Science Automation with the Pegasus Workflow Management System

Ewa Deelman

USC Information Sciences Institute

http://www.isi.edu/~deelman

Funding from DOE, NSF, and NIH
The Problem

- Scientific data is being collected at an ever increasing rate
  - The “old days” -- big, focused experiments – LHC
  - Today also “cheap” DNA sequencers – and an increasing number of them

- The complexity of the computational problems is ever increasing

- Local compute resources are often not enough
  - Too small, limited availability
  - Data sets are distributed

- The computing infrastructure keeps changing
  - Hardware, software, but also computational models
Our approach

- Provide a way to structure applications in such a way that enables them to be automatically managed
  - In a portable way: same description that works on different resources
  - In a way that scientists can interpret the results

- Develop a system that
  - Maps the application description onto the available resources
  - Manages its execution on heterogeneous resources
  - Sends results back to the user or archive
  - Provides good performance, reliability, scalability
Outline

- Scientific Workflows and Application Examples
- Managing scientific workflows
- Pegasus and its features
- Conclusions
Science-grade Mosaic of the Sky
Science-grade Mosaic of the Sky

Montage Workflow

Amazon M1 large with 2 cores

<table>
<thead>
<tr>
<th>Size of mosaic</th>
<th>Number of input data files</th>
<th>Number of tasks</th>
<th>Number of intermediate files</th>
<th>Total data footprint</th>
<th>Cumulative wall time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>84</td>
<td>387</td>
<td>770</td>
<td>1.8 GB</td>
<td>11 mins</td>
</tr>
<tr>
<td>2</td>
<td>300</td>
<td>1442</td>
<td>2880</td>
<td>6.4 GB</td>
<td>43 mins</td>
</tr>
<tr>
<td>4</td>
<td>685</td>
<td>3738</td>
<td>7466</td>
<td>17 GB</td>
<td>1 hour, 56 mins</td>
</tr>
<tr>
<td>6</td>
<td>1461</td>
<td>7462</td>
<td>14904</td>
<td>35 GB</td>
<td>3 hours, 42 mins</td>
</tr>
<tr>
<td>8</td>
<td>2565</td>
<td>12757</td>
<td>25480</td>
<td>59 GB</td>
<td>6 hours, 45 mins</td>
</tr>
</tbody>
</table>
Some workflows are large-scale and data-intensive

- 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)

- Need to support hierarchical workflows and scale

John Good (Caltech)
Workflows can be simple!
Sometimes you want to “hide” the workflow

Credit: Frank McKenna
UC Berkeley, NEES, HUBzero
Sometimes the environment is complex

- Campus Cluster
- XSEDE
- NERSC
- ALCF
- OLCF
- Open Science Grid
- FutureGrid
- Amazon Cloud
Sometime you want to change or combine resources

You don’t want to recode your workflow
Workflow Management

- Assume a high-level workflow specification
- Assume the potential use of different resources within a workflow or over time
  - Need a planning capability to map from high-level to executable workflow
  - Need to manage the task dependencies
  - Need to manage the execution of tasks on the remote resources
- Need to provide provenance information
- Need to provide scalability, performance, reliability
Outline

- Scientific Workflows and Application Examples
- Managing scientific workflows
- Pegasus and its features
- Conclusions
Our Approach

- **Analysis Representation**
  - Support a declarative representation for the workflow
  - Represent the workflow structure as a Directed Acyclic Graph (DAG)
  - Tasks operate on files
  - Use recursion to achieve scalability

- **System (Plan for the resources, Execute the Plan, Manage tasks)**
  - Layered architecture, each layer is responsible for a particular function
  - Mask errors at different levels of the system
  - Modular, composed of well-defined components, where different components can be swapped in
  - Use and adapt existing graph and other relevant algorithms
Submit locally, compute Globally

Montage Galactic Plane Workflow

Local Resource

Data Storage

Work definition

Workflow Management System

Local Resource

Work

data

Campus Cluster
XSEDE
NERSC
ALCF
OLCF
Open Science Grid
FutureGrid
Amazon Cloud
Pegasus
Workflow Management System (est. 2001)

- A collaboration between USC and the Condor Team at UW Madison
- Maps a resource-independent “abstract” workflow onto resources and executes the “concrete” workflow
- Used by a number of applications in a variety of domains
- Provides reliability—can retry computations from the point of failure
- Provides scalability—can handle large data and many computations (kbytes-TB of data, 1-10^6 tasks)
- Infers data transfers, restructures workflows for performance
- Automatically captures provenance information
- Can run on resources distributed among institutions, laptop, campus cluster, Grid (OSG, XSEDE), Cloud (Amazon, FutureGrid)
Pegasus Workflow Management System

- **A workflow “compiler”**
  - Input: abstract workflow description, resource-independent
  - Auxiliary Info (catalogs): available resources, data, codes
  - Output: executable workflow with concrete resources
  - Automatically locates physical locations for both workflow tasks and data
  - Transforms the workflow for performance and reliability

