Before We Start the Tutorial

- Collect the USB sticks containing the VM from the instructors.
  - Start the transfer of the folder in the USB stick to the desktop. Will take about 5 minutes.

- Install the appropriate Virtual Box software for your OS. The USB sticks have copies of the software that you can use to install.

- Open up the tutorial notes in your browser.
  - Online at
Pegasus WMS Tutorial
CCGrid 2011

Karan Vahi, Gaurang Mehta
(vahi@isi.edu, gmehta@isi.edu)

Advanced Systems Division
USC Information Sciences Institute
Outline of Tutorial

- Introduction to scientific workflows
- Pegasus-WMS and Composition of a Simple Workflow In terms of DAX
- Pegasus Internals
- Mapping and Executing Simple Workflow Locally
- Monitoring, Debugging and Statistics
- Mapping and Executing Simple Workflow On the Grid
- Optimization techniques for mapping and executing Large Scale workflows
- Applications using Pegasus
- Current and Future Research
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
- Allows the scientists to modularize their application
- Scaled up execution over several computational resources
- Provide automation
- Foster Collaborations
Generating mosaics of the sky (Bruce Berriman, Caltech)

<table>
<thead>
<tr>
<th>Size of the mosaic in degrees square*</th>
<th>Number of jobs</th>
<th>Number of input data files</th>
<th>Number of Intermediate files</th>
<th>Total data footprint</th>
<th>Approx. execution time (20 procs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>232</td>
<td>53</td>
<td>588</td>
<td>1.2GB</td>
<td>40 mins</td>
</tr>
<tr>
<td>2</td>
<td>1,444</td>
<td>212</td>
<td>3,906</td>
<td>5.5GB</td>
<td>49 mins</td>
</tr>
<tr>
<td>4</td>
<td>4,856</td>
<td>747</td>
<td>13,061</td>
<td>20GB</td>
<td>1hr 46 mins</td>
</tr>
<tr>
<td>6</td>
<td>8,586</td>
<td>1,444</td>
<td>22,850</td>
<td>38GB</td>
<td>2 hrs. 14 mins</td>
</tr>
<tr>
<td>10</td>
<td>20,652</td>
<td>3,722</td>
<td>54,434</td>
<td>97GB</td>
<td>6 hours</td>
</tr>
</tbody>
</table>

*The full moon is 0.5 deg. sq. when viewed from Earth, Full Sky is ~ 400,000 deg. sq.*
Workflows – Launch and Forget

- A single workflow can take days, weeks or even months
- Automates tasks user *could* perform manually…
  …but **WMS** takes care of automatically
- Includes features such as retries in the case of failures – avoids the need for user intervention
- The workflow itself can include error checking
- The result: one user action can utilize many resources while maintaining complex job inter-dependencies and data flows
- Maximizes compute resources / human time
Types of Workflow Applications

- Providing a service to a community (Montage project)
  - Data and derived data products available to a broad range of users
  - For large numbers of requests or large requests need to rely on shared cyber infrastructure resources
  - On-the-fly workflow generation, portable workflow definition

- Supporting community-based analysis (SCEC project)
  - Codes are collaboratively developed and “strung” together to model complex systems
  - Ability to correctly connect components, scalability

- Processing large amounts of shared data on shared resources (LIGO project)
  - Data captured by various instruments and cataloged in community data registries. Data is spread over clusters.
  - Automation, scalability and reliability

- Automating work of a scientist
  - SIPHT, Epigenomics, Mass Matrix
So you have the “Perfect Pipeline”

- Does what it is supposed to do
- Integrates data from a reference DB
- Makes use of the local cluster—has customized pbs submit scripts, can run on many cores
- Uses the latest sequence mapping code
BUT..

- New compute resources become available (or old one upgraded)
- New algorithm is available and want to compare how well it does
- Reference DB is updated (again!)
- Data is coming at an ever faster rate
- The person that wrote the script moved to a new lab
Why Scientific Workflows?

- Workflows can be portable across platforms and scalable
- Workflows are easy to reuse, support reproducibility
- Can be shared with others
  - Gives a leg-up to new staff, GRAs, PostDocs, etc
- Workflow Management Systems (WMS) can help recover from failures and optimize overall application performance
- WMS can capture provenance and performance information
- WMS can provide debugging and monitoring tools
Workflow Lifecycle

**Reuse**
- Data Products
  - Data, Metadata, Provenance information
  - Workflow and Component Libraries
  - Workflow Template
    - Adapt, Modify
    - Execute
    - Executable Workflow
      - Distributed, Compute, Storage and Network Resources

**Creation**
- Workflow Instance
  - Populate with data
  - Map to available resources

**Planning**
- Resource, Application Component Descriptions

**Scheduling/Execution**
- Execution
  - Workflow Template
  - Workflow Instance

**Reuse**
Workflow Creation

- Design a workflow (semantics info needed)
  - Find the right components
  - Set the right parameters
  - Find the right data
  - Connect appropriate pieces together
  - Find the right fillers

- Support both experts and novices
Workflow Lifecycle

Data Products

Adapt, Modify

Workflow and Component Libraries

Workflow Template

Populate with data

Workflow Instance

Map to available resources

Resource, Application Component Descriptions

Planning

Data, Metadata, Provenance Information

Execute

Executable Workflow

Compute, Storage and Network Resources
Why mapping?

