Chapter 3. Running on different Grids

1. Introduction

Pegasus (pegasus.isi.edu) is a configurable system for mapping and executing abstract application workflows over the Grid. An abstract application workflow is represented as a directed acyclic graph where the vertices are the compute tasks and the edges represent the data dependencies between the tasks. The input to Pegasus is a description of the abstract workflow in XML format. The mapping of tasks to the execution resources is done based on information derived for static and/or dynamic sources (site catalog, transformation catalog, MDS (http://www.globus.org/toolkit/mds/) RLS (www.globus.org/toolkit/rls), etc). The output is an executable workflow (also called the concrete workflow) that can be executed over the Grid resources. In case the workflow tasks are mapped to multiple resources that do not share a file system, explicit nodes are added to the workflow for orchestrating data transfer between the tasks. The format of the concrete workflow is determined by the execution engine used for executing the concrete workflow over the mapped resources. For example, if Condor DAGMan (http://cs.wisc.edu/condor/dagman) is used as the enactment engine, then the concrete workflow consists of a DAG description file which lists the dependencies and Condor submit files for each task in the concrete workflow. Pegasus has been mostly used with Condor DAGMan. DAGMan can schedule jobs onto Globus (www.globus.org) and Condor (http://cs.wisc.edu/condor)-managed resources. Condor-G can be used to execute any task in the workflow on any Grid resource that provides a Globus GRAM interface. The purpose of this document is to give an overview of the various resource configurations that can be used with Pegasus when Condor DAGMan is used as the workflow enactment engine. The classification is done based on the mechanism used for submitting jobs to these resources.

