Part IV

Workflow Mapping and Execution in Pegasus

(Thanks to Ewa Deelman)
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 to executable workflows
  - Correct mapping
  - Performance enhanced mapping
- Provides a runtime engine to carry out the instructions
  - Scalable manner
  - Reliable manner

Ewa Deelman, Gaurang Mehta, Karan Vahi (USC/ISI) in collaboration with Miron Livny (UW Madison)
Underlying Grid Middleware Services

- Pegasus uses Globus ([www.globus.org](http://www.globus.org)) grid services:
  - Gridftp for efficient and reliable data transfer
  - Grid proxys/certificates
- Pegasus uses the Condor ([http://www.cs.wisc.edu/condor](http://www.cs.wisc.edu/condor)) job management system:
  - CondorG for job submission to shared resources
  - DAGman to manage job interdependencies
Basic Workflow Mapping

- Select where to run the computations
  - Apply a scheduling algorithm
    - HEFT, min-min, round-robin, random
    - 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
Basic Workflow Mapping

- Add nodes to create an execution directory on a remote site
- Add nodes that register the newly-created data products
- Add data cleanup nodes to remove data from remote sites when no longer needed
  - reduces workflow data footprint
- Provide provenance capture steps
  - Information about source of data, executables invoked, environment variables, parameters, machines used, performance
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
Some challenges in workflow mapping

- Automated management of data
  - Through workflow modification

- Efficient mapping the workflow instances to resources
  - Performance
  - Data space optimizations
  - Fault tolerance (involves interfacing with the workflow execution system)
    - Recovery by replanning
    - plan “B”

- Mapping not a one shot thing

- Providing feedback to the user
  - Feasibility, time estimates
Pegasus Deployment

- Wings
  - Abstract Workflow (Resource-independent)
- Pegasus
  - Executable Workflow (Resources Identified)
- DAGMan
  - Ready Tasks
- Condor Queue
  - Condor -G
  - Condor -C
- LOCAL SUBMIT HOST
  - Community resource
- Resource Information and Data Location Information
  - NMI: Globus MDS, RLS, SRB
- Distributed Resources
  - GridFTP
  - HTTP
  - Storage
  - Condor
  - PBS
  - LSF
  - Globus GRAM
Node clustering

Useful for small granularity jobs

Level-based clustering

Vertical clustering

Arbitrary clustering
Data Reuse

- When it is cheaper to access the data than to regenerate it
- Keeping track of data as it is generated supports workflow-level checkpointing

---

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 is 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
- deal with files on partition boundaries
Workflow Footprint

In order to improve the workflow footprint, we need to determine when data are no longer needed:

- Because data was consumed by the next component and no other component needs it
- Because data was staged-out to permanent storage
- Because data are no longer needed on a resource and have been stage-out to the resource that needs it
Cleanup Disk Space as Workflow Progresses

- For each node add dependencies to cleanup all the files used and produced by the node.
- If a file is being staged-in from r1 to r2, add a dependency between the stage-in and the cleanup node.
- If a file is being staged-out, add a dependency between the stage-out and the cleanup node.
LIGO Workflows

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

26% improvement

56% improvement
Montage application

- ~7,000 compute jobs in workflow instance
- ~10,000 nodes in the executable workflow
- same number clusters as processors
- speedup of ~15 on 32 processors
Pegasus + Condor + TeraGrid for Southern California Earthquake Center (SCEC) [Deelman et al 06]

(nice TeraGrid folks)

Condor Glide-ins

Provision the resources

Resource Descriptions

Map the Workflow onto the Grid resources

Executable workflow

Provenance Tracking Catalog

Record Information about the Workflow

Run the Workflow on the Grid Resources

Executable Workflow

Executable workflow on the Grid Resources

Tasks

Executable workflow on the Grid Resources

Tasks

Hazard Map

TeraGrid

Condor DAGMan

Globus

Executable workflow

Pegasus

Condor Glide-ins

Provision the resources

Resource Descriptions

Map the Workflow onto the Grid resources

Executable workflow

Provenance Tracking Catalog

Record Information about the Workflow

Run the Workflow on the Grid Resources

Tasks

Hazard Map

TeraGrid

Condor DAGMan

Globus
SCEC workflows run each week using Pegasus and DAGMan on the TeraGrid and USC resources. Cumulatively, the workflows consisted of over half a million tasks and used over 2.5 CPU Years, Largest workflow $O(100,000)$ nodes.