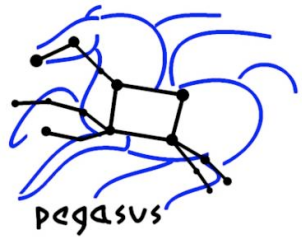
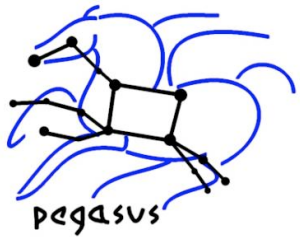


Grid Workflows - Pegasus WMS

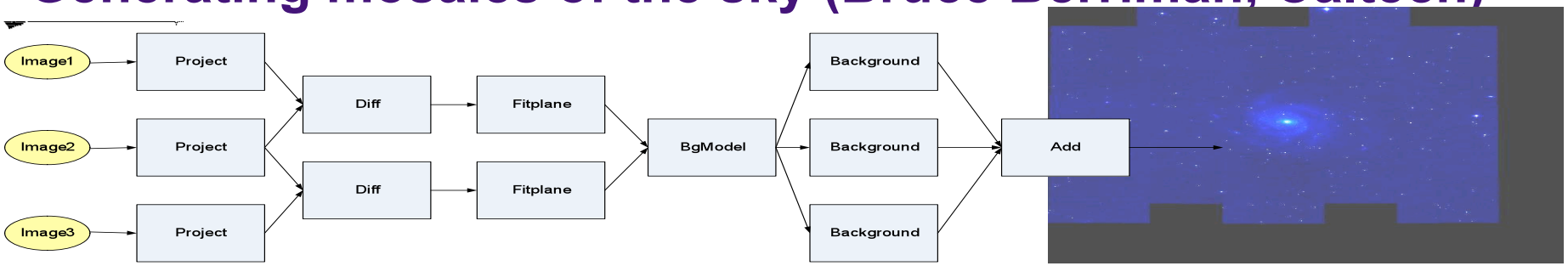
Gaurang Mehta[1], Ewa Deelman[1], Kent Wenger [2]
(gmehta@isi.edu, wenger@cs.wisc.edu)
[1] Center for Grid Technologies, USC Information Sciences Institute
[2] University of Wisconsin Madison, Madison, WI

Types of Workflow Applications



- **Providing a service to a community (Montage project)**
  - Data and derived data products available to a broad range of users
  - A limited number of small computational requests can be handled locally
  - For large numbers of requests or large requests need to rely on shared cyberinfrastructure resources
  - On-the fly workflow generation, portable workflow definition
- **Supporting community-based analysis (SCEC project)**
  - Codes are collaboratively developed
  - Codes are "strung" together to model complex systems
  - Ability to correctly connect components, scalability
- **Processing large amounts of shared data on shared resources (LIGO project)**
  - Data captured by various instruments and cataloged in community data registries.
  - Amounts of data necessitate reaching out beyond local clusters
  - Automation, scalability and reliability

Generating mosaics of the sky (Bruce Berriman, Caltech)

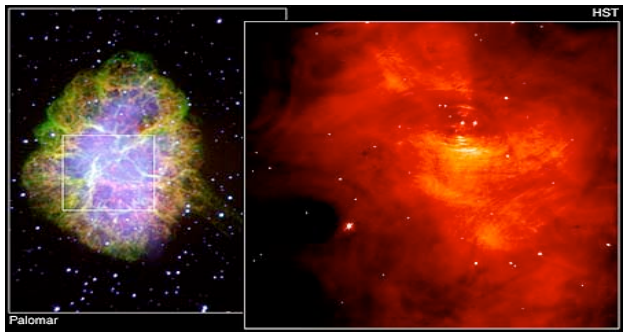


Size of the mosaic is degrees square*	Number of jobs	Number of input data files	Number of Intermediate files	Total data footprint	Approx. execution time (20 procs)
1	232	53	588	1.2GB	40 mins
2	1,444	212	3,906	5.5GB	49 mins
4	**4,856**	**747**	**13,061**	**20GB**	**1hr 46 mins**
6	8,586	1,444	22,850	38GB	2 hrs. 14 mins
10	20,652	3,722	54,434	97GB	6 hours

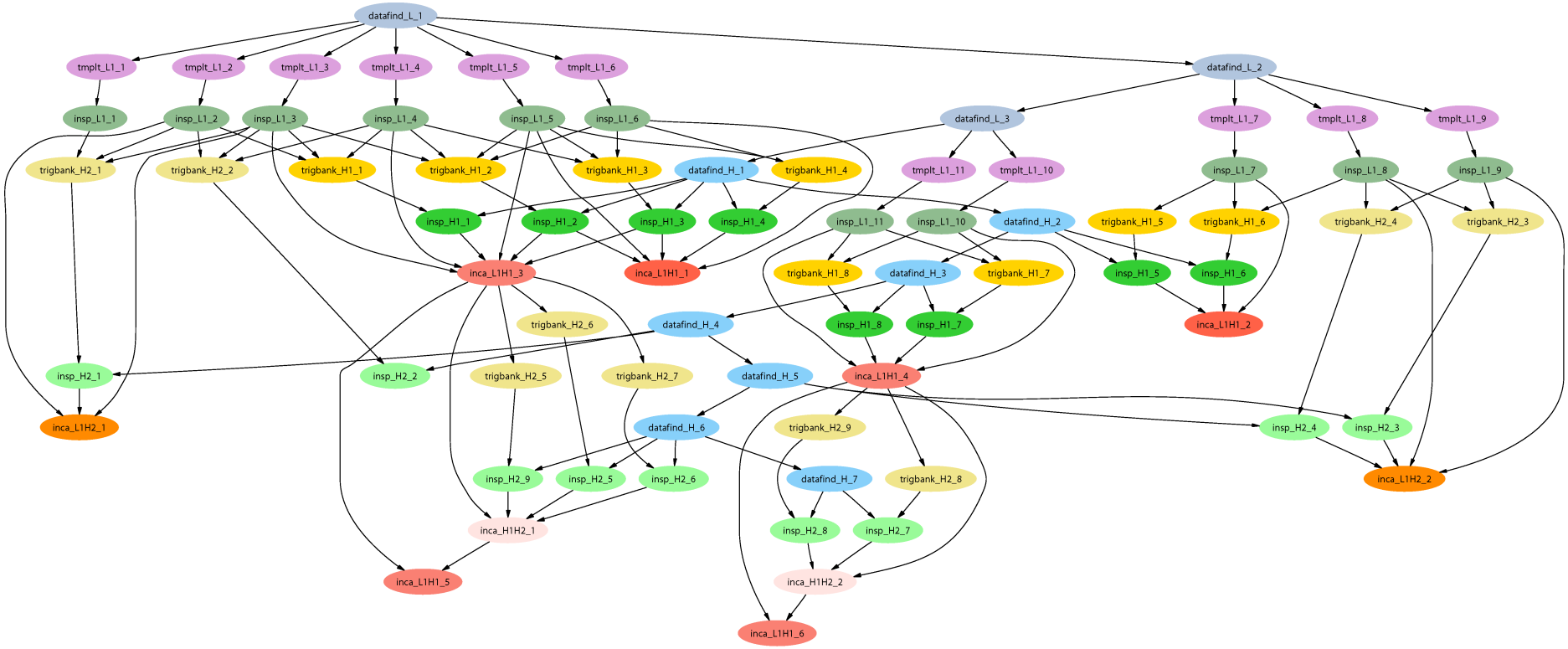
**The full moon is 0.5 deg. sq. when viewed form Earth, Full Sky is ~ 400,000 deg. sq.*

LIGO Scientific Collaboration

- Continuous gravitational waves are expected to be produced by a variety of celestial objects
- Only a small fraction of potential sources are known
- Need to perform blind searches, scanning the regions of the sky where we have no a priori information of the presence of a source
  - Wide area, wide frequency searches
- Search is performed for potential sources of continuous periodic waves near the Galactic Center and the galactic core
- Search for binary inspirals collapsing into black holes.
- The search is very compute and data intensive

