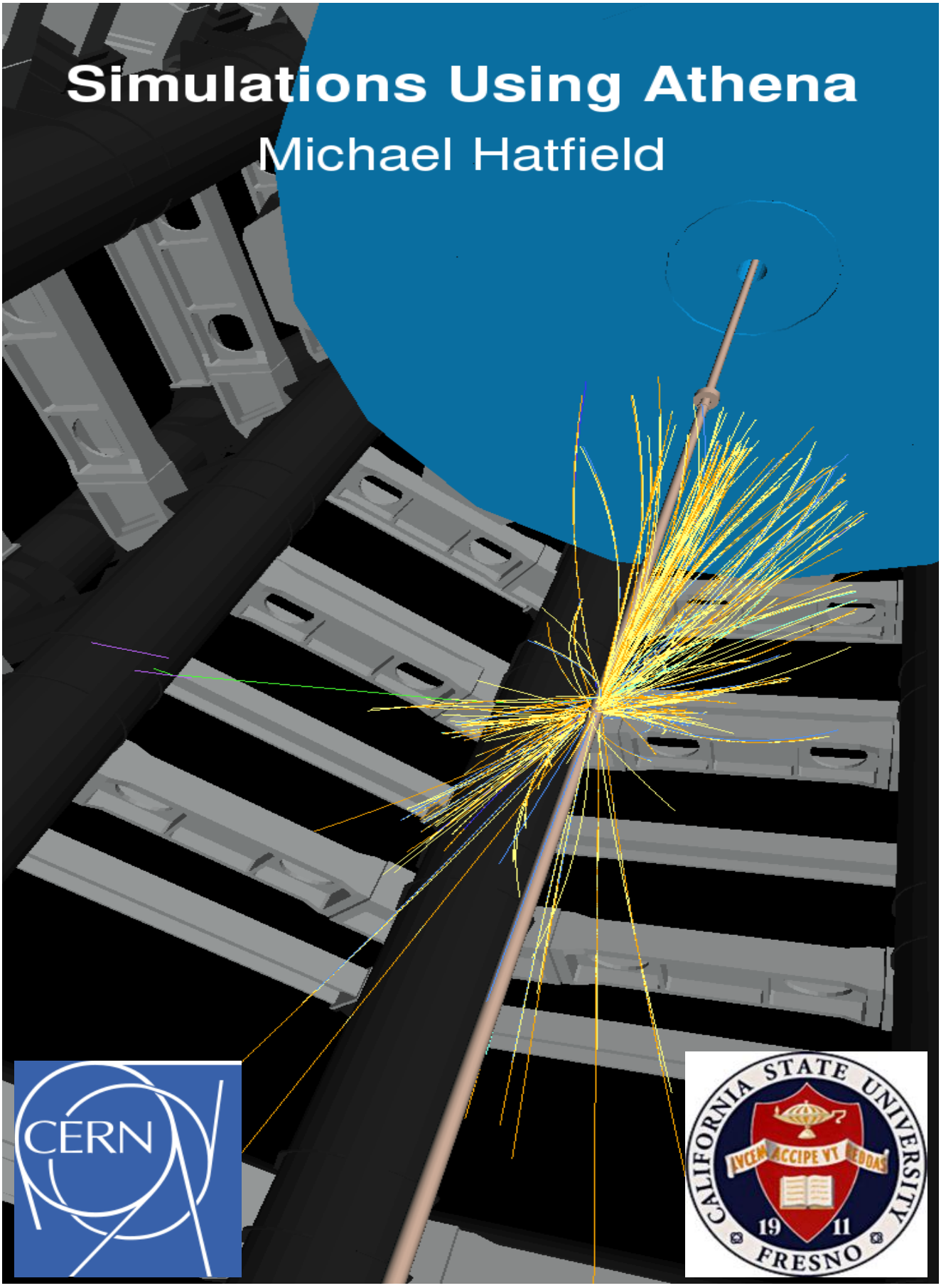
Simulations Using Athena

Michael Hatfield



Contents

Introduction:

Almost all large experiments in high-energy physics at LEP, HERA, Fermilab and LHC have included in their physical programs a search of excited fermions: leptons, and quarks, q*. Their interactions with ordinary matter proceed through a non-minimal tensor coupling $f^*\sigma\mu\nu f\cdot\partial\mu Z\nu$ instead of the minimal gauge coupling $f\gamma\mu f\cdot Z\mu$. Now it is natural to ask: "Why only excited fermions are being looked for, but not excited bosons, Z* , with a similar form of interaction $f\sigma\mu\nu f\cdot\partial\mu Z^*\nu$?" The project of such kind of search has been proposed by M.Chizhov, V.Bednyakov and J.Budagov in Phys. Atom. Nucl. 71 (2008) 2006.

The samples were generated with CalcHEP 2.6 at $\sqrt{s}$ = 7 TeV. Process $pp \rightarrow W^* \rightarrow qqbar$, with MW* = 800 GeV and mixing angle sin(X) = 1.0 . CalcHep gives lhe files which is interfaced with pythia for hadronisation. The generated events need to pass through full detector simulation , digitization and reconstruction under ATLAS framework in which the generated events are passed through the simulated ATLAS detector with specific geometry condition, so that the final reconstructed events can be compared directly with the real data events.

This is an overview of the full chain process for event simulation using Athena software on the CSU Fresno tier 3 system. Data analysis requires AOD or D3PD files, which this outline will describe how to create. The steps are Event Generation, Simulation, Digitization, Reconstruction, AOD production, and finally D3PD production. Each step is outlined below, where the command to run each step is given inside a box and the details of each argument in the command are explained afterward.

Simulation

```
csc_atlasG4_trf.py inputEvgenFile=Input_File_Name.pool.root
ouputHitsFile=Output_File_Name.pool.root maxEvents=-1 skipEvents=0
randomSeed=12345 geometryVersion=GeometryName
conditionsTag=ConditionsName > Log_File_Name 2>&1
```

The arguments for simulation are:

- csc_atlasG4_trf.py
  - This is the simulation command. Each additional argument leading up to the first angle bracket ">" are options for this command. The "csc_" prefix is older and more recent Athena versions will require "AtlasG4_trf.py". Details of each command and the acceptable arguments can be found in the following links:
  - csc_atlasG4_trf.py: http://alxr.usatlas.bnl.gov/lxr-stb4/source/atlas/Simulation/SimuJobTransforms/scripts/csc_atlasG4_trf.py
  - AtlasG4_trf.py: http://alxr.usatlas.bnl.gov/lxr-stb4/source/atlas/Simulation/SimuJobTransforms/scripts/AtlasG4_trf.py
- inputEvgenFile
  - This is the input file with the generated events
- outputHitsFile
  - This is the file that will be created with the simulated hits.
- maxEvents
  - This is the number of events to simulate. While a file with generated events may have many events (up to 5,000), not all events have to be simulated. You can specify the number of events to simulate with this option. Entering "-1" as the value means all events in the input file.
- SkipEvents
  - This is the number of events to skip. This allows you to break one input file into multiple jobs. Very important if submitting multiple jobs on a batch system for one input file.
- RandomSeed
  - This is the initial value for the random number generator. To have the computer automatically set this value, use randomSeed=$RANDOM.
- GeometryVersion
  - This is the ATLAS detector layout. The available tags can be found here: https://twiki.cern.ch/twiki/bin/viewauth/Atlas/AtlasGeomDBTags#ATLAS_Geom_DB_Tag_contents
- ConditionsTag
  - This is a COOL tag and accompanies the GeometryVersion tag. It describes the conditions of the detector for the given detector geometry. A full list of tags can be found here: https://twiki.cern.ch/twiki/bin/viewauth/Atlas/CoolProdTags
- Log_File_Name
  - This is the name of the log file which has all the output of the simulation. If there is ever a problem with running a simulation, this is the first file you will want to read to see what happened.
- 2>&1
  - This is a bash command which dumps any error messages into the log file.

