Chapter 1: Pegasus Job Clustering

1 Motivation

A large number of workflows executed through the Pegasus Workflow Management System, are composed of several jobs that run for only a few seconds or so. The overhead of running any job on the grid is usually 60 seconds or more. Hence, it makes sense to cluster small independent jobs into a larger job. This is done while mapping an abstract workflow to a concrete workflow. Site specific or transformation specific criteria are taken into consideration while clustering smaller jobs into a larger job in the concrete workflow. The user is allowed to control the granularity of this clustering on a per transformation per site basis.

2 Overview

The abstract workflow is mapped onto the various sites by the Site Selector. This semi executable workflow is then passed to the clustering module. The clustering of the workflow can be either be

- level based (horizontal clustering)
- label based (label clustering)

The clustering module clusters the jobs into larger/clustered jobs, that can then be executed on the remote sites. The execution can either be sequential on a single node or on multiple nodes using MPI. To specify which clustering technique to use the user has to pass the **--cluster** option to **pegasus-plan** .

3 Generating Clustered Concrete DAG

The clustering of a workflow is activated by passing the **--cluster**|**-C** option to **pegasus-plan**. The clustering granularity of a particular logical transformation on a particular site is dependant upon the clustering techniques being used. The executable that is used for running the clustered job on a particular site is determined as explained in section 7.

```
#Running pegasus-plan to generate clustered workflows

$ pegasus-plan —-dax example.dax --dir ./dags —p siteX —-output local
               --cluster [ comma separated list of clustering techniques]  —verbose↵

Valid clustering techniques are horizontal and label.
```

The naming convention of submit files of the clustered jobs is **merge_NAME_IDX.sub** . The NAME is derived from the logical transformation name. The IDX is an integer number between 1 and the total number of jobs in a cluster. Each of the submit files has a corresponding input file, following the naming convention **merge_NAME_IDX.in** . The input file contains the respective execution targets and the arguments for each of the jobs that make up the clustered job.

4 Horizontal Clustering

In case of horizontal clustering, each job in the workflow is associated with a level. The levels of the workflow are determined by doing a modified Breadth First Traversal of the workflow starting from the root nodes. The level associated with a node, is the furthest distance of it from the root node instead of it being the shortest distance as in normal BFS. For each level the jobs are grouped by the site on which they have been scheduled by the Site Selector. Only jobs of same type (txnamespace, txname, txversion) can be clustered into a larger job. To use horizontal clustering the user needs to set the **--cluster** option of **pegasus-plan to horizontal** .

4.1 Controlling Clustering Granularity

The number of jobs that have to be clustered into a single large job, is determined by the value of two parameters associated with the smaller jobs. Both these parameters are specified by the use of a PEGASUS

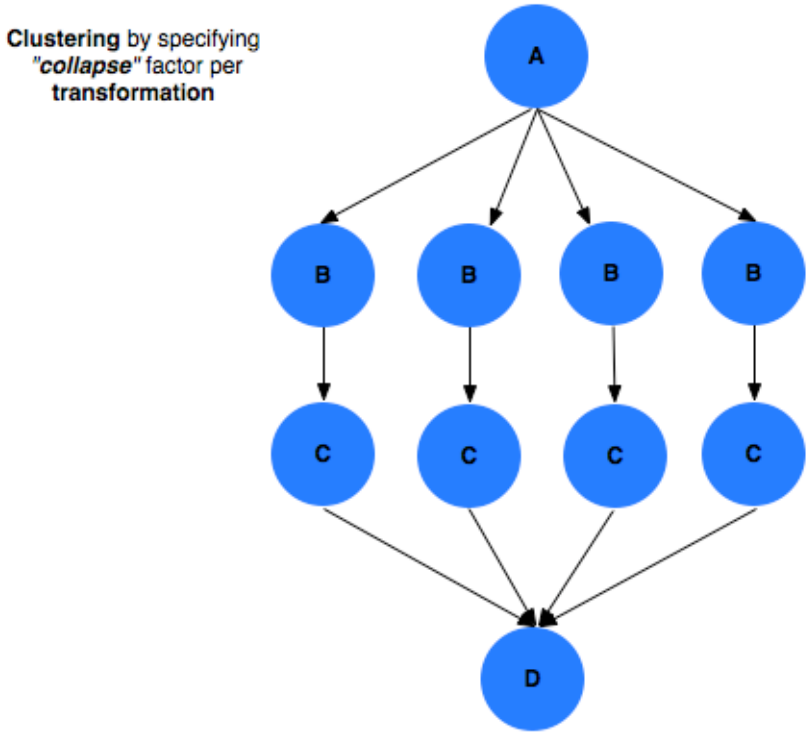
namespace profile keys. The keys can be specified at any of the placeholders for the profiles (abstract transformation in the DAX, site in the site catalog, transformation in the transformation catalog). The normal overloading semantics apply i.e. profile in transformation catalog overrides the one in the site catalog and that in turn overrides the one in the DAX. The two parameters are described below.

