1. How to submit jobs on IHEP cluster

For more details, see http://afsapply.ihep.ac.cn/cchelp/en/local-cluster/jobs/HTCondor/

1) Environment Setup

On IHEP cluster, users should use **HepJob** toolkit to manage jobs.

The **HepJob** toolkit is installed in the following directory.

```
/afs/ihep.ac.cn/soft/common/sysgroup/hep_job/bin
```

To make life easier, you could set the directory in your PATH environment variable and add this line to `.bash_profile` in your home directory:

```
$ export PATH=/afs/ihep.ac.cn/soft/common/sysgroup/hep_job/bin:$PATH
```

If you don't have `.bash_profile` in your home directory, simply create one by:

```
$ touch .bash_profile
```

and paste that command into the newly created `.bash_profile` file.

2) Job Submission

After 1) is done, logout and login again, then simply run:

```
$ hep_sub job.sh
```

to submit your job.

Make sure your job script `job.sh` is excutable.

3) Job Querying

Use the following command to query your jobs.

```
$ hep_q -u
```

4) Job Removing Use

```
$ hep_rm JobID
```

For example, with the command below

```
$ hep_rm 3745232 3745233.0
```

all jobs with clusterid 3745232, and the job with JobID 3745233.0 will be removed at the same time.

Use the command below to remove all your jobs

```
$ hep_rm -a
```

For more information, please see http://afsapply.ihep.ac.cn/cchelp/en/local-cluster/jobs/HTCondor/

2. Analysis examples of cepcsoft

The example I gave you last week might be too complicated for us as beginners. So I asked around and managed to find the analysis examples on http://cepcsoft.ihep.ac.cn/guides/Example/docs/introduction/. I put them on IHEP cluster, you can find them here:

```
/cefs/higgs/wangshudong/cepcsoft_AnaExamples
```

You will find: `ee_zz_vvqq-master.zip` `llH_Rec-master.zip` `validation-master.zip` `vvHgg-master.zip` `vvHmumu-master.zip` `vvHWW-master.zip`, you can use `unzip` command to unzip them:

```
unzip xxx.zip
```

You can find `README.md` file in each directory after you unzip those files above, you could learn how to run those examples with these `README.md` files.

The data is not included in those `.zip` file, you can find the data(`.slcio` file) in:

```
/cefs/data/DstData/CEPC240/CEPC_v4_update
```

For instance, for `vvHmumu` analysis, data files are stored here:

```
/cefs/data/DstData/CEPC240/CEPC_v4_update/higgs/E240.Pnnh_e2e2.e0.p0.whizard195/
```

To setup cepcsoft environment (cepcsoft chain), you can use `cepcenv.sh` in:

```
/cefs/higgs/wangshudong/cepcsoft_AnaExamples/
```

Just copy it to your directory and `source` it:

```
$ source <path/to/your/directory>/cepcenv.sh
```

Then follow `## Build and Install` in `README.md` in `vvHmumu-master`, in `vvHmumu-master` directory do:

```
$ mkdir build

$ cd build

$ cmake ..

$ make install
```