Simulation example:

```
csc_atlasG4_trf.py inputEvgenFile=EVNT.448090._000001.pool.root.1
ouputHitsFile=mc10_7TeV.119666_CalcHepPythia_WStar_800_dijets.pool.root_e843_
s933 maxEvents=-1 skipEvents=0 randomSeed=$RANDOM geometryVersion=ATLAS-
GEO-16-00-00 conditionsTag=OFLCOND-SDR-BS7T-02 > Sim.out 2>&1
```

Digitization

```
Digi_trf.py inputHitsFile=Input_File_Name.pool.root
ouputRDOFile=Output_File_Name.pool.root maxEvents=-1 skipEvents=0
geometryVersion=GeometryName conditionsTag=ConditionsName > Log_File_Name
2>&1
```

The arguments for digitization are:

- Digi_trf.py
  - This is the digitization command. Each additional argument leading up to the first angle bracket ">" are options for this command. Details of the command and the acceptable arguments can be found here: http://alxr.usatlas.bnl.gov/lxr-stb4/source/atlas/Simulation/SimuJobTransforms/scripts/Digi_trf.py
- inputHitsFile
  - This is the input file for digitization. The output from the previous step (simulation) will be used as the input.
- outputRDOFile
  - This is the file that will be created in the digitization process. The RDO (Raw Data Object) file is not in a form that can be used for analysis, but contains all the information of the simulated collision.
- maxEvents
  - This is the number of events. Entering "-1" as the value means all events in the input file.
- SkipEvents
  - This is the number of events to skip. This allows you to break one input file into multiple jobs. Very important if submitting multiple jobs on a batch system for one input file.
- GeometryVersion
  - This is the ATLAS detector layout. The available tags can be found here: https://twiki.cern.ch/twiki/bin/viewauth/Atlas/AtlasGeomDBTags#ATLAS_Geom_DB_Tag_contents
- ConditionsTag
  - This is a COOL tag and accompanies the GeometryVersion tag. It describes the conditions of the detector for the given detector geometry. A full list of tags can be found here: https://twiki.cern.ch/twiki/bin/viewauth/Atlas/CoolProdTags
- Log_File_Name
  - This is the name of the log file which has all the output of the digitization step. If there is ever a problem, this is the first file you will want to read to see what happened.
- 2>&1
  - This is a bash command which dumps any error messages into the log file mentioned above (Log_File_Name).

Digitization example:

```
Digi_trf.py
inputHitsFile=mc10_7TeV.119666_CalcHepPythia_WStar_800_dijets.pool.root_e843_s
933
ouputRDOFile=mc10_7TeV.119666_CalcHepPythia_WStar_800_dijets.pool.root_e843
_s933_r2302 maxEvents=-1 skipEvents=0 geometryVersion=ATLAS-GEO-16-00-00
conditionsTag=OFLCOND-SDR-BS7T-04-13 > Digi.out 2>&1
```

Reconstruction

```
Reco_trf.py inputRDOFile=Input_File_Name.pool.root
ouputESDFile=Output_File_Name.pool.root maxEvents=-1 skipEvents=0
geometryVersion=GeometryName conditionsTag=ConditionsName > Log_File_Name
2>&1
```

The arguments for reconstruction are:

- Reco_trf.py
  - This is the reconstruction command. Each additional argument leading up to the first angle bracket ">" are options for this command. Details of the command and the acceptable arguments can be found here: http://alxr.usatlas.bnl.gov/lxr/source/atlas/PhysicsAnalysis/PATJobTransforms/scripts/Reco_trf.py
- inputRDOFile
  - This is the input file for reconstruction. The output from the previous step (digitization) will be used as the input.
- outputESDFile
  - This is the file that will be created in the reconstruction process. The ESD (Event Summary Data) is the base data format which holds most of the initial information (hits, calorimeter clusters, etc) as well as the reconstructed information (tracks, jets, etc). Although providing the most information for the event display, it is very large and not as practical for analysis.
- maxEvents
  - This is the number of events. Entering "-1" as the value means all events in the input file.
- SkipEvents
  - This is the number of events to skip. This allows you to break one input file into multiple jobs. Very important if submitting multiple jobs on a batch system for one input file.
- GeometryVersion
  - This is the ATLAS detector layout. The available tags can be found here: https://twiki.cern.ch/twiki/bin/viewauth/Atlas/AtlasGeomDBTags#ATLAS_Geom_DB_Tag_contents
- ConditionsTag
  - This is a COOL tag and accompanies the GeometryVersion tag. It describes the conditions of the detector for the given detector geometry. A full list of tags can be found here: https://twiki.cern.ch/twiki/bin/viewauth/Atlas/CoolProdTags
- Log_File_Name
  - This is the name of the log file which has all the output of the digitization step. If there is ever a problem, this is the first file you will want to read to see what happened.
- 2>&1
  - This is a bash command which dumps any error messages into the log file mentioned above (Log_File_Name).

Reconstruction example:

```
Reco_trf.py
inputRDOFile=mc10_7TeV.119666_CalcHepPythia_WStar_800_dijets.pool.root_e843_
s933_r2302
ouputESDFile=mc10_7TeV.119666_CalcHepPythia_WStar_800_dijets.ESD.pool.root_e
843_s933_r2302 maxEvents=-1 skipEvents=0 geometryVersion=ATLAS-GEO-16-00-
00 conditionsTag=OFLCOND-SDR-BS7T-04-13 > Reco.out 2>&1
```

AOD Production

```
Reco_trf.py inputESDFile=Input_File_Name.pool.root
ouputAODFile=Output_File_Name.pool.root maxEvents=-1 skipEvents=0
geometryVersion=GeometryName conditionsTag=ConditionsName > Log_File_Name
2>&1
```

The arguments for AOD production are:

- Reco_trf.py
  - This is the reconstruction command, but used here to convert the ESD fiel to an AOD file. Each additional argument leading up to the first angle bracket ">" are options for this command. Details of the command (including possible file types that can be produced) and the acceptable arguments can be found here: http://alxr.usatlas.bnl.gov/lxr/source/atlas/PhysicsAnalysis/PATJobTransforms/scripts/Reco_trf.py
- inputESDFile
  - This is the input file for AOD production. The output from the previous step (reconstruction) will be used as the input.
- outputAODFile
  - This is the file that will be created in the AOD production process. The AOD (Analysis Object Data) format is a slimmed down version of the ESD format (no hits, calorimeter cell information, etc.) and only a limited amount of information can be extracted for the event display. This makes analysis much easier than working with ESD files.
- maxEvents
  - This is the number of events. Entering "-1" as the value means all events in the input file.
- SkipEvents
  - This is the number of events to skip. This allows you to break one input file into multiple jobs. Very important if submitting multiple jobs on a batch system for one input file.
- GeometryVersion
  - This is the ATLAS detector layout. The available tags can be found here: https://twiki.cern.ch/twiki/bin/viewauth/Atlas/AtlasGeomDBTags#ATLAS_Geom_DB_Tag_contents
- ConditionsTag
  - This is a COOL tag and accompanies the GeometryVersion tag. It describes the conditions of the detector for the given detector geometry. A full list of tags can be found here: https://twiki.cern.ch/twiki/bin/viewauth/Atlas/CoolProdTags
- Log_File_Name
  - This is the name of the log file which has all the output of the digitization step. If there is ever a problem, this is the first file you will want to read to see what happened.
- 2>&1
  - This is a bash command which dumps any error messages into the log file mentioned above (Log_File_Name).

AOD production example:

```
Reco_trf.py
inputESDFile=mc10_7TeV.119666_CalcHepPythia_WStar_800_dijets.ESD.pool.root_e
843_s933_r2302
ouputAODFile=mc10_7TeV.119666_CalcHepPythia_WStar_800_dijets.AOD.pool.root
_e843_s933_r2302 maxEvents=-1 skipEvents=0 geometryVersion=ATLAS-GEO-16-
00-00 conditionsTag=OFLCOND-SDR-BS7T-04-13 > AOD.out 2>&1
```

D3PD Production

```
AODToPhysicsD3PD.py
```

or to have a log file with the output and errors:

```
AODToPhysicsD3PD.py > Log_File_Name 2>&1
```

D3PD production simply requires getting the AODtoPhysicsD3PD.py script or creating it by copying it from this location into a text file:
http://alxr.usatlas.bnl.gov/lxr/source/atlas/PhysicsAnalysis/D3PDMakerConfig/share/AODToPhysicsD3PD.py
The input file name needs to be given in line 15 by substituting “AOD.pool.root” with the actual file name. The output file name needs to be given in line 22 by substituting “physics.root” with the actual output name.

Problems/Solutions

A few solutions to problems encountered during the full chain are listed here.