- Many workflow systems support only executable workflow composition
- 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)
Mapping Specification:
Place $Y = F(x)$ at $L$

- Find where $x$ is--- \{S1,S2, \ldots\}
- Find where $F$ can be computed--- \{C1,C2, \ldots\}
- Choose $c$ and $s$ subject to constraints (performance, space availability, \ldots)
- Move $x$ from $s$ to $c$
  - Move $F$ to $c$
- Compute $F(x)$ at $c$
- Move $Y$ from $c$ to $L$
- Register $Y$ in data registry
- Record provenance of $Y$, performance of $F(x)$ at $c$

Errors:
- $x$ was not at $s$!
- $F(x)$ failed!
- $c$ crashed!
- there is not enough space at $L$!
Challenges in Workflow Mapping

- Automated management of data
- Efficient mapping of workflow instances to resources
  - Runtime Performance
  - Data space optimizations
  - Fault tolerance (involves interfacing with the workflow execution system)
    - Recovery by re-planning
    - plan “B”
  - Scalability
- Providing feedback to the user
  - Feasibility, time estimates
Workflow Lifecycle

Data Products

Adapt, Modify

Workflow Template

Workflow and Component Libraries

Populate with data

Workflow Instance

Data, Metadata Catalogs

Resource, Application Component Descriptions

Data, Metadata, Provenance Information

Scheduling/Execution

Compute, Storage and Network Resources

Executable Workflow

Map to available resources
Execution Environment

APPLYING

Pegasus
WMS
Local machine

Distributed Resources:
Open Science Grid
TeraGrid

Resource Information and Data Location Information
NMI: Globus MDS, RLS, SRB

Issues
Select resource
Manage data xfer, registration
Reliability and Performance
Challenges in Workflow Execution

- **Resource provisioning**
  - Which resources to provision if many possibilities?
  - How many resources to provision?
  - For how long?

- **Fault Tolerance**
  - How to recognize different types of failures
  - How to recover from failures?

- **Efficient collaboration between the data and computation management systems**

- **Debugging**
  - How to relate the workflow result (outcome) to workflow specification
Challenges in user experiences

- Users’ expectations vary greatly
  - High-level descriptions
  - Detailed plans that include specific resources

- Users interactions can be exploratory
  - Or workflows can be iterative
    - Modifying portions of the workflow as the computation progresses

- Users need progress, failure information at the right level of detail

- There is no ONE user but many users with different knowledge and capabilities
Outline of Tutorial

- Introduction to scientific workflows
- Pegasus-WMS and Composition of a Simple Workflow In terms of DAX
- Pegasus Internals
- Mapping and Executing Simple Workflow Locally
- Monitoring, Debugging and Statistics
- Mapping and Executing Simple Workflow On the Grid
- Optimization techniques for mapping and executing Large Scale workflows
- Applications using Pegasus
- Current and Future Research
Pegasus WMS

COMMERCIAL AND SCIENCE CLOUDS

LOCAL CLUSTER

DISTRIBUTED RESOURCES (TERAGRID / OSG)

PEGASUS PLANNER

CONDOR DAGMAN

Workflow management system

Visualization tools

Pegasus Portal

Pegasus GUI

USERS
Pegasus workflow

❖ DAX
❖ What it describes
❖ How to read a DAX
❖ How to generate a DAX
  • Describe the various methods
    – Direct XML
    – DAX API
    – Behind portals
  • Migrating from a DAG to DAX
Abstract Workflow (DAX)
Exercise: 1.2.1

- Pegasus 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

- Exercise:
  - Use CreateDAX.java to generate a diamond dax
Comparison of abstract and executable workflows
Understanding DAX

<?xml version="1.0" encoding="UTF-8"?>

<!-- Section 1: Files - Acts as a Replica Catalog (can be empty) →
<file name="f.a">
  <pfn url="file:///scratch/tutorial/inputdata/diamond/f.a" site="local"/>
</file>

<!-- Section 2: Executables - Acts as a Transformation Catalog (can be empty) →
<executable namespace="pegasus" name="preprocess" version="4.0" installed="true" arch="x86"
  os="linux">
  <pfn url="file:///opt/pegasus/default/bin/keg" site="local"/>
</executable>

…

<!-- Section 4: Job's, DAX's or Dag's - Defines a JOB or DAX or DAG (Atleast 1 required) -->

