Offline tutorial
Working in AliEn

Raffaele Grosso

Raffaele.Grosso@cern.ch

March 22, 2013

Outline

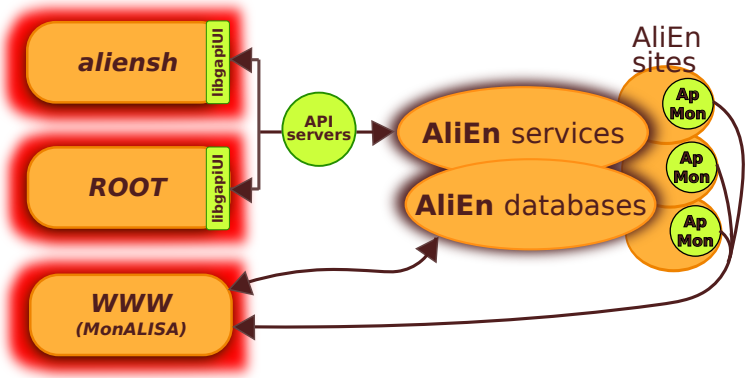
- Prerequisites
- Installation of the AliEn Grid client
- Connection and Login/Authentication
- The AliEn shell
  - Functionality
  - Basic commands
  - View, edit and copy files within the catalogue and from/to the catalogue

Outline

- The ROOT interface
- The MonALISA interface
- Grid Jobs
  - Job submission, status and control
  - Overview of the JDL files (Job Description Language)
- Working with the file catalogue
  - Copying files from/to the catalogue.
  - Creating collections of files
- Working with the AliEn plug-in
  - Configure your own plug-in
  - Run the analysis in grid via the plug-in

Aim

- Getting familiar with AliEn user interfaces:
  1. the API client gapi and the command line interface (aliensh, the AliEn shell)
  2. the AliEn interface for the ROOT/AliROOT framework
  3. the MonALISA interface to AliEn



- Getting a feeling of runnig production and analysis jobs as an ALICE user in AliEn

Prerequisites

- To be able to follow this hands-on tutorial you need to have a valid PEM keypair (`usercert.pem` and `userkey.pem` files) in your `$HOME/.globus` directory.
- For that you need to have followed the user registration to the end.
- The registration was supposed to be done at: `http://alien2.cern.ch/index.php?option=com_content&view=article&id=17&Itemid=90`
- You can check the validity of your certificate (one year validity for users) with: `openssl x509 -in usercert.pem -dates -noout`
- The browser manages X.509 certificates as PKCS12 files (.p12) instead. The .p12 certificate must be loaded in the browser to access the MonALISA interface to alien.

more

Installing the AliEn client

- Get the alien client installer:

```
wget http://alien.cern.ch/alien-installer
```

or:

```
curl http://alien.cern.ch/alien-installer
-o ./alien-installer
```

- Make it executable:

```
mod +x alien-installer
```

- Launch it and wait:

```
./alien-installer
```

Installing the AliEn client

- Get the alien client installer:

```
wget http://alien.cern.ch/alien-installer
```

or:

```
curl http://alien.cern.ch/alien-installer
-o ./alien-installer
```

- Make it executable:

```
mod +x alien-installer
```

- Launch it and wait:

```
./alien-installer
```

Installing the AliEn client

- Get the alien client installer:

```
wget http://alien.cern.ch/alien-installer
```

or:

```
curl http://alien.cern.ch/alien-installer
-o ./alien-installer
```

- Make it executable:

```
mod +x alien-installer
```

- Launch it and wait:

```
./alien-installer
```

Installing the AliEn client

You can make use of the flags for e.g.

- specifying the installation directory (`-install-dir <path>`)
- installing a specific version of the AliEn client (`-version <version>`)

`./alien-installer -h` ... if interested in additional options.

However the default installation is generally just what you need.

Installing the AliEn client

If you didn't have the folder `$HOME/bin`, the installer created it for you. To have it included in the `$PATH` shell environment variable, set it in the appropriate configuration file for your shell, e.g. in `$HOME/.bashrc`. Otherwise you will have to set it each time you open a new shell:

```
export PATH=$PATH:$HOME/bin
```

Installation – hands-on

- Copy grid certificates to the computer in front of you
  - `mkdir .globus`
  - `scp <username>@lxplus:.globus/*.pem .globus`
- Verify location (`.globus`) of your (certificate,key) pair and their permissions (400 for `userkey.pem`, 640 or 440 for `usercert.pem`).
- Download the alien installer from `http://alien.cern.ch/alien-installer`
- Make the file executable
- Run the installer (specify alternative installation location)
- Check that the installation went fine (e.g. check that you see inside the alien directory subdirectories like `api` and `src`, check the log-file of the installation in case)
- Do `export PATH=$PATH:~/bin`

Issues during installation

- The most common: "Secssl: Certficate is not yet valid" Solution: adjust the clock on the local host (or ask the sysadmin).
- For less common installation issues check the alien pages or the analysis mailing-list `alice-analysis-operations@cern.ch`
- If you think the issue is new or is not on your side, write to that mailing-list

Authentication

- To access the AliEn shell you have to have authenticated not sooner than 24h before.
- Authentication is done by creating your access token with `alien-token-init <grid-cert-username>`
- You don't need to specify your grid username if it is the same as your `$USER`
- You need to remember the PEM passphrase associated to your certificate ☺

```
rgrosso@pcalice72:~$ alien-token-init
------------------------------------------------------------
Setting central config:
============================================================
export alien_API_SERVER_LIST="pcapiserv03.cern.ch:10000|"
export TERMINFO=/usr/share/terminfo
============================================================
------------------------------------------------------------
Setting closest site to: CERN
============================================================
Using X509_CERT_DIR=/home/rgrosso/alien/api//../globus/share/certificates
*******************************************************************************
Attention: You don't have a valid grid proxy ( or less than 1 hour left ) - doing xrdgsiproxy init for you ...
*******************************************************************************
Enter PEM pass phrase:
```