1) Problem:
During simulation, the following error was in the log file:
"ValueError: ATLAS-GEO-16-00-00 is not the expected type and/or the value is not allowed for JobProperties.SimFlags.SimLayout"

Solution:
When there is a problem with the geometry, you can always try using "_VALIDATION" after the geometry version: ATLAS-GEO-16-00-00_VALIDATION. If that doesn't work, in your job options file you can set GeoModelSvc.IgnoreTagDifference=True. In this particular case it was a problem with the database release for the Athena setup. When setting up Athena with asetup, use –dbrelease=<release number>: "asetup 16.6.4.3,here –dbrelease=15.3.1". If you do not specifiy a dbrelease the default release will be used. After Athena is setup, the db release can be checked with "echo $DBRELEASE".

2) Problem:
Athena would not setup for batch jobs and gave the following errors:
a) /var/lib/condor/execute/dir_6235/condor_exec.exe:
/nfs/t3nfs/share/atlas/ATLASLocalRootBase/user/atlasLocalSetup.sh: No such file or directory
b) /var/lib/condor/execute/dir_6235/condor_exec.exe:
/nfs/t3nfs/share/atlas/ATLASLocalRootBase/packageSetups/atlasLocalGccSetup.sh: No such file or directory

Solution:
/nfs/t3nfs/share/atlas was not mounted on nodes that were trying to run the job. You can check if the path to setup files is mounted with
"ssh pt3wrk3 ls /nfs/t3nfs/share/atlas/ATLASLocalRootBase", where "pt3wrk3" is the node that you are running a job on.

3) Problem:
The following error was in the digitization log file: "Index Error: list index out of range"

Solution:
This was caused because the simulation did not complete successfully. There are a few ways to check the output file from any step to see how many events are in the file and what tags the file contains:
a) dump-athfile.py <filename>
b) checkFile.py <filename>
c) dumpVersionTags.py <filename>
These are useful commands to inspect your input file if you have any problems during a step of the full chain. These should work with any Athena version.
4) A good source for help with problems is the Hypernews Forum:
https://groups.cern.ch/group/hn-atlas-offlineSWHelp/default.aspx

Batch Jobs

Simulations take a long time to run, especially if there are a lot of events. It is best to use a batch system to submit your jobs. The batch system used at CSU Fresno is Condor. A few basic Condor commands are listed here:

condor_q – show active batch jobs
condor_submit – submit a batch job
condor_rm – remove a batch job

To submit a job, you would use

```
condor_submit /path/to/file/filename.cmd
```

Condor_q shows a list of all jobs and the job number. If you need to remove a job, you use

```
condor_rm <job number>
```

For example, if we have a submission script called "test.cmd" located in the directory "/xdata/user/batch", then we would submit the job with

```
condor_submit /xdata/user/batch/test.cmd
```

If condor_q showed the job number to be 1155, then we could cancel the job with

```
condor_rm 1155
```

There are 2 files needed for Condor; a cmd file and a sh file. The cmd file tells Condor what to do and the sh file is what actually runs. Each file is described below.

*.cmd file

The cmd file for CSU Fresno has these components:

```
requirements            = (Arch == "INTEL"||Arch == "X86_64")&& \
                        ((Machine == "pt3wrk0.atlas.csufresno.edu")|| \
                         (Machine == "pt3wrk1.atlas.csufresno.edu")|| \
                         (Machine == "pt3wrk2.atlas.csufresno.edu")|| \
                         (Machine == "pt3wrk3.atlas.csufresno.edu")|| \
                         (Machine == "pt3wrk4.atlas.csufresno.edu"))
executable                      = /xdata/user/batch/filename.sh
output                          = /xdata/user/batch/results/filename.out
error                           = /xdata/user/batch/results/filename.err
log                             = /xdata/user/batch/results/filename.log
universe                        = vanilla
should_transfer_files           = YES
when_to_transfer_output         = ON_EXIT
notify_user                     = user@pt3nfs.atlas.csufresno.edu
queue
```

The requirements part are the user defined requirements for the job to run. There are many options that can be entered here but a partial overview can be found at this website:
http://www.mi.infn.it/condor/manual/2_6Submitting_Job.html
The executable is the path to your sh file.

Output, error, and log are the Condor log files for the job.
notify_user is the user name where a message will appear when a job finishes. This is typically a message saying “You have new mail in /var/spool/username”, or something along those lines.

*.sh file

The sh file is the executable and and typically follows this template:
A project area is made so the temporary files can be deleted after the job finishes. Then Athena is setup in the project area, the necessary files are copied to the project area, and the appropriate Athena command is called. The output file is then copied to the results area and the project area is deleted.

Here is a sample sh file:

```
#!/bin/bash

JOBNAME="119666_5k1"
export USER="$(id -nu)"
echo $USER
DATE="$(date | sed 's/:/ /g' | awk '{print $2$3"_"$4_$5_$6}')"
JOBID="$(echo $PWD | sed 's/\// /g' | awk '{print $NF}')"
LOCAL_SCRATCH="/disk/$USER"
NFS_PROJECT_AREA="/xdata/${USER}/batch"
NFS_RESULTS_AREA="/xdata/${USER}/batch/results"

#==need these for atlas setup scripts
export HOME="/home/${USER}"
export LOCAL_JOB_AREA="${JOBNAME}_${DATE}_$JOBID"
export ATLAS_TEST_AREA="$LOCAL_SCRATCH/${LOCAL_JOB_AREA}/${ATLAS_RELEASE}"

#==set atlas environment
mkdir -p ${ATLAS_TEST_AREA}
source /cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase/user/atlasLocalSetup.sh > /dev/null
source /nfs/t3nfs/share/atlas/ATLASLocalRootBase/packageSetups/atlasLocalGccSetup.sh --gccVersion=gcc432_x86_64_slc5
source /cvmfs/atlas.cern.ch/repo/sw/software/i686-slc5-gcc43-opt/15.6.12/AtlasSetup/scripts/asetup.sh 15.6.12.9,here --dbrelease=12.2.1

#==prepare local job area
cd  $ATLAS_TEST_AREA
cp -rfvp $NFS_PROJECT_AREA/* ./

#==simulation
  csc_atlasG4_trf.py inputEvgenFile=EVNT.448090._000001.pool.root.1 outputHitsFile=119666.1.pool.root maxEvents=-1 skipEvents=0
randomSeed=94751 geometryVersion=ATLAS-GEO-16-00-00 JobConfig=VertexFromCondDB.py conditionsTag=OFLCOND-SDR-8S7T-02 >
s933_5k_1.out 2>&1

#==Copy files and clean up
env | sort
cp -rfvp s933_5k_1.out $NFS_RESULTS_AREA/
cp -rfvp 119666.1.pool.root $NFS_RESULTS_AREA/
cd $LOCAL_SCRATCH
ls -d  $LOCAL_JOB_AREA
rm -rf $LOCAL_JOB_AREA
```

The simulation section of this file (after the label #==simulation) is actually all one line.

Full Chain Script

The full chain process can be ran with one Condor submission by submitting the next job at the end of a script. As an example, the following script called jobCreator.sh uses several generic scripts to create all the necessary steps for the full chain except for event generation. After event generation, a user can enter the criteria for each step in the appropriate sections and all the cmd and sh scripts will be generated and submitted. Below is the jobCreator.sh file, followed by the generic scripts which this file edits to create the necessary cmd and sh files.

