Managing large-scale workflows with Pegasus

Karan Vahi (vahi@isi.edu)

Collaborative Computing Group
USC Information Sciences Institute

Funded by the National Science Foundation under the OCI SDCI program, grant #0722019
Pegasus Workflow Management System

- Takes in a workflow description and can map and execute it on a wide variety of environments:
  - Local desktop
  - Local Condor Pool
  - Local Campus Cluster
  - Grid
  - Commercial or Academic Clouds
Pegasus
Workflow Management System

- NSF funded Project and developed since 2001
- A collaboration between USC and the Condor Team at UW Madison (includes DAGMan)
- Used by a number of applications in a variety of domains
- Builds on top of Condor DAGMan.
  - Provides reliability—can retry computations from the point of failure
  - Provides scalability—can handle many computations (1-10⁶ tasks)
- Automatically captures provenance information
- Can handle large amounts of data (order of Terabytes)
- Provides workflow monitoring and debugging tools to allow users to debug large workflows
Pegasus WMS

Users

Scripting Tools
- Python
- Java

CGSMD Portal

Pegasus GUI
- Other Workflow Composition Tools: Xbaya, Wings, Triana

COMMERCIAL AND SCIENCE CLOUDS

Local Clusters
- GRAM
- LSF
- SGE
- CONDOR

MIDDLEWARE

COMPUTE

STORAGE

Distributed Resources (XSede, Open Science Grid)
- GRAM
- PBS
- LSF
- SGE
- CONDOR

MIDDLEWARE

COMPUTE

STORAGE
Abstract Workflow (DAX)

- Pegasus Input Workflow description—DAX
  - workflow “high-level language”
  - devoid of resource descriptions
  - devoid of data locations
  - refers to codes as logical transformations
  - refers to data as logical files

- You can use Java, Perl, Python APIs to generate DAXes
Comparison of DAX and Condor DAG

- **Abstraction provides**
  - **Ease of Use** (do not need to worry about low-level execution details)
  - **Portability** (can use the same workflow description to run on a number of resources and/or across them)
  - **Gives opportunities for optimization** and fault tolerance
    - automatically restructure the workflow
    - automatically provide fault recovery (retry, choose different resource)

Diagram showing the comparison between abstract and final executable workflows.
Issues for Large Scale Workflows

❖ Debug and Monitor Workflows
  ✴ 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?

❖ Data Management
  ✴ How do you ship in the large amounts data required by the workflows?

❖ Restructure Workflows for Improved Performance
  ✴ Can have lots of short running jobs
  ✴ Leverage MPI
Workflow Monitoring - Stampede

- Leverage Stampede Monitoring framework with DB backend
  - Separates DB loading infrastructure and log representation
  - Populates data at runtime. A background daemon monitors the logs files and populates information about the workflow to a database
  - Supports SQLite or MySQL
  - Python API to query the framework
  - 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 workflow
  - pegasus-plots
    - Visualization of your workflow execution
Workflow Monitoring - Stampede

Hosts Over Time – Distribution of Different Job Types on Hosts

Workflow Gantt Chart

Invocation breakdown by count grouped by transformation name
s6c_lowmass_hope

Jobs and Runtime over Time

https://pegasus.isi.edu
name: ligo/data_trigbank.1.0
Total count: 1764
Succeeded count: 164
Failed count: 0
Min Runtime: 0.025
Max Runtime: 1.0
Avg Runtime: 0.23074721875
Total Runtime: 88.655

Breakdown by
- count
- runtime
Workflow Debugging Through Pegasus

- After a workflow has completed, we can run `pegasus-analyzer` to analyze the workflow and provide a summary of the run.

- `pegasus-analyzer's` output contains:
  - A brief summary section
    - Showing how many jobs have succeeded
    - And how many have failed.
  - For each failed job
    - Showing its last known state
    - Exitcode
    - Working directory
    - The location of its submit, output, and error files.
    - Any stdout and stderr from the job.
Workflow and Task Notifications

- Users want to be notified at certain points in the workflow or on certain events.

- Support for adding Notification to Workflow and Tasks
  - Event based callouts
    - On Start, On End, On Failure, On Success
  - Provided with email and jabber notification scripts
  - Can run any user provided script as notification.
  - Defined in the DAX.
Three General Configurations Supported

- **Shared Filesystem setup (Typical of Xsede sites)**
  - Worker nodes and the Head Node have a shared filesystem.
  - Can leverage symlinking against existing datasets.