Authentication

- Do `alien-token-init <your-grid-cert-username>`
  - If asked about compiling the gapi and xrootd libs say no
  - Later, to install on your own machine and do analysis you will have to say yes.
  - If you want to force Alien recompilation you have to `export GSHELL_NO_GCC=0`. Alien will prompt for recompilation on the next `alien-token-init`.
  - no problem if you took the binaries from `http://alimonitor.cern.ch/packages`

At the end you should have a valid token.

```
Enter PEM pass phrase:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
file        : /tmp/x509up_u1000
issuer      : /DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=rgrosso/CN=605916/CN=Raffaele Grosso
subject     : /DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=rgrosso/CN=605916/CN=Raffaele Grosso/CN=811099136
path length : 0
bits        : 512
time left   : 12h:0m:0s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
=> Trying to connect to Server [0] root://pcapiserv03.cern.ch:10000 as User rgrosso
/alice/cern.ch/user/p/plpetrov/pidDedxTof_LHC10b/
/alice/cern.ch/user/r/rgrosso/
Your identity: rgrosso
Creating token ..................................... Done
Your token is valid until: Thu May  3 11:14:51 2012
```

Issues during authentication

- Permissions on `~/.globus/userkey.pem` are not private to your user.
  Solution: `chmod 400 userkey.pem`
- Your certificate authority is exotic and not known to the server
- Your certificate has expired.
  Solution: check its validity. Renew it.
- You have not given the AliEn user name as an argument to the token init command and your local user name is not identical to the AliEn user name.
  Solution: `alien-token-init alienusername`

Issues during authentication

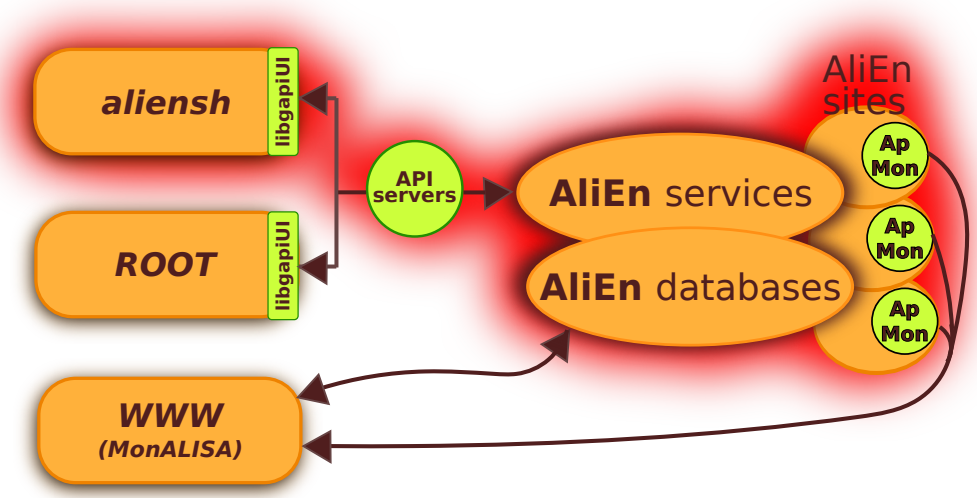
- Clock skew - your local computer time is out of the validity time of your certificate.
  (the error message contains something like: `Secssl: certificate is not yet valid;`).
- In case of any issue, remove the files created by the authentication in the `/tmp` dir:

  ```
  rm /tmp/*${UID}*
  ```

  and relaunch `alien-token-init`. Recreating the files in /tmp related to the alien session allows not to propagate previous errors (e.g. misconfiguration of the alien environment).

The AliEn shell: aliensh

- It's a standard bash shell including grid commands.
- Main shell features are available, in particular command-, file- and path- completion.
- Launch it with `aliensh`.




```
rgrosso@pcalice72:~$ aliensh
 [ aliensh 1.0.14nx (C) ARDA/Alice: Andreas.Joachim.Peters@cern.ch/Derek.Feichtinger@cern.ch]
aliensh:[alice] [1] /alice/cern.ch/user/r/rgrosso/ >ls
aCDB
alien-job-133636048
bin
collection
GridTutorial
ITS
ITSCOSMICS2008
MakeAndReadSnapshot
```

aliensh basic commands

- Standard Unix Shell commands work as usual: `ls`, `cd`, `mkdir`/`rmdir`, `cat`, `more`, `pwd`, `whoami` ... (with some less available flags)
- There is a `help` command to list all internally-defined commands
- Get a complete command list by typing `<tab><tab>`
- For internally-defined commands you can type `help <command>` or `<command> -h`
- for standard shell commands `man -k <command>` or `info -k <command>`

Editing in aliensh

- Use the `edit` command (`edit -c` for a new file)
- By default `edit` maps to vi. You can set it to your preferred editor adding the following line to the configuration file `/.alienshrc`: `export EDITOR=<editor>`, where editor is one of emacs, emacs -nw, xemacs, xemacs -nw, pico, vi, vim
- the previous version of the edited file is saved as a hidden file `.<filename>~`
- `purge` is meant to delete the old version (leaving in place the last one)