<job id="j1" namespace="pegasus" name="preprocess" version="4.0">
  <argument>-a preprocess -T 60 -i <file name="f.a"/> -o <file name="f.b1" />
  <file name="f.b2"/></argument>
  <uses name="f.a" link="input" transfer="true" register="true"/>
  <uses name="f.b1" link="output" transfer="false" register="false"/>
  <uses name="f.b2" link="output" transfer="false" register="false"/>
</job>

…

<!-- Section 5: Dependencies - Parent Child relationships (can be empty) -->

<child ref="j4">
  <parent ref="j2"/>
  <parent ref="j3"/>
</child></adag>

(excerpted for display) x
Creating Abstract Workflows (DAX)

- Majority of the users use the DAX API to integrate the DAX generation into their pipelines/portals etc.

- Pegasus provides DAX API’s in following languages
  - JAVA
  - PYTHON
  - PERL

- Easy to use
  - Tutorial will cover how to create an abstract workflow using the JAVA API
Outline of Tutorial

- Introduction to scientific workflows
- Pegasus-WMS and Composition of a Simple Workflow In terms of DAX
- Pegasus Internals
- Mapping and Executing Simple Workflow Locally
- Monitoring, Debugging and Statistics
- Mapping and Executing Simple Workflow On the Grid
- Optimization techniques for mapping and executing Large Scale workflows
- Applications using Pegasus
- Current and Future Research
Pegasus WMS

Workflow Description in XML

Properties

- Replica Catalog
- Site Catalog
- Transformation Catalog

Submit Host

Clouds
- TeraGrid
- Open Science Grid
- Campus resources
- Local machine

Pegasus WMS restructures and optimizes the workflow, provides reliability
Simple Steps to Run Pegasus

1. Specify your computation in terms of DAX
   - Write a simple DAX generator
   - Java based API provided with Pegasus or GUI

2. Set up your catalogs
   - Use `pegasus-sc-client` to generate site catalog and transformation catalog for your environment
   - Record the locations of your input files in a replica client using `pegasus-rc-client`

3. Plan and Submit your workflow
   - Use `pegasus-plan` to generate your executable workflow that is mapped onto the target resources and submits it for execution

4. Monitor and Analyze your workflow
   - Use `pegasus-status | pegasus-analyzer` to monitor the execution of your workflow
Discovery

❖ Data
  ✦ Where do the input datasets reside?

❖ Executables
  ✦ Where are the executables installed?
  ✦ Do binaries exist somewhere that can be staged to remote grid sites?

❖ Site Layout
  ✦ What does a grid site look like?
Replica Catalog Overview—finding data

- Replica Catalog stores mappings between logical files and their target locations

- Used to
  - discover input files for the workflow
  - track data products created
  - data reuse

- Data is replicated for scalability, reliability and availability
Pegasus interfaces with a variety of replica catalogs

- **File based Replica Catalog**
  - useful for small datasets (like this tutorial)
  - cannot be shared across users

- **Database based Replica Catalog**
  - useful for medium sized datasets
  - can be used across users

- **Globus Replica Location Service**
  - useful for large scale data sets across multiple users
  - LIGO’s LDR deployment
Optional Replica Catalog
Exercise: 1.2.2

- The pegasus-rc-client is a command line tool to interact with Replica Catalog
  ✩ One client talks to all types of Replica Catalog

- Practical exercise (Refer Exercise 1.2.2):
  ✩ Use the pegasus-rc-client to
    ✩ Populate the Replica Catalog
      - Single insert of an entry
      - Bulk inserts
    ✩ Query the Replica Catalog
    ✩ Remove entries (Offline exercise)
Site Catalog—finding resources

- Contains information about various sites on which workflows may execute

- For each site following information is stored
  - grid gateways
  - head node filesystem
  - worker node filesystem
  - scratch and shared file systems on the head nodes and worker nodes
  - replica catalog URL for the site
  - site wide information like environment variables to be set when a job is run.
Optional Site Catalog Exercise
(Ex 1.2.3 10 minutes)

Ʌ Two clients for generating a site catalog

Ʌ pegasus-sc-client
   ✝ Allows you to generate a site catalog
      • For OSG grid sites by querying OSGMM
<site handle="cluster" arch="x86" os="LINUX" osrelease="" osversion="" glibc="">
  <grid type="gt2" contact="pegasus/jobmanager-fork" scheduler="Fork" jobtype="auxillary"/>
  <grid type="gt2" contact="pegasus/jobmanager-sge" scheduler="SGE" jobtype="compute"/>

  <head-fs>
    <scratch>
      <shared>
        <file-server protocol="gsiftp" url="gsiftp://pegasus" mount-point="/home/pegasus/cluster-scratch"/>
        <internal-mount-point mount-point="/home/pegasus/cluster-scratch"/>
      </shared>
    </scratch>
    <storage>
      <shared>
        <file-server protocol="gsiftp" url="gsiftp://pegasus" mount-point="/home/pegasus/cluster-storage"/>
        <internal-mount-point mount-point="/home/pegasus/cluster-storage"/>
      </shared>
    </storage>
  </head-fs>