- **collapse factor**

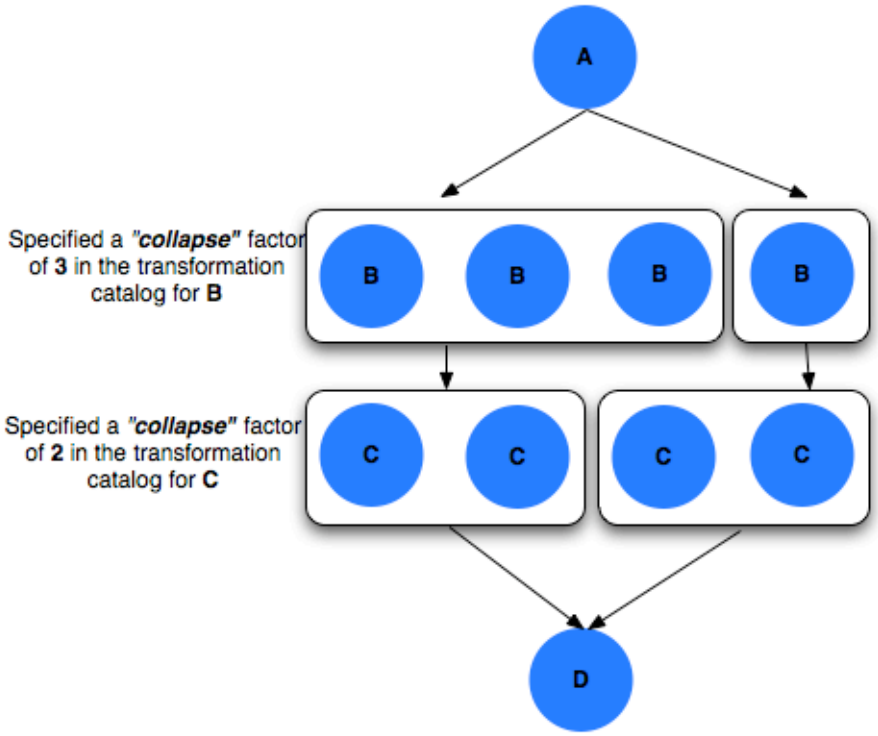
  The collapse factor denotes how many jobs need to be merged into a single clustered job. It is specified via the use of a PEGASUS namespace profile key “collapse”. for e.g. if at a particular level, say 4 jobs referring to logical transformation B have been scheduled to a siteX. The collapse factor associated with job B for siteX is say 3. This will result in 2 clustered jobs, one composed of 3 jobs and another of 2 jobs. The collapse factor can be specified in the transformation catalog as follows

```
#site   transformation   pfn              type              architecture  profiles↵

siteX    B     /shared/PEGASUS/bin/jobB INSTALLED       INTEL32::LINUX  PEGASUS::coll
siteX    C     /shared/PEGASUS/bin/jobC INSTALLED       INTEL32::LINUX  PEGASUS::coll
```



1. Original Workflow



2. Workflow After Clustering

- **bundle factor**

  The bundle factor denotes how many clustered jobs does the user want to see per level per site. It is