Copying files in the File Catalogue

The cp command works just like the Unix Shell command but is operating among files in the AliEn File Catalogue.

```
aliensh:[alice] [6] /alice/cern.ch/user/r/rgrosso/GridTutorial/ >ls
.tutorial.jdl~
grid_tutorial.pdf
JobOutput
pippo.txt
tutConfig.C
tutorial.jdl
tut_validation.sh
aliensh:[alice] [7] /alice/cern.ch/user/r/rgrosso/GridTutorial/ >cp pippo.txt foobar.txt
Overriding 'TransactionTimeout' with value 60.  Final value: 60
Overriding 'RequestTimeout' with value 30.  Final value: 30
Overriding 'ConnectTimeout' with value 10.  Final value: 10
Overriding '' with value FirstConnectMaxCnt.  Final value: 0
Error getting my hostname from $HOSTNAME or $HOST. Taking the machine's name, please verify if this is good or not.
The machine's hostname is 'pcalice72'
[xrootd] Total 0.00 MB  |====================| 100.00 % [inf MB/s]
Overriding 'TransactionTimeout' with value 60.  Final value: 60
Overriding 'RequestTimeout' with value 30.  Final value: 30
Overriding 'ConnectTimeout' with value 10.  Final value: 10
Overriding '' with value FirstConnectMaxCnt.  Final value: 0
Error getting my hostname from $HOSTNAME or $HOST. Taking the machine's name, please verify if this is good or not.
The machine's hostname is 'pcalice72'
[xrootd] Total 0.00 MB  |====================| 100.00 % [inf MB/s]
Overriding 'TransactionTimeout' with value 60.  Final value: 60
Overriding 'RequestTimeout' with value 30.  Final value: 30
Overriding 'ConnectTimeout' with value 10.  Final value: 10
Overriding '' with value FirstConnectMaxCnt.  Final value: 0
Error getting my hostname from $HOSTNAME or $HOST. Taking the machine's name, please verify if this is good or not.
The machine's hostname is 'pcalice72'
[xrootd] Total 0.00 MB  |====================| 100.00 % [inf MB/s]
aliensh:[alice] [8] /alice/cern.ch/user/r/rgrosso/GridTutorial/ >ls
.tutorial.jdl~
foobar.txt
grid_tutorial.pdf
JobOutput
pippo.txt
tutConfig.C
tutorial.jdl
tut_validation.sh
```

Copying from/to the File Catalogue

- To specify files on your local disk, use the prefix `file:`
- e.g. copying the file from AliEn to local current working directories:

```
aliensh:[alice] [5] /alice/cern.ch/user/r/rgrosso/GridTutorial/ >cp pippo.txt file:foobar.txt
```

- e.g. copying the file from the local to AliEn current working directories:

```
aliensh:[alice] [10] /alice/cern.ch/user/r/rgrosso/GridTutorial/ >cp file:foo.txt bar.txt
```

- You may need to define the environment variable `alien_CLOSE_SE` pointing to a SE that is accessible and close to your location:
  `export alien_CLOSE_SE=ALICE::GSI::SE`
- You can always specify the SE location in the copy commands
  `cp <localfile> <alienFC_path>@someSE`
  `cp <alienFC_path>@someSE <localfile>`

whereis

- What you see with `ls` is the the Logical File Name, the name of the file in the AliEn domain (in the AliEn File Catalogue)
- In the FC the LFN is associated to one or more Physical File Names, the actual file(s) residing on some particular Storage Element.
- To see where a file (LFN) is actually stored, use the command `whereis <LFN>`.

```
aliensh:[alice] [12] /alice/cern.ch/user/r/rgrosso/GridTutorial/ >whereis foobar.txt
Apr 24 15:10:16  info    The file rgrosso/GridTutorial/foobar.txt is in
     SE => ALICE::CERN::ALICEDISK  pfn =>root://voalice16.cern.ch:1094//14/08464/5eee7474-8e08-11e1-8d33-00266cfacaac
   SE => ALICE::Legnaro::SE  pfn =>root://t2-xrdrd.lnl.infn.it:1094//14/08464/5eee7474-8e08-11e1-8d33-00266cfacaac
```

LFN

first replica

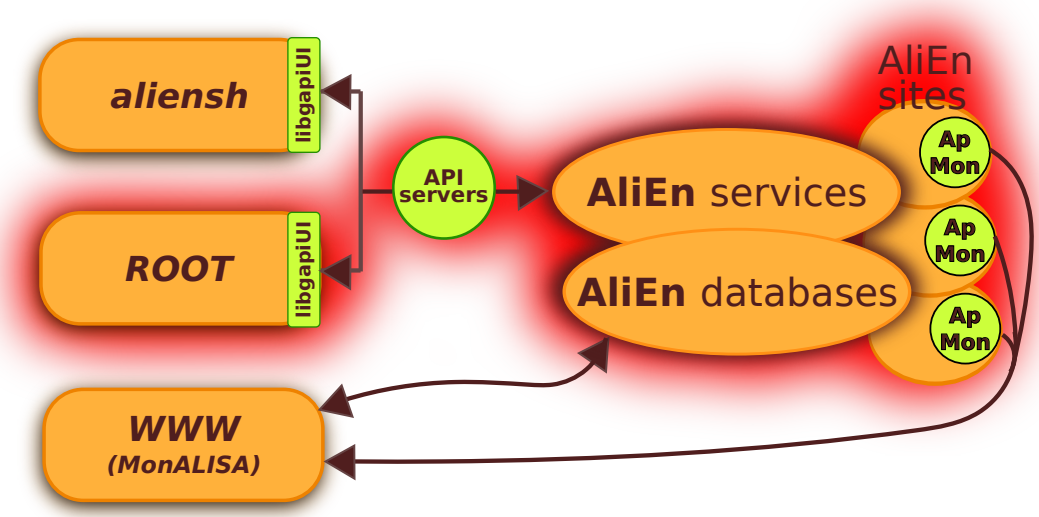
second replica

aliensh hands-on

- Access the alien shell
- Check your user name by typing `whoami`
- List the contents of your home directory
- Do the following in your AliEn space:
  - Create the directories `~/bin` , `~/tutorial/`
  - `cp /alice/cern.ch/user/r/rgrosso/GridTutorial/alientutorial.pd ~/tutorial`
  - Now copy the `alientutorial.pdf` to your local machine
- Get the information on the replicas of the file `alientutorial.pdf`
- Edit a textfile called `foo.txt`. Close and append something to it. Verify the creation of a backup of the previous version.
- Copy `foo.txt` to your home directory on your local machine and check that the file is there and that you can open it

