trigger studies

https://github.com/marco-link/trigger-studies

Marco Link

February 22, 2018

Contents

1 Prerequisites

- git
- CERN computing account
- GRID certificate (setup ready to use)
- write permission on any storage site
- python3 (tested with 3.5, scipt to setup an environment with the needed packages is included)
- doxygen (optional to generate documentation)
- LaTeX (optional, but highly recommended)

2 Setting up the analyzer

2.1 Setup the CMSSW framework

Setup a new CMSSW framework with the following commands:

```
source /cvmfs/cms.cern.ch/cmsset_default.sh
mkdir trigger_studies
cd trigger_studies

cmsrel CMSSW_9_3_0
cd CMSSW_9_3_0/src/
cmsenv

git clone https://github.com/marco-link/trigger-studies
mv trigger-studies/analyzer/ analyzer
scram b
```

2.2 Setup the analyzer

```
source /cvmfs/cms.cern.ch/cmsset_default.sh
cd <your CMSSW_9_3_0 folder>/src/analyzer
cmsenv
```

In *process_data.py* you can enable and disable some preselections. For more control over the preselections you can edit *TriggerAnalyzer/plugins/TriggerAnalyzer.cc*. After editing, run a test with:

```
scram b
cmsRun process_data.py
```

This starts a run over 10 000 Run2017B JetHT events. This can be changed in *process_data.py*.

2.3 Send task to GRID

```
source /cvmfs/cms.cern.ch/cmsset_default.sh
source /cvmfs/cms.cern.ch/crab3/crab.sh
voms-proxy-init -voms cms --valid 200:00
cd <your CMSSW_9_3_0 folder>/src/trigger_studies/analyzer
cmsenv
```

Now edit *crab.py* to fit your dataset, JSON-file and storageSite. Your dataset can be taken from the DAS (`https://cmsweb.cern.ch/das/`). For more details on the crab-config file see:
`https://twiki.cern.ch/twiki/bin/view/CMSPublic/CRAB3ConfigurationFile`

Then submit the task with:

```
crab submit -c crab.py
```

you can check the status of your submitted task with:

```
crab status
```

after some time the task also should show up at the Task Monitoring:
`http://dashb-cms-job.cern.ch/dashboard/templates/task-analysis`

if some jobs of your task failed, you can resubmit them:

```
crab resubmit crab_projects/<taskname>
```

kill your task with:

```
crab kill crab_projects/<taskname>
```

for more details on CRAB3 commands see:
`https://twiki.cern.ch/twiki/bin/view/CMSPublic/CRAB3Commands`

2.4 Get the data

After your tasks are finished, you need to get the data from your storage site. You can use a FTP-Client or gfal (`https://dmc.web.cern.ch/projects/gfal-2/documentation`).

3 Generate a basic trigger report

In this chapter *<your report folder>* corresponds to *<your CMSSW_9_3_0 folder>/src/trigger-studies/report*.

3.1 Get luminosity data

To calculate the luminosity of your datasets, you need to generate a file containing the luminosity for the runs defined by the JSON-file.

First we need to setup a environment like described here:
`https://cms-service-lumi.web.cern.ch/cms-service-lumi/brilwsdoc.html`

```
1 ssh <your lxplus username>@lxplus.cern.ch
2 export
     PATH=$HOME/.local/bin:/afs/cern.ch/cms/lumi/brilconda-1.1.7/bin:$PATH
     (bash)
3 pip install --install-option="--prefix=$HOME/.local" brilws
```

Then generate the luminosity file for your JSON file with:

```
1 brilcalc lumi -i <path to your JSON.txt> -o lumi.csv
```

Move *lumi.csv* to *<your report folder>/data* and remove the # second line (the line beginning with run:fill,).

3.2 Merging the data

After generating the needed data like described in chapter 2, you get a *fakeroot_csv*.root*-file for each job in the task. Now you can merge this files into a single one. Therefore

move the *fakeroot_csv*.root*-files into *<your report folder>/IN/crab/<task name>*. If needed rename the dataset in *packer.py*.

For merging then run:

```
python3 packer.py
```

After that place the **.csv*-files in *<your report folder>/IN*. They are automatically read in with the next generation of a report that loads the dataset. Therefore it is important for the **.csv*-file to be named after the dataset. If there were no conflicts with the readin process, the files are then moved to *<your report folder>/data/raw/*. Be sure to give it a unique name or otherwise it will override other files in *<your report folder>/data/raw/*, when moved.

3.3 Generate trigger report

Run the *install_dependencies.sh* script to generate a python environment with the needed packages. Then activate the environment.

```
./install_dependencies.sh
source py_venv/bin/activate
```

To generate a trigger report you can alter the working example *SingleMuon.py* to your needs. Then run it with:

```
python3 SingleMuon.py
```

Also have a look at the documentation that can be generated by running:

```
doxygen Doxyfile
```

or look at the already generated *documentation.pdf*.