Basic description and usage of the automatized crab job submission script for the IBs.

IDEA

Run a script triggered by a cronjob, which sends some 'standard' CMSSW jobs on real data to the grid.

OPERATION

The files are located here: http://cmssw.cvs.cern.ch/cgi-bin/cmssw.cgi/UserCode/bbozsogi/CRABCAF/

The cronjob starts `Tester/standaloneTester-caf.py` every 12h. Next to the usual statusFile (containing the time it started, finished), I set up a global statusFile (***now it's in my AFS area***), preventing to start new CAF jobs before the latest has finished.

important:
For the script to run, cmssw and grid environmental variables has to be set (done by standaloneTester).
For the IB runs, we have to use custom CRAB build, since they're not supported by default. I set up one on vocms101 (/build/bbozsogi/CRAB_2_7_5), currently the script runs on this machine.

Basically the script:

- creates a crabCAF directory
- checks out http://cmssw.cvs.cern.ch/cgi-bin/cmssw.cgi/CMSSW/Configuration/PyReleaseValidation/data/cmsDriver_star
- searches for jobs meant to be run on the caf
- sets up directories for each of them
- runs cmsDriver.py with --no_exec option
- creates the crab.cfg file
- send the jobs
- waits for the results
- copy the logs and a pickled dictionary with some basic parsed info about the jobs to the IB's AFS area, under *cafQAlogs//*
- sends a mail containing all the output from crab and the parsed info (crabCAF.log)

USAGE

I'll try to keep the `crabRun.py --help` up-to-date 🙂
By running the script without parameters, it does the things listed above.

You can change the default crab.cfg parameters (see CafQADefaults) by:

```
crabRun.py -c 'USER:email=bbozsogi@cern.ch,CMSSW:events_per_job=2000'
```

You can send custom jobs to the CAF by using option `-inputCmd='cmsDriver.py ...' -d=''` .
In this case it runs the cmsDriver.py with -no_exec option and use that config file as an input. For this, you also have to

```
crabRun.py -d '/Electron/Run2010B-WZEG-v2/RAW-RECO' -i 'cmsDriver.py step2 -s RAW2DIGI,L1Reco,REC
--data --datatier RECO --eventcontent RECO --conditions auto:com10 --scenario pp --no_exec --magF
--process reRECO --customise Configuration/DataProcessing/RecoTLR.py --cust_function customisePPD
```

PICKLE

The pickle file contains a dictionary(`logData`) with the most important infos about the job.
it has a dictionary for every logFile in a stucture like:

```
logData[logName] = {
   'JobExitCode' : exitcode
   'files' : ['file1.root', 'file2.root', ...]
   'SE_PATH' : '/castor/.../'
   'MSG' : [((startNum, endNum), 'errorMessage'), ...]
   'TotalMSG' : {MSGType: Number_of_Errors, MSGType2: Number_of_errors...}
}
```

The 'MSG' part is the most important, it's basically every block in the logs between `%MSG` tags.

A way to parse it (used it for the crabCAF.log):

```
        for key, value in logData.items():
            self.log.write(key+': {\n')
            for key2, value2 in value.items():
                if key2 == 'MSG':
                    map(lambda x: self.log.write('---'+key2+': '+'('+str(x[0][0])+'-'+str(x[0][1]
                elif key2 == 'files':
                    self.log.write('---'+key2+': '+', '.join(value2).split()[0])
                    self.log.write('\n')
                elif key2 == 'TotalMSG':
                    for key3, value3 in value2.items():
                        self.log.write('---Total number of '+key3+': '+str(value3)+'\n')
                else:
                    self.log.write('---'+key2+': '+value2)
                    self.log.write('\n')
            self.log.write('}\n')
            self.log.flush()
```

ISSUES

Now, the cronjob is running under my user. We have to port it at some point to cmsbuild maybe.
Some more error protection...

-- BalazsBozsogi - 17-Nov-2010

This topic: Sandbox > CafQA
Topic revision: r2 - 2010-12-09 - BalazsBozsogi