- **A workflow engine (DAGMan)**
  - Executes the workflow on local or distributed resources (HPC, clouds)
  - Task executables are wrapped with `pegasus-kickstart` and managed by Condor `schedd`

- Monitoring tools
- Provenance and execution traces collection
Generating executable workflows

APIs for workflow specification (DAX---DAG in XML)

Java, Perl, Python

Legend:
- Unmapped Job
- Compute Job mapped to a site
- Stage-in Job
- Stage-Out Job
- Registration Job
- Make Dir Job
- Cleanup Job
How do workflows start?
Time to solution: 2 weeks - 3 months

Execution on USC resources

Number of jobs

- 146 filterContains
- 146 fastq2blq
- 146 sort2anger
- 146 map2merge
- 146 mapindex
- 146 pileup
Pegasus optimizations address issues of:

- Failures in the execution environment or application
- Data storage limitations on execution sites
- Performance
  - Small workflow tasks
- Heterogeneous execution architectures
  - Different file systems (shared/non-shared)
  - Different system architectures (Cray XT, Blue Gene, ...)

USC Viterbi
School of Engineering
Sometimes fatal errors occur during workflow execution

Want to restart the workflow from where it left off
Sometimes intermediate data is already available

Workflow Reduction
Data Reuse
Workflow-level checkpointing
Storage limitations

“Small” amount of space

Automatically add tasks to “clean up” data no longer needed
Montage

1.25GB versus 4.5 GB
Storage limitations

Variety of file system deployments: shared vs non-shared

User workflow
pegasus-transfer subsystem

- Command line tool used internally by Pegasus workflows
- Input is a list of source and destination URLs
- Transfers the data by calling out to tools – provided by the system (cp, wget, ...) Pegasus (pegasus-gridftp, pegasus-s3) or third party (gsutil)
- Transfers are parallelized
- Transfers between non-compatible protocols are split up into two transfers using the local filesystem as a staging point – for example: GridFTP->GS becomes GridFTP->File and File->GS

Supported URLs
- GridFTP
- SRM
- iRods
- S3
- GS
- SCP
- HTTP
- File
- Symlink
Sometimes the environment is just not exactly right

Single core workload

XSEDE HPC Resources

https://www.tacc.utexas.edu/resources/hpc

Designed for MPI codes
Each site in the input map corresponds to one workflow. Each workflow has 820,000 tasks.

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).
Solutions

Cluster tasks

-- also good for performance

Use an MPI-based workflow management engine to manage sub-workflows

Use “pilot” jobs to dynamically provision a number of resources at a time

Use an MPI-based workflow management engine to manage sub-workflows
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, used on XSEDE
  - Requires: MPI, a shared file system, and fork()
- Allows sub-graphs of a Pegasus workflow to be submitted as monolithic grid jobs to remote resources
Enables earthquake scientists (SCEC) to run post-processing (single core) computations on new architectures (Stampede, Titan, Blue Waters)
The green partition boxes show user-defined partitions of the workflow which are used when mapping the workflow to TACC Stampede. The walltime of a partition will not exceed the 48-hour wall clock limit, given a certain number of compute nodes.

SoyKB: Bioinformatics analysis of 1000+ resequenced soybean germplasm lines selected for major traits including oil, protein, soybean cyst nematode resistance (SCN), abiotic stress resistance (drought, heat and salt) and root system architecture.
The green jobs are MPI container jobs of the partitions shown in the earlier slide. These are run on the remote supercomputer. The blue auxiliary tasks are run on the workflow submit host, and handle things like data staging and cleanup.

Input data is fetched from iPlant Data Store, or if already replicated, from the TACC iPlant Data Store node for close to computation access.

Outputs are automatically put back into the Data Store for easy access and further analysis in the iPlant Discovery Environment.
Pegasus-kickstart

- Lightweight C based executable to launch jobs
- Captures job runtime provenance and logs it as a XML record
- Following information is captured about each job on all supported platforms
  - exit code with which the job it launched exited
  - start time and duration of the job
  - hostname and IP address of the host the job ran on
  - stdout and stderr of the job
  - arguments with which it launched the job
  - directory in which the job was launched
  - environment that was set for the job before it was launched
- Additional profiling
  - peak memory usage (resident set size, and vm size)
  - total I/O read and write,
  - Pid
  - all files accessed (total read and write per file)
Workflow Monitoring Dashboard – *pegasus-dashboard*

Status, statistics, timeline of jobs

Helps pinpoint errors
If you are interested in Pegasus

- Pegasus: http://pegasus.isi.edu
- Tutorial and documentation: http://pegasus.isi.edu/wms/docs/latest/
- Virtual Machine with all software and examples http://pegasus.isi.edu/downloads
- Take look at some Pegasus applications: http://pegasus.isi.edu/applications
- User Support available: pegasus-users@isi.edu
If you get stuck...

And you can draw....

The XSEDE ECSS and Pegasus teams can help you!