The ROOT interface

- The AliEn services can be accessed via the API implemented in the library libgapiUI not only by the AliEn shell but also by ROOT, which needs just to be compiled with alien support.



- For that include the appropriate lines in the configuration file, like: `--with-alien-incdir=${ALIEN_ROOT}/api/include` ...
- You can find `here` the full configuration for compiling ROOT against AliEn.
- And check that `alien` is among `root-config --features`

The ROOT interface

- Establish the connection
  - get authenticated and set the environment:
    - `alien-token-init`
    - `source /tmp/gclient_env_$UID`
  - `TGrid::Connect("alien://");` connect, creating a token, to a default API service
  - `TGrid::Connect("alien://",0,0,"t");` connect using the existing token (created by `alien-token-init`)
- You can execute predefined TGrid commands, for example:
  - `Bool_t res = gGrid->Cd("/alien/cern.ch/user/r/rgrosso");`
  - `Bool_t res = gGrid->Mkdir("/alice/cern.ch/user/r/rgrosso/aDir");`
  - `TGridResult *result = gGrid->Ls("/alice/cern.ch/user/r/rgrosso");`
  - `for(Int_t i=0; i<result->GetEntries(); i++){`
  - `Printf("%s",result->GetFileName(i));}`
- Other TGrid commands: `Mkdir`, `Rmdir`, `Rm`, `Cd`, `Pwd`

The ROOT interface

- You can also execute every aliensh command by means of the `TAlien::Command` method. For example:
```
TGridResult *result = gGrid->Command("ls
/alice/cern.ch/user/r/rgrosso/GridTutorial");
```
- The TGrid command returns a TGridResult. You can then access the single items specifying a key, e.g.:
```
Printf("%s",result->GetKey(0,"name"))
Printf("%s",result->GetKey(0,"md5"))
```
- To know which keys are available for the result of a given command you can do: `result->Print("all");`
- E.g. for the `ls -la` in our example we would see the following keys: `name`, `md5`, `size`, `group`, `path`, `permissions`, `user`, `date`

The ROOT interface

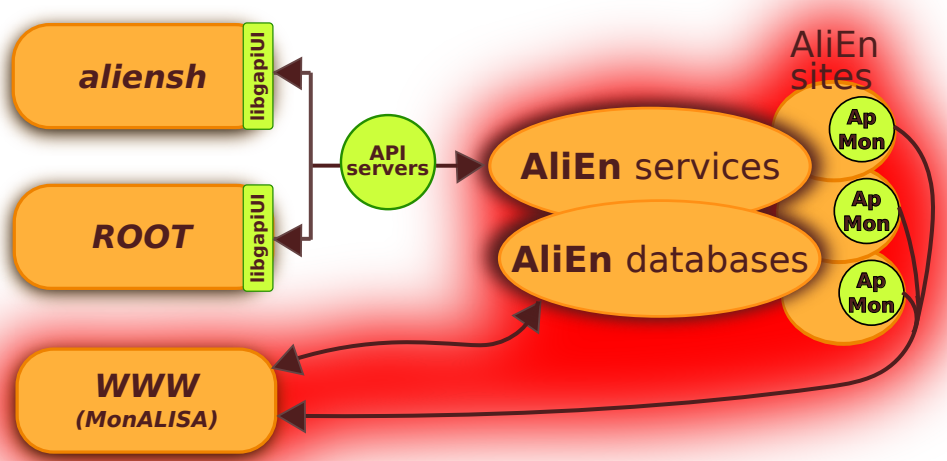
A convenient way of producing a file list, in case easily convertable in a `TChain`, is by using the `TGrid::Query` command. For example:

```
TGridResult *result =
gGrid->Query(TString("/alice/.../mydir"),TString("*.root"),TSt
10");
```

- would get all ROOT files in the directory tree under `"/alice/.../mydir"`;
- the `"-l 10"` option limits to 10 the number of returned files;
- an additional string argument can be passed for imposing a condition on the metadata.

