Introduction to running batch analysis on stoomboot

Panos.Christakoglou@nikhef.nl

What is stoomboot?

- ✓ Stoomboot is the name of the local Nikhef cluster
- ✓ It is the local batch computing facility at Nikhef
- ✓ accessible for users from scientific groups to perform
  - data analysis
  - Monte Carlo calculations/simulations

- ✓ The Stoomboot facility consists of
  - 3 interactive nodes and
  - a batch cluster with 93 nodes with 8 cores each,
- ✓ The cluster is running on Scientific Linux CERN 6 as operating system

Interacting with the batch system

- ☑ Interacting with the batch system means (among other operations)
  - submit a job,
  - query a job status
  - delete a job

- ☑ Login on a linux machine
  - Desktops
  - interactive Stoomboot nodes.

Job submission

- Use the command qsub

```
qsub [-q <queue>] [-l resource_name[=[value]][,resource_name[=[value]],...]] [script]
```

- The optional argument *script* is the user-provided script that does the work
  - If no ***script*** is provided, the input is read from the console (STDIN)
  - The argument ***queue*** allows you to select the desired queue (see next slides)
  - The -l option is used for demanding jobs
    - **-l nodes=1:ppn=4** requests 4 cores on 1 node
    - **-l walltime=32:10:05** requests a wall time of 32 hours, 10 minutes and 5 seconds

- More detailed information can be found in the manual page for qsub:

```
man qsub
```

Checking the status of a job

```
qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
1001.allier               script.sh        user1           21:22:33 R generic
1002.allier               myscript.csh     user2           08:15:38 C long
1003.allier               myscript.csh     user2           00:02:13 R long
1004.allier               myscript.csh     user2                  0 Q long
```

```
qstat -n1

                                                                  Req'd  Req'd   Elap
Job ID                  Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
----------------------- -------- -------- ---------------- ------ ----- --- ------ ----- - -----
1001.allier.nikh        user1    generic      script.sh     10280   --    --    --  36:00 R 21:22   stbc-081
1002.allier.nikh        user2    long         myscript.csh  28649   --    --    --  96:00 R 08:15   stbc-043
1003.allier.nikh        user2    long         myscript.csh  12365   --    --    --  96:00 R 00:02   stbc-028
1004.allier.nikh        user2    long         myscript.csh    --    --    --    --  96:00 R   --     --
```

```
man qstat
```

Job deletion

- If a user wants to kill a job with id 1001
  - qdel 1001
- Note that you are able to kill only your own jobs
- For more information about qdel

```
man qdel
```

Queues

queue	walltime [HH:MM]	remarks
generic	24:00	general purpose; default queue
short	04:00	
long	48:00	max. walltime of 96:00 via resource list
multicore	96:00	multicore jobs only
legacy	09:00	gluster access
iolimited	09:00	gluster access
express	00:10	intended for test jobs

Submission and scheduling

- The system is a shared facility
  - e.g. if I get a machine stuck you also pay
- Some considerations:
  - large output files (like logs) going to /tmp might fill up /tmp if many of your jobs land on the same node, and this will hang the node.
  - submitting lots of jobs, each of which opens lots of files, can cause problems on the storage server. Organizing information in thousands of small files is problematic.
- very short jobs are inefficient
  - If your average job run time is not at least 1 minute, please consider how to re-pack your work into jobs.
- Do not run a single core job to the multicore queue

Scheduling

- ☑ the system works on a fair-share scheduling basis.
  - Each group at Nikhef gets an equal share allocated, and within each group, all users are equal. The scheduler makes its decision based on:
    - how much time has your group used over the last 8 hours
    - how much time have you used over the last 8 hours.
- ☑ The group number is converted to a group ranking component
  - if our group has used less than the standard share in the last 8 hours, this number is positive, getting larger the less the group has used.
  - If you've used more than the standard share, the number is negative, getting more negative the more you've used.
  - The algorithm is absolutely the same for all groups at Nikhef.
  - There is a similar conversion for the user number, the scale of the group number being larger than the group one. The two components are added, resulting in a ranking ... the jobs that have the highest ranking run first.

Scheduling (cont.)

- ✓ Jobs are essentially run in this order:
  - low group usage in the past 8 hours, also low user usage
  - low group usage in the past 8 hours, higher user usage
  - higher group usage in the past 8 hours, lower user usage
  - higher group usage in the past 8 hours, higher user usage

Prerequisites for batch running

- ✓ A “submit” script (see later)
- ✓ A text file with runs for each raw data period
- ✓ A “run” macro adopted to take the directory name (base dir + run number) as an argument
  - Base dir is the directory where the raw data samples are stored (see lecture of last week)
  - The macro creates a TChain of AOD files residing under this run and passes it to the manager
- ✓ An AddTask macro (see lecture of last week)
- ✓ A task i.e. header and source files (see lecture of last week)