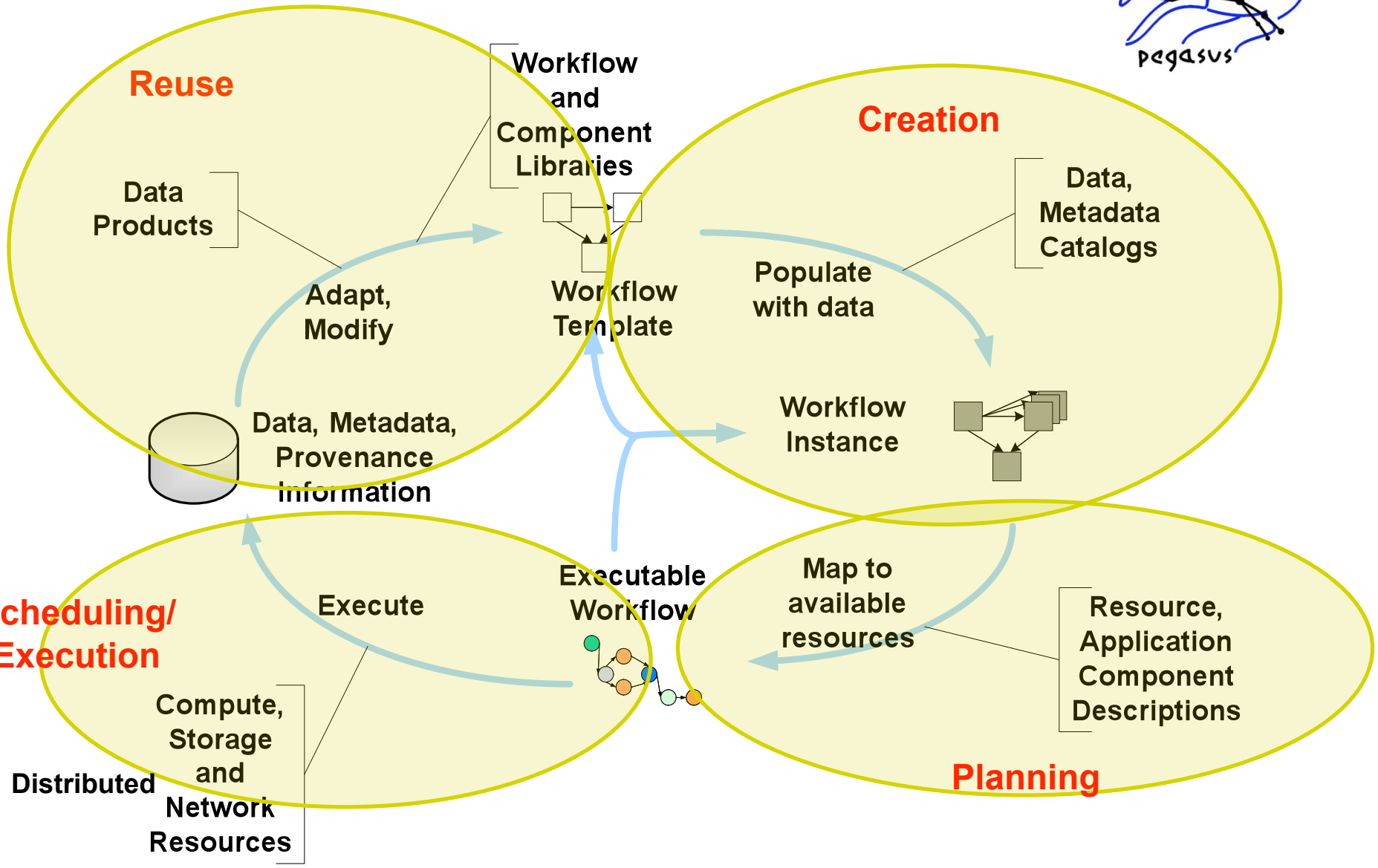


LIGO Inspiral Search Application

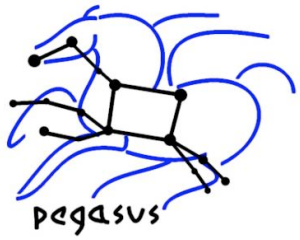


Inspiral workflow application is the work of Duncan Brown, Caltech, Scott Koranda, UW Milwaukee, and the LSC Inspiral group

Workflow Lifecycle

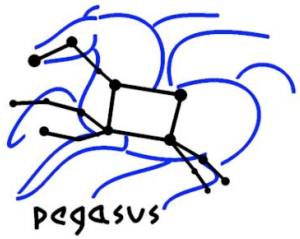


Workflow Creation



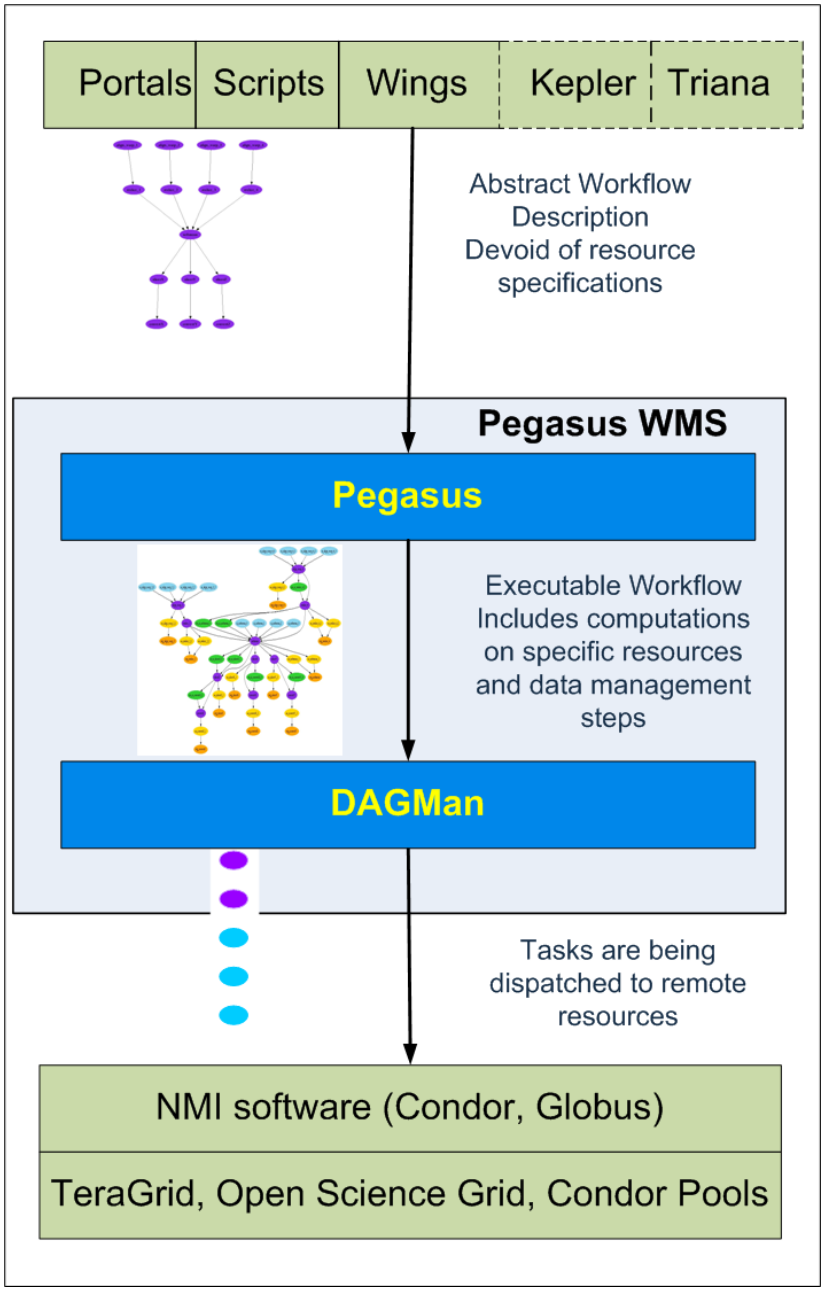
- Design a workflow (semantics info needed)
  - Find the right components
  - Set the right parameters
  - Find the right data
  - Connect appropriate pieces together
  - Find the right fillers
- Support both experts and novices

Challenges in user experiences

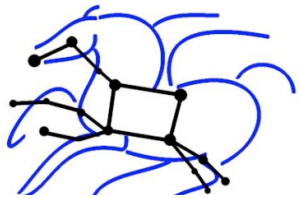


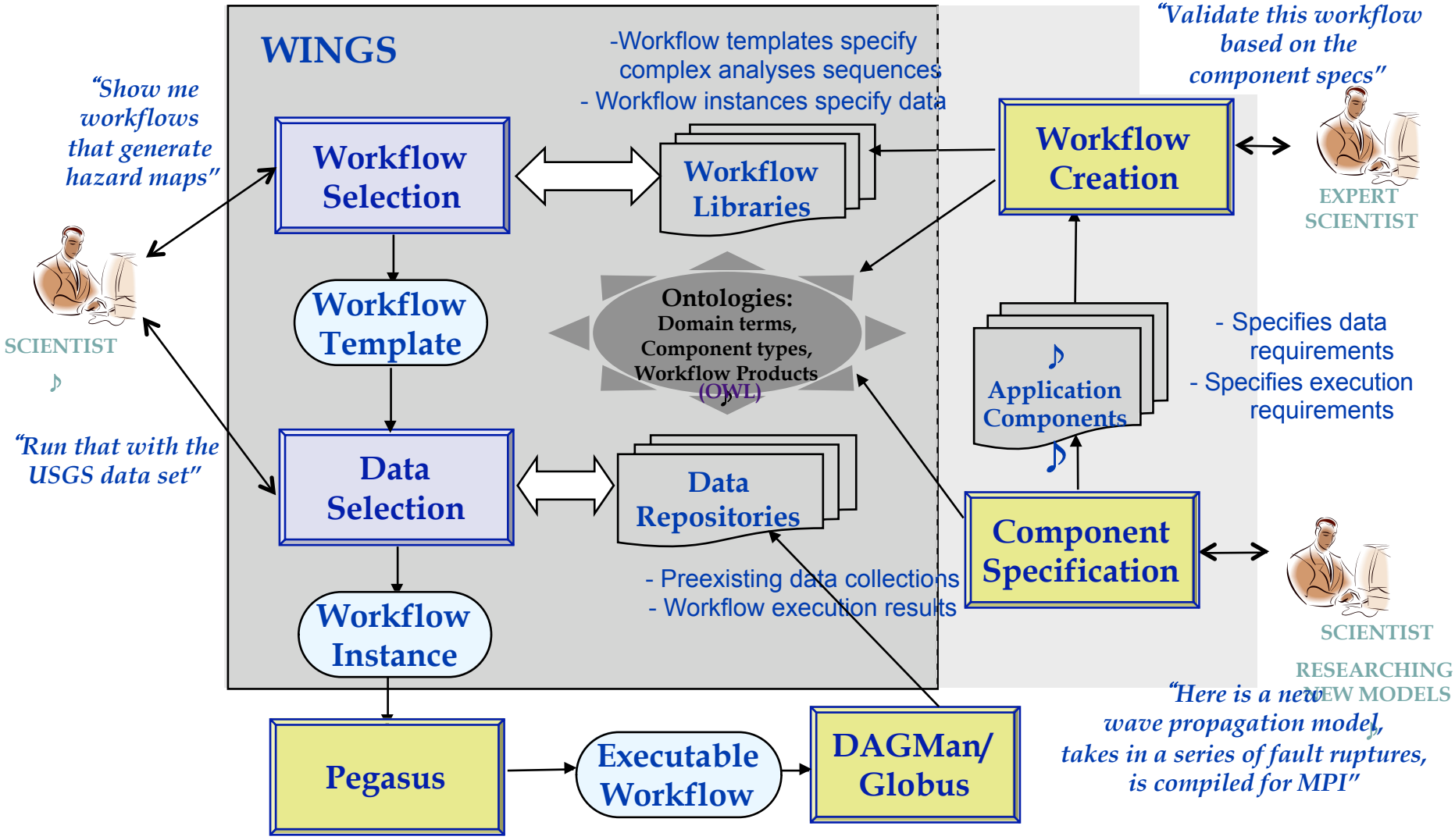
- Users' expectations vary greatly
  - High-level descriptions
  - Detailed plans that include specific resources
- Users interactions can be exploratory
    - Or workflows can be iterative
  - Modifying portions of the workflow as the computation progresses
- Users need progress, failure information at the right level of detail
- There is no ONE user but many users with different knowledge and capabilities