After doing this, follow `## Make the plots` in `README.md` step by step, when it comes to editing `vvHmumu.xml`, open `vvHmumu.xml` and edit it like(notice line 12 and line 24, (e2 means muon, n means neutrino,)):

```xml
<?xml version="1.0" encoding="us-ascii"?>
<!-- ?xml-stylesheet type="text/xsl" href="http://ilcsoft.desy.de/marlin/marlin.xsl"? -->
<!-- ?xml-stylesheet type="text/xsl" href="marlin.xsl"? -->

<marlin xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://ilcsoft.desy.de/marlin/marlin.xsd">
<execute>
   <processor name="vvHmumu"/>
</execute>

<global>
    <parameter name="LCIOInputFiles" type="string">
    /cefs/data/DstData/CEPC240/CEPC_v4_update/higgs/E240.Pnnh_e2e2.e0.p0.whizard195/nnh_e2e2.e0.p0.00001_000000_dst.slcio
    /cefs/data/DstData/CEPC240/CEPC_v4_update/higgs/E240.Pnnh_e2e2.e0.p0.whizard195/nnh_e2e2.e0.p0.00001_001000_dst.slcio
    /cefs/data/DstData/CEPC240/CEPC_v4_update/higgs/E240.Pnnh_e2e2.e0.p0.whizard195/nnh_e2e2.e0.p0.00002_000000_dst.slcio
    /cefs/data/DstData/CEPC240/CEPC_v4_update/higgs/E240.Pnnh_e2e2.e0.p0.whizard195/nnh_e2e2.e0.p0.00002_001000_dst.slcio
    /cefs/data/DstData/CEPC240/CEPC_v4_update/higgs/E240.Pnnh_e2e2.e0.p0.whizard195/nnh_e2e2.e0.p0.00003_000000_dst.slcio
    /cefs/data/DstData/CEPC240/CEPC_v4_update/higgs/E240.Pnnh_e2e2.e0.p0.whizard195/nnh_e2e2.e0.p0.00002_001000_dst.slcio
    /cefs/data/DstData/CEPC240/CEPC_v4_update/higgs/E240.Pnnh_e2e2.e0.p0.whizard195/nnh_e2e2.e0.p0.00004_000000_dst.slcio
    /cefs/data/DstData/CEPC240/CEPC_v4_update/higgs/E240.Pnnh_e2e2.e0.p0.whizard195/nnh_e2e2.e0.p0.00004_001000_dst.slcio
    /cefs/data/DstData/CEPC240/CEPC_v4_update/higgs/E240.Pnnh_e2e2.e0.p0.whizard195/nnh_e2e2.e0.p0.00005_000000_dst.slcio
    /cefs/data/DstData/CEPC240/CEPC_v4_update/higgs/E240.Pnnh_e2e2.e0.p0.whizard195/nnh_e2e2.e0.p0.00005_001000_dst.slcio
    </parameter>
    <!-- limit the number of processed records (run+evt): -->
    <parameter name="GearXMLFile"> /besfs5/groups/higgs/data/SimReco/wo_BS/CEPC_v4/GearOutput.xml </parameter>
    <parameter name="MaxRecordNumber" value="-1" />
    <parameter name="SkipNEvents" value="-1" />
    <parameter name="SupressCheck" value="false" />
    <!--parameter name="RandomSeed" value="1234567890" /-->
    <parameter name="PrintEventNumber" type="int" value="0" /> <!-- 0 for not printing event number, n for printing every n events -->
    <parameter name="Verbosity" options="DEBUG0-4,MESSAGE0-4,WARNING0-4,ERROR0-4,SILENT"> MESSAGE </parameter>
</global>


<processor name="vvHmumu" type="vvHmumu">
    <parameter name="InputPandoraPFOsCollection" type="string" lcioInType="ReconstructedParticle"> ArborPFOs  </parameter>
    <parameter name="Output" type="string"> vvHmumu.root  </parameter>
</processor>
```

Figure 1: vvHmumu.xml

Then continue with the remaining steps in `## Build and Install` in `README.md`.

3. Top quark electroweak coupling precision measurement

3.1 Samples

Location of signal samples:

```
/cefs/higgs/wangshudong/top_coupling/reconstruction/output/signal/sm_em
ep_ttbar_*/DST_DATA
```

There are 4 categories :

```
sm_emep_ttbar_hh
sm_emep_ttbar_ll
sm_emep_ttbar_sl_m
sm_emep_ttbar_sl_p
```

Here, `hh` means full hadronic channel, `ll` means full leptonic channel, `sl_p` means W^+ from t quark decay to charged lepton and neutrino and `sl_m` means W^- from $\bar{t}$ quark decay to charged lepton and neutrino

NB: Ideally, there should be 2000 events in each file for ll channel and 1000 events for others. But there are files which contain events way less than 2000/1000 (e.g. $\sim$ 400). This is due to the bugs of reconstruction software and no one wants to fix it. The existing events are reliable, so we don't need to worry too much about this.

