Running LCSim Analysis Jobs on the Grid with DIRAC

These instructions will show how to run an lcsim-based user analysis job on the grid using the DIRAC client.

Setup the Grid Session

This setup needs to be performed anytime you want to do any work with DIRAC.

Start by sourcing the setup script for your local copy of DIRAC.

```
source /nfs/slac/g/lcd/mc/prj/sw/x86_64/dirac/bashrc
```

The DIRAC installation location is site dependent. The above will work if you have access to SLAC NFS.

Now obtain a proxy.

```
dirac-proxy-init
```

The default user should be *ilc_user* which should work fine.

Now check that your proxy is valid.

```
dirac-proxy-info
```

The output of that command will look something like this:

```
[1556 $] dirac-proxy-info
subject      : /DC=org/DC=doegrids/OU=People/CN=Jeremy McCormick 123456/CN=proxy/CN=proxy
issuer       : /DC=org/DC=doegrids/OU=People/CN=Jeremy McCormick 123456/CN=proxy
identity     : /DC=org/DC=doegrids/OU=People/CN=Jeremy McCormick 123456
timeleft     : 21:14:04
DIRAC group  : ilc_prod
path         : /tmp/x509up_u8286
username     : jeremy
properties   : NormalUser, ProductionManagement, CSAdministrator, JobSharing, JobAdministrator
VOMS         : True
VOMS fqan    : ['/ilc/Role=production', '/ilc', '/ilc/sid']
```

If there is a problem, then you will need to fix it before continuing as DIRAC does not function correctly without a valid proxy.

Configuring the Working Directory

The following setup will need to be done just once.

Create some directory where you will work with user jobs.

```
mkdir ~/dirac_jobs
cd ~/dirac_jobs
```

Scripts from AFS will be used to launch the user jobs. Symlink to these in order to use the master copies.

```
ln -s /afs/cern.ch/work/c/cgrefe/public/jobSubmission/SiDChainJob.py
ln -s /afs/cern.ch/work/c/cgrefe/public/jobSubmission/bannedSites.py
```

Or you may copy them to your local directory.

```
cp /afs/cern.ch/work/c/cgrefe/public/jobSubmission/SiDChainJob.py .
cp /afs/cern.ch/work/c/cgrefe/public/jobSubmission/bannedSites.py .
```

These scripts will expect a few directories to exist so create these now.

```
mkdir repositoryFiles lcsimSteeringFiles trackingStrategies
```

You will need a local copy of the tracking strategies that will be used by lcsim. Technically, this may not be used in the job, but the script expects it to exist.

```
cvs co lcsim # or use a local up to date copy
cp lcsim/resources/org/lcsim/recon/tracking/seedtracker/strategies/sidloi3_trackingStrategies_default.xml .
/trackingStrategies
```

Finally, you will need an LCSim steering file to specify the drivers used in your user job. This is covered in the next section.

LCSim Steering File

This is best understood by an example.

The following is a very simple test steering file.

```
<lcsim xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
        xs:noNamespaceSchemaLocation="http://www.lcsim.org/schemas/lcsim/1.0/lcsim.xsd">
    <inputFiles>
        <file>${inputFile}</file>
    </inputFiles>
    <control>
        <numberOfEvents>-1</numberOfEvents>
    </control>
    <classpath>
        <jarUrl>http://www.lcsim.org/maven2/org/lcsim/lcsim-analysis/1.0-SNAPSHOT/lcsim-analysis-1.0-
20130426.224545-1.jar</jarUrl>
    </classpath>
    <execute>
        <driver name="EventMarkerDriver"/>
        <driver name="AidaSaveDriver"/>
        <driver name="SimpleTrackAnalysis" />
    </execute>
    <drivers>
        <driver name="EventMarkerDriver" type="org.lcsim.job.EventMarkerDriver">
            <eventInterval>10</eventInterval>
        </driver>
        <driver name="AidaSaveDriver" type="org.lcsim.job.AidaSaveDriver">
            <outputFileName>${outputAidaFile}</outputFileName>
        </driver>
        <driver name="SimpleTrackAnalysis" type="org.lcsim.analysis.SimpleTrackAnalysis" />
    </drivers>
</lcsim>
```

The **jarUrl** should point to the jar file containing your analysis code. In this example, I have pointed to a deployed copy of the lcsim-analysis project, a module from SLAC CVS containing lcsim-based analysis code. The URL here can in theory be any jar file that is publically accessible via HTTP, though it is good to use unique, deployed versions on lcsim.org when possible.