jobCreator.sh

```
#!/bin/bash
export USER="$(id -nu)"

#==Edit the variable names in the following sections || Make sure all paths exist

#============================#
#== ENTER JOB INFORMATION ==#
#============================#
JOB='119666_1'  #Job name
DATA='/xdata/$USER/batch'  #Path where data is located (i.e. EVNT files)
RESULTS='/xdata/$USER/batch/results'  #Path where results need to go
NUM_EVENTS='5000'  #Number of events to process || Explicit number, do NOT use "-1" for all events!

#============================#
#== ENTER SIMULATION VALUES ==#
#============================#
S_ATHENA_RELEASE='15.6.12'  #Athena core (i.e. 16.6.4)
S_ATHENA_VERSION='15.6.12.9'  #Specific Athena version (i.e. 16.6.4.X) || If same as ATHENA_RELEASE, enter that (i.e. 16.6.4)
S_DATABASE='12.2.1'  #dbRelease -> use DATABASE='current' if unsure
S_COMMAND='csc_atlasG4_trf.py'  #Athena command for simulation
S_INPUT_FILE='EVNT.448090._000001.pool.root.1'  #Generated event file name
S_OUTPUT_FILE='mc10_7TeV.119666_1.CalcHepPythia_WStar_800_dijets.pool.root.e843_s933_s946'  #Output file name
S_LOG_FILE='SIM.out'  #Name of log file for Athena output
S_TIME_LOG='SIM.time'  #Name of log file for the simulation time
S_GEOMETRY_VERSION='ATLAS-GEO-16-00-00'  #Geometry Version
S_CONDITION_TAG='OFLCOND-SDR-BS7T-02'  #Conditions Tag
#==Enter any additional options || Leave blank if none -> OTHER_TAG_i="
S_OTHER_TAG_1='JobConfig=VertexFromCondDB.py'
S_OTHER_TAG_2='PhysicsList=QGSP_BERT'
S_OTHER_TAG_3="
S_OTHER_TAG_4="
S_OTHER_TAG_5="


#============================#
#== ENTER DIGITIZATION VALUES ==#
#============================#
D_ATHENA_RELEASE='16.6.4'  #Athena core (i.e. 16.6.4)
D_ATHENA_VERSION='16.6.4.3'  #Specific Athena version (i.e. 16.6.4.X) || If same as ATHENA_RELEASE, enter that (i.e. 16.6.4)
D_DATABASE='15.3.1'  #dbRelease -> use DATABASE='current' if unsure
D_COMMAND='Digi_trf.py'  #Athena command for digitization
D_OUTPUT_FILE='mc10_7TeV.119666_1.CalcHepPythia_WStar_800_dijets.pool.root.e843_s933_s946_r2302'  #Output file name
D_LOG_FILE='DIGI.out'  #Name of log file for Athena output
D_TIME_LOG='DIGI.time'  #Name of log file for the simulation time
D_GEOMETRY_VERSION='ATLAS-GEO-16-00-00'  #Geometry Version
D_CONDITION_TAG='OFLCOND-SDR-BS7T-04-13'  #Geometry Version
#==Enter any additional options || Leave blank if none -> OTHER_TAG_i="
D_OTHER_TAG_1="autoConfiguration='everything'"
D_OTHER_TAG_2="
D_OTHER_TAG_3="
D_OTHER_TAG_4="
D_OTHER_TAG_5="


#================================#
#== ENTER RECONSTRUCTION VALUES ==#
#================================#
R_ATHENA_RELEASE='16.6.4'  #Athena core (i.e. 16.6.4)
R_ATHENA_VERSION='16.6.4.3'  #Specific Athena version (i.e. 16.6.4.X) || If same as ATHENA_RELEASE, enter that (i.e. 16.6.4)
```

```
R_DATABASE='15.3.1'  #dbRelease -> use DATABASE='current' if unsure
R_COMMAND='Reco_trf.py'  #Athena command for reconstruction
R_OUTPUT_FILE='mc10_7TeV.119666_1.CalcHepPythia_WStar_800_dijets.ESD.pool.root.e843_s933_s946_r2302'  #Output file name
R_LOG_FILE='RECO.out'  #Name of log file for Athena output
R_TIME_LOG='RECO.time'  #Name of log file for the simulation time
R_GEOMETRY_VERSION='ATLAS-GEO-16-00-00'  #Geometry Version
R_CONDITION_TAG='OFLCOND-SDR-BS7T-04-13'  #Conditions Tag
#==Enter any additional options || Leave blank if none -> OTHER_TAG_i="
R_OTHER_TAG_1="autoConfiguration='everything'"
R_OTHER_TAG_2="
R_OTHER_TAG_3="
R_OTHER_TAG_4="
R_OTHER_TAG_5="


#========================#
#== ENTER AOD FILE NAME ==#
#========================#
AOD_OUTPUT_FILE='mc10_7TeV.119666_1.CalcHepPythia_WStar_800_dijets.AOD.pool.root.e843_s933_s946_r2302'  #AOD file name


#=========================#
#== ENTER D3PD FILE NAME ==#
#=========================#
D3PD_OUTPUT_FILE='mc10_7TeV.119666_1.CalcHepPythia_WStar_800_dijets.D3PD.pool.root.e843_s933_s946_r2302'  #D3PD file name


#== END OF USER INPUT ==#
#!!!!!!!!!!!!!!!!!!!!!!!!#


#=================================================================#
#!!!!!!!!!!DO NOT EDIT ANYTHING BELOW THIS!!!!!!!!!!!!!!!!!!!!!!!!#
#=================================================================#

SW=50
if [ ${NUM_EVENTS} -gt ${SW} ]
then
	let MAX=$(( $NUM_EVENTS / $SW ))
else
	let MAX=$(( ${NUM_EVENTS} / 2 ))
	SW=2
fi
let LAST=$MAX-1

for ((j=0; j<${MAX}; j++))
do
	if [ ${j} -eq ${LAST} ]
	then
		echo -n "${S_LOG_FILE}_${j}" >> simloglist
			echo -n "${S_OUTPUT_FILE}_${j}" >> simoutlist
	else
		echo -n "${S_LOG_FILE}_${j}"","  >> simloglist
			echo -n "${S_OUTPUT_FILE}_${j}"","  >> simoutlist
	fi
done
echo -n "" >> jobnamelist
echo -n "${JOB}" >> jobnamelist
echo -n "" >> jobnamelist
echo -n "" >> datanamelist
echo -n "${DATA}" >> datanamelist
echo -n "" >> datanamelist
echo -n "" >> resnamelist
echo -n "${RESULTS}" >> resnamelist
echo -n "" >> resnamelist
echo -n "" >> mergeoutlist
echo -n "${S_OUTPUT_FILE}" >> mergeoutlist
echo -n "" >> mergeoutlist
echo -n "" >> datrlist
echo -n "${D_ATHENA_RELEASE}" >> datrlist
echo -n "" >> datrlist
echo -n "" >> datvlist
echo -n "${D_ATHENA_VERSION}" >> datvlist
```

```
echo -n "" >> datvlist
echo -n "" >> ddblist
echo -n "${D_DATABASE}" >> ddblist
echo -n "" >> ddblist
echo -n "" >> ratrlist
echo -n "${R_ATHENA_RELEASE}" >> ratrlist
echo -n "" >> ratrlist
echo -n "" >> ratvlist
echo -n "${R_ATHENA_VERSION}" >> ratvlist
echo -n "" >> ratvlist
echo -n "" >> rdblist
echo -n "${R_DATABASE}" >> rdblist
echo -n "" >> rdblist
echo -n "" >> dloglist
echo -n "${D_LOG_FILE}" >> dloglist
echo -n "" >> dloglist
echo -n "" >> dtimelist
echo -n "${D_TIME_LOG}" >> dtimelist
echo -n "" >> dtimelist
echo -n "" >> doutputlist
echo -n "${D_OUTPUT_FILE}" >> doutputlist
echo -n "" >> doutputlist
echo -n "" >> rloglist
echo -n "${R_LOG_FILE}" >> rloglist
echo -n "" >> rloglist
echo -n "" >> rtimelist
echo -n "${R_TIME_LOG}" >> rtimelist
echo -n "" >> rtimelist
echo -n "" >> routputlist
echo -n "${R_OUTPUT_FILE}" >> routputlist
echo -n "" >> routputlist
echo -n "" >> aodlist
echo -n "${AOD_OUTPUT_FILE}" >> aodlist
echo -n "" >> aodlist
echo -n "" >> atrlist
echo -n "${S_ATHENA_RELEASE}" >> atrlist
echo -n "" >> atrlist
echo -n "" >> atvlist
echo -n "${S_ATHENA_VERSION}" >> atvlist
echo -n "" >> atvlist
echo -n "" >> dblist
echo -n "${S_DATABASE}" >> dblist
echo -n "" >> dblist
echo -n "" >> sinlist
echo -n "${S_INPUT_FILE}" >> sinlist
echo -n "" >> sinlist
S_IN="`cat sinlist`"
S_ATHENA_RELEASEQ="`cat atrlist`"
S_ATHENA_VERSIONQ="`cat atvlist`"
S_DATABASEQ="`cat dblist`"
rm sinlist
rm atrlist
rm atvlist
rm dblist
JOBQ="`cat jobnamelist`"
DATAQ="`cat datanamelist`"
RESULTSQ="`cat resnamelist`"
INPUT_LOG_FILES="`cat simloglist`"
INPUT_MERGE_FILES="`cat simoutlist`"
M_OUTPUT_FILEQ="`cat mergeoutlist`"
D_ATHENA_RELEASEQ="`cat datrlist`"
D_ATHENA_VERSIONQ="`cat datvlist`"
D_DATABASEQ="`cat ddblist`"
D_LOG_FILEQ="`cat dloglist`"
D_TIME_LOGQ="`cat dtimelist`"
D_OUTPUT_FILEQ="`cat doutputlist`"
R_ATHENA_RELEASEQ="`cat ratrlist`"
R_ATHENA_VERSIONQ="`cat ratvlist`"
R_DATABASEQ="`cat rdblist`"
R_LOG_FILEQ="`cat rloglist`"
R_TIME_LOGQ="`cat rtimelist`"
R_OUTPUT_FILEQ="`cat routputlist`"
AOD_OUTPUT_FILEQ="`cat aodlist`"
```

```
rm aodlist
rm dloglist
rm dtimelist
rm doutputlist
rm rloglist
rm rtimelist
rm routputlist
rm simloglist
rm simoutlist
rm jobnamelist
rm datanamelist
rm resnamelist
rm mergeoutlist
rm datrlist
rm datvlist
rm ddblist
rm ratrlist
rm ratvlist
rm rdblist

cp ./jobMergeHits.sh ./${JOB}_Merge.sh
cp ./jobDigi.sh ./${JOB}_Digi.sh
cp ./jobSubmit.cmd ./${JOB}_Digi.cmd
cp ./jobReco.sh ./${JOB}_Reco.sh
cp ./jobSubmit.cmd ./${JOB}_Reco.cmd
cp ./jobReco.sh ./${JOB}_AOD.sh
cp ./jobSubmit.cmd ./${JOB}_AOD.cmd
cp ./jobD3PD.py ./${JOB}_D3PD.py
cp ./jobSubmit.cmd ./${JOB}_Merge.cmd

M_LOG_FILEQ='"MERGE.out"'
M_TIME_LOGQ='"MERGE.time"'
sed -i '/JOBNAME=/ c\JOBNAME='${JOBQ}" ${JOB}_Merge.sh
sed -i '/NFS_PROJECT_AREA=/ c\NFS_PROJECT_AREA='${DATAQ}" ${JOB}_Merge.sh
sed -i '/NFS_RESULTS_AREA=/ c\NFS_RESULTS_AREA='${RESULTSQ}" ${JOB}_Merge.sh
sed -i '/ATLAS_RELEASE=/ c\ATLAS_RELEASE='${S_ATHENA_RELEASEQ}" ${JOB}_Merge.sh
sed -i '/ATLAS_VERSION=/ c\ATLAS_VERSION='${S_ATHENA_VERSIONQ}" ${JOB}_Merge.sh
sed -i '/DBRELEASE=/ c\DBRELEASE='${S_DATABASEQ}" ${JOB}_Merge.sh
sed -i '/OUT_FILE=/ c\OUT_FILE='${M_OUTPUT_FILEQ}" ${JOB}_Merge.sh
sed -i '/LOG=/ c\LOG='${M_LOG_FILEQ}" ${JOB}_Merge.sh
sed -i '/TIME=/ c\TIME='${M_TIME_LOGQ}" ${JOB}_Merge.sh
sed -i '/executable=/ c\executable='${DATA}'/'${JOB}_Merge.sh" ${JOB}_Merge.cmd
sed -i '/output=/ c\output='${RESULTS}'/'${JOB}_Merge.out" ${JOB}_Merge.cmd
sed -i '/error=/ c\error='${RESULTS}'/'${JOB}_Merge.err" ${JOB}_Merge.cmd
sed -i '/log=/ c\log='${RESULTS}'/'${JOB}_Merge.log" ${JOB}_Merge.cmd
sed -i '/YES/ a\when_to_transfer_output=ON_EXIT' ${JOB}_Merge.cmd
sed -i '/notify_user=/ c\notify_user='$USER'@pt3nfs.atlas.csufresno.edu' ${JOB}_Merge.cmd
sed -i '/#==EXTRAS/ a\cd $ATLAS_TEST_AREA' ${JOB}_Merge.sh
sed -i '/#==EXTRAS/ a\condor_submit '${DATA}'/'${JOB}_Digi.cmd" ${JOB}_Merge.sh
sed -i '/#==EXTRAS/ a\cd $HOME' ${JOB}_Merge.sh
sed -i '/ALL=/ c\ALL='${MAX}" ${JOB}_Merge.sh
