Rethinking Data Management For Big Data Scientific Workflows

Karan Vahi , Mats Rynge, Gideon Juve,

Rajiv Mayani, Ewa Deelman

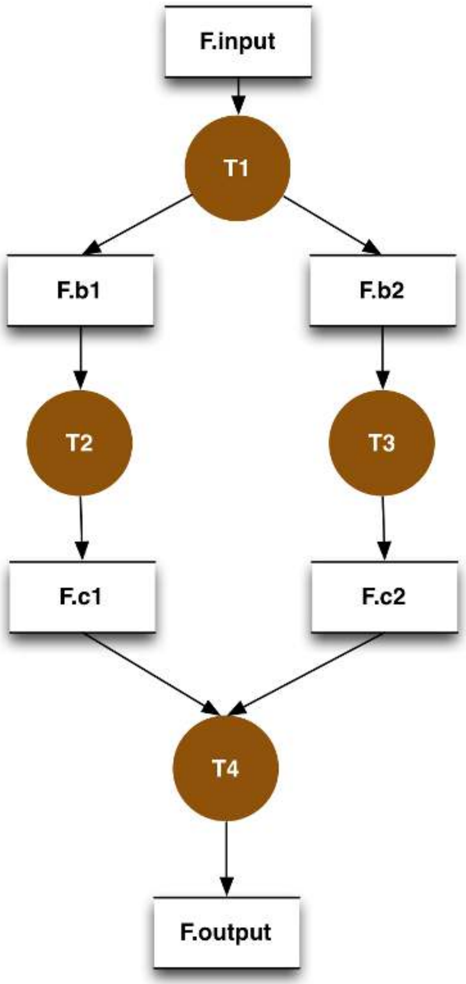