  <replica-catalog type="LRC" url="rlsn://localhost"/>

  <profile namespace="env" key="GLOBUS_LOCATION">
    /usr/local/globus/gt-5.0.2
  </profile>
  <profile namespace="env" key="JAVA_HOME">
    /usr/java/default
  </profile>
  <profile namespace="pegasus" key="clusters.num">
    1
  </profile>
  <profile namespace="pegasus" key="stagein.clusters">
    1
  </profile>
</site>
Transformation Catalog ---- finding codes

- Transformation Catalog maps logical transformations to their physical locations

- Used to
  - Discover application codes installed on the grid sites
  - Discover statically compiled codes, that can be deployed at grid sites on demand
Transformation Catalog Overview

- For each transformation following are stored
  - **tr** - A transformation identifier. (Normally a Namespace::Name:Version.. The Namespace and Version are optional.)
  - **pfn** - URL or file path for the location of the executable. The pfn is a file path if the transformation is of type INSTALLED and generally a url (file:/// or http:// or gridftp://) if of type STAGEABLE
  - **site** - The site identifier for the site where the transformation is available
  - **type** - The type of transformation. Whether it is installed ("INSTALLED") on the remote site or is available to stage ("STAGEABLE").
  - **arch os, osrelease, osversion** - The arch/os/osrelease/osversion of the transformation. osrelease and osversion are optional. ARCH can have one of the following values x86, x86_64, sparcv7, sparcv9, ppc, aix. The default value for arch is x86 OS can have one of the following values linux,sunos,macosx. The default value for OS if none specified is linux
  - **Profiles** - One or many profiles can be attached to a transformation for all sites or to a transformation on a particular site.
Transformation Catalog Entry

```plaintext
tr example::keg:1.0 {

    # specify profiles that apply for all the sites for the transformation
    # in each site entry the profile can be overridden
    profile env "JAVA_HOME" "/opt/java/1.6"

    site isi {
        profile env "HELLO" "WORLD"
        profile env "JAVA_HOME" "/bin/java.1.6"
        pfn "/path/to/keg"
        arch "x86"
        os "linux"
        osrelease "fc"
        osversion "4"
        type "INSTALLED"
    }

    site wind {
        profile condor "universe" "condor"
        pfn "file:///path/to/keg"
        arch "x86"
        os "linux"
        osrelease "fc"
        osversion "4"
        type "STAGEABLE"
    }
}
```
pegasus-tc-client is a command line client that is primarily used to configure the database TC.

Works even for file based transformation catalog.

Practical exercise (Refer Exercise 1.3.4):

- Insert an entry
- Query for a single entry
- Query for all the entries
Component Configuration using Properties File

Most of the configuration of Pegasus is done by properties

Properties can be specified
- On the command line
- In $HOME/.pegasusrc file

All properties are described in $PEGASUS_HOME/doc/advanced-properties.pdf

For the tutorial the properties are configured in the $HOME/.pegasusrc
Outline of Tutorial

- Introduction to scientific workflows
- Pegasus-WMS and Composition of a Simple Workflow In terms of DAX
- Pegasus Internals
- Mapping and Executing Simple Workflow Locally
- Monitoring, Debugging and Statistics
- Mapping and Executing Simple Workflow On the Grid
- Optimization techniques for mapping and executing Large Scale workflows
- Applications using Pegasus
- Current and Future Research
Map and Execute Workflow Locally

- Take a 4 node diamond abstract workflow (DAX) and map it to an executable workflow that runs locally
Basic Workflow Mapping

❖ Select where to run the computations
   ✦ Change 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 that register the newly-created data products
- Add nodes to create an execution directory on a remote site
- Write out the workflow in a form understandable by a workflow engine
  - Include provenance capture steps
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)
- 13 data stage-in nodes
- 8 inter-site data transfers
- 14 data stage-out nodes to long-term storage
- 14 data registration nodes (data cataloging)
Exercise: 1.2.6

- Plan using Pegasus and submit the workflow to Condor DAGMan/CondorG for local job submissions

- `$ pegasus-plan --dax <dax file> --dir <dags directory> -s local –o local --nocleanup`

- `$ pegasus-run –Dpegasus.user.properties=<properties file> --nodatabase <dag directory>`
Run (pegasus-run)
Exercise: 1.2.6 (cont.)