sed -i '/#==Wait/ c\                while [ ! -e '${RESULTS}'/'${S_LOG_FILE}'_${k} ]; do' ${JOB}_Merge.sh
sed -i '/#==Found/ c\echo "Found file '${S_LOG_FILE}'_${k}"' ${JOB}_Merge.sh
sed -i '/#==Copy/ a\cp -rfvp ${OUT_FILE} $NFS_PROJECT_AREA/' ${JOB}_Merge.sh
sed -i '/asdf/ c\  csc_mergeHIT_trf.py inputHitsFile='${INPUT_MERGE_FILES}' outputHitsFile='${S_OUTPUT_FILE}' maxEvents=-1
skipEvents=0 geometryVersion='${S_GEOMETRY_VERSION}' InputLogsFile='${INPUT_LOG_FILES}' > MERGE.out 2>&1'
${JOB}_Merge.sh

sed -i '/JOBNAME=/ c\JOBNAME='${JOBQ}" ${JOB}_Digi.sh
sed -i '/NFS_PROJECT_AREA=/ c\NFS_PROJECT_AREA='${DATAQ}" ${JOB}_Digi.sh
sed -i '/NFS_RESULTS_AREA=/ c\NFS_RESULTS_AREA='${RESULTSQ}" ${JOB}_Digi.sh
sed -i '/ATLAS_RELEASE=/ c\ATLAS_RELEASE='${D_ATHENA_RELEASEQ}" ${JOB}_Digi.sh
sed -i '/ATLAS_VERSION=/ c\ATLAS_VERSION='${D_ATHENA_VERSIONQ}" ${JOB}_Digi.sh
sed -i '/DBRELEASE=/ c\DBRELEASE='${D_DATABASEQ}" ${JOB}_Digi.sh
sed -i '/IN_FILE=/ c\IN_FILE='${M_OUTPUT_FILEQ}" ${JOB}_Digi.sh
sed -i '/OUT_FILE=/ c\OUT_FILE='${D_OUTPUT_FILEQ}" ${JOB}_Digi.sh
sed -i '/LOG=/ c\LOG='${D_LOG_FILEQ}" ${JOB}_Digi.sh
sed -i '/TIME=/ c\TIME='${D_TIME_LOGQ}" ${JOB}_Digi.sh
sed -i '/executable=/ c\executable='${DATA}'/'${JOB}_Digi.sh" ${JOB}_Digi.cmd
sed -i '/output=/ c\output='${RESULTS}'/'${JOB}_Digi.out" ${JOB}_Digi.cmd
sed -i '/error=/ c\error='${RESULTS}'/'${JOB}_Digi.err" ${JOB}_Digi.cmd
sed -i '/log=/ c\log='${RESULTS}'/'${JOB}_Digi.log" ${JOB}_Digi.cmd
sed -i '/YES/ a\when_to_transfer_output=ON_EXIT' ${JOB}_Digi.cmd
```

```
sed -i '/notify_user=/ c\notify_user='$USER'@pt3nfs.atlas.csufresno.edu' ${JOB}_Digi.cmd
sed -i '/#==EXTRAS/ a\cd $ATLAS_TEST_AREA' ${JOB}_Digi.sh
sed -i '/#==EXTRAS/ a\condor_submit '${DATA}'/'${JOB}_Reco.cmd" ${JOB}_Digi.sh
sed -i '/#==EXTRAS/ a\cd $HOME' ${JOB}_Digi.sh
sed -i '/#==Copy/ a\cp -rfvp ${OUT_FILE} $NFS_PROJECT_AREA/' ${JOB}_Digi.sh
sed -i '/asdf/ c\  '${D_COMMAND}' inputHitsFile='${S_OUTPUT_FILE}' outputRDOFile='${D_OUTPUT_FILE}' maxEvents=-1 skipEvents=0
geometryVersion='${D_GEOMETRY_VERSION}' '${D_OTHER_TAG_1}' '${D_OTHER_TAG_2}' '${D_OTHER_TAG_3}'
'${D_OTHER_TAG_4}' '${D_OTHER_TAG_5}' conditionsTag='${D_CONDITION_TAG}' > '${D_LOG_FILE}' 2>&1' ${JOB}_Digi.sh

sed -i '/JOBNAME=/ c\JOBNAME='${JOBQ}'" ${JOB}_Reco.sh
sed -i '/NFS_PROJECT_AREA=/ c\NFS_PROJECT_AREA='${DATAQ}'" ${JOB}_Reco.sh
sed -i '/NFS_RESULTS_AREA=/ c\NFS_RESULTS_AREA='${RESULTSQ}'" ${JOB}_Reco.sh
sed -i '/ATLAS_RELEASE=/ c\ATLAS_RELEASE='${R_ATHENA_RELEASEQ}'" ${JOB}_Reco.sh
sed -i '/ATLAS_VERSION=/ c\ATLAS_VERSION='${R_ATHENA_VERSIONQ}'" ${JOB}_Reco.sh
sed -i '/DBRELEASE=/ c\DBRELEASE='${R_DATABASEQ}'" ${JOB}_Reco.sh
sed -i '/IN_FILE=/ c\IN_FILE='${D_OUTPUT_FILEQ}'" ${JOB}_Reco.sh
sed -i '/OUT_FILE=/ c\OUT_FILE='${R_OUTPUT_FILEQ}'" ${JOB}_Reco.sh
sed -i '/LOG=/ c\LOG='${R_LOG_FILEQ}'" ${JOB}_Reco.sh
sed -i '/TIME=/ c\TIME='${R_TIME_LOGQ}'" ${JOB}_Reco.sh
sed -i '/executable=/ c\executable='${DATA}'/'${JOB}_Reco.sh" ${JOB}_Reco.cmd
sed -i '/output=/ c\output='${RESULTS}'/'${JOB}_Reco.out" ${JOB}_Reco.cmd
sed -i '/error=/ c\error='${RESULTS}'/'${JOB}_Reco.err" ${JOB}_Reco.cmd
sed -i '/log=/ c\log='${RESULTS}'/'${JOB}_Reco.log" ${JOB}_Reco.cmd
sed -i '/notify_user=/ c\notify_user='$USER'@pt3nfs.atlas.csufresno.edu' ${JOB}_Reco.cmd
sed -i '/YES/ a\when_to_transfer_output=ON_EXIT' ${JOB}_Reco.cmd
sed -i '/#==EXTRAS/ a\cd $ATLAS_TEST_AREA' ${JOB}_Reco.sh
sed -i '/#==EXTRAS/ a\condor_submit '${DATA}'/'${JOB}_AOD.cmd" ${JOB}_Reco.sh
sed -i '/#==EXTRAS/ a\cd $HOME' ${JOB}_Reco.sh
sed -i '/#==Copy/ a\cp -rfvp ${OUT_FILE} $NFS_PROJECT_AREA/' ${JOB}_Reco.sh
sed -i '/asdf/ c\  '${R_COMMAND}' inputRDOFile='${D_OUTPUT_FILE}' outputESDFile='${R_OUTPUT_FILE}' maxEvents=-1
skipEvents=0 geometryVersion='${R_GEOMETRY_VERSION}' '${R_OTHER_TAG_1}' '${R_OTHER_TAG_2}' '${R_OTHER_TAG_3}'
'${R_OTHER_TAG_4}' '${R_OTHER_TAG_5}' conditionsTag='${R_CONDITION_TAG}' > '${R_LOG_FILE}' 2>&1' ${JOB}_Reco.sh

sed -i '/JOBNAME=/ c\JOBNAME='${JOBQ}'" ${JOB}_AOD.sh
sed -i '/NFS_PROJECT_AREA=/ c\NFS_PROJECT_AREA='${DATAQ}'" ${JOB}_AOD.sh
sed -i '/NFS_RESULTS_AREA=/ c\NFS_RESULTS_AREA='${RESULTSQ}'" ${JOB}_AOD.sh
sed -i '/ATLAS_RELEASE=/ c\ATLAS_RELEASE='${R_ATHENA_RELEASEQ}'" ${JOB}_AOD.sh
sed -i '/ATLAS_VERSION=/ c\ATLAS_VERSION='${R_ATHENA_VERSIONQ}'" ${JOB}_AOD.sh
sed -i '/DBRELEASE=/ c\DBRELEASE='${R_DATABASEQ}'" ${JOB}_AOD.sh
sed -i '/IN_FILE=/ c\IN_FILE='${R_OUTPUT_FILEQ}'" ${JOB}_AOD.sh
sed -i '/OUT_FILE=/ c\OUT_FILE='${AOD_OUTPUT_FILEQ}'" ${JOB}_AOD.sh
sed -i '/LOG=/ c\LOG="'${JOB}_AOD.out'"" ${JOB}_AOD.sh
sed -i '/TIME=/ c\TIME="'${JOB}_AOD.time'"" ${JOB}_AOD.sh
sed -i '/executable=/ c\executable='${DATA}'/'${JOB}_AOD.sh" ${JOB}_AOD.cmd
sed -i '/output=/ c\output='${RESULTS}'/'${JOB}_AOD.out" ${JOB}_AOD.cmd
sed -i '/error=/ c\error='${RESULTS}'/'${JOB}_AOD.err" ${JOB}_AOD.cmd
sed -i '/log=/ c\log='${RESULTS}'/'${JOB}_AOD.log" ${JOB}_AOD.cmd
sed -i '/notify_user=/ c\notify_user='$USER'@pt3nfs.atlas.csufresno.edu' ${JOB}_AOD.cmd
sed -i '/YES/ a\when_to_transfer_output=ON_EXIT' ${JOB}_AOD.cmd
sed -i '/#==EXTRAS/ a\cp -rfvp D3PD.out $NFS_RESULTS_AREA/' ${JOB}_AOD.sh
sed -i '/#==EXTRAS/ a\cp -rvfp '${D3PD_OUTPUT_FILE}' $NFS_RESULTS_AREA/' ${JOB}_AOD.sh
sed -i '/#==EXTRAS/ a\athena.py '${JOB}_D3PD.py' > D3PD.out 2>&1' ${JOB}_AOD.sh
sed -i '/#==Copy/ a\cp -rfvp ${OUT_FILE} $NFS_PROJECT_AREA/' ${JOB}_AOD.sh
sed -i '/asdf/ c\  '${R_COMMAND}' inputESDFile='${R_OUTPUT_FILE}' outputAODFile='${AOD_OUTPUT_FILE}' maxEvents=-1
skipEvents=0 geometryVersion='${R_GEOMETRY_VERSION}' '${R_OTHER_TAG_1}' '${R_OTHER_TAG_2}' '${R_OTHER_TAG_3}'
'${R_OTHER_TAG_4}' '${R_OTHER_TAG_5}' conditionsTag='${R_CONDITION_TAG}' > 'AOD.out' 2>&1' ${JOB}_AOD.sh

sed -i '/InAodFile/ c\athenaCommonFlags.FilesInput=["""${AOD_OUTPUT_FILE}"""]' ${JOB}_D3PD.py
sed -i '/OutD3PDFile/ c\       tupleFileOutput = """${D3PD_OUTPUT_FILE}""" ${JOB}_D3PD.py

for ((i=0; i<${MAX}; i++))
do
        cd $DATA
        JOBX=${JOB}_${i}
        S_LOG_FILEX=${S_LOG_FILE}_${i}
        S_OUTPUT_FILEX=${S_OUTPUT_FILE}_${i}
        S_TIME_LOGX=${S_TIME_LOG}_${i}
        echo -n "" >> jobQnamelist
        echo -n "${JOB}_${i}" >> jobQnamelist
        echo -n "" >> jobQnamelist
        echo -n "" >> lognamelist
        echo -n "${S_LOG_FILE}_${i}" >> lognamelist
        echo -n "" >> lognamelist
```

```
        echo -n "" >> timelist
        echo -n "${S_TIME_LOG}_${i}" >> timelist
        echo -n "" >> timelist
        echo -n "" >> outputlist
        echo -n "${S_OUTPUT_FILE}_${i}" >> outputlist
        echo -n "" >> outputlist
        JOBQX="`cat jobQnamelist`"
        S_LOG_FILEQX="`cat lognamelist`"
        S_OUTPUT_FILEQX="`cat outputlist`"
        S_TIME_LOGQX="`cat timelist`"
        let SKP=${SW}*${i}
        rm jobQnamelist
        rm lognamelist
        rm timelist
        rm outputlist
        cp ./jobSim.sh ./${JOBX}_Sim.sh
        cp ./jobSubmit.cmd ./${JOBX}_Sim.cmd
        sed -i '/JOBNAME=/ c\JOBNAME='${JOBQX}'' ${JOBX}_Sim.sh
        sed -i '/NFS_PROJECT_AREA=/ c\NFS_PROJECT_AREA='${DATAQ}'' ${JOBX}_Sim.sh
        sed -i '/NFS_RESULTS_AREA=/ c\NFS_RESULTS_AREA='${RESULTSQ}'' ${JOBX}_Sim.sh
        sed -i '/ATLAS_RELEASE=/ c\ATLAS_RELEASE='${S_ATHENA_RELEASEQ}'' ${JOBX}_Sim.sh
        sed -i '/ATLAS_VERSION=/ c\ATLAS_VERSION='${S_ATHENA_VERSIONQ}'' ${JOBX}_Sim.sh
        sed -i '/DBRELEASE=/ c\DBRELEASE='${S_DATABASEQ}'' ${JOBX}_Sim.sh
        sed -i '/asdf/ c\  '${S_COMMAND}' inputEvgenFile='${S_INPUT_FILE}' outputHitsFile='${S_OUTPUT_FILEX}' maxEvents='${SW}'
skipEvents='${SKP}' randomSeed=$RANDOM geometryVersion='${S_GEOMETRY_VERSION}' '${S_OTHER_TAG_1}'
'${S_OTHER_TAG_2}' '${S_OTHER_TAG_3}' '${S_OTHER_TAG_4}' '${S_OTHER_TAG_5}' conditionsTag='${S_CONDITION_TAG}' >
'${S_LOG_FILEX}' 2>&1' ${JOBX}_Sim.sh
        sed -i '/IN_FILE=/ c\IN_FILE='${S_IN}'' ${JOBX}_Sim.sh
        sed -i '/OUT_FILE=/ c\OUT_FILE='${S_OUTPUT_FILEQX}'' ${JOBX}_Sim.sh
        sed -i '/LOG=/ c\LOG='${S_LOG_FILEQX}'' ${JOBX}_Sim.sh
        sed -i '/TIME=/ c\TIME='${S_TIME_LOGQX}'' ${JOBX}_Sim.sh
      sed -i '/#==Copy/ a\cp -rfvp ${LOG} $NFS_PROJECT_AREA/' ${JOBX}_Sim.sh
        sed -i '/#==Copy/ a\cp -rfvp ${OUT_FILE} $NFS_PROJECT_AREA/' ${JOBX}_Sim.sh
        if [ ${i} == ${LAST} ]
        then
                sed -i '/#==EXTRAS/ a\cd $ATLAS_TEST_AREA' ${JOBX}_Sim.sh
                sed -i '/#==EXTRAS/ a\condor_submit '${DATA}'/'${JOB}_Merge.cmd'' ${JOBX}_Sim.sh
                sed -i '/#==EXTRAS/ a\cd $HOME' ${JOBX}_Sim.sh
        fi
        sed -i '/executable=/ c\executable='${DATA}'/'${JOBX}_Sim.sh'' ${JOBX}_Sim.cmd
        sed -i '/output=/ c\output='${RESULTS}'/'${JOBX}_Sim.out'' ${JOBX}_Sim.cmd
        sed -i '/error=/ c\error='${RESULTS}'/'${JOBX}_Sim.err'' ${JOBX}_Sim.cmd
        sed -i '/log=/ c\log='${RESULTS}'/'${JOBX}_Sim.log'' ${JOBX}_Sim.cmd
        sed -i '/notify_user=/ c\notify_user='$USER'@pt3nfs.atlas.csufresno.edu' ${JOBX}_Sim.cmd
        sed -i '/YES/ a\when_to_transfer_output=ON_EXIT' ${JOBX}_Sim.cmd
        cd /home/$USER
        condor_submit ${DATA}/${JOBX}_Sim.cmd
done
```