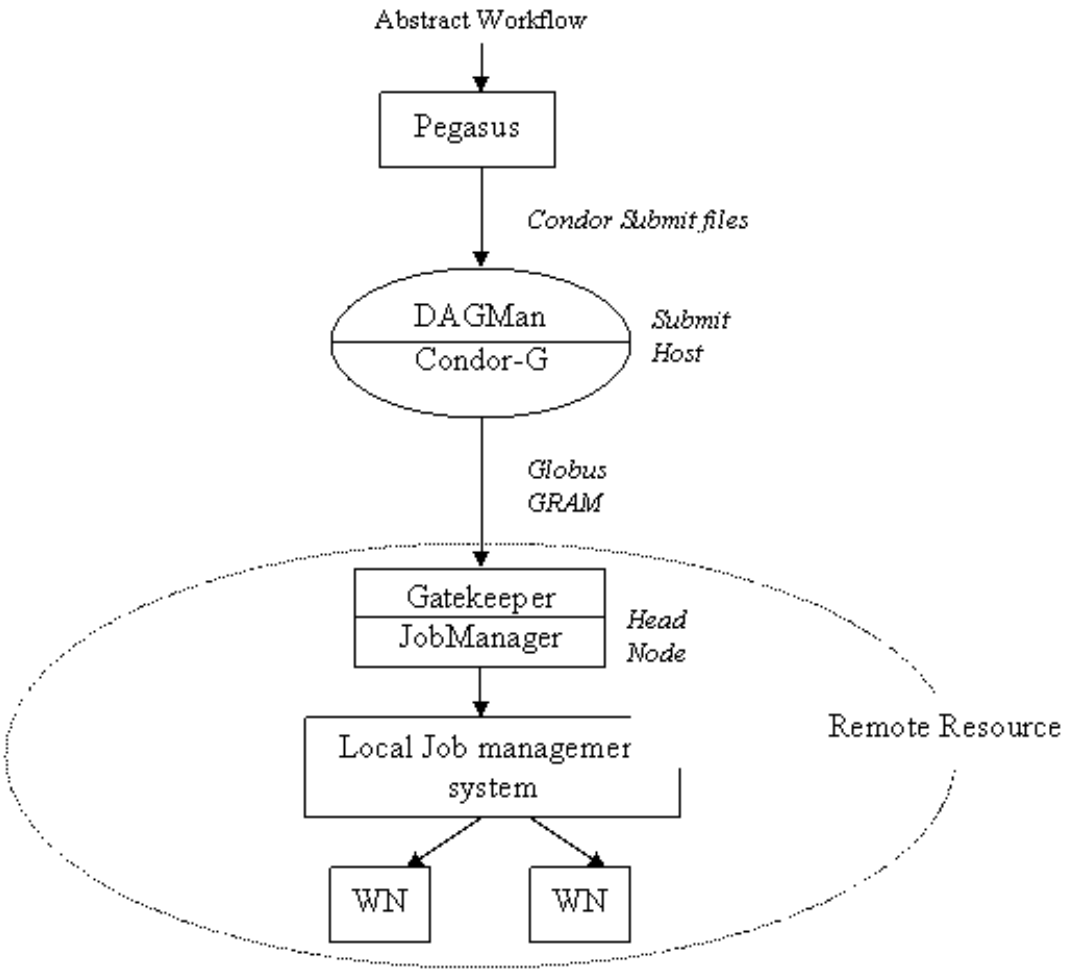
2. Resource Configurations

This section discusses the various resource configurations that can be used with Pegasus when Condor DAGMan is used as the workflow execution engine. It is assumed that there is a submit host where the workflow is submitted for execution. The following classification is done based the mechanism used for submitting jobs to the Grid resources and monitoring them. The classifications explored in this document are using Globus GRAM and using a Condor pool. Both of the configurations use Condor DAGMan to maintain the dependencies between the jobs, but differ in the manner as to how the jobs are launched. A combination of the above mentioned approach is also possible where some of the tasks in the workflow are executed in the Condor pool and the rest are executed on remote resources using Condor-G.

3. Using Globus GRAM

In this configuration, it is assumed that the target execution system consists of one or more Grid resources. These resources may be geographically distributed and under various administrative domains. Each resource might be a single desktop computer or a network of workstations (NOW) or a cluster of dedicated machines. However, each resource must provide a Globus GRAM interface which allows the users to submit jobs remotely. In case the Grid resource consists of multiple compute nodes, e.g. a cluster or a network of workstations, there is a central entity called the head node that acts as the single point of job submissions to the resource. It is generally possible to specify whether the submitted job should run on the head node of the resource or a worker node in the cluster or NOW. In the latter case, the head node is responsible for submitting the job to a local resource management system (PBS, LSF, Condor etc) which controls all the machines in the resource. Since, the head node is the central point of job submissions to the resource it should not be used for job execution since that can overload the head node delaying further job submissions. Pegasus does not make any assumptions about the configuration of the remote resource; rather it provides the mechanisms by which such distinctions can be made.