Location of background samples: (not generated yet)

3.2 Some useful commands

After activating cepcsoft environment, you can use a command, `anajob`, to check the contents of the data:

```
$ anajob sm_emep_ttbar_ll_02_004000_dst.slcio
```

It can tell you the collections in each file:

COLLECTION NAME	COLLECTION TYPE	NUMBER OF ELEMENTS
AncientPFOs	ReconstructedParticle	88
ArborCharged	Cluster	35
ArborNeutral	Cluster	51
ArborPFOs	ReconstructedParticle	88
CluAB_1st	Cluster	129
ClupatraTrackSegments	Track	159
ClupatraTracks	Track	61
ClusterChargedCore	Cluster	31
EHBushes	Cluster	206
ForwardTracks	Track	0
LSBranches	Cluster	1441
MCParticle	MCParticle	1139
MCParticlesSkimmed	MCParticle	198
MarlinTrkTracks	Track	43
RecoMCTruthLink	LCRelation	86
SiTracks	Track	28
SubsetTracks	Track	27

Another command is `lcio_event_counter`, it can tell you the number of events in each file, just do:

```
$ lcio_event_counter sm_emep_ttbar_ll_02_004000_dst.slcio
```

3.3 Processing the reconstructed data

Our analysis start from ParticleFlowObject(PFO), we need to derive some physics objects (leptons, jets...) from PFOs. We need to use `Marlin` and `FSClasser` package to achieve this.

`Marlin` is a analysis framework, it can define the processors (i.e. modules that analyse data) and their order that are executed at runtime in a simple steering file. You can check Marlin to get more information. The `FSClasser` package is composed of many processors, you can copy it from my directory:

```
/workfs2/bes/wangshudong/cepc/FSClasser
```

then activate `cepcsoft` environment by `source load_cepcsoft.sh` and compile it:

```
$ mkdir build
```

```
$ cd build

$ cmake ..

$ make install
```

In this package, the main processors we need to use are `IsolatedLeptonFinderProcessor` and `FSClasserProcessor`. You can dig these two processors a little bit since we might need to modify them.

Moving to our physics problem, considering the semi-leptonic final state, there will be 1 charged lepton, 2 b-jets, and 2 quark jets decayed from W boson. So the first thing we need to do is find that charged lepton from all PFOs using `IsolatedLeptonFinderProcessor`, then we need to cluster all the remain PFOs into four jets and tag their flavors using `LcfiplusProcessor`. The source code of `LcfiplusProcessor` is located in

```
/cvmfs/cepc.ihep.ac.cn/software/cepcsoft/x86_64-centos7-gcc49/cepcsoft/
0.1.1/Reconstruction/HighLevelObjectFinding/Jets/LCFIPlus/00-05-02
```

After doing so, we can dump all these objects into `root` files and start our analysis. To achieve this, we use `Marlin`. As mentioned above, we need to write a steering file to tell `Marlin` the processors (and their order) that need to be excuted and define named parameters for every processor as well as for the global scope.

You can find an example steering file `uuh.xml` in:

```
/cefs/higgs/wangshudong/top_coupling/Analysis
```

This file is written for $\mu\mu h$ analysis so you need to modify it to apply it to our scenario.

The `.xml` steering file is organised as follows:

The first part controls the processors and their order that are executed:

```
<execute>
  <processor name="RootFileProcessor"/>
  <processor name="MyISOlatedLeptonFinderProcessor"/>
  <processor name="VertexFinder"/>
  <processor name="JetClusteringAndFlavorTag"/>
  <processor name="FSClasserProcessor"/>
</execute>
```

The second part defines all the global parameters

```
<global>
  <parameter name="LCIOInputFiles">
  ${FILE1}
  ${FILE2}

  </parameter>
  <parameter name="MaxRecordNumber" value="100001" />  <!-- should be (number of output events you want +1) -->
  <parameter name="SkipNEvents" value="0" />
  <parameter name="SupressCheck" value="false" />
  <parameter name="GearXMLFile"> /cefs/higgs/wangshudong/top_coupling/simulation/GearOutput.xml </parameter>
  <parameter name="RandomSeed" value="1234567890" />
  <parameter name="PrintEventNumber" type="int" value="0" /> <!-- 0 for not printing event number, n for printing every n events -->
  <parameter name="Verbosity" options="DEBUG0-4,MESSAGE0-4,WARNING0-4,ERROR0-4,SILENT"> WARNING </parameter>
</global>

<processor name="RootFileProcessor" type="RootFileProcessor">
  <parameter name="OutputRootFile" type="string"> ${OUT} </parameter>
</processor>
```