High-level system view



WINGS/Pegasus: Workflow Instance Generation and Selection, Using semantic technologies for workflow generation





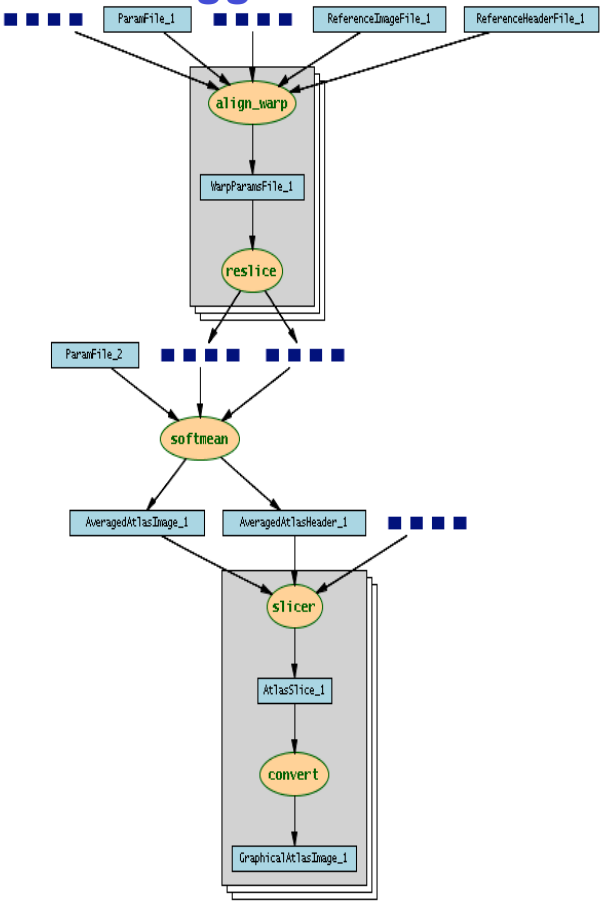
Wings for Pegasus: A Semantic Approach to Creating Very Large Scientific Workflows

Yolanda Gil, Varun Ratnakar, Ewa Deelman, Marc Spraragen, and Jihie Kim, *OWL: Experiences and Directions 2006*

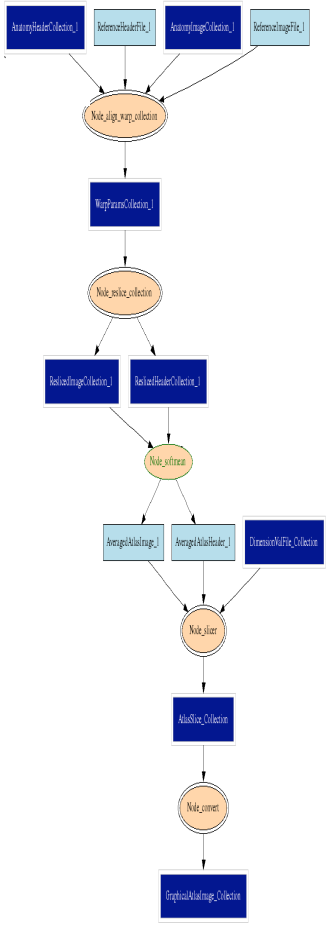
Editing and Creating Workflows

Wings Editor

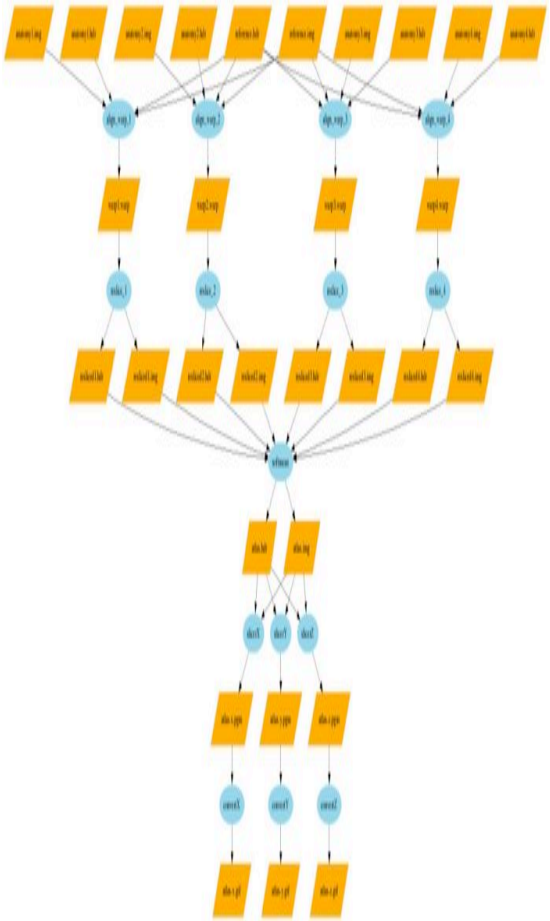
Users get feedback and suggestions



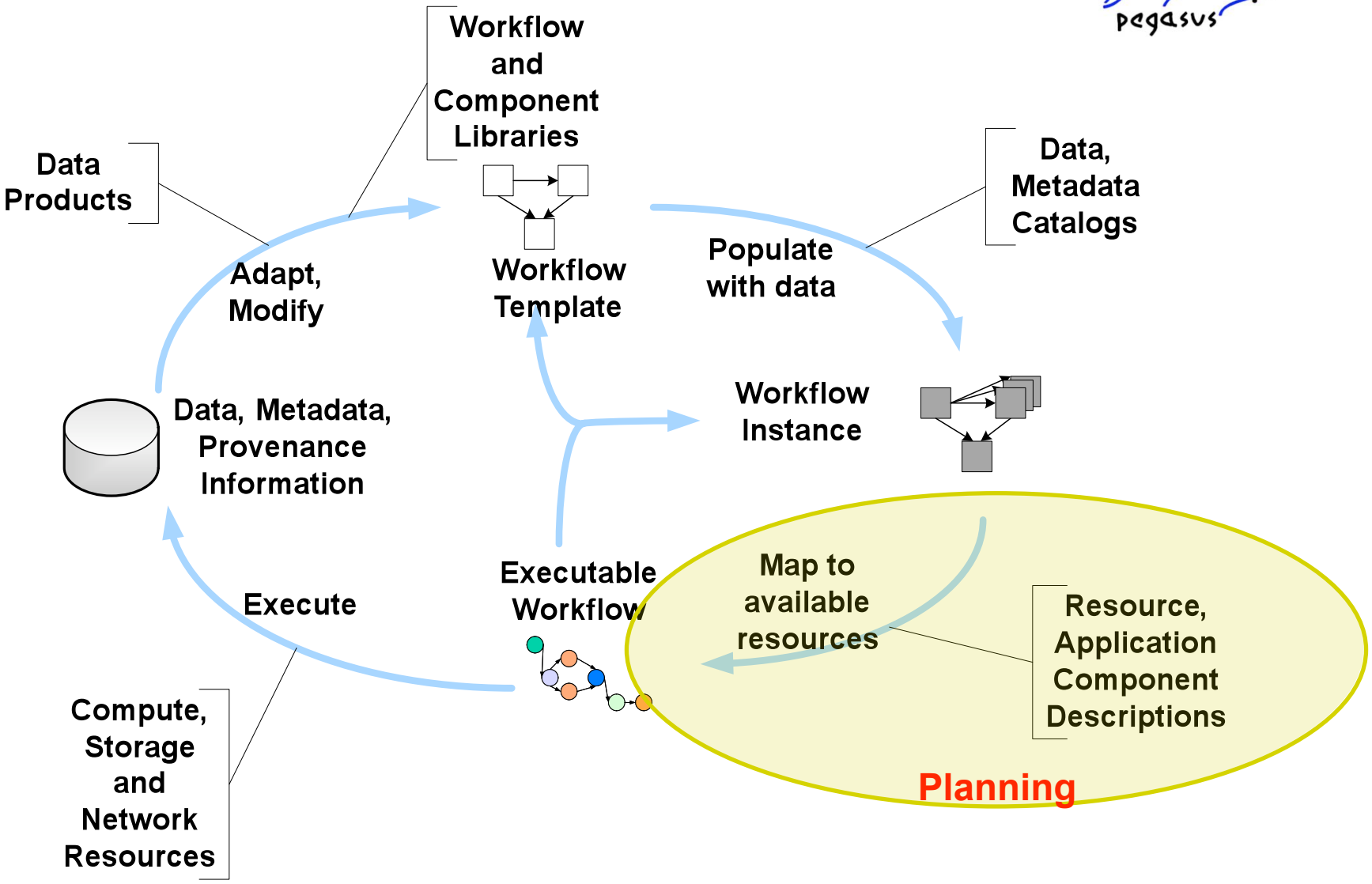
Workflow template



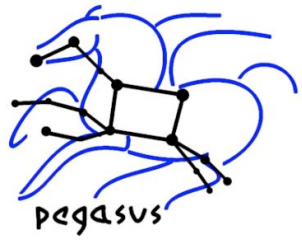
Workflow instance



Workflow Lifecycle



Specification: Place Y = F(x) at L
Execution Environment: Distributed



- 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

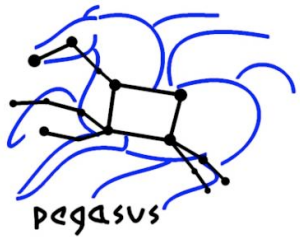
Error! *x* was not at *s!*

Error! *F(x)* failed!