- Submits the workflow to Condor DAGMAN/CondorG for local job submissions
Outline of Tutorial

- Introduction to scientific workflows
- Pegasus-WMS and Composition of a Simple Workflow In terms of DAX
- Pegasus Internals
- Mapping and Executing Simple Workflow Locally
- Monitoring, Debugging and Statistics
- Mapping and Executing Simple Workflow On the Grid
- Optimization techniques for mapping and executing Large Scale workflows
- Applications using Pegasus
- Current and Future Research
Exercise: 1.3.1 - (Monitor) pegasus-status

- A perl wrapper around condor_q
- Allows you to see only the jobs of a particular workflow
- Also can see what different type of jobs that are executing

Usage
- pegasus-status <submit-directory>
- pegasus-status -l <submit-directory>
Exercise 1.3.2 – Workflow Debugging Tool

- 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, and any stdout and stderr from the job.

- Usage:
  - `pegasus-analyzer <dag directory>`
Exercise 1.3.3 – Condor DAGMan

- Treated as a unit
- Job or POST script determines node success or failure
DAG file

- Defines the DAG shown previously
- Node names *are* case-sensitive
- Keywords are not case-sensitive

MAXJOBS projection 2

JOB create_dir_blackdiamond_0_local create_dir_blackdiamond_0_local.sub
SCRIPT POST create_dir_blackdiamond_0_local /opt//bin/pegasus-exitcode
/home/pegasus/pegasus-wms/dags/pegasus/pegasus/blackdiamond/run0001/
create_dir_blackdiamond_0_local.out
RETRY create_dir_blackdiamond_0_local 2

JOB stage_in_local_local_0 stage_in_local_local_0.sub
SCRIPT POST stage_in_local_local_0 /opt/pegasus/default/bin/pegasus-exitcode
/home/pegasus/pegasus-wms/dags/pegasus/pegasus/blackdiamond/run0001/
stage_in_local_local_0.out
RETRY stage_in_local_local_0 2

PARENT create_dir_blackdiamond_0_local CHILD stage_in_local_local_0
PARENT create_dir_blackdiamond_0_local CHILD preprocess_j1
Exercise 1.3.4- pegasus-remove

- Remove your workflow and associated jobs
- In future, would cleanup the remote directories that are created during workflow execution.

Usage
- pegasus-remove <submit directory>
Exercise 1.3.5 - pegasus-statistics

- Generates useful statistics about a workflow run
  - Total workflow execution time
  - Total workflow execution wall time
  - Total jobs
  - Total tasks
  - Jobs succeeded
  - Jobs failed
  - Jobs unsubmitted
  - Jobs unknown

- Usage
  - pegasus-statistics <submit directory>
Exercise 1.3.5 - pegasus-plots

- Generates useful plots about a workflow run
  - Image of the DAX
  - Image of the DAG
  - Workflow Gantt Chart
  - Workflow Host Time Chart

- Usage
  - pegasus-plots <submit directory>
Outline of Tutorial

- Introduction to scientific workflows
- Pegasus-WMS and Composition of a Simple Workflow In terms of DAX
- Pegasus Internals
- Mapping and Executing Simple Workflow Locally
- Monitoring, Debugging and Statistics
- Mapping and Executing Simple Workflow On the Grid
- Optimization techniques for mapping and executing Large Scale workflows
- Applications using Pegasus
- Current and Future Research
Map and Execute Montage Workflow on Grid

- Take a montage abstract workflow (DAX) and map it to an executable workflow that runs on the Grid
- The available grid site in the VM is named cluster
Exercise: 1.4 - Planning and Executing Workflow against a Remote Resource

- Plan using Pegasus and submit the workflow to Condor DAGMan/CondorG for remote job submissions.

- pegasus-run starts the monitoring daemon (pegasus-monitor) in the directory containing the condor submit files.

- pegasus-monitor parses the condor output and updates the status of the workflow to a database.

- pegasus-monitor updates job status to a text file jobstate.log in the directory containing the condor submit files.
Outline of Tutorial

- Introduction to scientific workflows
- Pegasus-WMS and Composition of a Simple Workflow In terms of DAX.
- Pegasus Internals
- Mapping and Executing Simple Workflow Locally
- Mapping and Executing Simple Workflow On the Grid
- Optimization techniques for mapping and executing Large Scale workflows
- Applications using Pegasus
- Current and Future Research
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

Useful for small granularity jobs

Level-based clustering

Vertical clustering

Arbitrary clustering
Exercise 1.5.1
Optional clustering exercise

- To trigger specify --cluster horizontal option to pegasus-plan

- The granularity of clustering configured via pegasus profile key clusters.num
  - Can be specified with a transformation in the transformation catalog, or with sites in the site catalog
  - Pegasus profile clusters.num specified in the site catalog
  - This means how many clustered jobs for that transformation you need on a particular site
**Ex 1.5.2 - WF Reduction (Data Reuse)**

**How to:** Files need to be cataloged in replica catalog at runtime. The registration flags for these files need to be set in the DAX.
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

How to: remove the –nocleanup option to the pegasus-plan invocation in exercises 1.5.
Storage Improvement for Montage Workflows

