Scientific Workflows with Pegasus WMS

Gideon Juve

Science Automation Technologies Group
USC Information Sciences Institute
Common Workflow Challenges

- **Portability**
  - Can you run a pipeline on Amazon EC2 one day, and a PBS cluster the next?

- **Performance and Scalability**
  - How can you manage large workflows with thousands of tasks and TBs of files?

- **Data Management**
  - What about complex data flows across multiple sites?

- **Provenance**
  - Can you go back and find out how and where data was produced?

- **Reliability**
  - How do you handle failures and retries?

- **Monitoring and Troubleshooting**
  - If something fails, can you identify the problem quickly (at all)?
Pegasus Workflow Management System (WMS)

- Under development since 2001
- A collaboration between USC/ISI and the Condor Team at UW Madison
  - USC/ISI develops Pegasus planner
  - UW Madison develops DAGMan and Condor
- Used by many applications in a variety of domains
  - Earth science, physics, astronomy, bioinformatics
Example: SNS Parameter Refinement

- Spallation Neutron Source at ORNL
- Parameter sweeps of MD and neutron scattering simulations
  - Fit simulation to experimental data
  - e.g. temperature, charge, force
- Nanodiamond Workflow
  - Feb 2015 on Hopper using GRAM and GridFTP
  - 19 parameter values for nonbonded interactions between ND and H2O
  - 800 core NAMD jobs x 22 hrs
  - 400 core Sassena jobs x 3 hrs
  - ~380,000 CPU hours
  - ~1/2 TB output
Example: Periodogram Exoplanet Workflow

- Kepler continuously monitors the brightness of over 175,000 stars
  - Search for periodic dips in signals as Earth-like planets transit in front of host star.
- For each star, Kepler data is used to create a “light curve”
- Need to perform a bulk analysis of all the data to search for these periodic signals

2012 Run at SDSC
- 1.1M tasks, 180 jobs
- 1.1M input, 12M output files
- ~101,000 CPU hours
- 16 TB output data
Example: Montage Image Mosaics

- **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)
  - Run on Amazon EC2 2013-2014

- Need to support hierarchical workflows and scale

John Good (Caltech)
Example: CyberShake PSHA Workflow

- 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)

2014: 286 Sites, 4 models
- Each site = one workflow
- Each workflow has 420,000 tasks in 21 jobs using task clustering w/ PMC
- NCSA BlueWaters, TACC Stampede
Why use Pegasus?

- Maps abstract workflows to diverse computing infrastructure
  - Desktop, Condor Pool, HPC Cluster, Grid, Cloud
- Supports large-scale, data-intensive workflows
  - Experience up to $O(1M)$ tasks and $O(10TB)$ of data
- Automatically plans and executes data transfers
  - Supports complex data flows
- Manages failures to provide reliability
  - Including retries, checkpointing and re-planning
- Provides tools to allow users to monitor and troubleshoot large workflows
- Technical support
  - Funding to support users, mailing lists, chat room, public bug tracker, open source, regular releases, decent documentation
Key Pegasus Concepts

- **Pegasus WMS == Pegasus planner (mapper) + DAGMan workflow engine + Condor scheduler/broker**
  - Pegasus maps workflows to infrastructure
  - DAGMan manages dependencies and reliability
  - Condor is used as a broker to interface with different schedulers

- **Workflows are DAGs (or hierarchical DAGs)**
  - Nodes: jobs, edges: dependencies
  - No while loops, no conditional branches

- **Planning occurs ahead of execution**
  - (Except hierarchical workflows)

- **Planning converts an abstract workflow into a concrete, executable workflow**
  - Planner is like a compiler
Pegasus Workflows are Directed Acyclic Graphs

- **Nodes are tasks**
  - Typically, executables with arguments
  - Nodes can also be other workflows

- **Edges are dependencies**
  - Represent data flow
  - Can also be control dependencies
  - Pegasus can infer edges from data use

- **No loops, no branches**
  - Recursion is possible
  - Can generate workflows in a workflow
  - Can conditionally skip tasks with wrapper
Abstract to Executable Workflow Mapping

- 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)
**General Workflow Execution Model**

- **Input Data Site**, **Compute Site** and **Output Data Sites** can be co-located
  - Example: Input data is already present on the compute site.
  - Each task in the workflow opens one or more input files
  - Read or write a portion of it and then close the file.