Error! *c* crashed!

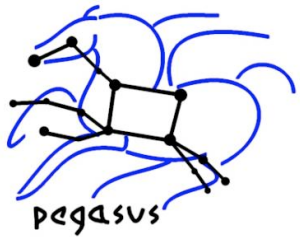
Error! *there is not enough* space at L!

Some challenges in workflow mapping



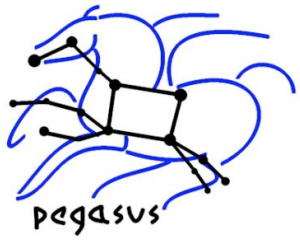
- Automated management of data
- Efficient mapping of workflow instances to resources
  - Runtime Performance
  - Data space optimizations
  - Fault tolerance (involves interfacing with the workflow execution system)
    - Recovery by replanning
    - plan “B”
  - Scalability
- Providing feedback to the user
  - Feasibility, time estimates

Mapping Correctly



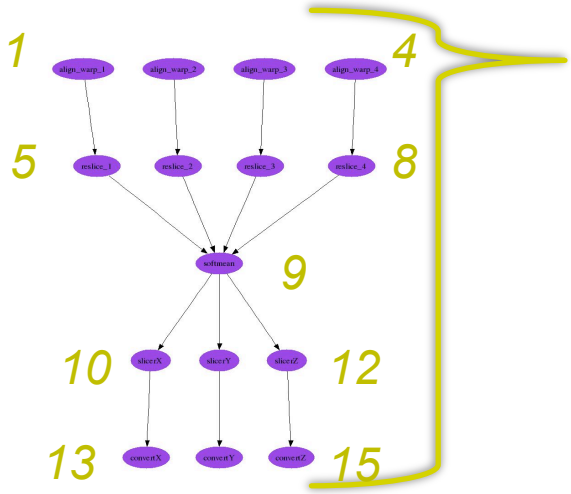
- Select where to run the computations
  - Apply a scheduling algorithm
    - HEFT, min-min, round-robin, random
    - Schedule in a data-aware fashion (data transfers, amount of storage)
    - The quality of the scheduling depends on the quality of information
  - Transform task nodes into nodes with executable descriptions
    - Execution location
    - Environment variables initializes
    - Appropriate command-line parameters set
- Select which data to access
  - Add stage-in nodes to move data to computations
  - Add stage-out nodes to transfer data out of remote sites to storage
  - Add data transfer nodes between computation nodes that execute on different resources
- Add nodes to create an execution directory on a remote site

Additional Mapping Elements



- Cluster compute nodes in small granularity applications
- Add data cleanup nodes to remove data from remote sites when no longer needed
  - reduces workflow data footprint
- Add nodes that register the newly-created data products
- Provide provenance capture steps
  - Information about source of data, executables invoked, environment variables, parameters, machines used, performance
- Scale matters--today we can handle:
  - 1 million tasks in the workflow instance (SCEC)
  - 10TB input data (LIGO)

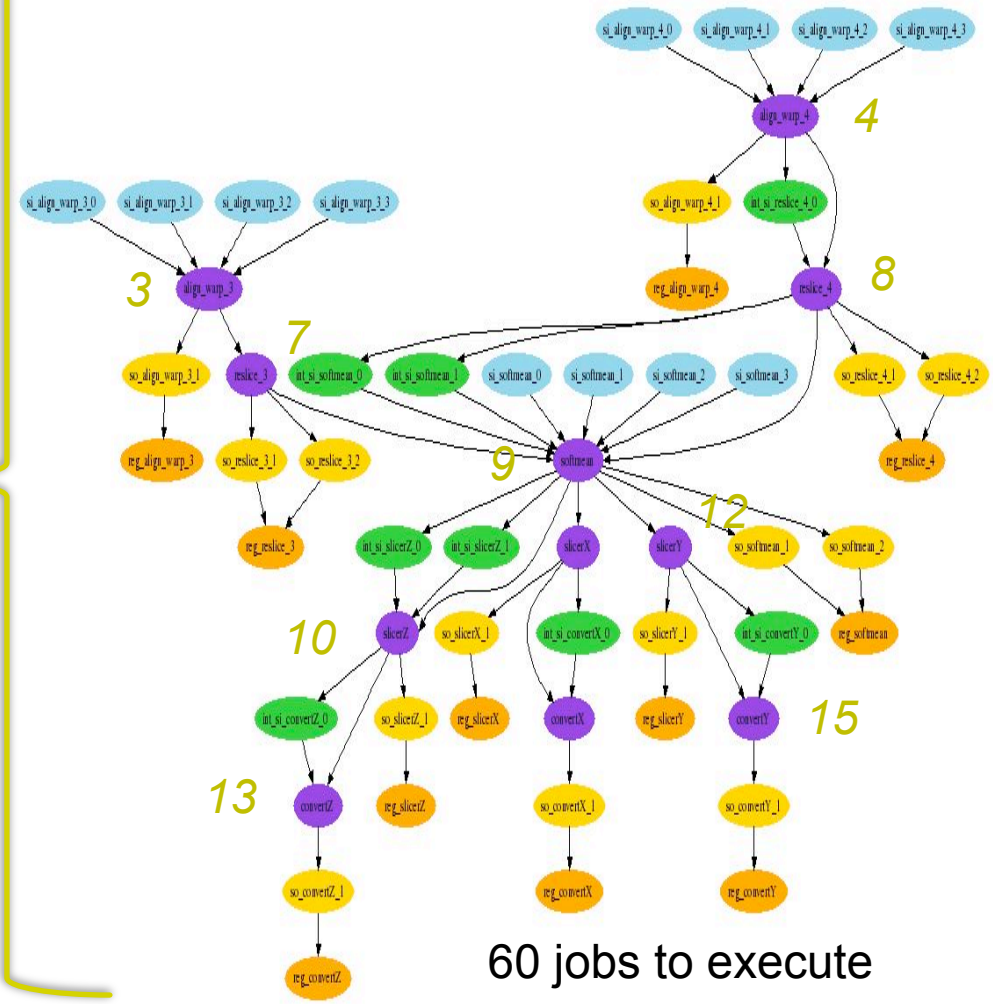
Pegasus Workflow Mapping



Original workflow: 15 compute nodes devoid of resource assignment

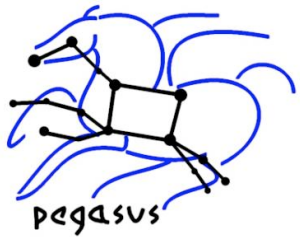
Resulting workflow mapped onto 3 Grid sites:

11 compute nodes (4 reduced based on available intermediate data)
12 data stage-in nodes
8 inter-site data transfers
14 data stage-out nodes to long-term storage
14 data registration nodes (data cataloging)



60 jobs to execute

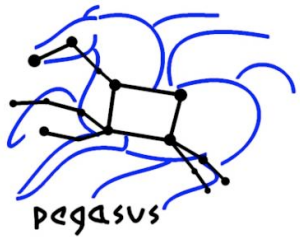
Pegasus-Workflow Management System



- Leverages abstraction for workflow description to obtain ease of use, scalability, and portability
- Provides a compiler to map from high-level descriptions (workflow instances) to executable workflows
  - Correct mapping
  - Performance enhanced mapping
- Provides a runtime engine to carry out the instructions (Condor DAGMan)
  - Scalable manner
  - Reliable manner

In collaboration with Miron Livny, UW Madison, funded under NSF-OCI SDCI

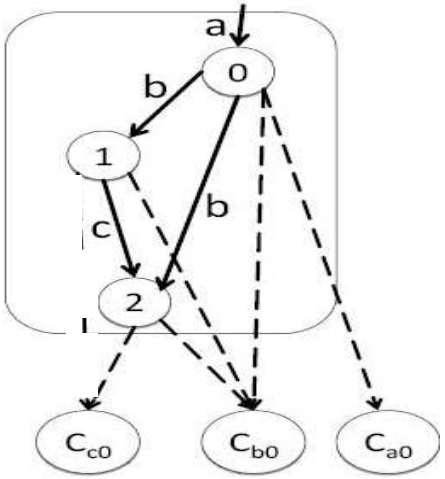
Efficient data handling



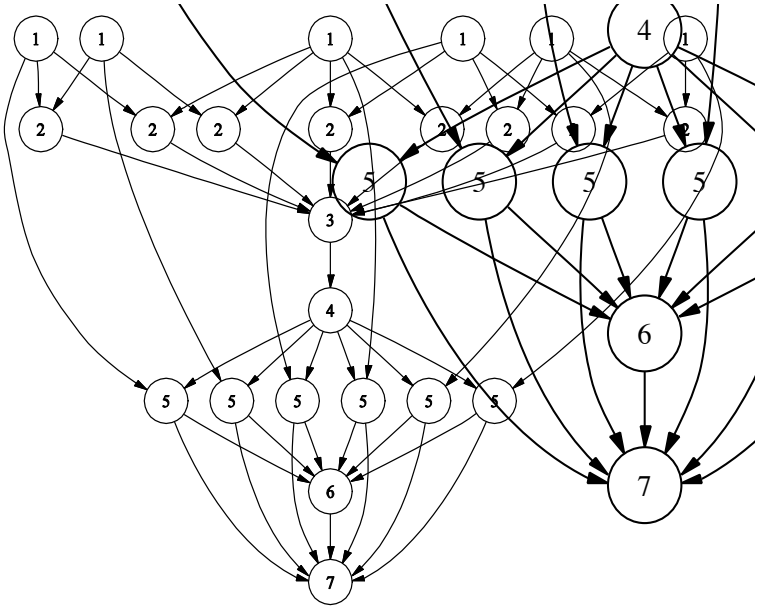
- Input data is staged dynamically
- New data products are generated during execution
- For large workflows 10,000+ files
  - Similar order of intermediate and output files
  - Total space occupied is far greater than available space—failures occur
- Solution:
  - Determine which data are no longer needed and when
  - Add nodes to the workflow do cleanup data along the way
- Issues:
  - minimize the number of nodes and dependencies added so as not to slow down workflow execution
  - deal with portions of workflows scheduled to multiple sites