The generic scripts jobCreator.sh relies on are jobSim.sh for simulation, jobMergeHits.sh for merging hit files after simulation, jobDigi.sh for digitization, jobReco.sh for reconstruction and producing AODs, jobD3PD.py for producing D3PDs, and jobSubmit.cmd for submitting jobs with Condor. Each one is listed below.

jobSim.sh

```
#!/bin/bash

JOBNAME=
export USER="$(id -nu)"
DATE="$(date | sed 's/:/ /g' | awk '{print $2$3"_"$4_$5_$6}')"
JOBID="$(echo $PWD | sed 's/\// /g' | awk '{print $NF}')"
LOCAL_SCRATCH="/disk/$USER"
NFS_PROJECT_AREA=
NFS_RESULTS_AREA=

#==Athena job info
export ATLAS_RELEASE=
export ATLAS_VERSION=
export DBRELEASE=
```

```
export IN_FILE=
export OUT_FILE=
export LOG=
export TIME=

#==need these for atlas setup scripts
export HOME="/home/${USER}"
export ATLAS_LOCAL_ROOT_BASE="/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase"
export LOCAL_JOB_AREA="${JOBNAME}_${DATE}_$JOBID"
export ATLAS_TEST_AREA="$LOCAL_SCRATCH/${LOCAL_JOB_AREA}/${ATLAS_RELEASE}"

#==set atlas environment
mkdir -p ${ATLAS_TEST_AREA}
cd  $ATLAS_TEST_AREA
source /cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase/user/atlasLocalSetup.sh
source /nfs/t3nfs/share/atlas/ATLASLocalRootBase/packageSetups/atlasLocalGccSetup.sh --gccVersion=gcc432_x86_64_slc5
source /cvmfs/atlas.cern.ch/repo/sw/software/i686-slc5-gcc43-opt/${ATLAS_RELEASE}/AtlasSetup/scripts/asetup.sh
${ATLAS_VERSION},here --dbrelease=${DBRELEASE}

#==prepare local job area
cp -rfvp $NFS_PROJECT_AREA/${IN_FILE} ./

#==Athena job
  export START=$(date +%s)
asdf
  export END=$(date +%s)
  export DIFF=$(( $END - $START))
  echo ${DIFF} >> ${TIME}

#==Copy files and clean up
env | sort
cp -rfvp ${TIME} $NFS_RESULTS_AREA/
cp -rfvp ${LOG} $NFS_RESULTS_AREA/
cp -rfvp ${OUT_FILE} $NFS_RESULTS_AREA/

#==Submit another job
#==EXTRAS

cd $LOCAL_SCRATCH
ls -d  $LOCAL_JOB_AREA
rm -rf $LOCAL_JOB_AREA
```