Figure 3.1. Resource Configuration using GRAM

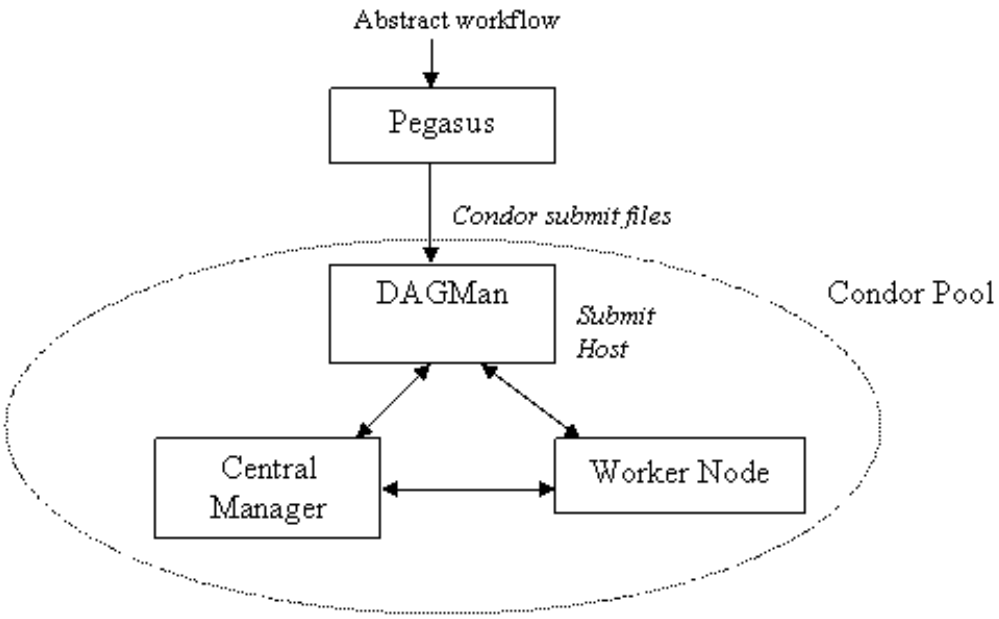


In this configuration, Condor-G is used for submitting jobs to these resources. Condor-G is an extension to Condor that allows the jobs to be described in a Condor submit file and when the job is submitted to Condor for execution, it uses the Globus GRAM interface to submit the job to the remote resource and monitor its execution. The distinction is made in the Condor submit files by specify the universe as Globus or grid and the grid_resource or globusscheduler attribute is used to indicate the location of the head node for the remote resource. Thus, Condor DAGMan is used for maintaining the dependencies between the jobs and Condor-G is used to launch the jobs on the remote resources using GRAM. The implicit assumption in this case is that all the worker nodes on a remote resource have access to a shared file system that can be used for data transfer between the tasks mapped on that resource. This data transfer is done using files.

4. Condor pool

A Condor pool is a set of machines that use Condor for resource management. A Condor pool can be a cluster of dedicated machines or a set of distributively owned machines. Pegasus can generate concrete workflows that can be executed on a Condor pool.

Figure 3.2. The Grid resources appear to be part of a Condor pool.



The workflow is submitted using DAGMan from one of the job submission machines in the Condor pool. It is the responsibility of the Central Manager of the pool to match the task in the workflow submitted by DAGMan to the execution machines in the pool. This matching process can be guided by including Condor specific attributes in the submit files of the tasks. If the user wants to execute the workflow on the execution machines (worker nodes) in a Condor pool, there should be a resource defined in the site catalog which represents these execution machines. The universe attribute of the resource should be vanilla. There can be multiple resources associated with a single Condor pool, where each resource identifies a subset of machine (worker nodes) in the pool. Pegasus currently uses the *FileSystemDomain* classad[] attribute to restrict the set of machines that make up a single resource. To clarify this point, suppose there are certain execution machines in the Condor pool whose *FileSystemDomain* is set to "ncsa.teragrid.org". If the user wants to execute the workflow on these machines, then there should be a resource, say "NCSA_TG" defined in the site catalog and the *FileSystemDomain* and universe attributes for this resource should be defined as "ncsa.teragrid.org" and "*vanilla*" respectively. When invoking Pegasus, the user should select NCSA_TG as the compute resource.

```
<sitecatalog xmlns="http://pegasus.isi.edu/schema/sitecatalog"
 xsi:schemaLocation="http://pegasus.isi.edu/schema/sitecatalog
 http://pegasus.isi.edu/schema/sc-2.0.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0">
 ……
  <site handle="ncsa_tera" gridlaunch="/opt/Pegasus/bin/kickstart" >
    <profile namespace="condor" key="universe">vanilla</profile>
    <profile namespace="condor" key="FileSystemDomain">ncsa.teragrid.org</profile>
    <gridftp url="gsiftp://ncsa.teragrid.org/" storage="/pegasus/data/"
       major="2" minor="4" patch="0" />
    <jobmanager universe="transfer" url="ncsa.teragrid.org/jobmanager-condor"
       major="2" minor="4" patch="0" />
    <jobmanager universe="vanilla" url="ncsa.teragrid.org/jobmanager-condor"
       major="2" minor="4" patch="0" />
    <workdirectory >/opt/workspace/CONDOR/exec</workdirectory>
  </site>
</sitecatalog>
```