Joint work with Rizos Sakellariou, Manchester University, CCGrid 2007, Scientific Programming Journal, 2007

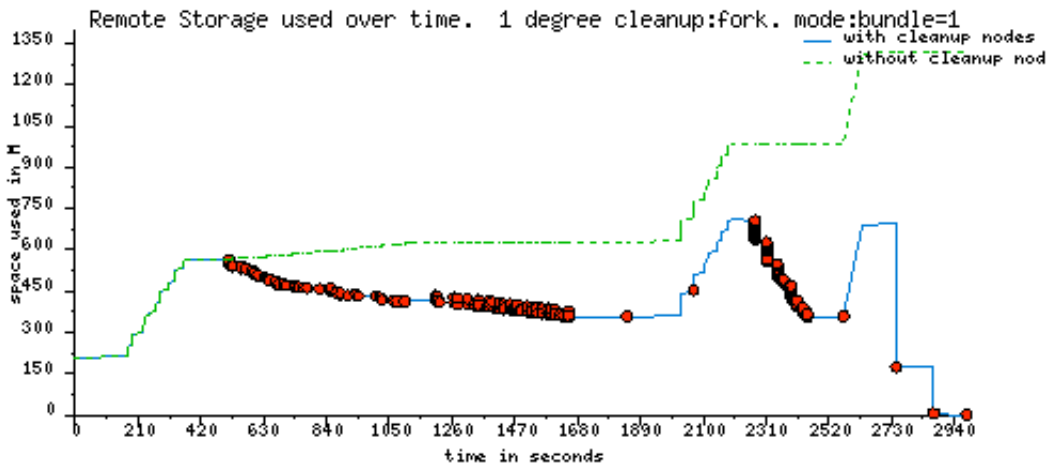
LIGO's gravitational-wave physics application and Montage



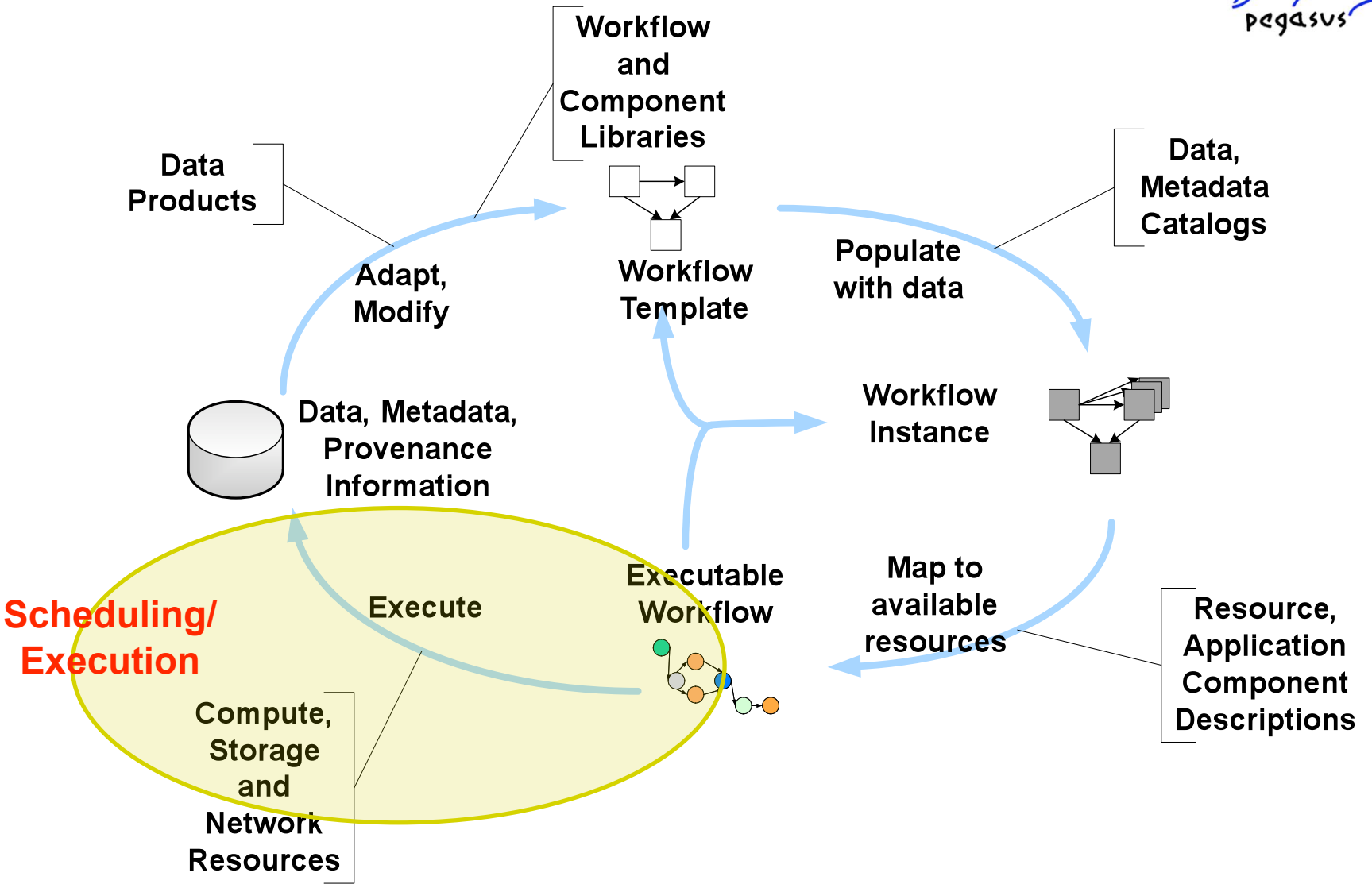
Adding cleanup nodes to the workflow



1.25GB versus 4.5 GB



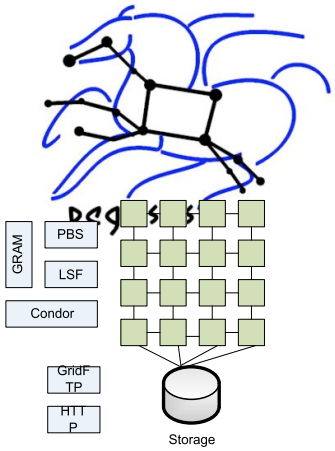
Workflow Lifecycle

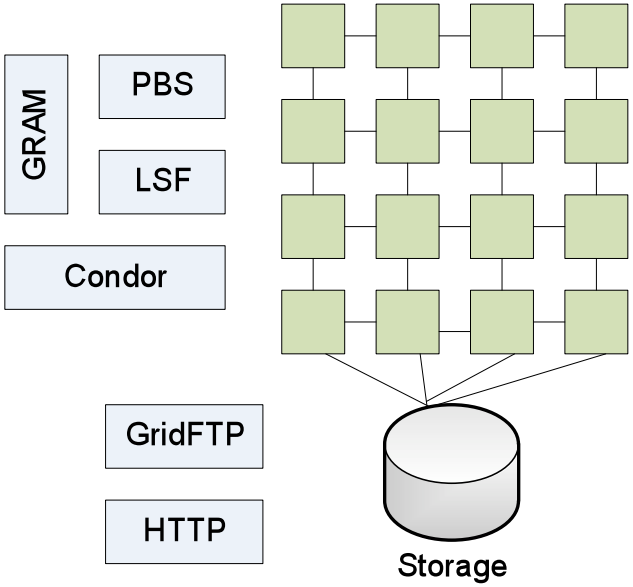


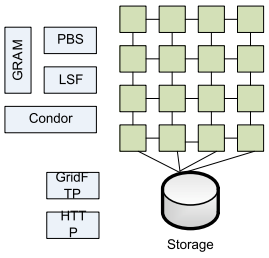
Execution Environment

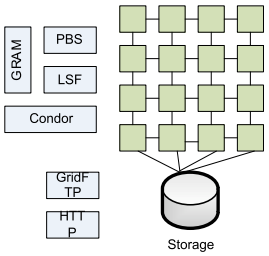
Globus and Condor Services for job scheduling
Globus Services for data transfer and Cataloging

Information Services:
--- information about data location
--- information about the execution sites

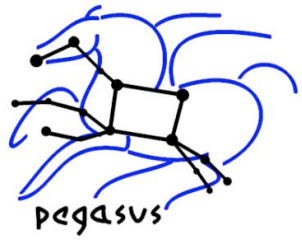






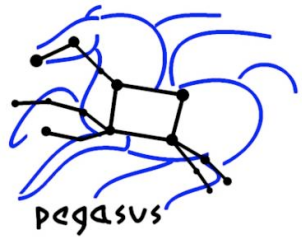


Challenges in Workflow Execution



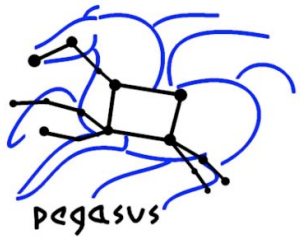
- Resource provisioning
  - Which resources to provision if many possibilities?
  - How many resources to provision?
  - For how long?
- Fault Tolerance
  - How to recognize different types of failures
  - How to recover from failures?
- Efficient collaboration between the data and computation management systems
- Debugging
  - How to relate the workflow result (outcome) to workflow specification