The submit script

pchrist — ssh panosch@stbc-i1 — 133×56

```
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
SCRIPT="runAnalysis.sh"
while IFS='' read -r runNumber || [[ -n "$runNumber" ]]; do
    echo "Adding run number from file: $runNumber"

if [ "$2" == "2010" ]
then
    gDirectory="/dcache/alice/panosch/alice/data/2010/LHC10h/AOD160/$runNumber"
elif [ "$2" == "2015" ]
then
    gDirectory="/dcache/alice/panosch/alice/data/2015/LHC15o/000$runNumber"
else
    exit
fi

#make the script to submit
    (echo "#!/bin/bash"
echo "source /cvmfs/alice.cern.ch/etc/login.sh"
echo "eval $(alienv printenv VO_ALICE@AliPhysics::vAN-20161025-1)"
echo "which aliroot || exit 1"
if [ "$2" == "2010" ]
then
    echo "mkdir -p /glusterfs/alice1/alice2/pchrist/Flow/HighHarmonics/2010/LHC10h/AOD160/$runNumber"
    echo "cd /glusterfs/alice1/alice2/pchrist/Flow/HighHarmonics/2010/LHC10h/AOD160/$runNumber"
elif [ "$2" == "2015" ]
then
    echo "mkdir -p /glusterfs/alice1/alice2/pchrist/Flow/HighHarmonics/2015/LHC15o/000$runNumber"
    echo "cd /glusterfs/alice1/alice2/pchrist/Flow/HighHarmonics/2015/LHC15o/000$runNumber"
else
    exit
fi
echo "pwd"
echo "if [ -f AnalysisResults.root ]"
echo "  then "
echo "rm -rf AnalysisResults.root"
echo "fi"
echo "if [ ! -f runFlowPIDSPTask.C ]"
echo " then "
echo "ln -s /user/panosch/ALICE/Flow/HigherHarmonics/Stoomboot/runFlowPIDSPTask.C ."
echo "fi"
echo "exec aliroot -b -q runFlowPIDSPTask.C'(\"$gDirectory\")'"
    ) > $SCRIPT

qsub -q generic $SCRIPT

done < "$1"
```

The run list

pchrist — ssh panosch@login.nikhef.nl — 121×47

```
139510
139507
139505
139503
139465
139438
139437
139360
139329
139328
139314
139310
139309
139173
139107
139105
139038
139037
139036
139029
139028
138872
138871
138870
138837
138732
138730
138666
138662
138653
138652
138638
138624
138621
138583
138582
138579
138578
138534
138469
138442
138439
138438
138396
138364
138275
--More--(50%)
```

The run macro

```
void runFlowPIDSPTask(TString gdirectory,
                      Int_t mode = mLocal, Bool_t DATA = kTRUE) {
  //void runEventPlaneTask(Int_t mode = mGrid, Bool_t DATA = kTRUE) {
  // Time:
  TStopwatch timer;
  timer.Start();

  // Load needed libraries:
  LoadLibraries(mode);

  // Create and configure the AliEn plug-in:
  if(mode == mGrid || mode == mGridPAR) {
    gROOT->LoadMacro("CreateAlienHandler.C");
    AliAnalysisGrid *alienHandler = CreateAlienHandler(lhcPeriod,
                                                       particleSpecies.Data());
    if (!alienHandler) return;
    //gROOT->LoadMacro("AliAnalysisTaskEventPlaneAOD.cxx++");
  }
```

```
TSystemDirectory dir(gdirectory.Data(),gdirectory.Data());
TList *files = dir.GetListOfFiles();
if (files) {
  TSystemFile *file;
  TString filename;
  TIter next(files);
  while ((file=(TSystemFile*)next())) {
    filename = gdirectory.Data();
    filename += file->GetName();
    filename += "/AliAOD.root";
    //if (!file->IsDirectory() && fname.EndsWith(ext)) {
    cout << "Adding: "<<filename.Data() << endl;
    chain->Add(filename.Data());//}
  }
}
```

Submitting the jobs

- Please change the names of the
  - Input directory
  - Run macro
  - AddTask
  - Task

- Use an ascii file with just one run (e.g. from LHC10h)

- Go to the directory where the submit macro is stored and type:
  - source submit.sh lhc10h.txt 2010

Merging

- ✓ At the end, your batch jobs have produced one output rot file per run
- ✓ To get full statistics for your analysis you need to merge the output
- ✓ Assuming that your output contains simple ROOT objects e.g. histograms, or list of histograms then
  - Go to where your output is stored one directory up from each run-number directory
  - Issue the following command on the terminal

Final merged filename

ROOT command

```
hadd  mergedAnalysisResults.root  */AnalysisResults.root
```

Name of the analysis output stored for every run

  - The final root file will have the same structure as the individual AnalysisResults.root files of every run but it will contain histograms with all the statistics of all the runs
    - The histograms and thus their entries of every run are added and stored in the merged file

Good luck…

- ✓ https://www.nikhef.nl/grid/stats/stbc/