- **NonShared Filesystem setup with a staging site (Typical of OSG or Campus Condor Pools)**
  - Worker Nodes don’t share a filesystem.
  - Data is pulled from an external staging site.

- **Condor IO**
  - Worker Nodes don’t share a filesystem.
  - Data is pulled from the submit host.
Data Flow For Pegasus Workflows

**LEGEND**
- Orange: Directory Setup Job
- Green: Data Stageout Job
- Red: Directory Cleanup Job
- Blue: Data Stagein Job

**SUBMIT HOST**
- Abstract Workflow
- Pegasus Planner
- Workflow Setup Job
- Workflow Stagein Job
- Executable Workflow
- Workflow Stageout Job
- Data Cleanup Job
- Condor DAGMan

**OSG COMPUTE ELEMENT - 1**
- Head Node
- Pegasus Lite Instance
- WN

**STAGING SITE**
- SRM
- GridFTP
- irods
- Storage

**OUTPUT SITE**
- SRM
- GridFTP
- irods
- Storage

1. Submit Site 1 SRM GridFTP irods S3
2. Execute Site 1 Job
3. Staging Site SRM GridFTP irods S3
4. Submit Site 1 Job

**Data Flow Diagram**
- Workflow execution flow from Submit Host to Staging Site and back to Submit Host.
Tip: Set pegasus.data.configuration = sharedfs
WF Reduction (Data Reuse)

Abstract Workflow

File f.d exists somewhere. Reuse it. Mark Jobs D and B to delete

Delete Job D and Job B

Users can use this to move their computations to a different cluster in case of failure
File cleanup

- Problem: Running out of space on shared scratch
  - In OSG scratch space is limited to 30Gb for all users

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

- 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
Storage Improvement for Montage Workflows

Montage 1 degree workflow run with cleanup on OSG-PSU
Workflow Restructuring to improve Application Performance

- 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 job on the grid that takes at least 10 minutes to execute
Job Clustering

- **Level-based clustering**
- **Vertical clustering**
- **Arbitrary clustering**

Useful for small granularity jobs
Previous solution: Glideins

- Pegasus clusters the jobs in a workflow and runs these jobs on a dynamic Condor pool
  - Pool is grown by submitting condor_startd daemons to remote cluster

- Works great on “regular” clusters
  - XSEDE: Ranger, …
  - OSG

- Not so great on some newer Cray/IBM/… architectures
  - Problem 1: no/limited networking on compute nodes
  - Problem 2: queuing system optimized for large jobs
pegasus-mpi-cluster

- Planner creates subgraph based on user assigned labels
- Subgraph is expressed as DAG (simplified Condor DAGMan format)
- Submitted to remote resource (usually GRAM and CondorG)
- Executed with MPI master/worker DAG engine
Large Workflows on Xsede using PMC
Summary –
What Does Pegasus provide an Application - I

❖ All the great features that DAGMan has!
   ♦ Scalability - Hierarchal Workflows. Pegasus runs workflows ranging from few computational tasks upto 1 million
   ♦ Retries in case of failure.

❖ Portability / Reuse
   ♦ User created workflows can easily be run in different environments without alteration.

❖ Performance
   ♦ The Pegasus mapper can reorder, group, and prioritize tasks in order to increase the overall workflow performance.
Summary – 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.

❖ **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.

❖ **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.

❖ **Error Recovery**
   - Reuse existing output products to prune the workflow and move computation to another site.
Some Applications using Pegasus

- **Astronomy**
  - Montage, Galactic Plane, Periodograms

- **Bio Informatics**
  - Brain Span, RNA Seq, SIPHT, Epigenomics, Seqware

- **Earthquake Science**
  - Cybershake, Broadband from Southern California Earthquake Center

- **Physics**
  - LIGO

Complete Listing: [http://pegasus.isi.edu/applications](http://pegasus.isi.edu/applications)
Relevant Links

- Pegasus WMS: [http://pegasus.isi.edu/wms](http://pegasus.isi.edu/wms)
- Tutorial and VM: [http://pegasus.isi.edu/tutorial/](http://pegasus.isi.edu/tutorial/)
- Ask not what you can do for Pegasus, but what Pegasus can do for you: [pegasus@isi.edu](mailto:pegasus@isi.edu)

Acknowledgements

- Pegasus Team, Condor Team, all the Scientists that use Pegasus, Funding Agencies NSF, NIH..