DAGMan ("under the hood" of Pegasus)



- Pegasus uses DAGMan to run the executable workflow
- Users may not have to interact with DAGMan directly…
- …but they may (for debugging, optimization)
- Pegasus doesn't expose all DAGMan features

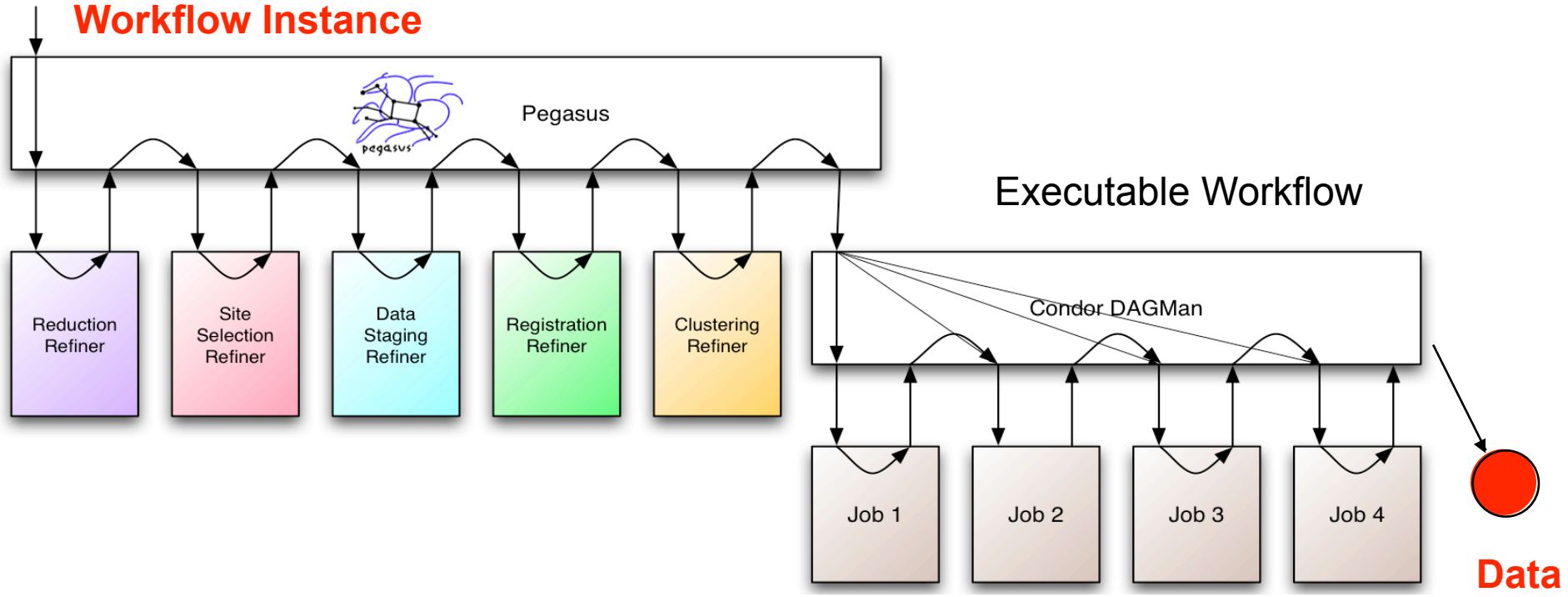
DAGMan (Directed Acyclic Graph MANager)



- Runs workflows that can be specified as Directed Acyclic Graphs
- Enforces DAG dependencies
- Progresses as far as possible in the face of failures
- Provides retries, throttling, etc.
- Runs on top of Condor (and is itself a Condor job)
- Doesn't "care" whether node jobs are local or Grid jobs

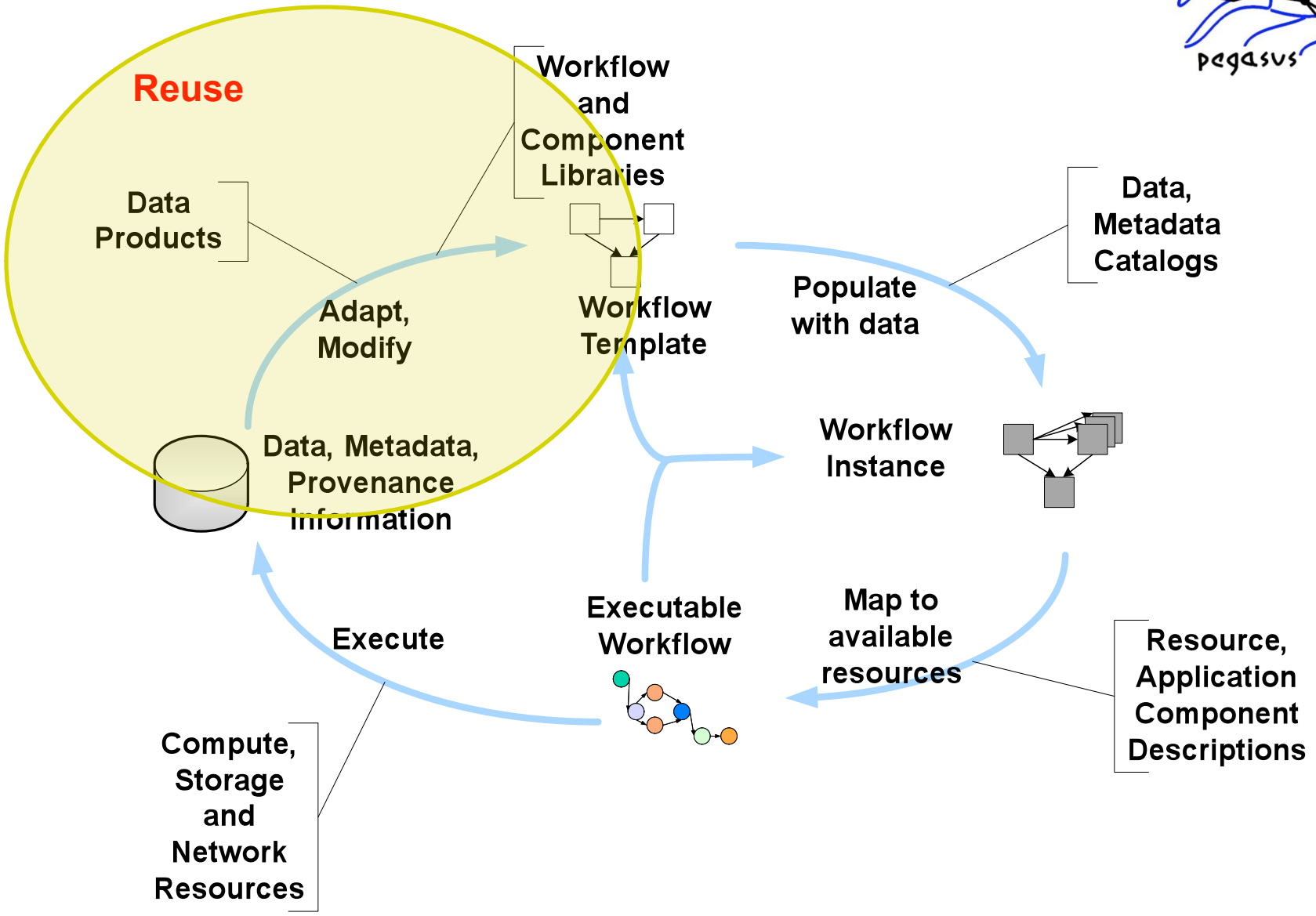
Workflow Mapping and Execution Connected

- For each data item, we can find the executable workflow steps that produced it and other data items that contributed to those steps.
- For each workflow step, we can find its connection to the workflow instance jobs from which it was refined.



