Okla_SkimSlim Usage Information:

Use dq2-ls (or ls -d) to create a list datasets to use:

"dq2-ls data12.*SMWZ*_p1067/ > dq2_data12.list "

"dq2-ls MC12.*SMWZ*_p1067/ > dq2_data12.list "

Use a common text editor to reduce the list to the desired datasets.

Reduce again to unique wild card identifiers and add a label to distinguish data from MC:

Data12: Egamma Muons
MC12a: ggH500 VBFH500 WZ WW ZZ

The identifier will be used in the output dataset name if running on the grid, or in the output folder name if running locally. It will also be used with a wildcard specification when running or submitting jobs:

Generated a "slim_list", a list of branch names:

Example "get_leaflist.C"

```
{
  gROOT->ProcessLine(".L Okla_SkimSlim.C+");
  TChain t("HWWTree_lvqq");
  t.Add("HWWSkimmedNTUP_lvqq.root");
  if ( t.GetNtrees() < 1 ){
    cout << endl
         << "File HWWSkimmedNTUP_lvqq.root not found." << endl
         << "Please run in a directory that contains this file." << endl
         << "(or modify this script to point to a valid file)" << endl
         ;
  }
  else{
    Okla_SkimSlim::make_slim_list(&t,"full_list.txt");
    cout << "Now edit the file full_list.txt and remove unwanted branches." << endl
         << "copy the file to \"slim_list.txt\" to work with Okla_SkimSlim.C macro." << endl
         ;
  }
}
```

Edit the output file and reduce the list of branch names to a list that identifies the branches that you wish to keep in the output ntuple.

```
RunNumber
EventNumber
lbn
bcid
etc
```

Running locally:

Change directory to the “Okla_SkimSlim/code” subfolder.

Use “ls -d /path/to/my/folder/with/data/containers/in/it/* > data.list”.

Reduce the list as described above.

Create a branch name list for Data12 and for MC12a. Name the files “Data12_slim_list.txt” “MC12a_slim_list.txt” respectively.

Use the run script “. ./code.sh” to execute the “code_Okla_SkimSlim.C” macro.

Modify the script and/or macro to meet you specific needs.

Execute the script and find output in a subfolders under the ./output/*identifier*

Running on the grid

Change the directory to the “Okla_SkimSlim/prun” sub folder.

Use “dq2-ls”, as described above, to generated and reduce to a list of dataset identifiers.

Generate a “slim_list” for Data12 and MC12a, as described above, and place them in the Okla_SkimSlim/prun/code subfolder.

Use the script “setup.sh” to set up PanDA Run (prun).

Establish a voms proxy using your grid certificate.

From the prun folder, source the script test_nosubmit.sh to create a “tar-ball” to be used with prun submission.

Modify the script “test_submit.sh” to reflect your desired output dataset name. Also, embellish the input dataset name to accommodate the wild-card identifier names.

Use the script “bash_prun.sh” to spawn off the execution of the script “test_submit.sh”.

Example: “. ./bash_prun.sh data.list “cut0 cut1 cut2 cut3 cut4”

> Press enter when prompted, and repeat this a few times. Watch the first few grid jobs start and begin running using PanDA monitor. If convinced that there are no errors in the process, then continue to press enter until all jobs are submitted. There is no need to wait for one job to be submitted before starting the next one, so you can hold down the enter key until all jobs are submitted.

Notice *identifier*.out logfiles being created in the prun folder while the submission process is in progress.

> Use top to watch the multitude of processes involved in the submission process. While submission is in progress, and after it completes, you can examine the .out log files to make sure everything is working.

Monitor the jobs on the grid using PanDA.

When complete, down load the output ntuples using dq2-get.

Identify failed jobs and use PanDA Moniter to determine if the failure is related to a problem with a specific grid site.

> If so, then create a new “data.list” to target the failed jobs. A grid site exclusion parameter can be added to the prun command located in the test_submit.sh script. Repeat the execution of bash_prun.sh to retry the jobs on a new grid site.