Montage 1 degree workflow run with cleanup on OSG-PSU
Transfer of Executables

- Allows the user to dynamically deploy scientific code on remote sites
- Makes for easier debugging of scientific code
- The executables are transferred as part of the workflow
- Currently, only statically compiled executables can be transferred
- Also we transfer any dependant executables that maybe required. In your workflow, the mDiffFit job is dependant on mDiff and mFitplane executables
Staging of executable exercise

- All the workflows that you ran had staging of executables

- In your transformation catalog, the entries were marked as STAGEABLE on site “local”

- Selection of what executable to transfer
  - pegasus.transformation.mapper property
  - pegasus.transformation.selector property
Hierarchical Workflows

Recursive DAX

Increasing Level of Recursion

DAX A
- A1
- A2
- A3
- A4

DAX B
- B1
- B2
- B3
- B4

DAX C
- C1
- C2
- C3
- C4

DAX D
- D1
- D2
- D3
- D4

Recursion ends when DAX with only compute jobs is encountered.
Hierarchal Workflows

RECURSIVE DAX EXECUTION TIMELINE

TIME

DAGMAN execution of DAG corresponding to DAX A

P(A)  A1  A2  A3  A4

P(B)  B1  B2

P(C)  C1  C2  C3  C4

P(D)  D1  D2  D3  D4

Legend:
- Pegasus Planning
- Compute Job
- Pegasus Plan And Execute DAGMAN Job
Exercise 1.5.3 – Hierarchal Workflows

- Run the superdiamond workflow
  - pegasus-plan --dax `pwd`/dax/superdiamond.dax --force --submit --dir dags -s local -o local --nocleanup -v
Transfer Throttling

- Large Sized Workflows result in large number of transfer jobs being executed at once. Results in
  - Grid FTP server overload (connection refused errors etc)
  - May result in a high load on the head node if transfers are not configured for being executed as third party transfers

- Need to throttle transfers
  - Set pegasus.transfer.refiner property
  - Allows you to create chained transfer jobs or bundles of transfer jobs
  - Looks in your site catalog for pegasus profile "stagein.clusters"
Throttling in DAGMan

- Maxjobs (limits jobs in queue/running)
- Maxidle (limits idle jobs)
- Maxpre (limits PRE scripts)
- Maxpost (limits POST scripts)
- All limits are *per DAGMan*, not global for the pool

The above parameters can be configured in Pegasus Properties
Throttling by category

- CATEGORY *JobName CategoryName*
- MAXJOBS *CategoryName MaxJobsValue*
- Applies the maxjobs setting to only jobs assigned to the given category
- Global throttles still apply
- Useful with different types of jobs that cause different loads
Pegasus Job Throttling properties

- Specifying for the whole workflow
  - `dagman.maxidle` (limits jobs in queue/running)
  - `dagman.maxjobs` (limits idle jobs)
  - `dagman.maxpre` (limits PRE scripts)
  - `dagman.maxpost` (limits POST scripts)

- Specifying per category
  - `dagman.[category-name].maxjobs`
Outline of Tutorial

- Introduction to scientific workflows
- Pegasus-WMS and Composition of a Simple Workflow
  In terms of DAX.
- Pegasus Internals
- Mapping and Executing Simple Workflow Locally
- Monitoring, Debugging and Statistics
- Mapping and Executing Simple Workflow On the Grid
- Optimization techniques for mapping and executing
  Large Scale workflows
- Applications Using Pegasus
- Current and Future Research
Each Curve/workflow
• ~800,000 jobs
• 18.3 ± 3.9 hours on 400 processors (December 2008)
• 90 GB of output
• 7.5 million data files

CyberShake Application

Breaks up large simulation data into smaller chunks for parallel processing
Large Scale Workflows through Pegasus - SCEC Cybershake (2009)

- TACC’s Ranger – Teragrid
- 223 sites
  - Curve produced every 5.4 hrs
- 1207 wallclock hrs
  - 4,420 cores on average
  - 14,540 peak (23% of Ranger)
- 189 million tasks
  - 43 tasks/sec
  - 3.8 million Condor jobs
    - 289 failures
    - 3952 Ranger queue jobs
- 189 million files
  - 11 TB output, 165 TB temp
LIGO Scientific Collaboration

- Continuous gravitational waves are expected to be produced by a variety of celestial objects
- Only a small fraction of potential sources are known
- Need to perform blind searches, scanning the regions of the sky where we have no a priori information of the presence of a source
  - Wide area, wide frequency searches
- Search is performed for potential sources of continuous periodic waves near the Galactic Center and the galactic core
- Search for binary inspirals collapsing into black holes.
- The search is very compute and data intensive
Support for LIGO on Open Science Grid
LIGO Workflows:
185,000 nodes,
466,000 edges
10 TB of input data, 1 TB of output data.
Ocean Modeling (JPL)