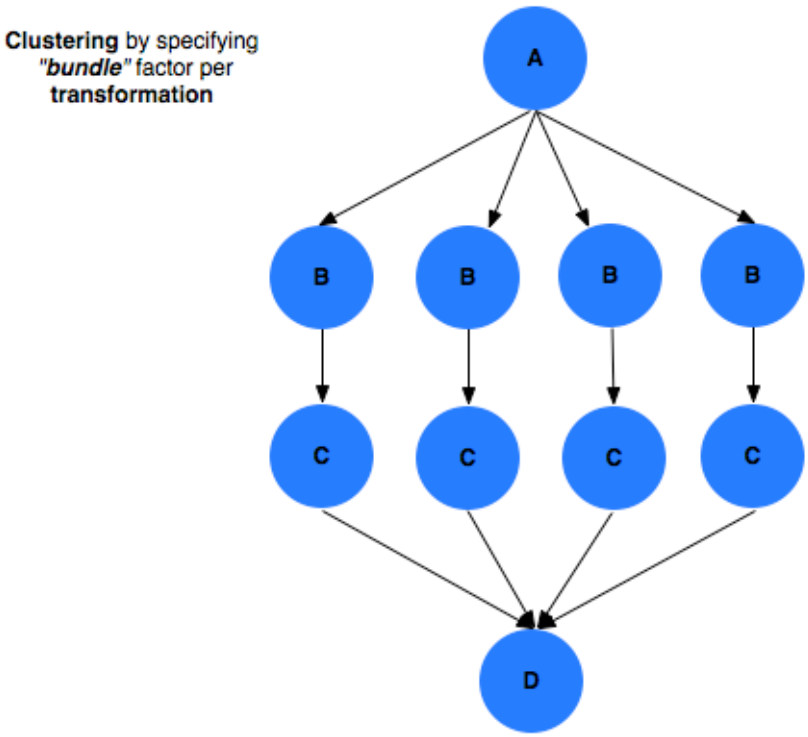
specified via the use of a PEGASUS namespace profile key “bundle”. for e.g. if at a particular level, say 4 jobs referring to logical transformation B have been scheduled to a siteX. The “bundle” factor associated with job B for siteX is say 3. This will result in 3 clustered jobs, one composed of 2 jobs and others of a single job each. The bundle factor in the transformation catalog can be specified as follows

```
#site  transformation      pfn            type            architecture    profiles
siteX    B     /shared/PEGASUS/bin/jobB INSTALLED       INTEL32::LINUX  PEGASUS::bund
siteX    C     /shared/PEGASUS/bin/jobC INSTALLED       INTEL32::LINUX  PEGASUS::bund
```

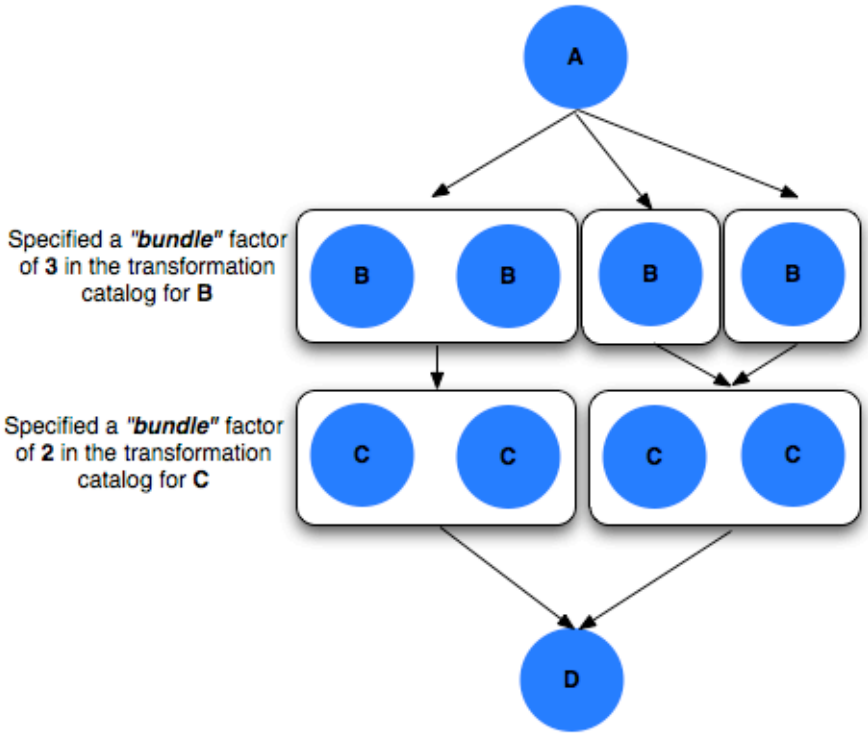
In the case, where both the factors are associated with the job, the bundle value supersedes the collapse value.

```
#site  transformation   pfn              type            architecture   profiles
siteX    B     /shared/PEGASUS/bin/jobB INSTALLED       INTEL32::LINUX PEGASUS::colla
```

In the above case the jobs referring to logical transformation B scheduled on siteX will be clustered on the basis of “bundle” value. Hence, if there are 4 jobs referring to logical transformation B scheduled to siteX, then 3 clustered jobs will be created.