Joint work with Luc Moreau, Southampton University e-Science 2007

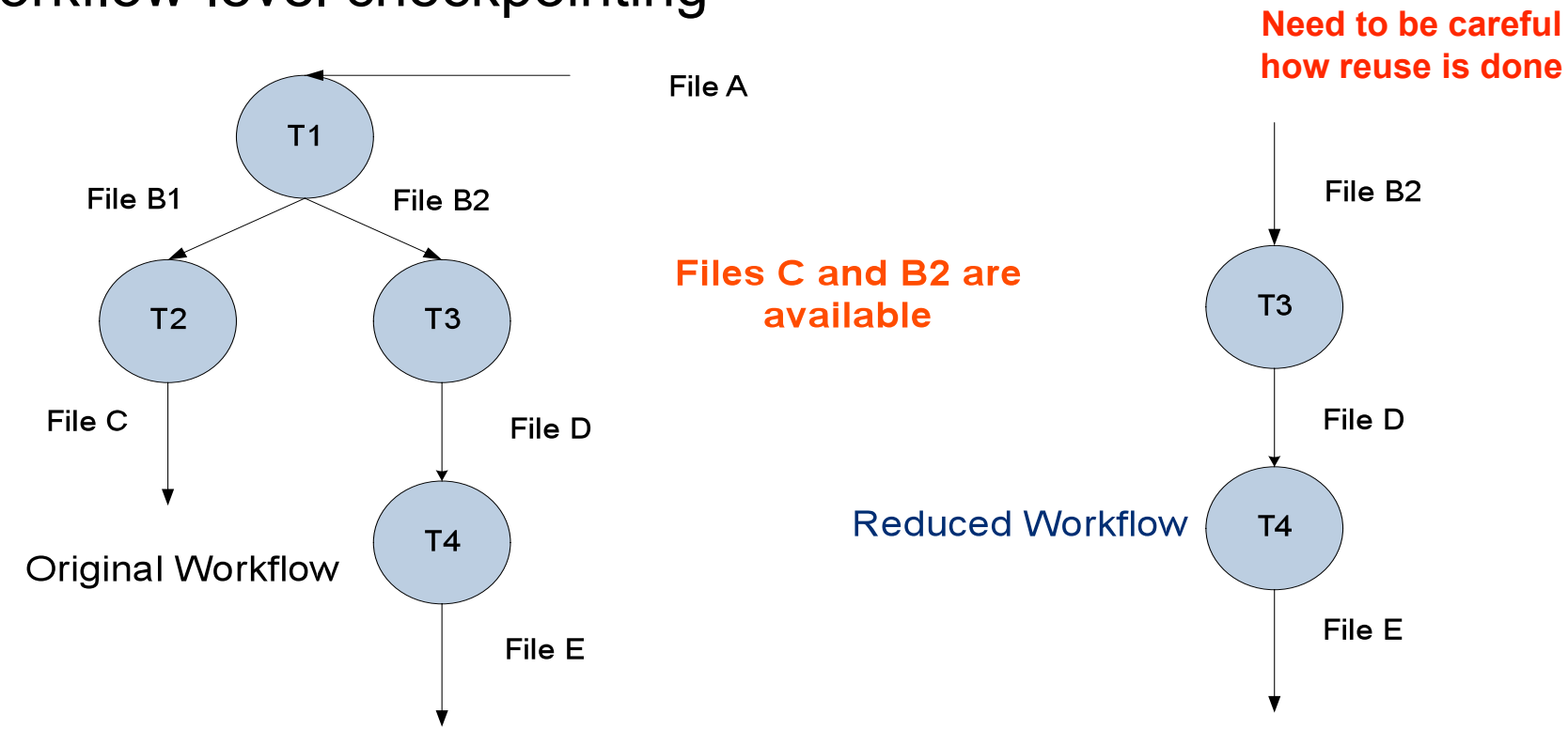
Workflow Lifecycle



Data Reuse

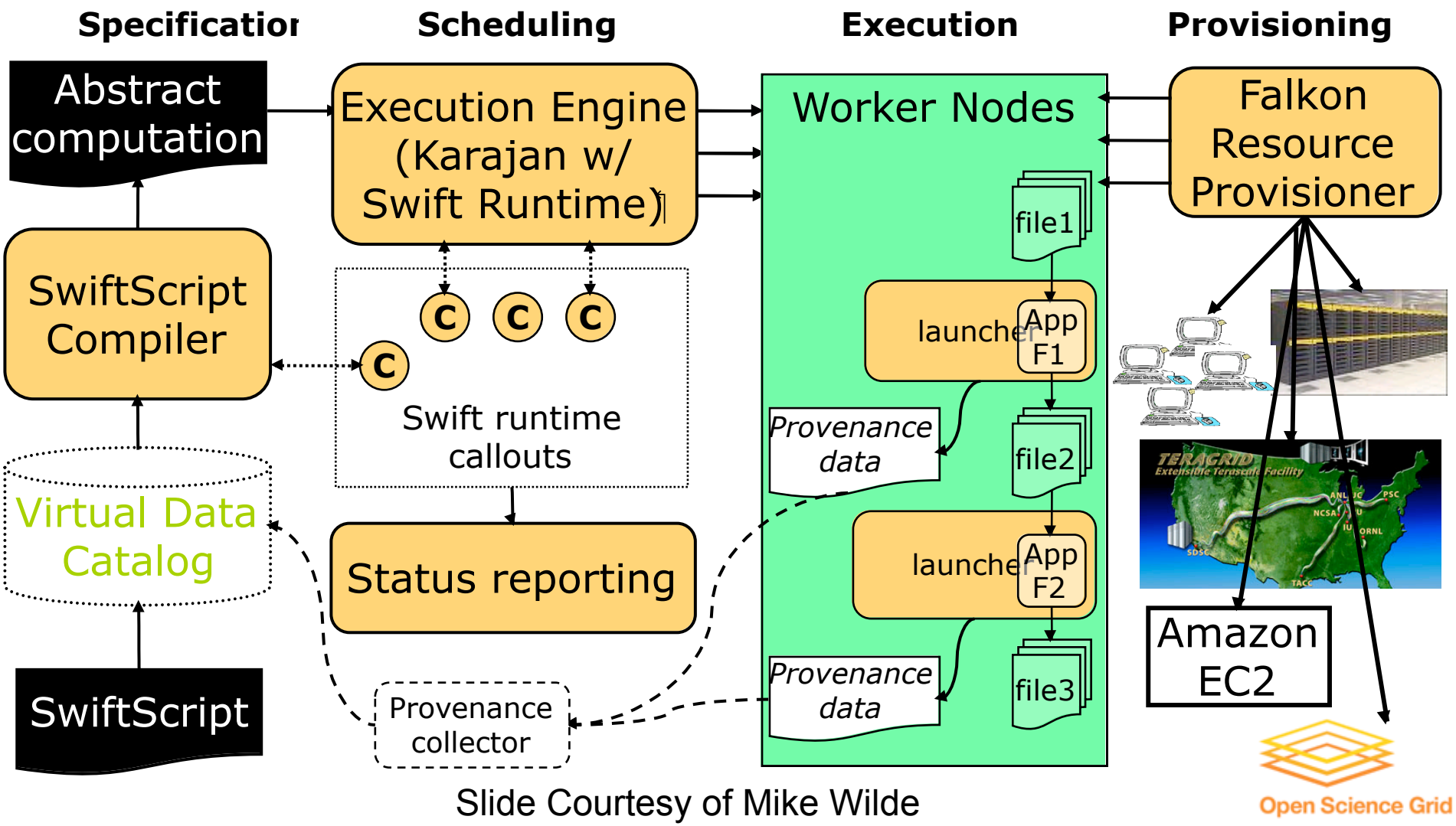
Sometimes it is cheaper to access the data than to regenerate it

Keeping track of data as it is generated supports workflow-level checkpointing



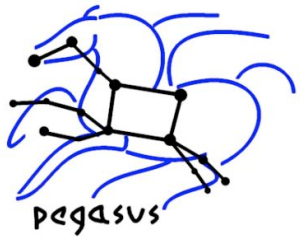
Mapping Complex Workflows Onto Grid Environments, E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Backburn, A. Lazzarini, A. Arbee, R. Cavanaugh, S. Koranda, *Journal of Grid Computing, Vol.1, No. 1, 2003., pp25-39.*

Swift Architecture



Slide Courtesy of Mike Wilde

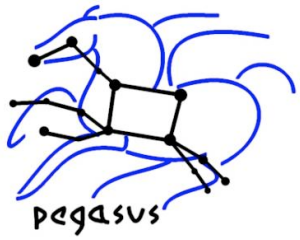
Swift programs



- A Swift script is a set of **functions**
  - Atomic functions wrap & invoke application programs
  - Composite functions invoke other functions
- Data is **typed** as composable arrays and structures of files and simple scalar types (int, float, string)
- Collections of **persistent file structures** (datasets) are **mapped** into this data model
- Members of datasets can be processed in **parallel**
- Statements in a procedure are executed in **data-flow** dependency order and concurrency
- Variables are **single assignment**
- **Provenance** is gathered as scripts execute