And the rest of the file define parameters for every processor:

```
<processor name="RootFileProcessor" type="RootFileProcessor">
  <parameter name="OutputRootFile" type="string"> ${OUT} </parameter>
</processor>

<processor name="MyISOlatedLeptonFinderProcessor" type="LGISOlatedLeptonFinderProcessor">
    <!--Input collection of ReconstructedParticles-->
    <parameter name="InputCollection" type="string" lcioInType="ReconstructedParticle">ArborPFOs </parameter>
    <!--Output collection of isolated leptons-->
    <parameter name="OutputCollectionIsolatedLeptons" type="string" lcioOutType="ReconstructedParticle">IsoLeps </parameter>
    <!--Copy of input collection but without the isolated leptons-->
    <parameter name="OutputCollectionWithoutIsolatedLepton" type="string" lcioOutType="ReconstructedParticle">RemainPFOs </parameter>
    <!--Output collection of isolated leptons from Z -->
    <parameter name="OutputCollectionIsolatedZLeptons" type="string" lcioOutType="ReconstructedParticle">IsoZLeps </parameter>
    <!--Copy of input collection but without the isolated leptons of Z-->
    <parameter name="OutputCollectionWithoutIsolatedZLepton" type="string" lcioOutType="ReconstructedParticle">RemainZPFOs </parameter>
```

You can find the definitions in the corresponding processor's source file.

After finish writing this steering file (assume its name is `top.xml`), you can finally process events and dump them into root file by:

```
$ Marlin top.xml
```

Before excute `Marlin` command, remember to excute `load_cepcsoft.sh` to activate `cepcenv` and add two lines:

```
source /cvmfs/cepc.ihep.ac.cn/software/cepcenv/setup.sh
cepcenv use 0.1.1
export FSClasser_HOME=/workfs2/bes/lig/higgs/FSClasser
export MARLIN_DLL=${MARLIN_DLL}:$FSClasser_HOME/lib/libFSClasser.so
```

Line 3 and line 4 should direct to your own `FSClasser` package

NB: If you want to process a large amount of events, you should submit a job to the cluster.

NB: The current `IsolatedLeptonFinderProcessor` might not work very well in our scenario. If that's the case, we need to modify this processor.