There are three drivers that are run by this analysis.

EventMarkerDriver - prints event numbers
AidaSaveDriver - saves AIDA to an output file
SimpleTrackAnalysis - a very basic test analysis that plots track momenta

Any Driver can be included here, as long as its implementation is present in lcsim or the jars listed in the classpath section. In this case, the SimpleTrackAnalysis Driver is found in the lcsim-analysis jar listed in the classpath section.

The **${outputAidaFile}** is a special variable that the job script uses to insert the name of the AIDA output from your analysis. The name is automatically generated from the input files and your job information.

Running the Jobs

Java Version

The lcsim package currently depends on Java 1.6 whereas the current release is 1.7 so incompatibilities may occur. These will show up as error messages in your job log like the following.

```
2013-04-25 23:56:05 UTC dirac-jobexec/LCSIMAnalysis
ERROR: java version "1.6.0_33"Java(TM)
SE Runtime Environment (build 1.6.0_33-b04)Java HotSpot(TM) 64-Bit Server VM (build 20.8-b03, mixed
mode)
Exception in thread "main" java.lang.UnsupportedClassVersionError: org/lcsim/analysis
/SimpleTrackAnalysis :
Unsupported major.minor version 51.0
```

This means that the jar with the external classes, in this case lcsim-analysis, was compiled with Java 1.7 but the JVM on the grid node is 1.6. This problem can be solved by using a 1.6 compiler locally or specifying 1.6 as the target in the Maven POM file.

Banning Sites

There is a default list of sites NOT to use for the jobs in the *bannedSites.py* script. The default set is perhaps not restrictive enough. This is the contents of mine.

```
bannedSites = [
"LCG.DESY-HH.de",
"LCG.KEK.jp",
"LCG.PNNL.us",
"LCG.UKI-LT2-IC-HEP.uk",
"LCG.IN2P3-CC.fr"
]
```

These are banned for being unreliable or from not being able to install necessary software, etc.

The job script has a lot of different options. For user analysis jobs, most of this functionality will be turned off, so that only the lcsim portion of the job chain executes with a user steering file.

Here is a sample command to submit a user job:

```
python SiDChainJob.py --prodid=1946 --merge=10 --maxfiles=100 --slic="" --marlin="" --pandora="" \
--lcsim="2.8-SNAPSHOT" --lcsimxml=template.lcsim --postlcsimxml="" \
--title="JM_test_analysis_prodID_1946" --detector="sidloi3"
```

If you have setup your working directory as previously described then this command should create a new set of jobs to perform the analysis.

It assumes that your lcsim steering file is present in the working directory with the name **template.lcsim**. The script will actually use this file to create a number of steering files in the *lcsimSteeringFiles* directory.

The *prodid* is used to find input files for the job. Generally these should contain full reconstruction output.

The *merge* argument specifies how many input files to merge into one job. Generally 10 is a good value here. Any more can cause issues on the nodes.

The *slic*, *marlin*, and *pandora* arguments are set to empty strings to indicate that these applications shouldn't be run. The *lcsim* argument specifies a version of lcsim to run, which much be present in the DIRAC software catalog. The DIRAC experts will need to be consulted if you need a special or updated version of lcsim for your job, so that it can be put into the catalog.

The *title* will be used for the job description in the job monitor as well as for naming output files.

The *detector* should be a valid lcsim detector from LCDetectors.