Slide Courtesy of Mike Wilde

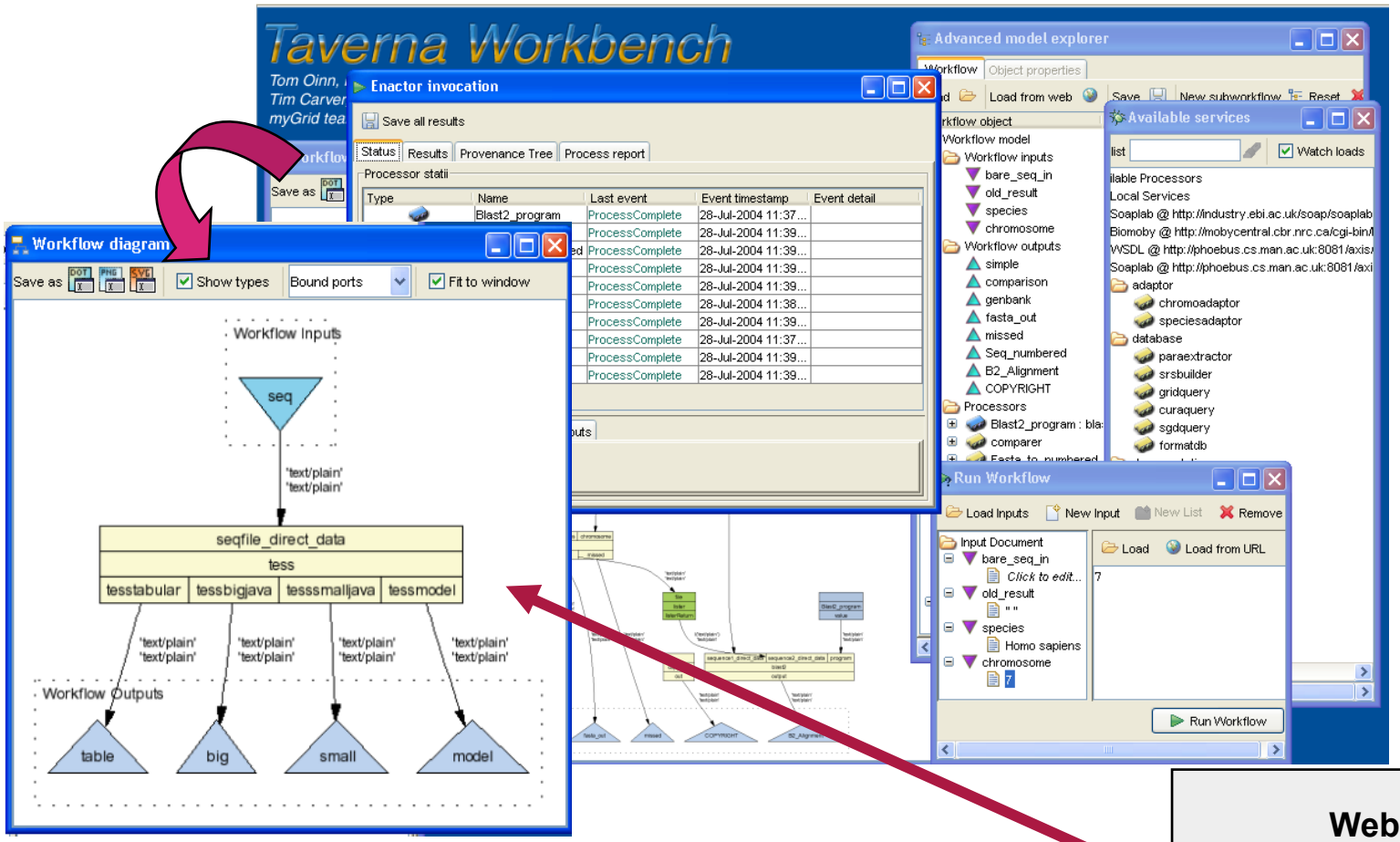
Swift



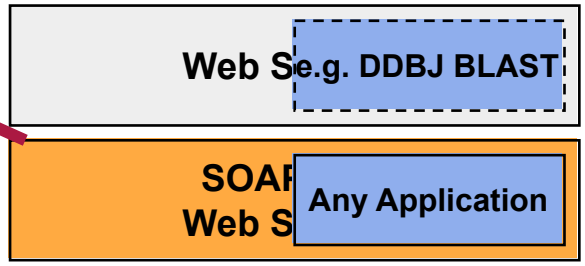
- Clean separation of logical/physical concerns
  - XDTM specification of logical data structures
- Concise specification of parallel programs
  - SwiftScript, with iteration, etc.
- Efficient execution (on distributed resources)
  - **Karajan+Falkon**:
  - Grid Interface, light dispatch, pipelining, clustering, provisioning
- Rigorous provenance tracking and query
  - Records provenance data of each job executed

Slide Courtesy of Mike Wilde

Taverna Workbench

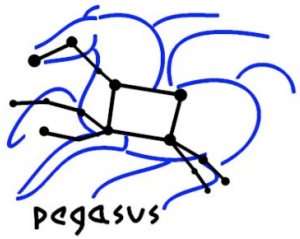


Scufl Simple Conceptual Unified Flow Language
Taverna Writing, running workflows & examining results
SOAPLAB Makes applications available



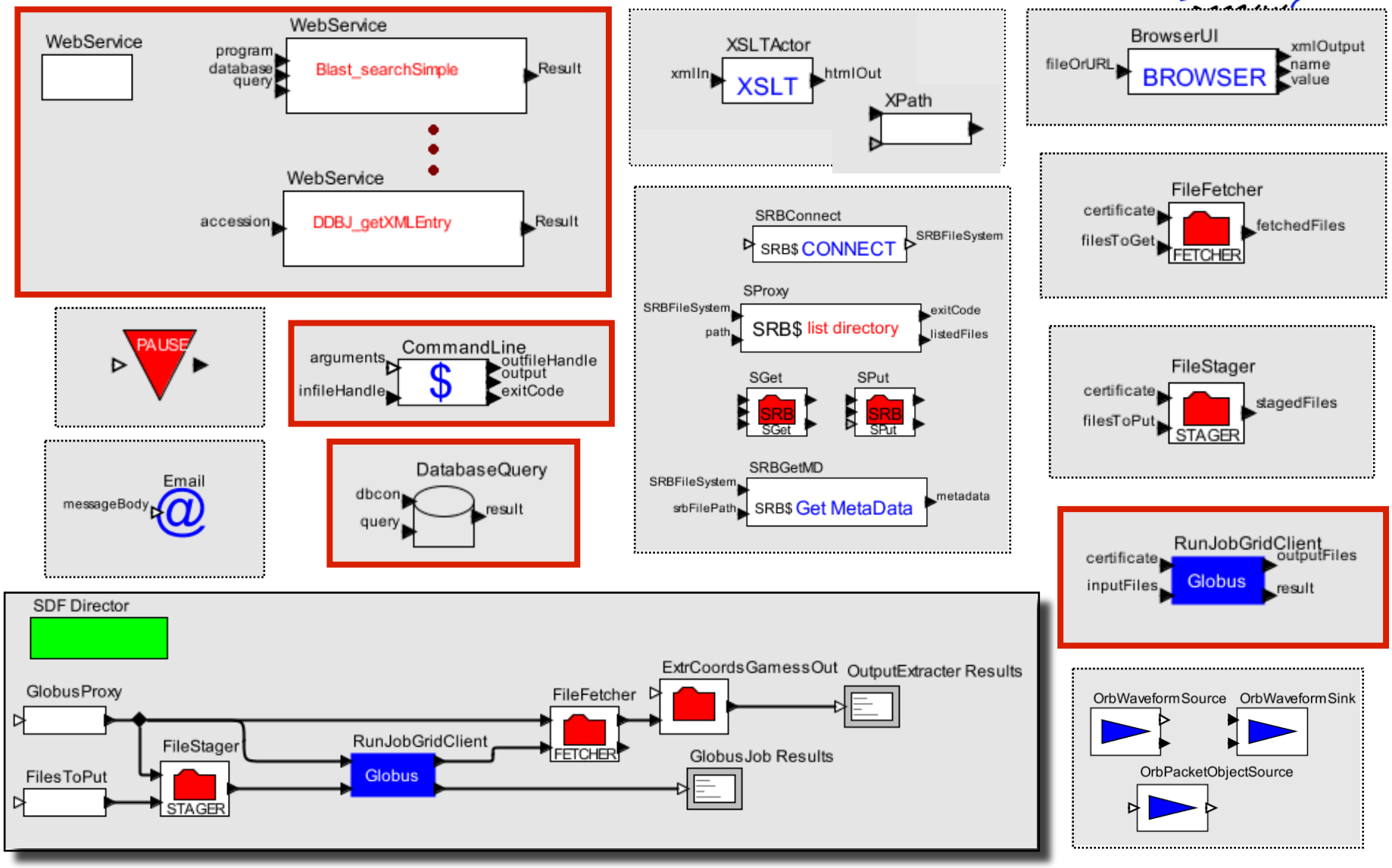
Slides courtesy of Katy Wolstencroft

Kepler (UCSD)



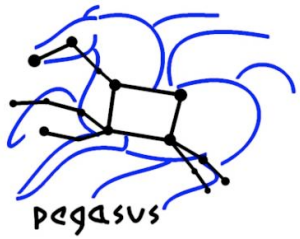
- Kepler is a software application for the analysis and modeling of scientific data
  - Builds on Ptolemy II framework and provides a GUI to construct workflows
- Actor Oriented Modeling
  - Each actor has input/output ports
  - Parameters are static ports
- Data Connections
  - Unidirectional communication channels connect output to input ports
- Composite Actors
  - Wrap sub workflows
  - Arbitrary Nesting
- Directors
  - Define the execution semantics of workflow graph
  - executes workflow graph (some schedule)
  - sub-workflows may have different directors promotes reusability

Everything is a service / actor…



Slides courtesy of Bertram Ludaesher

Pegasus Current and Future Research

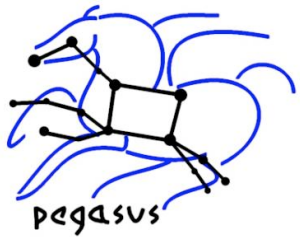


- Resource selection
- Resource provisioning
- Workflow restructuring
- Adaptive computing
  - Workflow refinement adapts to changing execution environment
- Workflow provenance (including provenance of the mapping process)
- Management and optimization across multiple workflows
- Workflow debugging
- Streaming data workflows
- Automated guidance for workflow restructuring
- Support for long-lived and recurrent workflows

Acknowledgments

- Pegasus: Ewa Deelman, Mei-Hui Su, Karan Vahi, Arun Ramakrishnan (USC)
- DAGMan (in Pegasus-WMS): Miron Livny, and the Condor team (Wisconsin Madison)
- Wings: Yolanda Gil, Jihie Kim, Varun Ratnakar, Paul Groth (USC)
- LIGO: Kent Blackburn, Duncan Brown, Stephen Fairhurst, Scott Koranda (Caltech)
- Montage: Bruce Berriman, John Good, Dan Katz, and Joe Jacobs (Caltech, JPL)
- SCEC: Tom Jordan, Robert Graves, Phil Maechling, David Okaya, Li Zhao (USC, UCSD, others)

Relevant Links



- Pegasus: pegasus.isi.edu
- DAGMan: www.cs.wisc.edu/condor/dagman
- For more questions: pegasus@isi.edu

Relevant Links

- NSF Workshop on Challenges of Scientific Workflows : www.isi.edu/nsf-workflows06, E. Deelman and Y. Gil (chairs)
- Workflows for e-Science, Taylor, I.J.; Deelman, E.; Gannon, D.B.; Shields, M. (Eds.), Dec. 2006

- Open Science Grid: www.opensciencegrid.org
- LIGO: www.ligo.caltech.edu/
- SCEC: www.scec.org
- Montage: montage.ipac.caltech.edu/
- Condor: www.cs.wisc.edu/condor/
- Globus: www.globus.org
- TeraGrid: www.teragrid.org