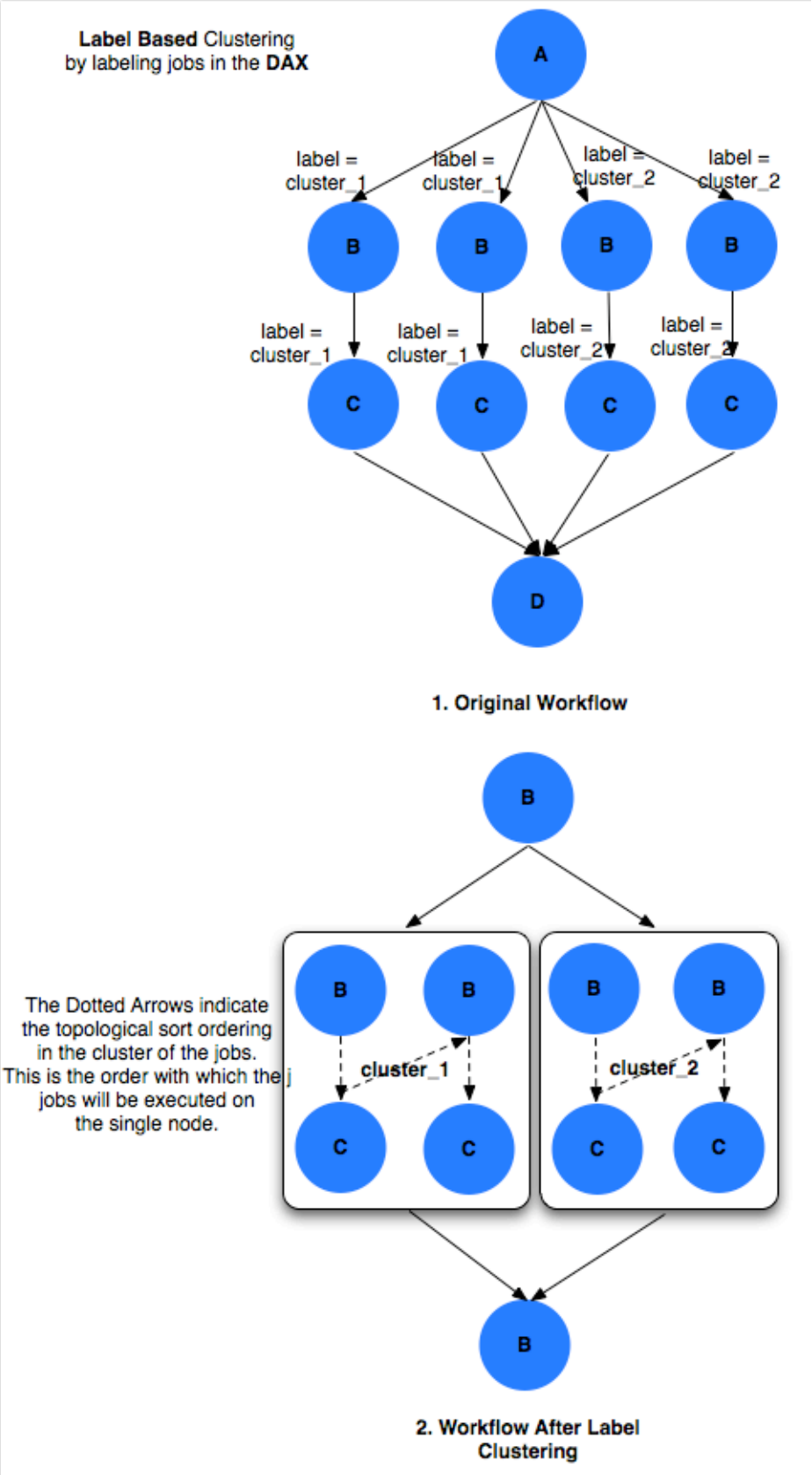
1. Original Workflow



2. Workflow After Clustering

5 Label Clustering

In label based clustering, the user labels the workflow. All jobs having the same label value are clustered into a single clustered job. This allows the user to create clusters or use a clustering technique that is specific to his workflows. If there is no label associated with the job, the job is not clustered and is executed as is



Since, the jobs in a cluster in this case are not independent, it is important the jobs are executed in the correct order. This is done by doing a topological sort on the jobs in each cluster. To use label based

clustering the user needs to set the **--cluster** option of **pegasus-plan** to label.