USC Information Sciences Institute

Outline

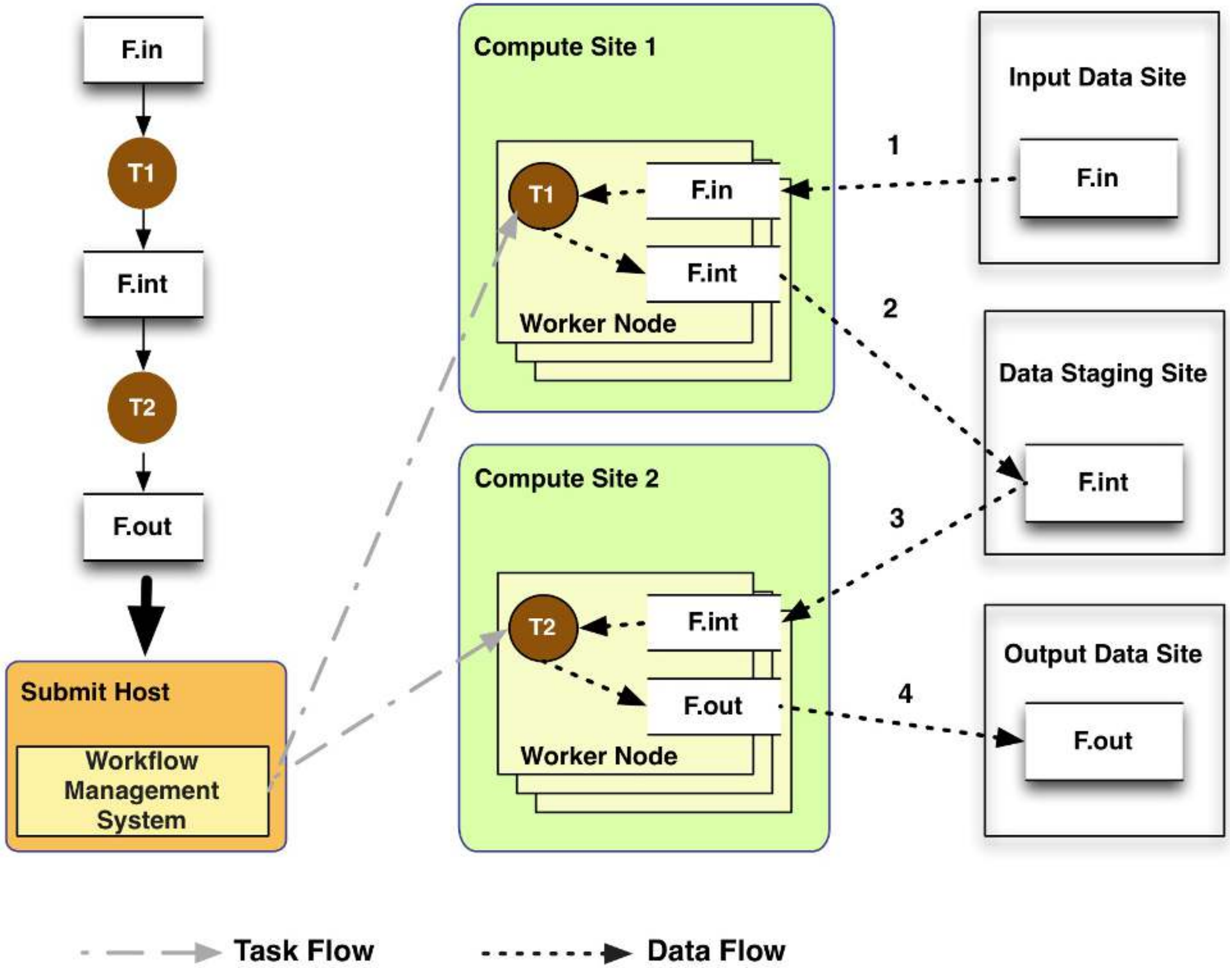
- Introduction
- Object Stores for Workflows
- Pegasus Data Management
- Experiments
- Conclusions and Future Work