- 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.
Data Staging Configurations

Shared File System (typical of XSEDE and HPC sites)
- Worker nodes and the head node have a shared filesystem, usually a parallel filesystem with high-performance I/O
- Can leverage symlinking against pre-staged datasets
- Staging site is the compute site

Non-shared File System (typical of OSG and EC2)
- Worker nodes don’t share a file system
- Uses a staging site separate from the compute site such as Amazon S3
- Data is pulled from / pushed to the staging site
Data Staging Configurations

Condor I/O (Typical of Condor Pools like OSG sites)

- Worker nodes don’t share a file system
- Data is pulled from / pushed to the submit host via Condor file transfers
- Staging site is the submit host

- Supports many different protocols
  - HTTP
  - SCP
  - GridFTP
  - IRODS
  - Amazon S3
  - SRM
  - cp
  - ln -s

Using Pegasus allows you to move from one configuration to another without changing the workflow description
Workflow Reduction (Data Reuse)

Done automatically when output files are discovered in replica catalog. Useful when you have done a part of computation and then realize the need to change the structure.
Data Cleanup

- **Problem:** Running out of disk space during workflow execution

- **Why does it occur**
  - Workflows could bring in large amounts of data
  - Data is generated during workflow execution
  - Applications don’t clean up after they are done

- **Solution**
  1. **Do cleanup after workflows finish**
     - Cleanup is last job in the workflow
  2. **Interleave cleanup automatically during workflow execution**
     - Analyze the workflow to determine when a file is no longer required
     - **Cluster the cleanup jobs by level for large workflows**

**Example:** Used by a UCLA genomics researcher to delete TB’s of intermediate data automatically during long running workflows
Data Cleanup Example

Montage 1 degree workflow run with cleanup

Use the --cleanup option for pegasus-plan
- “none” for no cleanup
- “inplace” to clean up as soon as possible
- “leaf” to clean up at the end of the workflow
Task Clustering

- 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 jobs that take at least 10/30/60/? minutes
  - Clustered tasks can reuse common input data – less data transfers

Also: time-based clustering
Workflow Monitoring and Reporting

- **Data collection**
  - Data extracted from log files and stored in a relational database
  - DB contains workflow structure, status information, runtimes, host info, task stdout/stderr

- **Reporting tools**
  - Status of the workflow
    - pegasus-status path/to/submit/directory
  - Detailed runtime statistics
    - pegasus-statistics -s all path/to/submit/directory

<table>
<thead>
<tr>
<th>Type</th>
<th>Succeeded</th>
<th>Failed</th>
<th>Incomplete</th>
<th>Total</th>
<th>Retries</th>
<th>Total+Retries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tasks</td>
<td>135002</td>
<td>0</td>
<td>0</td>
<td>135002</td>
<td>0</td>
<td>135002</td>
</tr>
<tr>
<td>Jobs</td>
<td>4529</td>
<td>0</td>
<td>0</td>
<td>4529</td>
<td>0</td>
<td>4529</td>
</tr>
<tr>
<td>Sub-Workflows</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

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)
Pegasus Dashboard

- Web-based workflow monitoring GUI
  - Data comes from monitoring database
  - Supports monitoring, troubleshooting, and reporting
Other Features

- Hierarchical Workflows
- Pegasus-MPI-Cluster
- Troubleshooting tools (pegasus-analyzer)
- Workflow and Task Notifications
- Job and Transfer Throttling
- Executable Staging
- Task Profiling via Kickstart
- Multi-site execution
- Shell Planner Mode
Workflow Infrastructure Requirements

1. A place to run
   - We call this the “submit host”
   - Needs to have reasonable uptime while running workflows
   - Policy may prevent us from using the cluster head node
   - Needs to have network access for job and data management
   - e.g. head node at USC, VM at OLCF, workflow.isi.edu

2. An interface for submitting jobs
   - Needs to be automatic: no manual RSA tokens for 2 factor auth
   - Needs to be fairly robust

3. A way to transfer data
   - Needs to get data into and out of scratch and project storage
   - High performance

   - Nice to have
     - Good infrastructure monitoring and testing
Summary – What Does Pegasus provide an Application

- **Portability / Reuse**
  - User created workflows can 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.

- **Scalability**
  - Pegasus can easily scale both the size of the workflow, and the resources that the workflow is distributed over. Pegasus runs workflows ranging from just a few computational tasks up to 1 million.
Summary – What Does Pegasus provide an Application

- **Provenance**
  - Provenance data is collected in a database, and the data can be summaries with tools such as pegasus-statistics, Pegasus Dashboard, 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**
  - Jobs and data transfers are automatically retried in case of failures. Debugging tools such as pegasus-analyzer help the user to debug the workflow in case of non-recoverable failures.
More Information

- **Website:**
  - [http://pegasus.isi.edu](http://pegasus.isi.edu)

- **Tutorial:**
  - [http://pegasus.isi.edu/wms/docs/latest/tutorial.php](http://pegasus.isi.edu/wms/docs/latest/tutorial.php)

- **Documentation:**
  - [http://pegasus.isi.edu/documentation](http://pegasus.isi.edu/documentation)

- **Contact:**
  - Pegasus users list (public): [pegasus-users@isi.edu](mailto:pegasus-users@isi.edu)
  - Pegasus support (private): [pegasus-support@isi.edu](mailto:pegasus-support@isi.edu)
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 jobs to remote resources