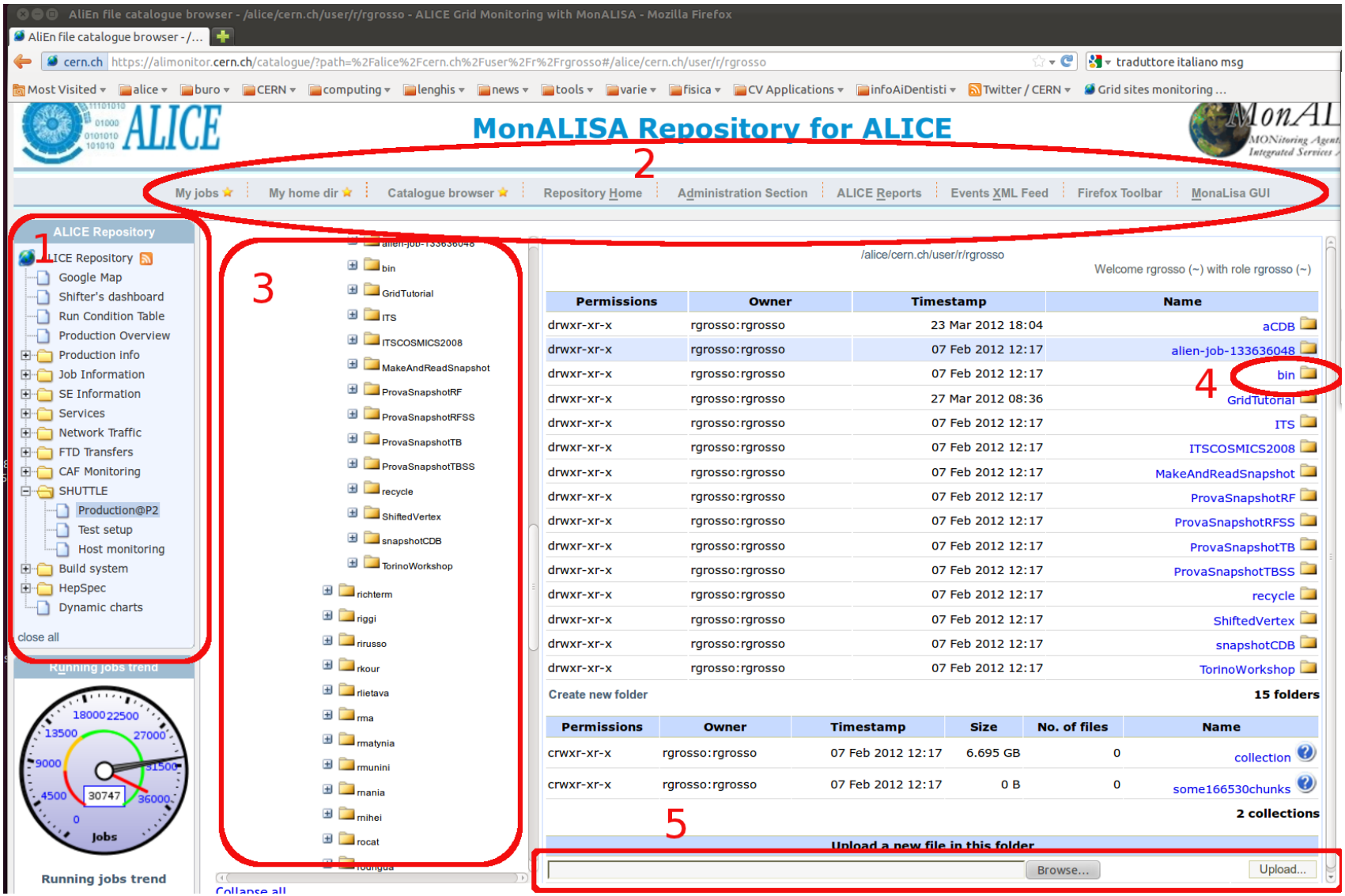
The MonALISA interface

- The AliEn services can be accessed also via WWW by means of MonALISA, whose main purpose is to monitor them, but which also allows to read and write on the file catalogue.
- the access (via ApMon) is different w.r.t. to the access for the AliEn shell and ROOT (peer-to-peer vs. client-server)



https://alimonitor.cern.ch

The MonALISA interface



The MonALISA interface

1: a local navigation bar to access information from MonALISA subsections for Productions (MC, RAW, analysis trains), Jobs, Storage Elements, analysis facilities, . . . ;

2: a toolbar for accessing your alien jobs, the AliEn catalogue from a fixed node or from your home directory, . . . ;

3: an interface to the catalogue;

4: an interface to view, edit or download single files;

5: an interface to upload a file to the AliEn catalogue from the local machine.

The MonALISA interface

Information often needed for analysis on AliEn:

- Job information→User views→Grid packages to make sure the `Packages` JDL attribute is set correctly
- Job information→User views→Quotas
- Job information→Task queue→Jobs in TQ table
- Job information→Memory profiles→Offending jobs
- SE Information→Status to avoid landing on SE in failed state.

Expand Series and Options to check your username.

The JDL files

- A JDL file is a text file using the Job Description Language syntax to specify all the information needed to run a job on the grid.
- It consists of statements of type `Attribute="value";` or: `Attribute={"value1",...,"valueN"};`
- Statements are ended by a semicolon (and jdl is sensitive to spaces and tabs!).
- Only the `Executable` attribute is mandatory, however many of the others are necessary for jobs to end up successfully.

You can get inspired by looking at the JDLs of RAW, MC and analysis productions e.g. in `/alice/cern.ch/user/a/alidaq/<LHCperiod>/` and `/alice/cern.ch/user/a/aliprod/<productionname>/`.

General JDL attributes

- `Executable` the name of the executable, which must be stored in `/bin`, `/alice/bin` or `~/bin`
- Arguments: arguments to be passed to the executable
  Special example to pass to the executable two arguments passed when submitting: `Arguments="$1 $2";`
- Packages: list the packages on AliEn needed by your job.
  Type `packages` in the shell to see what packages are installed.
  Check at `https://alimonitor.cern.ch/packages` the AliRoot/ROOT/GEANT3 dependencies
- Validationcommand: specifies the script to be used as a validation script
- TTL: The maximum run time of your job in seconds
  e.g. `TTL=36000;` sets the Time To Live to ten hours

JDL attributes for input

- InputFile: list of files that the job needs to have staged in the sandbox (copied locally where the job will run)
- InputData: requires that the job is executed close to where the file is.
  - allows to specify patterns, including for example `*`
  - add `,nodownload` to avoid files being staged in the sandbox
  - if you have many input files, use instead
- InputDataCollection: file containing the InputData in XML collection format