Scientific Workflows

- Capture individual data transformation and analysis steps
- Large monolithic applications broken down to smaller jobs
- Smaller jobs can be independent or connected by some control flow/ data flow dependencies
- Usually expressed as a Directed Acyclic Graph of tasks
- Files are classified as
  - **Input Files:** (F.input) not generated by any task.
  - **Intermediate Files:** (F.b1,F.b2,F.c1,F.c2) generated during workflow execution
  - **Output Files:** (F.output) – files generated that are of interest to the user.



General Workflow Execution Model



- Most of the tasks in scientific workflow applications require POSIX file semantics
  - Each task in the workflow opens one or more input files
  - Read or write a portion of it and then close the file.

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

Posix Access for Tasks in the workflow

- How do you ensure posix access for the tasks?
  - Place it directly on local filesystem of the worker node from the input site.
  - Place it on a shared filesystem shared across nodes.

- 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.
    - Users are not going to configure a shared FS across their VMs

Outline

- Introduction
- Object Stores for Workflows
- Pegasus Data Management
- Experiments
- Conclusions and Future Work

Object Storage for Workflows

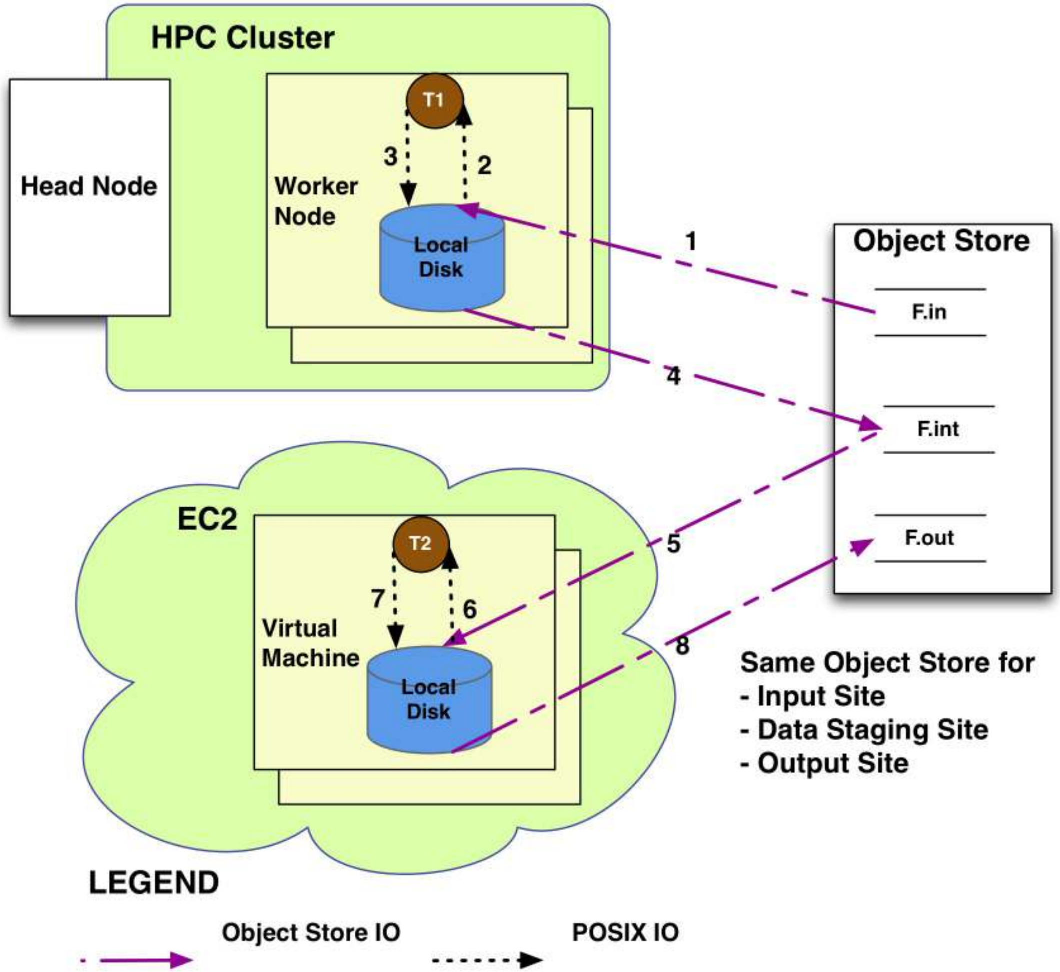
- Object Store: high level storage service with limited operations
  - Store, retrieve and delete data objects(files)
  - Don't provide byte level access
    - Cannot open a file in an object store, read and update it and then close it.
    - Instead a client needs to download the file, update it and then store as a new object.
  - Highly scalable and available such as Amazon S3
- Highly appealing for workflow systems to integrate object stores.
  - Support both late and early binding of tasks.
  - **Do all of this as generally as possible: Can we still support shared filesystem approach and traditional grid storage services and protocols?**

Leveraging Object Stores in Workflow Systems

- 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.

- Two general options for using object stores
  1. Use object stores for storing all 3 types of data
  2. When, available use a shared filesystem as a data staging site.