After this command is executed, something like the following should show in the terminal window:

```
################################
 SiD job submission to DIRAC
        christian.grefe@cern.ch
################################

Using production ID to define LFN list:
  Found production ID 1946. Associated meta data:
    EvtType: evW_eeZ_vvZ_semileptonic
    NumberOfEvents: 400
    Polarisation: m80p20
    Datatype: DST
    Energy: 1000
    MachineParams: B1b_ws
    DetectorType: sid
    Machine: ilc
    DetectorModel: sidloi3
  Found 33398 files associated with meta data


Jobs to submit:
  Number of input files: 33398
  Maximum input files to use: 101
  Merged input files per job: 10
  Events per job: all
  Total number of jobs: 10
  Maximum CPU time per job: 100000 sec

General parameters:
  Detector model: sidloi3
  Process name: evW_eeZ_vvZ_semileptonic
  Job title: JM_test_analysis_prodID_1946
  Banned sites: ['LCG.DESY-HH.de', 'LCG.KEK.jp', 'LCG.PNNL.us', 'LCG.UKI-LT2-IC-HEP.uk']
  Repository file: repositoryFiles/sidloi3.JM_test_analysis_prodID_1946.prod1946.cfg

Files:
  Input sand box: ['LFN:/ilc/prod/software/lcsim/lib.tar.gz']
  Output sand box: ['*.log', '*.xml', '*.lcsim']
  Output data: ['JM_test_analysis_prodID_1946_evW_eeZ_vvZ_semileptonic_m80p20_rec_1946_16000.aida']
  Output storage path: sidloi3/evW_eeZ_vvZ_semileptonic/JM_test_analysis_prodID_1946
  Output storage element: CERN-SRM

Steps executed:
  1) LCSim step:
    LCSim version: 2.8-SNAPSHOT
    LCSim file: lcsimSteeringFiles/JM_test_analysis_prodID_1946_evW_eeZ_vvZ_semileptonic_m80p20_rec_1946_16000.
xml
    Tracking strategies: trackingStrategies/sidloi3_trackingStrategies_default.xml
    Detector alias file: alias.properties

Proceed and submit job(s)? (Y/N):
```

Now press the Y key and Enter to launch the jobs.

Then you'll get more output and another prompt:

```
<=====v18r0p4=====>
Replica Lookup Time: 1.42 seconds
All LFN files have replicas available
lcsim 2.8-SNAPSHOT
Attribute list :
   outputSE : Not defined
   energy : 1000.0
   extraParams : Not defined
   outputRecFile : Not defined
   outputDstPath : Not defined
   trackingstrategy : trackingStrategies/sidloi3_trackingStrategies_default.xml
   outputPath : Not defined
   forget_about_Input : Not defined
   outputDstFile : Not defined
   aliasProperties : alias.properties
   accountInProduction : True
   appname : lcsim
   detectortype : SID
   inputSB : ['lcsimSteeringFiles/JM_test_analysis_prodID_1946_evW_eeZ_vvZ_semileptonic_m80p20_rec_1946_16000.
xml', 'trackingStrategies/sidloi3_trackingStrategies_default.xml']
   version : 2.8-SNAPSHOT
   outputFile : JM_test_analysis_prodID_1946_evW_eeZ_vvZ_semileptonic_m80p20_rec_1946_16000.slcio
   willBeCut : Not defined
   outputRecPath : Not defined
   logfile : lcsim_2.8-SNAPSHOT_Step_1.log
   detectorModel : sidloi3.zip
   addedtojob : True
   datatype : REC
   nbevts : -1
   prodparameters : {'JM_test_analysis_prodID_1946_evW_eeZ_vvZ_semileptonic_m80p20_rec_1946_16000.slcio':
{'datatype': 'REC', 'detectortype': 'SID'}}
   steeringfile : lcsimSteeringFiles
/JM_test_analysis_prodID_1946_evW_eeZ_vvZ_semileptonic_m80p20_rec_1946_16000.xml
   debug : Not defined
   inputfile : Not defined


Proceed and submit job(s)? y/[n] :
```

Hit the y key and Enter and the jobs should start.

The script will print messages like the following as it finds your files.

```
<=====v18r0p4=====>
Replica Lookup Time: 1.28 seconds
All LFN files have replicas available
```

Now you are finished with submission.