Other JDL arguments specifying the input, but less commonly used, are:

- InputDataList: collection filename to be processed on each worker node (where the Job Agent writes the InputData)
- InputDataListFormat: format of the InputDataList

JDL attributes for splitting

- Split: Split the jobs in several sub jobs
  There are three splitting modes of most frequent use:
  - `split="se";` – for analysis (subjob are created and sent to sites maximizing the number of input files found in the local SE)
  - `split="file";` – for RAW reco productions (one subjob per file, e.g. each subjob reconstructs a chunk)
  - `split="production:n1-n2";` – for MC productions (create n2-n1 subjobs, with increasing `alien-counter`, always the same JDL with unchanged input)
- SplitArguments: defines the arguments for each subjob.
  Here you can use the input arguments `$1`,`$2`,. . . and the placeholders like `#alien_counter#` or `#alienfilename#`, e.g.:
  ```
  splitarguments = "simrun.C --run $1 --event #alien_counter#
  ```
- SplitMaxInputFileNumber and SplitMaxInputFileSize: maximum input file number or input data volume (in MB) for each subjob

JDL attributes for output

- OutputFile declare files to be registered in the catalogue once the job finishes. Declare archives.
  `OutputFile={"myFilename","root_archive.zip:*.root"};`
  The number of copies defaults to two. You can change that with `@disk=n`, e.g. to produce the output file in three copies:
  `OutputFile={"myFilename@disk=3"};`
  Use instead `disk=1` for logfiles and leave the default for root files.
- OutputDir: Where the output files & archives will be stored
  `OutputDir="/alice/cern.ch/user/a/aliprod/analysis/output101";`

AliEn has a Storage Element discovery and failover mechanism, storing your output files by default always in the two topmost locations ⇒ Do not use static SE declarations (@<SE>) to store output files!

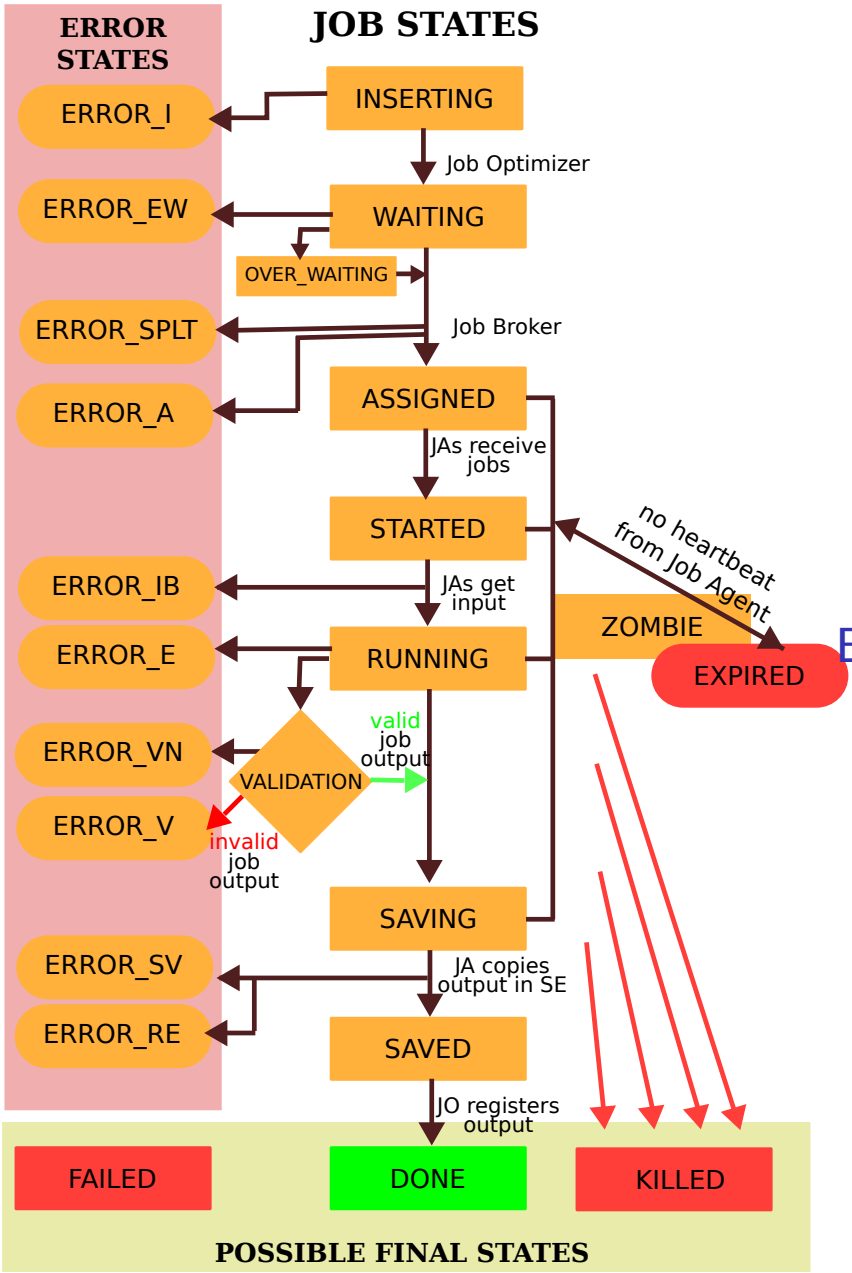
Other JDL attributes

- MasterResubmitThreshold: resubmit subjobs until the specified threashold of successful jobs is reached. E.g.:
  `MasterResubmitThreshold=99%";`
  or you can give an absolute number of jobs.
- Email: email address to receive an email when the job finishes
  `Email="alienmaster@cern.ch";`
- Jobtag: name with which the job appears e.g. in MonAlisa

