FOCAL analysis on grid

Ionut-Cristian Arsene (Oslo),
01/07/2022
Updated on 26/01/2023

Overview

- Have a working grid user account
- Have the FOCAL package installed and loaded in your environment
- Copy the scripts and JDL file from $FOCAL/aliroot/macros/ in a local working directory
  - analyzeFOCAL.jdl, copyFromGrid.sh, RunFOCALanalysis.sh
- Create your analysis macros according to description, or use the provided ones as examples (ClusterizeGrid.C, AnalyzeJpsiGrid.C) → **see slide 3**
- Modify the analyzeFOCAL.jdl file according to your needs → **see slide 4**
- Create a working directory on the grid
- Copy to the grid working directory the JDL, RunFOCALanalysis.sh, your analysis macro(s) and whatever other files required (e.g. geometry.txt, clustering parameters file)
- Create a collection of simulation input files to be analyzed → **see slide 5**
- Submit the grid job → **see slide 6**
- Monitor progress on Monalisa → **see slide 7**
- When your jobs are done, use the copyFromGrid.sh script to download the individual job outputs → **see slide 8**
- Instructions on how to run analysis with embedding, on the grid → **see slide 9,10,11**

Create an analysis macro

- The analysis jobs will be run on a worker node on the grid where all the needed files will exist in the same directory:
  - Input file: root_archive.zip containing galice.root, Kinematics.root and Hits files
  - Analysis macros
  - Other input files, e.g. geometry.txt, parameters.txt (input params for the clusterizer)
- Consequently, the analysis macros should assume the galice.root file is in the current directory (see example AnalyzeJpsiGrid.C)
- The main function in the macro has to have the same name as the macro, because it will be run like this: “aliroot -b -q -x MyMacro.C”

Modify the analyzeFOCAL.jdl

- The jdl files are files containing info required to submit jobs on the grid
- Open the provided example jdl file and modify it according to your needs (follow the comments inside the file for each required information)
- As in bash scripts, $1, $2, $3… refer to command line arguments provided to the jdl, so $1 is the first argument, $2 is the second argument, etc.
- The existing info in the provided example jdl is running first the clusterization, followed by a jpsi analysis. The clusterization macro is called from inside the jpsi analysis macro.

Create a collection of input files to be analyzed

- Collections can be created using the **find** command on alien, e.g.:
  - Go to alien: alien.py
  - find /alice/cern.ch/user/f/focal/sim/v1.1_root6/pythiaMBtrig/*/root_archive.zip -x collection.xml
- You can also edit the collection.xml file and remove some input files if you want to run on just a subsample
- You can also use already existing collections, typically created together with the simulation samples

Submit jobs on the grid

- In your alien working directory, run the **submit** command:
- submit analyzeFOCAL.jdl <inputCollection.xml> </your/grid/working/directory> <outputSubDir> <MyAnalysisMacro.C>
- <inputCollection.xml>: the input XML collection
- </your/grid/working/directory>: your working directory on the grid, where all the previously mentioned files should be present. Also, the outputs will be copied in a subdirectory in this same directory
- <outputSubDir>: output subdirectory which will contain the job outputs
- <MyAnalysisMacro.C>: The main analysis macro

Monitor job progress on grid

- After you submited your master job, you can go to the ALICE Monalisa page (https://alimonitor.cern.ch/users/jobs.jsp) and monitor the job progress
  - This can be found in the “My jobs” tab
- If you click on the master job number in the “PID” column, you then get a window with the status of the different subjobs. You can check their status, resubmit the failed ones, check logs, access the output directories, etc.

Copy the outputs from grid to your local machine

- You can use the copyFromGrid.sh script to copy the outputs using multiple threads:
  - ./copyFromGrid.sh </target/grid/directory/> </destination/local/directory> <filename> <n-threads>
  - <filename>: name of the file you want to copy
  - <n-threads>: number of threads to be used for copying (e.g. 20-30 threads are efficient when copying many small files)
- Note: if the copy script is run multiple times with the same destination directory, the already existing files will be skipped

Run an analysis on embedded MC samples

- The main purpose of this is to do embedding of special MC processes or particles into Pb-Pb events, but it can do embedding for any generic FOCAL MC production into any other FOCAL production
  - The point is to avoid simulating too many Pb-Pb events which are expensive as both CPU and disk usage
    - One Pb-Pb event may be re-used multiple times for embedding
  - It is done by summing the digits of two events selected from the two productions and then running the clusterization which produces clusters as if this is a single event
- The embedding analysis can be done either locally (own computer or local farm) or on the grid. In the next slide a description of running over grid is given

Run an embedding analysis on the grid (1)

- Pre-requisites:
  - scripts and macros (see below) which can be found in FOCAL/aliroot/macros/
  - Your analysis macro (see below for special requirements)
  - Valid alien token
  - Two identified datasets to be used, one as a signal and one as background
- **submitGridJobs.py**
  - Python macro that will steer the uploading of necessary files to grid and job submission
  - Look inside and modify the paths corresponding to you for the `jdlExecutable, gridHomeDir, gridWorkDir`. Everything else can be modified via command line parameters
- **analyzeFOCALWithEmbedding.jdl**
  - JDL file specially prepared for embedding (REMEMBER to change the “user” variable to your own grid user)
- **RunFOCALanalysisWithEmbedding.sh**
  - Shell executable which is needed by the JDL (it copies the designated background data file to the worker node and launches the analysis)
- **ClusterizeWithEmbedingGrid.C**
  - Similar to ClusterizeGrid.C, but in addition provides the background hits to the digitizer. The background event is selected randomly from the background file. The background file is assumed to be in the same directory, in `bkg/root_archive.zip`
- **AnalysisQAGrid.C**
  - Example macro which can run on grid (it must be similar to what is described at **slide3**, with the exception that the main function must have one integer parameter used to transmit the number of events in a given bkg data file). The ClusterizeWithEmbeddingGrid.C macro is called within this macro, please pay attention that a parameter is being used to call it, e.g. **`.x AnalysisQAGrid.C(10)`**

Run an embedding analysis on the grid (2)

- Syntax:
  - `./submitGridJobs.py analyzeFOCALWithEmbedding.jdl <output> <geometry-file> <parameters-file> <list-of-macros> <signal-production-path> <bkg-production-path> <number-of-events-in-bkg-file>`
  - `<output>` is the directory on the grid where output will appear. This will be created inside your gridHomeDir/gridWorkDir/, as specified inside the python script
  - `<list-of-macros>` is the **comma-separated** list of macros to be uploaded to grid. The first macro in the list must be the one called by the jdl, while the rest can be any number of macros eventually called by your main analysis macro. This list should include also the clusterizer macro.
  - `<signal-production-path>` is the path to the place where production files are stored
  - `<bkg-production-path>` path to background production files
  - `<number-of-events-in-bkg-file>` is the number of events which are in one single background file. This is needed in the digitization, where a random bkg event is selected from a given file.
- Running example:
  - `./submitGridJobs.py analyzeFOCALWithEmbedding.jdl test2PbPb geometry.txt parameters.txt AnalysisQAGrid.C,ClusterizeWithEmbeddingGrid.C /alice/cern.ch/user/f/focal/sim/v1.5_root6/box/singlePi0/ /alice/cern.ch/user/f/focal/sim/v1.5_root6/hijingPbPb/simulations 10`
  - This command will submit 200 separate grid jobs, one for each file in the singlePi0 production. For each job, a randomly chosen background file from the hijingPbPb production is used. In each bkg file, there are 10 PbPb events, so we provide 10 as input parameter.