Specifying FileSystemDomain and Universe in Site Catalog File

5. Running Jobs through Globus GRAM

This section describes how Pegasus runs jobs through Globus GRAM via Condor-G. Pegasus by default generates a concrete workflow in terms of condor submit files and DAG containing workflow dependencies, that when submitted to DAGMan ends up running on remote resources. In this configuration Pegasus generates jobs in the condor globus or grid universe. In addition the condor submit file, identifies what globusscheduler or

grid_resource (jobmanager contact or GRAM service end point) the job needs to be run on. The jobmanager contacts associated with a resource are identified in the site catalog. There are two types of jobmanager contacts associated with a site, vanilla and transfer. The vanilla job manager is identified by specifying the attribute universe as vanilla. Pegasus uses this jobmanager for executing compute jobs, when the site selector schedules compute jobs to that site. The transfer jobmanager is identified by specifying the attribute universe as transfer. Pegasus uses this jobmanager for executing transfer jobs on that particular site.

5.1. Pre-Webservices GRAM (GRAM2)

For GRAM2 the jobmanager url's need to be of the form <hostname[:port]/jobmanager[-schedulertype]>

E.g. smarty.isi.edu:2119/jobmanager-condor. Below is an example showing a site configured to run jobs on GRAM2

```
<sitecatalog xmlns="http://pegasus.isi.edu/schema/sitecatalog"
 xsi:schemaLocation="http://pegasus.isi.edu/schema/sitecatalog
 http://pegasus.isi.edu/schema/sc-2.0.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0">
 ……
  <site handle="isi_condor" gridlaunch="/opt/pegasus/bin/kickstart" >
    <profile namespace="env" key="GLOBUS_LOCATION">
        /nfs/v6/globus/GT2/linux/STABLE</profile>
    <profile namespace="env" key="LD_LIBRARY_PATH">
        /nfs/v6/globus/GT2/linux/STABLE/lib</profile>
    <gridftp url="gsiftp://smarty.isi.edu/" storage="/peasus/data/"
        major="2" minor="4" patch="0" />
    <jobmanager universe="transfer" url="columbus.isi.edu/jobmanager-fork"
        major="2" minor="4" patch="0" />
    <jobmanager universe="vanilla" url="columbus.isi.edu/jobmanager-condor"
        major="2" minor="4" patch="0" />
    <workdirectory >/nfs/cgt-scratch/vahi/CONDOR/exec</workdirectory>
  </site>
 …
</sitecatalog>
```

Specifying GRAM2 Job Managers in Site Catalog

5.2. Webservices GRAM (GRAM4)

For running jobs on GRAM4 the url to the jobmanager needs to be a End Point Reference (EPR) to the ManagedJobFactoryService. Additionally you need to provide two profiles under namespace=condor. The keys are grid_type=gt4 and jobmanager_type=<SCHEDULER> where scheduler is on of the GRAM4 supported schedulers. Below is an example of a site which is configured for running jobs on GRAM4 on the Condor scheduler. Currently it is not possible to define different types of SCHEDULER's for the different universe in GRAM4 configuration but will be available in the next site catalog.

```
<sitecatalog xmlns="http://pegasus.isi.edu/schema/sitecatalog"
 xsi:schemaLocation="http://pegasus.isi.edu/schema/sitecatalog
  http://pegasus.isi.edu/schema/sc-2.0.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0">
 ……
  <site handle="isi_skynet_gt4" gridlaunch="/opt/pegasus/bin/kickstart"
    sysinfo="INTEL32::LINUX">
    <profile namespace="env" key="GLOBUS_LOCATION">
      /nfs/software/globus/default</profile>
    <profile namespace="env" key="LD_LIBRARY_PATH">
      /nfs/software/globus/default/lib</profile>
    <profile namespace="condor" key="grid_type" >gt4</profile>
    <profile namespace="condor" key="jobmanager_type">Condor</profile>
    <lrc url="rlsn://smarty.isi.edu" />
    <gridftp  url="gsiftp://skynet-login.isi.edu" storage="/nfs/scratch01/vahi/DATA"
      major="2" minor="4" patch="0" />
    <jobmanager universe="transfer"
       url="https://skynet.isi.edu:8443/wsrf/services/ManagedJobFactoryService"
       major="4" minor="0" patch="0" />
    <jobmanager universe="vanilla"
       url="https://skynet.isi.edu:8443/wsrf/services/ManagedJobFactoryService"
       major="4" minor="0" patch="0" />
    <workdirectory>/nfs/scratch01/vahi/EXEC/WSGRAM</workdirectory>
  </site>
```