- Modeling the temperature and ocean current

~3 Hours on 8 Procs
Current Workflow at USC Epigenome Center

Automate the process of data analysis from sequencer to visualization

Sequencing and Real time base calling

Alignment

Visualization
BRAIN ATLAS RSEQ Workflows

- Computes the expression levels of genes, exons and splice junctions given human RNA-seq datasets (Illumina/Solexa) as input.
- Consists of two major components: the read-mapping and the expression quantification.
  - Read-mapping, done using ELAND to map RNA-Seq reads to the human transcriptome sequences.
  - Expression quantification, RPKM values (number of Reads mapped Per Kilobase of gene per Million reads) computed for both genes and exons using non-repetitive regions of gene transcripts as well as whole transcript (repetitive and non repetitive)
  - Splice junctions, RPM values computed (number of Reads mapped Per Million reads).
RSEQ based Brain Atlas
BRAIN ATLAS RSEQ Workflows

- Total jobs run: 1360 (about 106 samples)
- Total data generated: 4.9TB
- Total data saved: 84GB
- Total CPU hrs used: 595 hrs

<table>
<thead>
<tr>
<th>#Transformation</th>
<th>Count</th>
<th>Mean</th>
<th>Variance</th>
<th>Min</th>
<th>Max</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>eland</td>
<td>44</td>
<td>39683.04</td>
<td>448849052.7</td>
<td>8529.71</td>
<td>96996.06</td>
<td>1746053.74</td>
</tr>
<tr>
<td>transfer</td>
<td>169</td>
<td>22.69</td>
<td>1779.39</td>
<td>0.17</td>
<td>268.23</td>
<td>3835.3</td>
</tr>
<tr>
<td>dirmanager</td>
<td>41</td>
<td>0.1</td>
<td>0</td>
<td>0.06</td>
<td>0.2</td>
<td>4.09</td>
</tr>
<tr>
<td>export2mrf</td>
<td>158</td>
<td>1516.65</td>
<td>717957.98</td>
<td>318.37</td>
<td>3303.56</td>
<td>239630.55</td>
</tr>
<tr>
<td>mrfquantifier</td>
<td>948</td>
<td>162.29</td>
<td>69440.88</td>
<td>0.29</td>
<td>1537.18</td>
<td>153854.83</td>
</tr>
</tbody>
</table>
USC RNA Seq Brain Atlas Workflow

- Computes the expression levels of genes, exons and splice junctions given human RNA-seq datasets (Illumina/Solexa) as input.
- Consists of two major components: the read-mapping and the expression quantification.
  - Read-mapping, done using PerM[1] to map RNA-Seq reads to the human transcriptome sequences.
  - Expression quantification, RPKM values (number of Reads mapped Per Kilobase of gene per Million reads) computed for both genes and exons using non-repetitive regions of gene transcripts.
  - Splice junctions, RPM values computed (number of Reads mapped Per Million reads).
USC RNA Seq Workflow

Single Lane Statistics

- Execution Time: 250mins (Single Processor)
- Input Files: 3
- Input File Size = 20GB
- Output Files: 127
- Output File Size = 33GB

Currently in Production Testing

- Plans to analyze about 200+ samples using this pipeline.
- Pipeline being distributed for wider use as KVM and Vbox VM.
Mass Matrix
THE CANCER GENOME ATLAS
Seqware

- Computed over 800 sequences using SeqWare framework
- Built on Pegasus WMS.
- Provided as a VM technology
NVO’s Montage mosaic application: Transformed a single-processor code into a workflow and parallelized computations to process larger-scale images

- Pegasus mapped workflow of 4,500 nodes onto NSF’s TeraGrid
- Pegasus improved runtime by 90% through automatic workflow restructuring and minimizing execution overhead
- Montage is a collaboration between IPAC, JPL and CACR
Epigenomic Workflows
Mapping the epigenetic state of human cells on a genome-wide scale

- split sequence files into multiple parts to be processed in parallel
- convert sequence files to the appropriate file format
- filter out noisy and contaminating sequences
- map sequences to their genomic locations
- merge output from individual mapping steps into a single global map
- use sequence maps to calculate the sequence density at each position in the genome

~7 hours on 8 procs, 6GB of data footprint

Ben Berman and others at USC
Outline of Tutorial

- Introduction to scientific workflows
- Pegasus-WMS and Composition of a Simple Workflow In terms of DAX.
- Pegasus Internals
- Mapping and Executing Simple Workflow Locally
- Monitoring, Debugging and Statistics
- Mapping and Executing Simple Workflow On the Grid
- Optimization techniques for mapping and executing Large Scale workflows
- Current and Future Research
Current and Future Research

- Resource selection
- Resource provisioning (Corral)
- Workflow restructuring
- Adaptive computing
  - Workflow refinement adapts to changing execution environment
- Workflow provenance (including provenance of the mapping process)
- Management and optimization across multiple workflows
- Workflow debugging
- Streaming data workflows
- Automated guidance for workflow restructuring
- Support for long-lived and recurrent workflows
Grid Computing

- **Grids**
  - Benefit: Provide plenty of computing resources
  - Challenge: Using those resources effectively

- **Grid Overheads**
  - Queuing Delays
  - Software Overheads
  - Scheduling Delays
  - Scheduling Policies
  - => Bad performance for throughput applications!