Submitting jobs

- In order to submit a job, call `submit` together with the JDL file you have created
- You can also pass arguments, which will be seen in the JDL as `$1`, `$2`, `$3`, ...
- e.g.: `submit tutorial.jdl <#runnum>`
- As soon as the job enters the Task Queue it will be visible from the AliEn shell and from MonALISA

Job Status



- The submitted job is inserted in the AliEn Task Queue, in case splitted
- Requested by some JA, the jobs are sent somewhere in the world
- Error states help tracking down the issue for the failed job, e.g.:

ERROR_IB	e.g. some input file is missing
ERROR_EW	expired waiting (stayed in waiting for too long)
ERROR_E	TTL or memory exceeded
ERROR_RE	prevents storing twice the same result
ERROR_SV	e.g. some output file is already there (preventing saving)

- http://alimonitor.cern.ch/show?page=jobStatus.html

back

Job Status

There are two commands allowing the user to check the Grid jobs. Several parameters allow to focus on the information needed.

- `ps` will give you a list of your jobs or Job-ID specific info when used with flags
- `top` is more verbose than `ps`. By default returns a list of all jobs in the queue, thus use parameters to constrain the output, e.g.: `top -status DONE -user <myuser>`

Also useful (e.g. to check if the JDL can't be matched by any site):

- `gbbox jobListMatch <subjob-ID>`

Job Status: ps

- `ps` lists your active jobs
- `ps -jdl <Job-ID>`
  displays the jobs JDL during or after the jobs runtime. (during/after runtime the JDL contains more information, some arguments are added and filled automatically).
- `ps -trace <Job-ID>` traces the job, showing all the states it went through (during or after the job's runtime).
- `ps -trace <Job-ID> all` as the above, more verbose
- `ps -XA` information for all your active jobs

`ps --help` for all flags

Job Status: top and masterjob

- top -id <Job-ID> status, executable and submitting host for given Job-ID
- top -status <STATUS> -user <alienuser> limits the list of active jobs to given STATUS and owner
- top -host <host> -site <site> limits the list of jobs to a given host or site (useful for administrators)

- masterjob <JOB-Id> will print a status of the sub jobs of the specified master job
- This is only interesting/working, if you have a job that splits

Checking the job's output

- Normally, the log files `stdout` and `stderr` end up in the job's output directory.
- Not for jobs ending in EV (logs go to `home/recycle/alien-job-#`). In that case you can:
  - use `registerOutput <jobnum>`
  - use the My Jobs link in the MonALISA interface, from where you can also get the trace of the job execution (in particular the error message of the validation).
  - also, from MonALISA, Job Information→Task Queue →Jobs in TQ table→select the interesting owner→ (All→)click on the interesting PID→ on the popup window click on log files
- With `spy` you can check the output of a job while it is still running
  - `spy <filename>` or
  - `spy <Job-ID> stderr`
  - Never spy on large files, e.g. never spy on a `*.root` file

Job Control: killing a job

- You can also kill a job while it is running with: kill <Job-ID>

```
aliensh:[alice] [4] /alice/cern.ch/user/a/alienmas/ >submit TestJob.jdl
Submit submit TestJob.jdl
submit: Your new job ID is 42138635
aliensh:[alice] [5] /alice/cern.ch/user/a/alienmas/ >ps
  alienmas 42138635    __  SV                    testjob
aliensh:[alice] [6] /alice/cern.ch/user/a/alienmas/ >kill 42138635
Process 42138635 killed!!
aliensh:[alice] [7] /alice/cern.ch/user/a/alienmas/ >
```

Submitting/Running Jobs - Problems

- If everything is ok with your JDL then your job is submitted and a Job-ID is assigned to it.
- You get a submission error message, e.g. if
  - Your JDL contains errors , e.g. the syntax is not correct
  - A file listed in the JDL is missing
  - A package defined in the JDL is not listed in the packman
- The subjobs can go to error states at any step of the process

▸ see job states

- If you kill the jobs in error you cannot access their logs anymore (with registerOutput or MonAlisa), so . . .
- Look for similar issues and solutions browsing the analysis task-force mailing-list alice-project-analysis-task-force@cern.ch

Creating File Collections

There is more than one way to create a collection:

- Manually: make an empty collection in the catalogue with `createCollection`, add the files you want with `addFileToCollection`
- With `find` ←*Recommended*
  - With find you can create XML collections of files: `find -x <Coll-Name> <Path-To-Search> <Search-Tag> > <Local-Outp.-File>`
    E.g.: `find -z -l 100 -x sigma1385 /alice/sim/PDC_09/LHC09a4/80050/* *ESDs.root > prova.xml`
    will create an xml collection named `sigma1385` in the local file `prova.xml`, containing <100 ESDs files from the given directory.
  - Dont forget the output file is local on your machine, you need to upload it later.

Files and Grid Jobs: hands-on

- in your AliEn home dir, do `mkdir AliEnTutorial` and `cd` into it
- `cp /alice/cern.ch/user/r/rgrosso/AliEnTutorial/tutorial.jdl tutorial.jdl`
- `cp /alice/cern.ch/user/r/rgrosso/AliEnTutorial/simrun.sh simrun.sh`
- `cp /alice/cern.ch/user/r/rgrosso/AliEnTutorial/sim.C sim.C`
- `cp /alice/cern.ch/user/r/rgrosso/AliEnTutorial/Config.C Config.C`
- `cp /alice/cern.ch/user/r/rgrosso/AliEnTutorial/rec.C rec.C`
- fix the file locations inside `tutorial.jdl` (change each `r/rgrosso` to your `m/myalienuser`)
- submit the job
  - `submit tutorial.jdl 195391`

Files and Grid Jobs: hands-on

- You should get an error, the JDL file has syntax problems ⇒ find the errors and correct them!
- Submit the job again
- Follow the job's stages
- Once the job has finished with DONE
- Look at the content of the output directories `~/AliEnTutorial/<runnumber>/???`
- Check what the job did, where it was running, the files and their content.