5.1 Labelling the Workflow

The labels for the jobs in the workflow are specified by associated **pegasus** profile keys with the jobs during the DAX generation process. The user can choose which profile key to use for labeling the workflow. By default, it is assumed that the user is using the PEGASUS profile key label to associate the labels. To use another key, in the **pegasus** namespace the user needs to set the following property

- pegasus.clusterer.label.key

For example if the user sets **pegasus.clusterer.label.key** to **user_label** then the job description in the DAX looks as follows

```
<adag >
…
  <job id="ID000004" namespace="app" name="analyze" version="1.0" level="1" >
    <argument>-a bottom -T60  -i <filename file="user.f.c1"/>  -o <filename file="user.f.
    <profile namespace="pegasus" key="user_label">p1</profile>
    <uses file="user.f.c1" link="input" dontRegister="false" dontTransfer="false"/>
    <uses file="user.f.c2" link="input" dontRegister="false" dontTransfer="false"/>
    <uses file="user.f.d" link="output" dontRegister="false" dontTransfer="false"/>
  </job>
…
</adag>
```

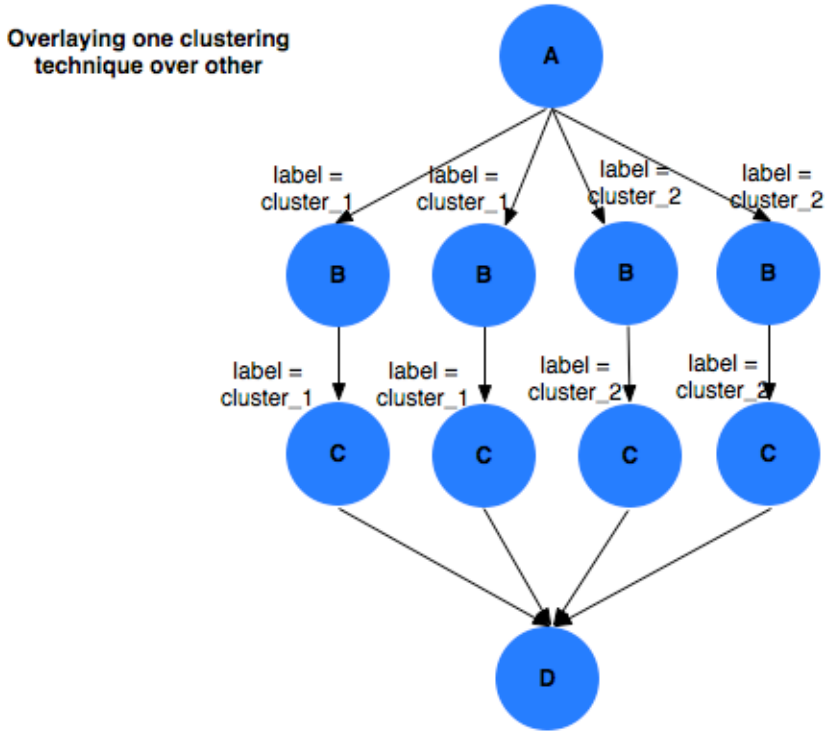
- The above states that the **pegasus** profiles with key as **user_label** are to be used for designating clusters.
- Each job with the same value for **pegasus** profile key **user_label** appears in the same cluster.