- **Some solutions**
  - Task clustering (workflows)
  - Advance reservations
Benefits of MLS and Glideins

- Running short jobs on the grid
  - Condor dispatches jobs faster than, e.g. Globus

- Bypass site scheduling policies
  - Use application-specific policies
  - e.g. prioritize jobs based on application needs

- Avoid competition for resources
  - Glideins reserve resources for multiple jobs
  - Minimizes queuing delays

- Better application scalability
  - Compared to GT2 GRAM, for example
  - Fewer jobmanagers => reduced load on gateway
Pilot Jobs

- Way for an application to use grid without the overheads
- Overlay a **personal cluster** on top of grid resources

- **Pilot jobs** install and run a **user-level resource manager**, which contacts an **application-specific scheduler** to be matched with application jobs
- **Glidein**: How to do this using Condor
Benefits of Glideins

• Running short jobs on the grid
  – Condor dispatches jobs faster than, e.g. Globus

• Bypass site scheduling policies
  – Use application-specific policies
  – e.g. prioritize jobs based on application needs

• Avoid competition for resources
  – Glideins reserve resources for multiple jobs
  – Minimizes queuing delays

• Better application scalability
  – Compared to GT2 GRAM, for example
  – Fewer jobmanagers => reduced load on gateway
GlideinWMS / Corral

- Resource provisioning system
  - Allocate resources explicitly rather than implicitly
  - Pay to allocate resources once and reuse them
  - Effectively minimizes grid overheads
  - Requires resource specification (# processors, time)

http://www.uscmsg.org/SoftwareComputing/Grid/WMS/glideinWMS/

http://pegasus.isi.edu/corralwms/
Montage with provisioning and without

Epigenomic Workflow

![Bar chart showing runtime (minutes) for different numbers of processors (1, 2, 4, 8, 16) for globus and corral.](chart)

![Bar chart showing number of processors (30, 128, 256, 512) vs. runtime.](chart)
Virtual Clusters

- A cluster of virtual machine instances
- Dynamic
  - Provision on-demand
  - Configure at runtime
- Mimic traditional clusters
  - Shared file systems, batch schedulers, etc.
- Setup is not trivial
  - Manual setup is error-prone and not scalable
  - Scripts work to a point, but break down for complex setups
Wrangler

- A service that automates the provisioning and configuration of virtual clusters

- **Features:** User-defined roles, dependencies, one-step provisioning

- **Interfaces:** RESTful service, Python API, command-line

- **Resource Providers:** Amazon EC2, Eucalyptus, OpenNebula

**Wrangler Coordinator:**
Negotiates with Resource Provider to provision Virtual Cluster Nodes

**Wrangler Agent:** Manages node, sends updates to coordinator, configures and launches services based on user-defined roles.
Virtual Cluster Configuration

- **Roles**: A set of scripts, data files and parameters that define the behavior of a node (transferred from client)
  - Examples: condor_worker, nfs_server, hadoop_peer, etc.

- **Dependencies**: Virtual cluster nodes may be dependent upon the services of other nodes (e.g. client-server)

- **Group**: A collection of nodes that are all dependent upon each other (e.g. parallel file system)
Virtual Clusters for Workflows

Condor with GlusterFS parallel file system

Condor with NFS file system

These nodes can all be deployed in one cloud, in different clouds, or both locally and in the cloud.
Stampede – Pegasus meets Netlogger

Pegasus Workflow Management System

NetLogger Log Pipeline
- Parser
- Loader
- DB

Execution Environment
- Task
- Execution logs

Log Analysis
Total number of tasks completed

Computational time by workflow partition
Detailed analysis of time spent in each stage of the hundreds of workflow tasks in each partition of the workflow.

Augmented “Gantt charts” can correlate errors with performance patterns.
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.
Relevant Links

- NSF Workshop on Challenges of Scientific Workflows: [www.isi.edu/nsf-workflows06](http://www.isi.edu/nsf-workflows06), E. Deelman and Y. Gil (chairs)

- TeraGrid: [www.teragrid.org](http://www.teragrid.org)
- Open Science Grid: [www.opensciencegrid.org](http://www.opensciencegrid.org)
- LIGO: [www.ligo.caltech.edu/](http://www.ligo.caltech.edu/)
- SCEC: [www.scec.org](http://www.scec.org)
- Montage: [montage.ipac.caltech.edu/](http://montage.ipac.caltech.edu/)
- Globus: [www.globus.org](http://www.globus.org)