Exclusive Use of Object Stores

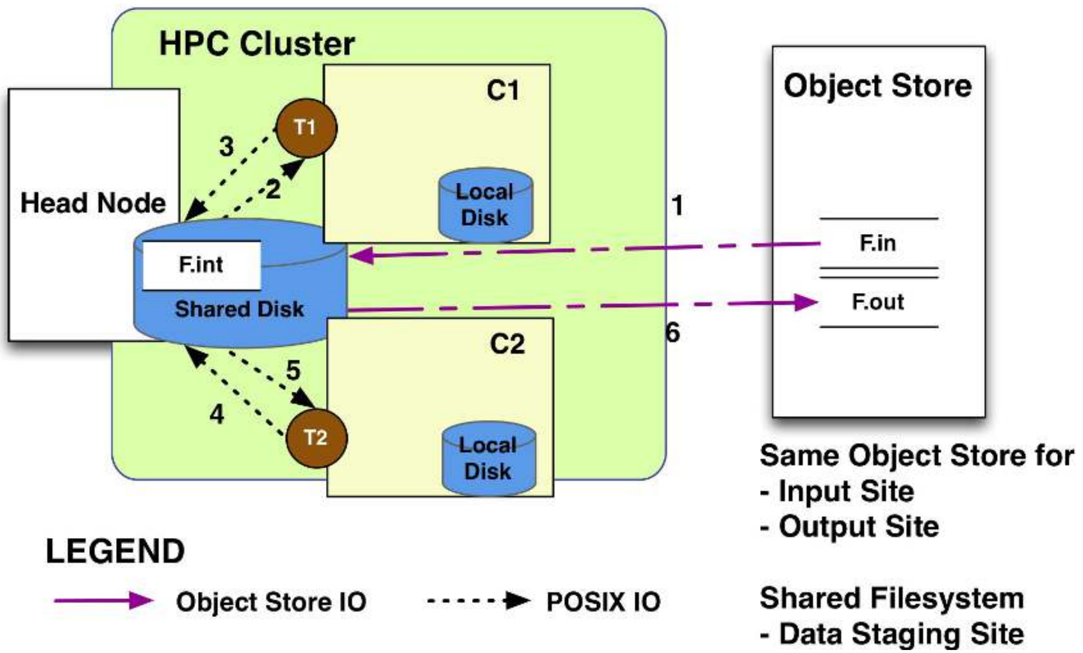


- Advantages
  - Can leverage scalable stores
  - Distribute computations across resources, such as supporting spillover from local resources to cloud resources.
  - Great bandwidth

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

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

Use of Shared Filesystem as Data Staging Site



- Advantages
  - No duplicate transfers for intermediate and input files
  - Lowers costs against a commercial object store as intermediate files are not put in the store
  - Works well in traditional supercomputing environment such as XSEDE.
- Disadvantages
  - Loss of flexibility where to place the tasks.
  - Setup not easy to recreate in the cloud.

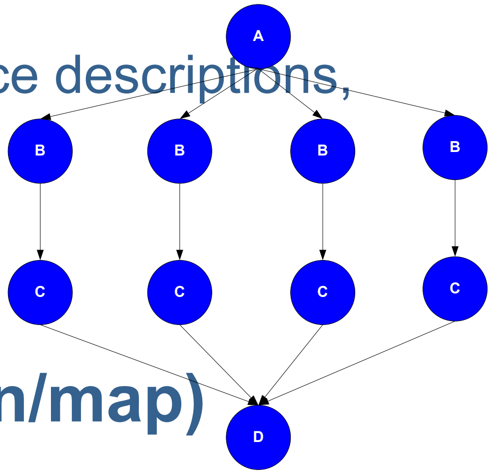
- Workflow stages the input data on demand to a shared POSIX compliant filesystem shared across worker nodes. Acts as data staging site.

Outline

- Introduction
- Object Stores for Workflows
- Pegasus Data Management
- Experiments
- Conclusions and Future Work

