

Miron Livny UWMadison, Bill Allcock ANL, Ewa Deelman USC, Doug Thain UND, Frank Wuerthwein UCSD)

Objectives

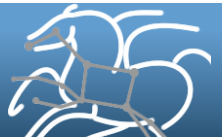
- Design a computational framework that enables computational experimentation at scale while supporting the model of "**submit locally, compute globally**"
- Focus on **Estimating** application resource needs, **Finding** the appropriate computing resources, **Acquiring** those resources, **Deploying** the applications and data on the resources, **Managing** applications and resources during run
- Task resource profiling and resource estimation



CMS factory
1 month of data

Task Characterization/Execution

- Collect and archive data from existing infrastructure deployments
- Understand the resource needs of a task (memory, disk, CPU)
- Establish expected values and limits for task resource consumption
- Launch tasks on the correct resources
- Monitor task execution and resource consumption, interrupt tasks that reach resource limits
- Possibly re-launch tasks on different resources
- Task characterization needs to be an online process



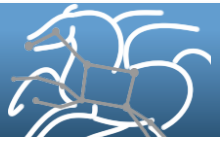
Predictive Modeling and Diagnostic Monitoring of Extreme Science Workflow (9/14-8/17, DOE)

Ewa Deelman USC, Chris Carothers RPI, Anirban Mandal RENCI
Brian Tierney LBNL, Jeff Vetter ORNL

Objective: Understand complex scientific workflow applications and infrastructure behaviors and to translate this understanding into flexible, end-to-end analytical models that can effectively predict the behavior of extreme scale workflows on current and future infrastructures

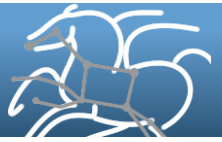
Approach:

- Engage DOE science teams from simulation (e.g., Earth Systems Modeling (ESM) and instrument facilities (e.g., Spallation Neutron Source(SNS) to create example workflow scenarios
- Develop a general analytical modeling methodology that captures the end-to-end performance of these workflow scenarios using a structured modeling approach
- Validate the analytical models using empirical measurement and simulation
- Employ the analytical performance models to facilitate prototype capabilities that include anomaly detection and diagnosis, resource management and adaptation, and infrastructure design and planning



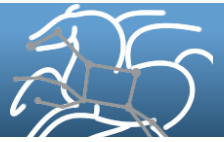
Benefits of Pegasus

- **Provides Support for Complex Computations**
  - Can be hidden behind a portal
- **Portability / Reuse**
  - User created workflows can easily be run in different environments without alteration (XSEDE, OSG, FutureGrid, Amazon)
- **Performance**
  - The Pegasus mapper can reorder, group, and prioritize tasks in order to increase the overall workflow performance
- **Scalability**
  - Pegasus can easily scale both the size of the workflow, and the resources that the workflow is distributed over.



Benefits of Pegasus

- **Provenance**
  - Performance and 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**
  - 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.
- **Analysis tools**



If you are interested in Pegasus

- Pegasus: http://pegasus.isi.edu
- Tutorial and documentation: http://pegasus.isi.edu/wms/docs/latest/
- Virtual Machine with all software and examples http://pegasus.isi.edu/downloads
- Take look at some Pegasus applications: http://pegasus.isi.edu/applications
- Support: pegasus-users@isi.edu pegasus-support@isi.edu