6 Recursive Clustering

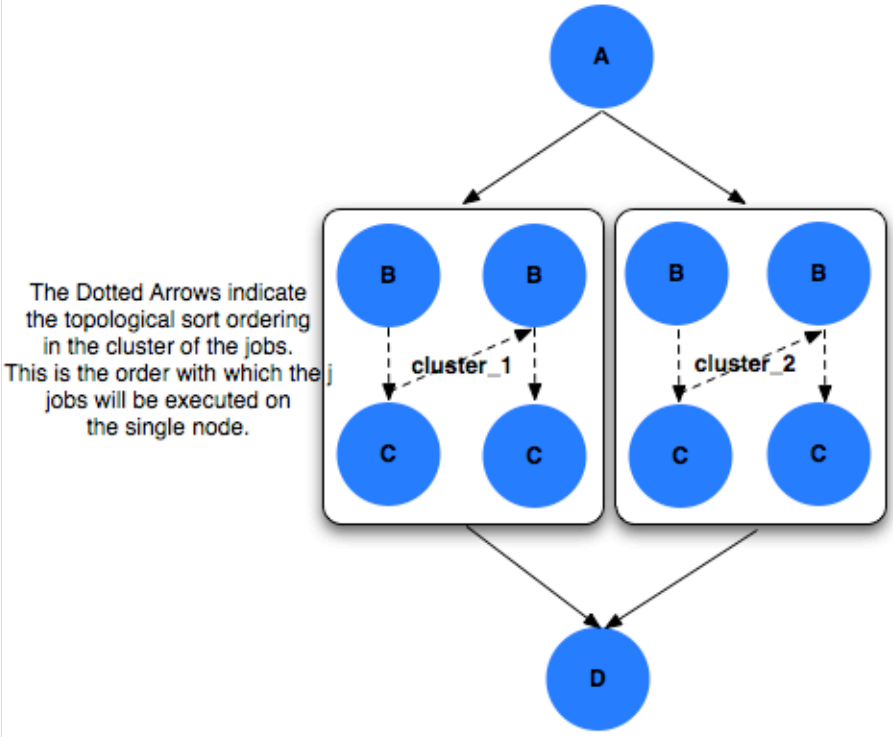
In some cases, a user may want to use a combination of clustering techniques. For e.g. a user may want some jobs in the workflow to be horizontally clustered and some to be label clustered. This can be achieved by specifying a comma separated list of clustering techniques to the **--cluster** option of **pegasus-plan**. In this case the clustering techniques are applied one after the other on the workflow in the order specified on the command line.

For example

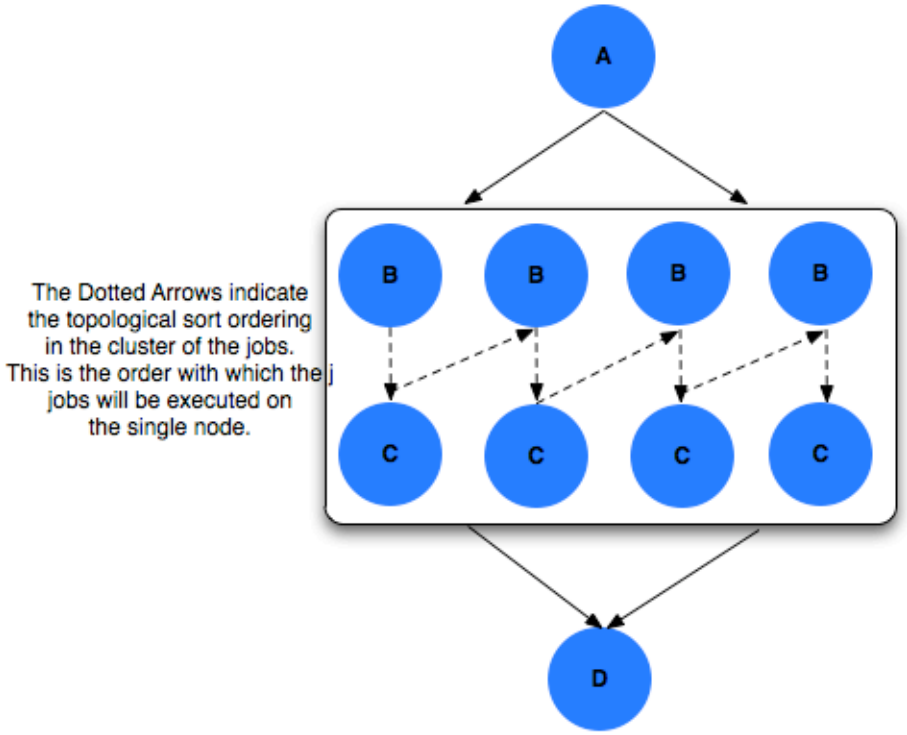
```
$ pegasus-plan --dax example.dax --dir ./dags --cluster label,horizontal -s siteX --outpu
```



1. Original Workflow



2. Workflow After Label Clustering



3. Workflow After Horizontal Clustering applied to Label based Clustered Workflow

7 Execution of the Clustered Job

The execution of the clustered job on the remote site, involves the execution of the smaller constituent jobs either

- **sequentially on a single node of the remote site**

  The clustered job is executed using **seqexec**, a wrapper tool written in C that is distributed as part of the PEGASUS. It takes in the jobs passed to it, and ends up executing them sequentially on a single node. To use "**seqexec**" for executing any clustered job on a siteX, there needs to be an entry in the transformation catalog for an executable with the logical name seqexec and namespace as pegasus.

```
#site  transformation   pfn              type                architecture    profiles

siteX    pegasus::seqexec     /shared/PEGASUS/bin/seqexec INSTALLED       INTEL32::LI
```

  By default, the entry for seqexec on a site is automatically picked up if $PEGASUS_HOME or $VDS_HOME is specified in the site catalog for that site.