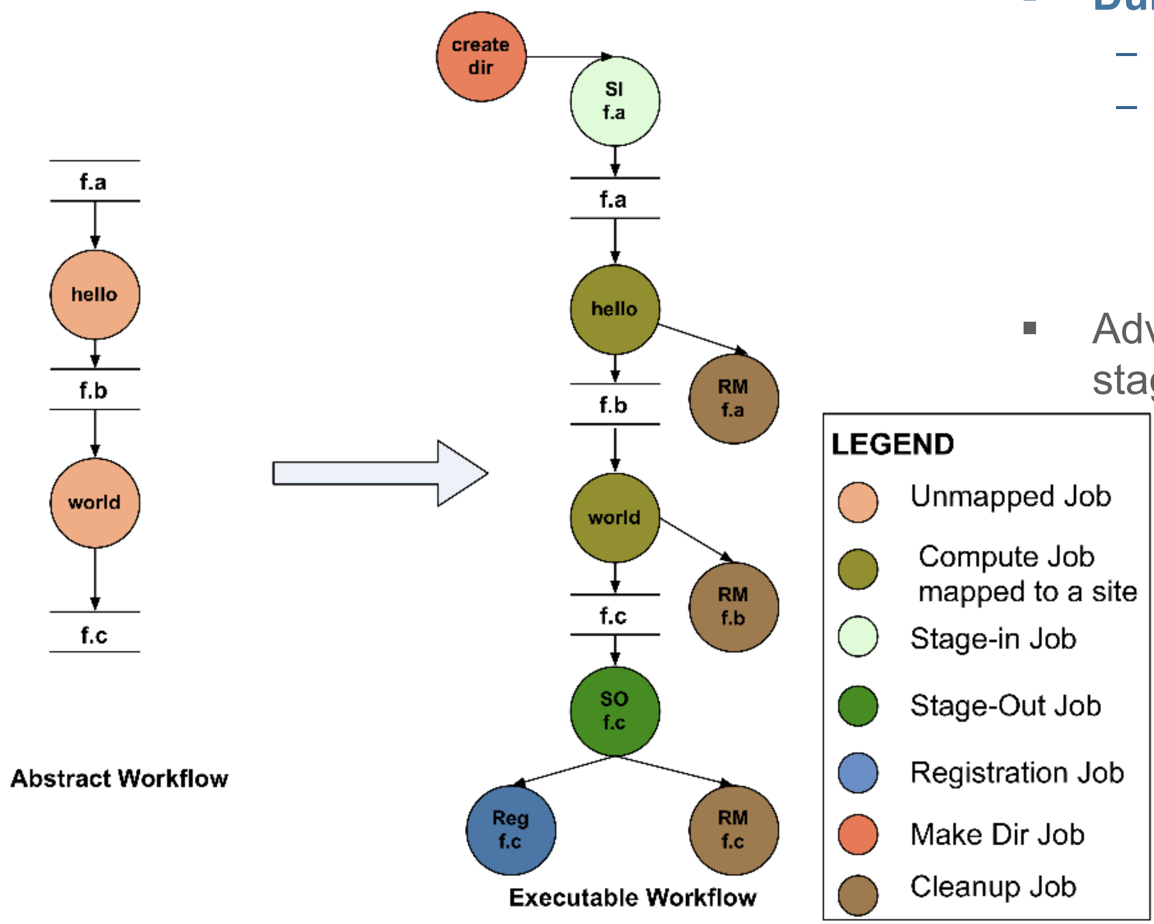
Pegasus Workflow Management System

- **Abstract Workflows - Pegasus input workflow description**
  - Workflow "high-level language"
  - Only identifies the computation, devoid of resource descriptions, devoid of data locations
  - File Aware



- **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

Abstract to Executable Workflow Mapping



- **During mapping process, Pegasus:**
  - Figures out where a job is run
  - What input data to use, adds data stagein and stageout to stage in and out the data.
- Advantage of having separate data stage-in and stage-out nodes
  - Optimizations like limiting the number of stage-in nodes for large workflows
  - No pre-staging of input data
  - Can symlink against existing data.
  - Allows for funneling in data when interfacing with low performance data servers.

Pegasus Data Management

- **Earlier Approach**
  - Stage-in nodes always staged input data to shared filesystem on compute site.
  - Static binding of jobs. Made it hard to support late binding of tasks.