Monitoring the Jobs

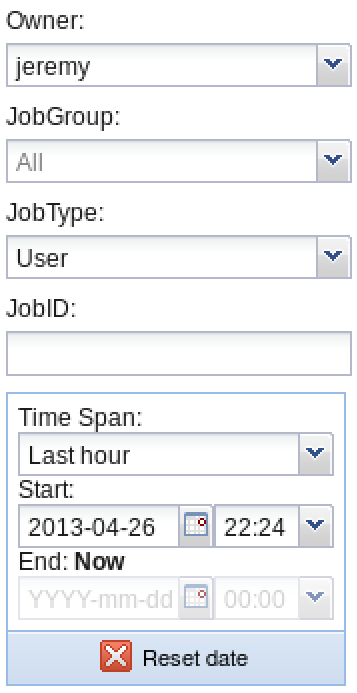
The ILCDIRAC web interface can be used to monitor the jobs.

Open the Job Monitor in a browser that has your grid certificate installed.

Now you will need to find your jobs in order to monitor them.

In the search box on the left side, select your name for *Owner*. Put the *JobType* as "User". For the *Time Span*, select "Last hour".

These selections will look something like this:


Owner: jeremy
JobGroup: All
JobType: User
JobID:
Time Span: Last hour
Start: 2013-04-26 22:24
End: Now
YYYY-mm-dd 00:00

Now click the "Submit" button.

The jobs will show on the right from where they can be monitored, killed, etc.

Getting the Job Output

The output, which in our example is one AIDA file per job, can be retrieved using DIRAC commands.

Create a directory where the job output will be stored.

```
mkdir output
cd output
```

Now you can use the local job repository that was automatically created by DIRAC when the jobs were submitted to retrieve the output in batch.

Below is an example command to retrieve the output data for a sample set of jobs:

```
dirac-repo-retrieve-jobs-output-data -r ../repositoryFiles/sidloi3.JM_test_analysis_prodID_1946.prod1946.cfg
```

This command will copy these output data to local directories named after the job IDs.

The specific name of the repository file will be different depending on what submission settings were used.

The job output data is also stored on the grid in your home directory.

For instance, the sample job that I ran put all the output files here:

```
/ilc/user/j/jeremy/sidloi3/evW_eeZ_vvZ_semileptonic/JM_test_analysis_prodID_1946/
```

The grid files can be obtained using the dirac-dms commands, etc.

Other output from the job is also available, including log files. These files can be obtained using this command:

```
dirac-repo-retrieve-jobs-output -r ../repositoryFiles/sidloi3.JM_test_analysis_prodID_1946.prod1946.cfg
```

Here is an example set of files from one job, not including the analysis output:

```
JM_test_analysis_prodID_1946_evW_eeZ_vvZ_semileptonic_m80p20_rec_1946_16062.xml
jobDescription.xml
job_1.lcsim
lcsim_2.8-SNAPSHOT_Step_1.log
localEnv.log
sidloi3_trackingStrategies_default.xml
std.out
```

In order, these are the following:

1. Steering file with actual output file names resolved.
2. XML file in DIRAC's Workflow schema descripting the job.
3. Steering file with input and output file names resolved.
4. Log output from the lcsim job.
5. Local shell environment of the worker node.
6. Tracking strategies.
7. All stdout IO including from DIRAC and lcsim.

These files can be useful when debugging any problems that may occur with the job execution.

Combining Output Files

Finally, you will need to combine the job output into a final analysis. This could be specific to your particular analysis. I will provide an example that combines the histograms from multiple AIDA files into one output file. These procedures may change in the future, and this is just an example.

Download a jar file from lcsim.org that contains a utility for merging AIDA files.

```
cd ~/work
wget http://www.lcsim.org/maven2/org/lcsim/hps-java/1.6/hps-java-1.6-bin.jar
```

Move all your output AIDA files to a single directory.

```
cd ~/work/output
mkdir scratch
cp */*.aida scratch
```

Finally, execute the utility to combine the AIDA files into a single output file:

```
cd ~/work
java -cp ./hps-java-1.6-bin.jar org.lcsim.hps.users.phansson.mergeSimpleAIDA -d ./output/scratch/ -r ".*.aida" -
o myanalysis.aida
```

Now you can open the final AIDA file in JAS3 or perform additional analysis on it.

The analysis on the final output is only an example. You will probably want to write your own code to combine your AIDA output if the utility's default behavior of simply adding histograms is insufficient for your needs.