jobMergeHits.sh

```
#!/bin/bash

JOBNAME=
export USER="$(id -nu)"
echo $USER
DATE="$(date | sed 's/:/ /g' | awk '{print $2$3"_"$4_$5_$6}')"
JOBID="$(echo $PWD | sed 's/\// /g' | awk '{print $NF}')"
LOCAL_SCRATCH="/disk/$USER"
NFS_PROJECT_AREA=
NFS_RESULTS_AREA=

#==Athena job info
export ATLAS_RELEASE=
export ATLAS_VERSION=
export DBRELEASE=
export OUT_FILE=
export LOG=
export TIME=

#==need these for atlas setup scripts
export HOME="/home/${USER}"
export ATLAS_LOCAL_ROOT_BASE="/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase"
export LOCAL_JOB_AREA="${JOBNAME}_${DATE}_$JOBID"
export ATLAS_TEST_AREA="$LOCAL_SCRATCH/${LOCAL_JOB_AREA}/${ATLAS_RELEASE}"

#==set atlas environment
mkdir -p ${ATLAS_TEST_AREA}
cd  $ATLAS_TEST_AREA
source /cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase/user/atlasLocalSetup.sh > /dev/null
```

```
source /nfs/t3nfs/share/atlas/ATLASLocalRootBase/packageSetups/atlasLocalGccSetup.sh --gccVersion=gcc432_x86_64_slc5
source /cvmfs/atlas.cern.ch/repo/sw/software/i686-slc5-gcc43-opt/${ATLAS_RELEASE}/AtlasSetup/scripts/asetup.sh
${ATLAS_VERSION},here --dbrelease=${DBRELEASE}

#==Make sure all simulations are done before starting merge
ALL=
echo "Checking if all simulations have finished"
for ((k=0; k<${ALL}; k++))
do
#==Wait
                        sleep 60
                        echo "Waiting 1 minute for jobs to finish"
        done
#==Found
done
echo "Found all simulation files: Beginning merge"

#==prepare local job area
cp -rfvp $NFS_PROJECT_AREA/* ./

#==Athena job
  export START=$(date +%s)
asdf
  export END=$(date +%s)
  export DIFF=$(( $END - $START))
  echo ${DIFF} >> ${TIME}

#==Copy files and clean up
env | sort
cp -rfvp ${TIME} $NFS_RESULTS_AREA/
cp -rfvp ${LOG} $NFS_RESULTS_AREA/
cp -rfvp ${OUT_FILE} $NFS_RESULTS_AREA/

#==Submit another job
#==EXTRAS

cd $LOCAL_SCRATCH
ls -d  $LOCAL_JOB_AREA
rm -rf $LOCAL_JOB_AREA
```