- **New Hybrid Approach**

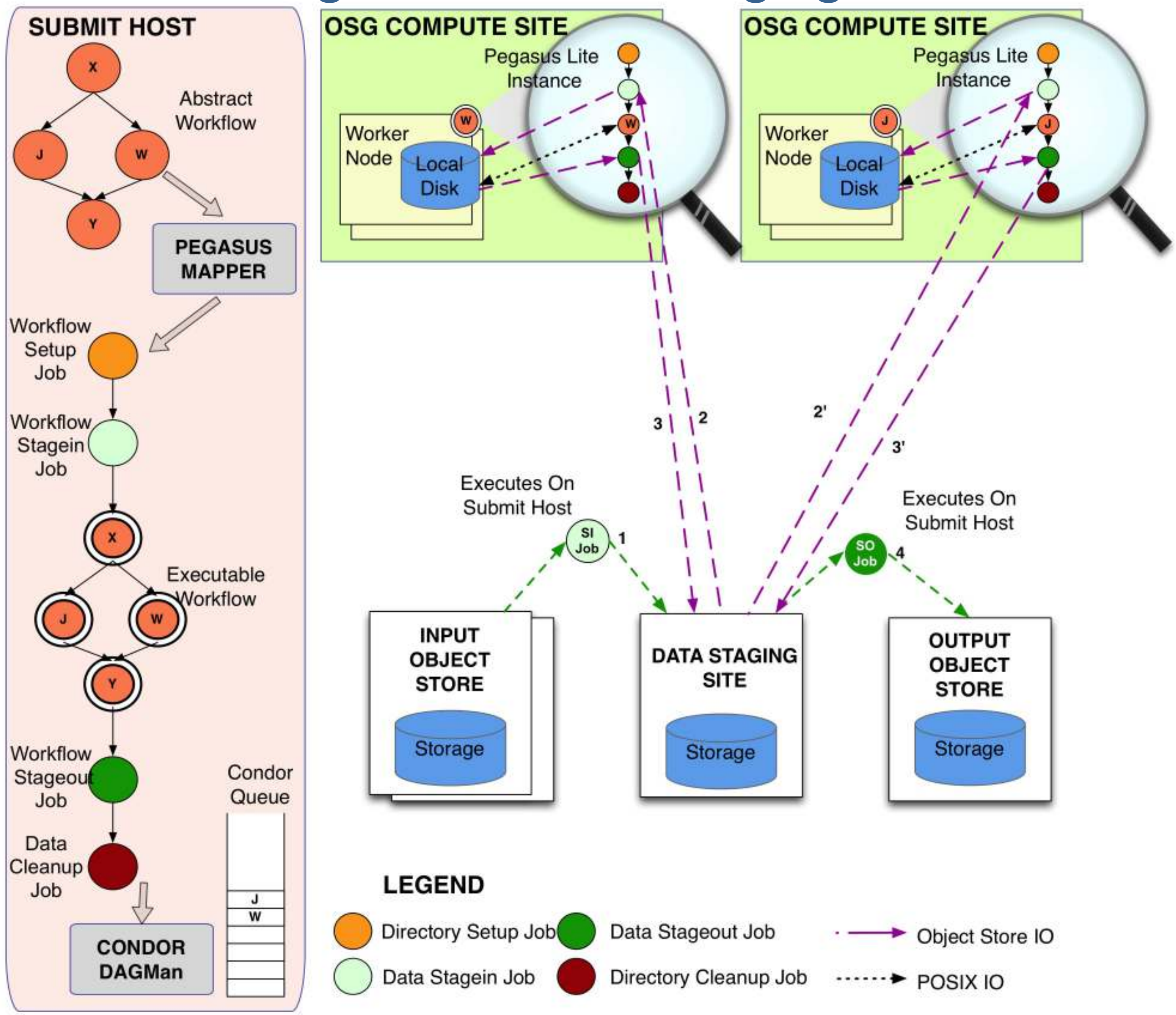
  Data Staging Site

  1. Still add data stage-in nodes and stage-out nodes, but don't tie to execution site. Instead place it on a data-staging site for the execution site.
  2. Stores input data and all intermediate data