3.4 Visualize detector geometry and event using Druid

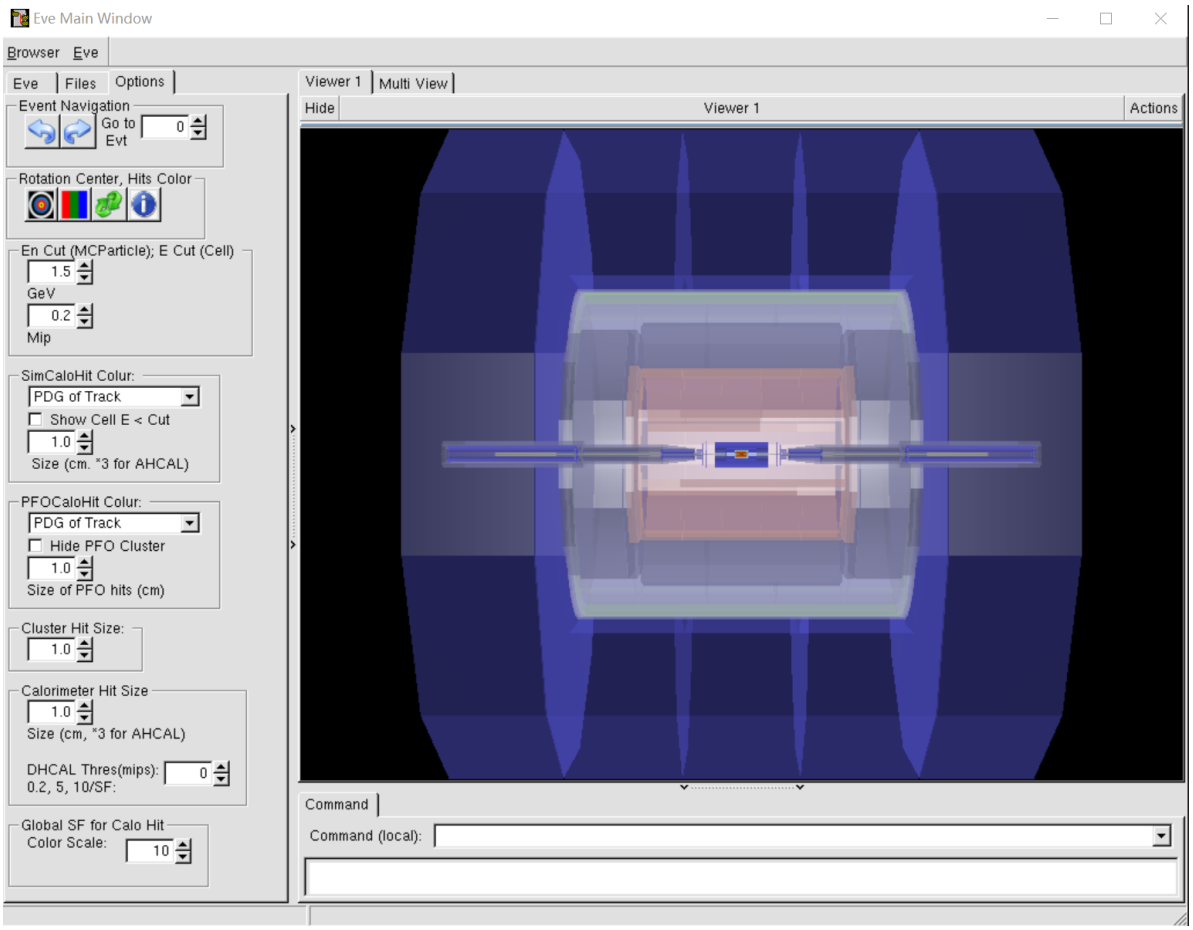
To visualize detector geometry(CEPC_v4) and particle generation, you can use `Druid` command. First, copy my directory:

```
/cefs/higgs/wangshudong/top_coupling/Vis
```

You can find a file named `geometry.root`. Then after loading cepcenv, just type:

```
$ Druid geometry.root
```

You can see the geometry of CEPC_v4:



To visualize events in parallel, just type:

```
$ Druid xxx.slcio geometry.root
```

`xxx.slcio` can be the sample files described in section 3.1, but instead of files under `*/REC_DATA`, should use files under `*/DST_DATA` which have more information.

(2023-04-19: Currently, when trying to visualize events in `*/DST_DATA`, you will see a warning *Detector type from gdml root file and slcio data file got conflict! Pls check! Will not make a geometry projection* and the program will crash a few seconds later, I tried to figure out why but I haven't)

3.5 Calculate the efficiency of lepton isolation

I can't tell you exactly how to do this because I don't know either, Prof.Gang Li just told

me a possible way to achieve this:

After lepton isolation and jet clustering etc., based on the root file you got, reconstruct the mass of top quark which its daughter W boson decayed hadronically. See whether the mass is around 173 GeV.

Update(2023-04-28): I consulted Gang about how to pick the lepton out today and he thinks that there are two ways to do the whole thing.

1. stick with using the current FSClasser package: using MC truth information to draw the kinematics distribution of lepton from W and lepton from other particles (e.g. a hadron which belong to a jet), then decide the cut. You can get the MC truth information from the collection "MCParticle", and in line 108 of FSClasserProcessor.cc you can find a parameter called "showMC", figure out how this parameter works and set it properly in the xml steering file.

But personally, for the first step I think may be we can just select the isolated lepton with the highest momentum and regard it as the lepton from W and check the efficiency.

2. Give up using this FSClasser package and write a processor yourself, a highly customized processor that you can finish all tasks needed. In this processor, after finishing the lepton isolation, we can get a vector containing all the isolated leptons, writing a loop, in a loop select one and regard it as the lepton from W then do all the reamining things : vertices finding, jet clustering, jet flavor taging and calculating the top quark mass, then enter the next loop and pick the next lepton in the vector as lepton from W and repeat..... After the loop, compare which calculated top mass is closest to the real top mass and take the corresponding isolated lepton as the lepton from W.