Specifying GRAM4 Job Managers in Site Catalog

6. Running Jobs in Condor Pool

This section describes the various changes required in the site catalog for running the workflow jobs in a Condor pool#. Pegasus by default uses Condor-G for job submissions i.e. It generates jobs in the condor Globus or grid universe. For running jobs directly in a Condor pool, the jobs need to be generated for the vanilla universe. This can be done using the **condor profile namespace** in Pegasus, and associating it with the job or with the execution site (compute resource). The recommended way is to tag your execution site as a vanilla pool, which would result in all the jobs scheduled on that site being run in the vanilla universe. Alternatively, the user could tag a particular job either in the transformation catalog or in the DAX (abstract representation of the workflow) as to be run in the vanilla universe.

In all the three cases, the user needs to insert a key **universe** with the value **vanilla** in **condor profile namespace**.

To tag the site as a vanilla universe pool, the user needs to update his Site catalog. Usually, this is the site catalog file in XML form.

```
<sitecatalog xmlns="http://pegasus.isi.edu/schema/sitecatalog"
 xsi:schemaLocation="http://pegasus.isi.edu/schema/sitecatalog
 http://pegasus.isi.edu/schema/sc-2.0.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0">
 ……
  <site handle="ncsa_tera" gridlaunch="/opt/pegasus/bin/kickstart">
    <profile namespace="condor" key="universe">vanilla</profile>
    <gridftp url="gsiftp://ncsa.teragrid.org/" storage="/pegasus/data/"
      major="2" minor="4" patch="0" />
    <jobmanager universe="transfer" url="ncsa.teragrid.org/jobmanager-condor"
      major="2" minor="4" patch="0" />
    <jobmanager universe="vanilla" url="ncsa.teragrid.org/jobmanager-condor"
      major="2" minor="4" patch="0" />
    <workdirectory >/opt/workspace/CONDOR/exec</workdirectory>
  </site>
 ……
</sitecatalog>
```

Site Catalog for use with a Condor Pool

In the above example, the site isi_condor has been tagged as a vanilla universe pool. Alternatively, the user can tag a particular job in the DAX as a vanilla universe job. This is not recommended but the facility exists. **Note even though the jobmanagers are specified for this site (due to XML schema constraints), they are not used in this mode.**

```
<adag xmlns="http://pegasus.isi.edu/schema/DAX"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://pegasus.isi.edu/schema/DAX
  http://pegasus.isi.edu/schema/dax-2.0.xsd"
  version="2.0" count="1" index="0" name="black-diamond">
 …….
   <job id="ID000004" namespace="pegasus" name="analyze" version="1.0" level="1"
      dv-namespace="pegasus" dv-name="bottom" dv-version="1.0">
      <argument>-a bottom -T 6 -i <filename file="f.c1"/>
         <filename file="f.c2"/> -o <filename file="f.d"/>
      </argument>
      <profile namespace="condor" key="universe">vanilla</profile>
      <uses file="f.c1" link="input" dontRegister="false" dontTransfer="false"/>
      <uses file="f.c2" link="input" dontRegister="false" dontTransfer="false"/>
      <uses file="f.d" link="output" dontRegister="false" dontTransfer="false"/>
   </job>
 …….
</adag>
```

Modified DAX job with profile universe=vanilla under condor namespace.