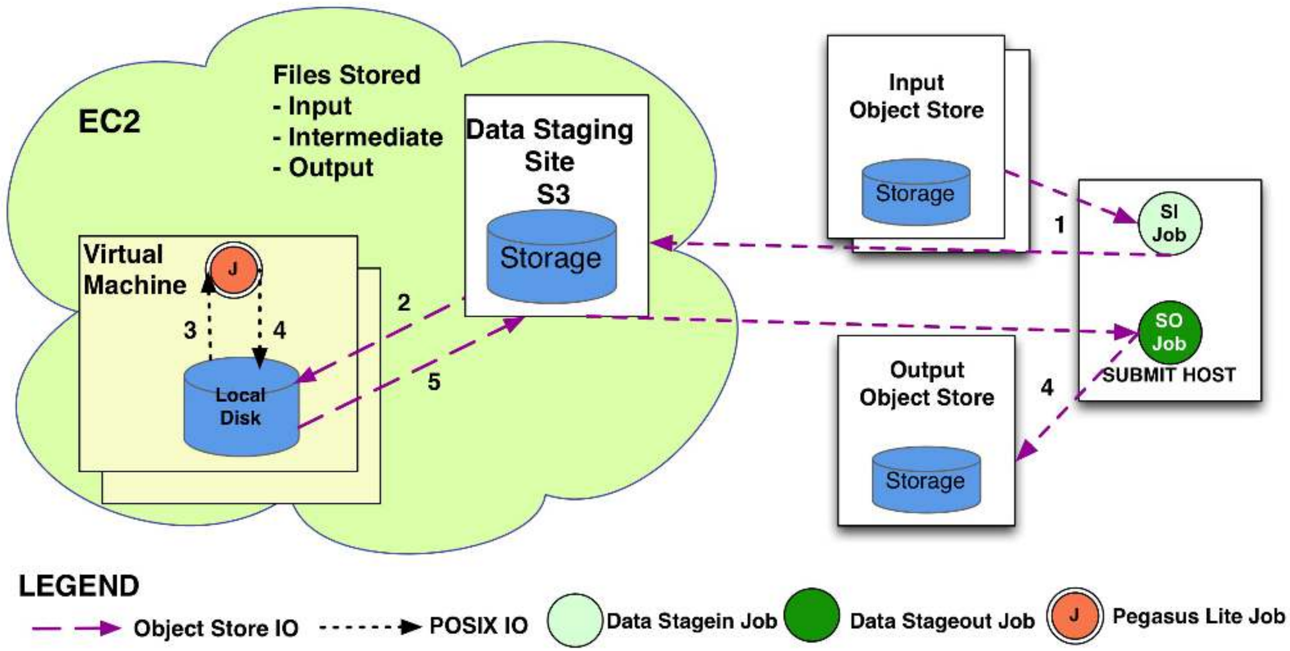
  Pegasus Lite

  1. Pegasus Mapper does workflow level reasoning and optimizations.
  2. Delagates set of runtime decisions to Pegasus Lite that runs on worker nodes
     - Discovers directory on which to run the tasks
     - Pulls in the data from input site or data staging site
     - interfaces with local transfer tools present on the nodes.
     - Runs the task
     - Stageout the data back to data staging site.