- **On multiple nodes of the remote site using MPI**

  The clustered job is executed using mpiexec, a wrapper mpi program written in C that is distributed as part of the PEGASUS. It is only distributed as source not as binary. The wrapper ends up being run on every mpi node, with the first one being the master and the rest of the ones as workers. The number of instances of mpiexec that are invoked is equal to the value of the globus rsl key nodecount. The master distributes the smaller constituent jobs to the workers.

  For e.g. If there were 10 jobs in the merged job and nodecount was 5, then one node acts as master, and the 10 jobs are distributed amongst the 4 slaves on demand. The master hands off a job to the

slave node as and when it gets free. So initially all the 4 nodes are given a single job each, and then as and when they get done are handed more jobs till all the 10 jobs have been executed.

To use “mpiexec” for executing the clustered job on a siteX, there needs to be an entry in the transformation catalog for an executable with the logical name mpiexec and namespace as pegasus.

```
#site  transformation   pfn            type                architecture    profiles

siteX    pegasus::seqexec     /shared/PEGASUS/bin/mpiexec INSTALLED       INTEL32::LI
```

Another added advantage of using mpiexec, is that regular non mpi code can be run via MPI.

Both the clustered job and the smaller constituent jobs are invoked via kickstart, unless the clustered job is being run via mpi (mpiexec). Kickstart is unable to launch mpi jobs. If kickstart is not installed on a particular site i.e. the gridlaunch attribute for site is not specified in the site catalog, the jobs are invoked directly.

7.1 Specification of Method of Execution for Clustered Jobs

The method execution of the clustered job(whether to launch via mpiexec or seqexec) can be specified

1. **globally in the properties file**

   The user can set a property in the properties file that results in all the clustered jobs of the workflow being executed by the same type of executable.

   ```
   #PEGASUS PROPERTIES FILE
   pegasus.clusterer.job.aggregator seqexec|mpiexec
   ```

   In the above example, all the clustered jobs on the remote sites are going to be launched via the property value, as long as the property value is not overridden in the site catalog.

2. **associating profile key “collapser” with the site in the site catalog**

   ```
   <site handle="siteX" gridlaunch = "/shared/PEGASUS/bin/kickstart">
       <profile namespace="env" key="GLOBUS_LOCATION" >/home/shared/globus</profile>
       <profile namespace="env" key="LD_LIBRARY_PATH">/home/shared/globus/lib</profil
       <profile namespace="pegasus" key="collapser" >seqexec</profile>
       <lrc url="rls://siteX.edu" />
       <gridftp  url="gsiftp://siteX.edu/" storage="/home/shared/work" major="2" mino
       <jobmanager universe="transfer" url="siteX.edu/jobmanager-fork" major="2" mino
       <jobmanager universe="vanilla" url="siteX.edu/jobmanager-condor" major="2" min
       <workdirectory >/home/shared/storage</workdirectory>
     </site>
   ```

   In the above example, all the clustered jobs on a siteX are going to be executed via seqexec, as long as the value is not overridden in the transformation catalog.

3. **associating profile key “collapser” with the transformation that is being clustered, in the transformation catalog**

   ```
   #site  transformation   pfn            type                architecture profiles

   siteX    B     /shared/PEGASUS/bin/jobB INSTALLED       INTEL32::LINUX pegasus::co
   ```

   In the above example, all the clustered jobs that consist of transformation B on siteX will be executed via mpiexec.

   Note: The clustering of jobs on a site only happens only if

   - there exists an entry in the transformation catalog for the clustering executable that has been determined by the above 3 rules
   - the number of jobs being clustered on the site are more than 1

8 Outstanding Issues

1. **Label Clustering**

   More rigourous checks are required to ensure that the labeling scheme applied by the user is valid.