In the above example, the job with logical name analyze has been tagged with the condor profile, which would result in it being run in the condor universe vanilla on whatever execution site it is mapped to.

6.1. Changes to the Compute Jobs

The tagging of job as vanilla ends up triggering a couple of changes in the submit files for the compute jobs that are done automatically by Pegasus. This section explains what those changes are, and why they are done.

Removal of the following “Globus” universe key value pairs from the condor submit files.

- globusscheduler
- remote_intialdir
- globusrsl

Addition of –w option to kickstart for the compute jobs Since, Pegasus is not specifying the remote directory in the submit file, condor runs each job in a unique spool directory on the remote execution pool. However, we need the jobs need to be run in the work directory specified by the user in his configuration files, as that is where all the data required by the job is staged in. Hence, the –w option is passed to kickstart that makes it change into that directory before executing the user executable as specified by the compute job. Addition of the following condor key value pairs to the submit files

- should_transfer_files=YES
- when_to_transfer_output=ON_EXIT

6.2. Changes to the transfer jobs

The transfer jobs are different from the compute jobs in the sense that they rely on the underlying grid mechanisms (globus-url-copy mainly) to transfer the files in and out of the execution pools. The running of the transfer jobs on the remote pools requires the use of the user proxy that in case of the “globus” universe jobs is transported to the remote end by CondorG from the submit host.

In case of condor “vanilla” universe jobs, the transfer of proxy is handled by using the Condor file transfer mechanisms. This is done by the user either specifying the path to the user proxy in their site catalog for the local site (submit host) or using the property `pegasus.local.env` in the properties file.

The user specifies the `X509_USER_PROXY` environment variable in the env profile namespace.

```
<sitecatalog xmlns="http://pegasus.isi.edu/schema/sitecatalog"
 xsi:schemaLocation="http://pegasus.isi.edu/schema/sitecatalog
 http://pegasus.isi.edu/schema/sc-2.0.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0">
 ……
 <site handle="local" gridlaunch="/nfs/asd2/vahi/test/pegasus/bin/kickstart">
   <lrc url="rls://localhost" />
   <gridftp url="gsiftp://smarty.isi.edu/cgt-scratch/gmehta/LOCAL"
     storage="/cgt-scratch/gmehta/LOCAL" major="2" minor="4" patch="0" />
   <jobmanager universe="transfer" url="localhost/jobmanager-fork"
     major="2" minor="4" patch="0" />
   <jobmanager universe="vanilla" url="localhost/jobmanager-condor"
     major="2" minor="4" patch="0" />
   <workdirectory >/nfs/cgt-scratch/gmehta/LOCAL</workdirectory>
   <profile namespace="env" key="GLOBUS_LOCATION">
     /nfs/v6/globus/GT2/linux/STABLE</profile>
   <profile namespace="env" key="LD_LIBRARY_PATH">
     /nfs/v6/globus/GT2/linux/STABLE/lib</profile>
   <profile namespace="env" key="X509_USER_PROXY">
     /tmp/pegasus/x509.proxy</profile>
 </site>
 ……
</sitecatalog>
```

Modifications to the Site Catalog for site local

The user can optionally edit the property file and specify the value of the environment variable.

```
#Pegasus PROPERTY FILE
pegasus.local.env X509_USER_PROXY=/tmp/pegasus/x509.proxy
```

Modifications to the Properties file

The tagging of job as vanilla ends up triggering the following changes in the submit files for the transfer jobs, that are done automatically by Pegasus. Removal of the following “globus” universe key value pairs from the condor submit files.

- globusscheduler
- remote_intialdir
- globusrsl

Addition of condor key value pair `transfer_input_files= <path to user proxy on submit node>`

The path to the user proxy is picked up from the site catalog or the properties files with the value from the properties file overriding the one in the site catalog. This tells Condor to transfer the proxy to the remote spool directory where it is going to run the job.

However, the job needs to know that the proxy is there. This is done by specifying the environment variable X509_USER_PROXY in the environment key in the submit file.