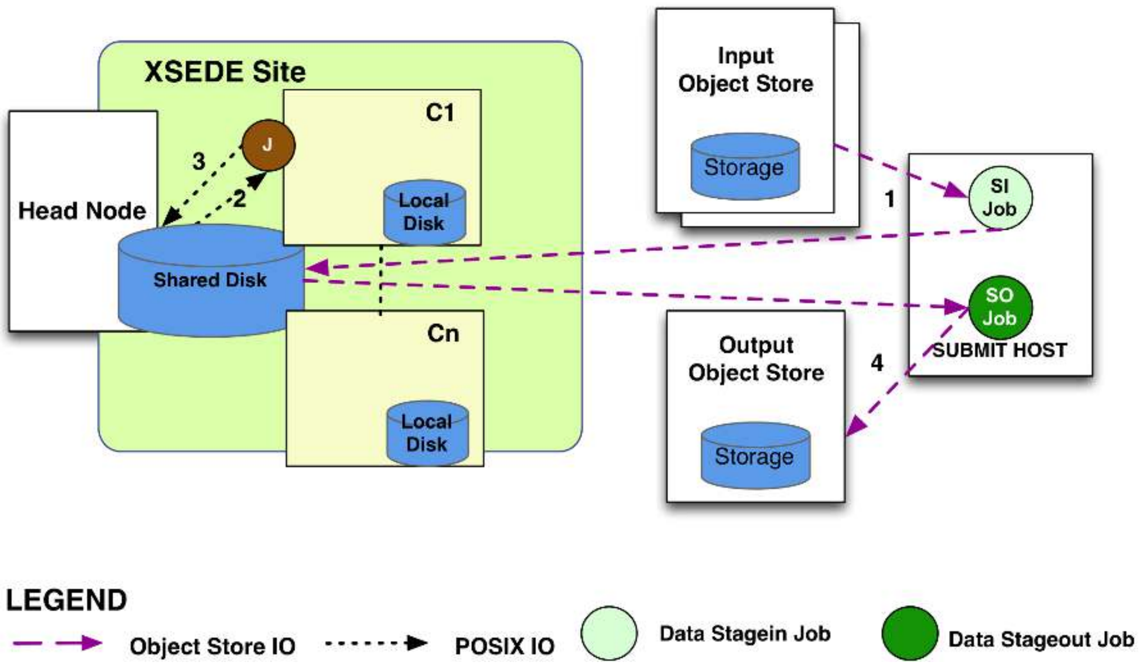
Pegasus Data Configuration: Workflows on OSG using SRM as Data Staging Site.



Pegasus Data Configuration: Workflows on EC2 with S3 as Data Staging Site.



Pegasus Data Configuration: Workflows on XSEDE with shared filesystem as Data Staging Site.



Outline

- Introduction
- Object Stores for Workflows
- Pegasus Data Management
- Experiments
- Conclusions and Future Work

Experiments

- Goal of this work
  - Provide easy to use solution to execute data intensive workflows in variety of different environments.
  - Not necessary to improve workflow performance.
- Workflow Experiments
  - 2 application workflows
    - Montage – I/O Intensive
    - Rosetta - Compute intensive
  - Execution environment
    - Executed on Amazon EC2,
      - dedicated NFS file server (m1.xlarge)
      - one submit node (c1.xlarge) and 8 worker instances (c1.xlarge)
    - Data Configuration
      - Shared File System setup with NFS as data staging site
      - Non Shared File System setup with S3 as data Staging Site

Experiments

	NFS Shared FS (minutes)	S3 – Nonshared FS (minutes)
Walltime	70	129
Cumulative Kickstart Time	921	220
Cumulative Job Time	1030	1196

Table1: Average runtimes for I/O intensive montage workflow.

	NFS Shared FS (minutes)	S3 – Nonshared FS (minutes)
Walltime	57	95
Cumulative Kickstart Time	2935	2966
Cumulative Job Time	2936	4557

Table2: Average runtimes for CPU bound Rosetta workflow.

Conclusions and Future Work

- Supporting different and varied execution and data setup environments is a challenging and important task for workflow systems.
- Our approach of decoupling a data staging site from the shared filesystem allows for great flexibility and can be used by other workflow systems.
  - Pegasus has implemented the above model allowing users the flexibility on running on varied infrastructure ranging from computation grids, supercomputing class machines to computational clouds.
- Put in hooks in Pegasus Lite to leverage application specific compute infrastructure such as LIGO, where data is replicated out of band.

Relevant Links

- **Pegasus: http://pegasus.isi.edu**
- **Tutorial and documentation: http://pegasus.isi.edu/wms/docs/latest/**
- **Support: pegasus-users@isi.edu**
  pegasus-support@isi.edu

Acknowledgements

Pegasus Team, Condor Team, funding agencies, NSF, NIH, and everybody who uses Pegasus.

Thank you!

USC Viterbi School of Engineering
Information Sciences Institute