jobDigi.sh

```
#!/bin/bash

JOBNAME=
export USER="$(id -nu)"
echo $USER
DATE="$(date | sed 's/:/ /g' | awk '{print $2$3"_"$4_$5_$6}')"
JOBID="$(echo $PWD | sed 's/\// /g' | awk '{print $NF}')"
LOCAL_SCRATCH="/disk/$USER"
NFS_PROJECT_AREA=
NFS_RESULTS_AREA=

#==Athena job info
export ATLAS_RELEASE=
export ATLAS_VERSION=
export DBRELEASE=
export IN_FILE=
export OUT_FILE=
export LOG=
export TIME=

#==need these for atlas setup scripts
export HOME="/home/${USER}"
export ATLAS_LOCAL_ROOT_BASE="/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase"
export LOCAL_JOB_AREA="${JOBNAME}_${DATE}_$JOBID"
export ATLAS_TEST_AREA="$LOCAL_SCRATCH/${LOCAL_JOB_AREA}/${ATLAS_RELEASE}"

#==set atlas environment
mkdir -p ${ATLAS_TEST_AREA}
cd  $ATLAS_TEST_AREA
source /cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase/user/atlasLocalSetup.sh > /dev/null
source /nfs/t3nfs/share/atlas/ATLASLocalRootBase/packageSetups/atlasLocalGccSetup.sh --gccVersion=gcc432_x86_64_slc5
```

```
source /cvmfs/atlas.cern.ch/repo/sw/software/i686-slc5-gcc43-opt/${ATLAS_RELEASE}/AtlasSetup/scripts/asetup.sh
${ATLAS_VERSION},here --dbrelease=${DBRELEASE}

#==prepare local job area
cp -rfvp $NFS_PROJECT_AREA/${IN_FILE} ./

#==Athena job
  export START=$(date +%s)
asdf
  export END=$(date +%s)
  export DIFF=$(( $END - $START))
  echo ${DIFF} >> ${TIME}

#==Copy files and clean up
env | sort
cp -rfvp ${TIME} $NFS_RESULTS_AREA/
cp -rfvp ${LOG} $NFS_RESULTS_AREA/
cp -rfvp ${OUT_FILE} $NFS_RESULTS_AREA/

#==EXTRAS

cd $LOCAL_SCRATCH
ls -d  $LOCAL_JOB_AREA
rm -rf $LOCAL_JOB_AREA
```

jobReco.sh

```
#!/bin/bash

JOBNAME=
export USER="$(id -nu)"
echo $USER
DATE="$(date | sed 's/:/ /g' | awk '{print $2$3"_"$4_$5_$6}')"
JOBID="$(echo $PWD | sed 's/\// /g' | awk '{print $NF}')"
LOCAL_SCRATCH="/disk/$USER"
NFS_PROJECT_AREA=
NFS_RESULTS_AREA=

#==Athena job info
export ATLAS_RELEASE=
export ATLAS_VERSION=
export DBRELEASE=
export IN_FILE=
export OUT_FILE=
export LOG=
export TIME=

#==need these for atlas setup scripts
export HOME="/home/${USER}"
export ATLAS_LOCAL_ROOT_BASE="/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase"
export LOCAL_JOB_AREA="${JOBNAME}_${DATE}_$JOBID"
export ATLAS_TEST_AREA="$LOCAL_SCRATCH/${LOCAL_JOB_AREA}/${ATLAS_RELEASE}"

#==set atlas environment
mkdir -p ${ATLAS_TEST_AREA}
cd  $ATLAS_TEST_AREA
source /cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase/user/atlasLocalSetup.sh > /dev/null
source /nfs/t3nfs/share/atlas/ATLASLocalRootBase/packageSetups/atlasLocalGccSetup.sh --gccVersion=gcc432_x86_64_slc5
source /cvmfs/atlas.cern.ch/repo/sw/software/i686-slc5-gcc43-opt/${ATLAS_RELEASE}/AtlasSetup/scripts/asetup.sh
${ATLAS_VERSION},here --dbrelease=${DBRELEASE}

#==prepare local job area
cp -rfvp $NFS_PROJECT_AREA/${IN_FILE} ./

#==Athena job
  export START=$(date +%s)
asdf
  export END=$(date +%s)
  export DIFF=$(( $END - $START))
  echo ${DIFF} >> ${TIME}

#==Copy files and clean up
env | sort
```

```
cp -rfvp ${TIME} $NFS_RESULTS_AREA/
cp -rfvp ${LOG} $NFS_RESULTS_AREA/
cp -rfvp ${OUT_FILE} $NFS_RESULTS_AREA/

#==Submit another job
#==EXTRAS

cd $LOCAL_SCRATCH
ls -d  $LOCAL_JOB_AREA
rm -rf $LOCAL_JOB_AREA
```

jobD3PD.py

```
# $Id$
#
# @file D3PDMakerConfig/share/AODToPhysicsD3PD.py
# @author scott snyder <snyder@bnl.gov>
# @date Aug, 2009
# @brief Example for building a physics D3PD from an AOD.
#


##############################################################################
# Define the input file here.
#

from AthenaCommon.AthenaCommonFlags import athenaCommonFlags
athenaCommonFlags.FilesInput= ["InAodFile"]

##############################################################################
# Define the output file here.
#

if not globals().get('tupleFileOutput'):
   tupleFileOutput = 'OutD3PDFile'

from D3PDMakerConfig.D3PDProdFlags import prodFlags
prodFlags.WritePhysicsD3PD = True
prodFlags.WritePhysicsD3PD.FileName = tupleFileOutput
prodFlags.WritePhysicsD3PD.lock()

##############################################################################
# Define other job options here.
#

athenaCommonFlags.EvtMax = -1

# Example of changing D3PD maker flags.
from D3PDMakerConfig.D3PDMakerFlags import D3PDMakerFlags
#D3PDMakerFlags.DoTrigger = False


##############################################################################
# Configure RecExCommon.
#

from RecExConfig.RecFlags import rec
rec.DPDMakerScripts.append( "D3PDMakerConfig/PhysicsD3PD_prodJobOFragment.py" )
rec.doCBNT.set_Value_and_Lock( False )
rec.doAOD.set_Value_and_Lock( False )
rec.doWriteTAG.set_Value_and_Lock( False )
rec.doWriteAOD.set_Value_and_Lock( False )
include ("RecExCommon/RecExCommon_topOptions.py")
```

jobSubmit.cmd

```
requirements            = (Arch == "INTEL"||Arch == "X86_64")&& \
                        ((Machine == "pt3wrk0.atlas.csufresno.edu")|| \
                         (Machine == "pt3wrk1.atlas.csufresno.edu")|| \
                         (Machine == "pt3wrk2.atlas.csufresno.edu")|| \
                         (Machine == "pt3wrk3.atlas.csufresno.edu")|| \
                         (Machine == "pt3wrk4.atlas.csufresno.edu"))
#
```

```
executable=
output=
error=
log=
universe=vanilla
should_transfer_files=YES
when_to_transfer_output=ON_EXIT
notify_user=
queue
```