For e.g the submit file would contain `environment=X509_USER_PROXY=x509.proxy`

There is no –w option to kickstart generated for transfer jobs as we want the jobs to be executed in the spool directory where condor launches them. This is because there is no means to determine in advance the spool directory that contains the transferred proxy. The input for the transfer jobs (transfer.in file) is staged via stdin and not via data files as is the case for compute jobs.

Addition of the following condor key value pairs to the submit files

- should_transfer_files=YES
- when_to_transfer_output=ON_EXIT

7. Condor GlideIn

As mentioned Pegasus can execute workflows over Condor pool. This pool can contain machines managed by a single institution or department and belonging to a single administrative domain. This is the case for most of the Condor pools. In this section we describe how machines from different administrative domains and supercomputing centers can be dynamically added to a Condor pool for certain timeframe. These machines join the Condor pool temporarily and can be used to execute jobs in a non preemptive manner. This functionality is achieved using a Condor feature called Glide-in http://cs.wisc.edu/condor/glidein that uses Globus GRAM interface for migrating machines from different domains to a Condor pool. The number of machines and the duration for which they are required can be specified.

In this case, we use the abstraction of a local Condor pool to execute the jobs in the workflow over remote resources that have joined the pool for certain timeframe. Details about the use of this feature can be found in the condor manual (http://cs.wisc.edu/condor/manual/).

A basic step to migrate in a job to a local condor pool is described below.

```
condor_glidein -count 10 gatekeeper.site.edu/jobmanager-pbs
```

GlideIn of Remote Globus Resources

The above step glides in 10 nodes to the user’s local condor pool, from the remote pbs scheduler running at gatekeeper.site.edu. By default, the glide in binaries are installed in the users home directory on the remote host.

It is possible that the Condor pool can contain resources from multiple Grid sites. It is normally the case that the resources from a particular site share the same file system and thus use the same FileSystemDomain attribute

while advertising their presence to the Central Manager of the pool. If the user wants to run his jobs on machines from a particular Grid site, he has to specify the FileSystemDomain attribute in the requirements classad expression in the submit files with a value matching the FileSystemDomain of the machines from that site. For example, the user migrates nodes from the NCSA Teragrid cluster (with FileSystemDomain ncsa.teragrid.org) into a Condor pool and specifies FileSystemDomain == “ncsa.teragrid.org”. Condor would then schedule the jobs only on the nodes from the NCSA Teragrid cluster in the local condor pool. The FileSystemDomain can be specifed for an execution site in the site catalog with condor profile namespace as follows

```
<sitecatalog xmlns="http://pegasus.isi.edu/schema/sitecatalog"
 xsi:schemaLocation="http://pegasus.isi.edu/schema/sitecatalog
 http://pegasus.isi.edu/schema/sc-2.0.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0">
 ……
  <site handle="ncsa_tera" gridlaunch="/opt/Pegasus/bin/kickstart">
    <profile namespace="condor" key="universe">vanilla</profile>
    <profile namespace="condor" key="FileSystemDomain">ncsa.teragrid.org</profile>
    <gridftp url="gsiftp://ncsa.teragrid.org/" storage="/pegasus/data/"
       major="2" minor="4" patch="0" />
    <jobmanager universe="transfer" url="ncsa.teragrid.org/jobmanager-condor"
       major="2" minor="4" patch="0" />
    <jobmanager universe="vanilla" url="ncsa.teragrid.org/jobmanager-condor"
       major="2" minor="4" patch="0" />
    <workdirectory >/opt/workspace/CONDOR/exec</workdirectory>
  </site> …….
</sitecatalog>
```

Specifying FileSystemDomain in Site Catalog

Specifying the FileSystemDomain key in condor namespace for a site, triggers Pegasus into generating the requirements classad expression in the submit file for all the jobs scheduled on that particular site. For example, in the above case all jobs scheduled on site isi_condor would have the following expression in the submit file.

```
requirements = FileSystemDomain == “ncsa.teragrid.org”
```