That's it. Mission accomplished!

The AliEn plug-in: a helper for the AliEn analysis

- It is the recommended way to structure your analysis on AliEn
- Works as a plugin for the analysis manager
- For detailed information see:
  http://aliceinfo.cern.ch/Offline/Activities/Analysis/AnalysisFramework/AlienPlugin.html
- Provides to the user a rich functionality, hiding the underlying complexity. In particular it can:
  - create
    - datasets
    - JDL
    - analysis macro
    - execution+validation scripts
  - copy needed files to AliEn, submit your job and merge the results

The AliEn plug-in

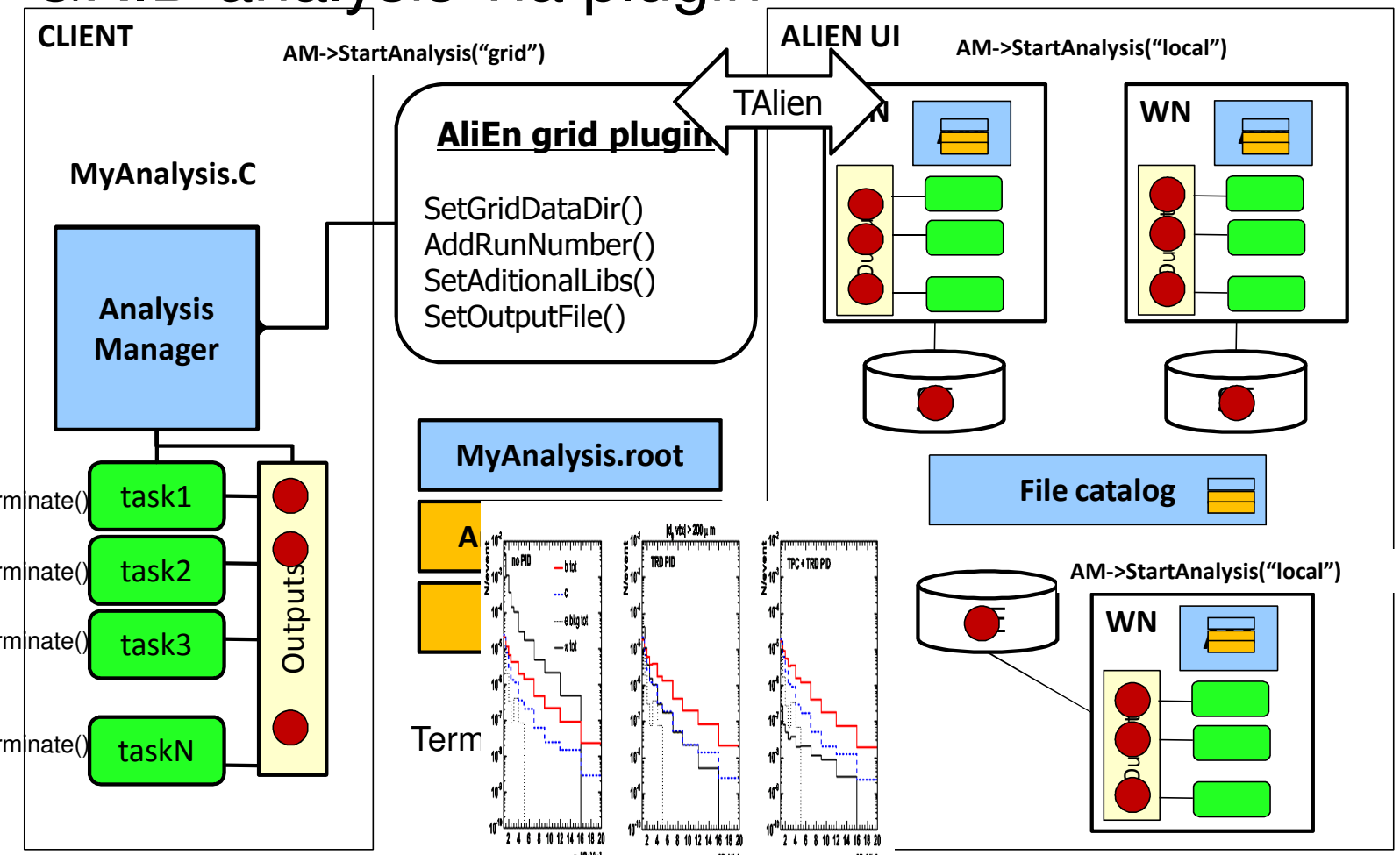
- The plugin is implemented in the class `AliAnalysisGrid` and direved classes inside libANALYSIS providing the interface to configure many custom parameters.
- As in the example macro `CreateAlienHandler.C` in the tarball, one has to:
  - create and configure an `AliAnalysisGrid` object:
    `AliAnalysisAlien *plugin = new AliAnalysisAlien();`
  - set the run mode to be `full`, `test`, `offline`, `submit` or `terminate`:
    `plugin->SetRunMode("full");`
  - customize input, output and grid processing in general (this provides an interface to the JDL)

The AliEn plug-in

- In your analysis macro, as done in the example macro `runGrid.C` in the tarball, you have to:
  - include the plugin in the analysis macro:
    ```
    gROOT->LoadMacro("CreateAlienHandler.C");
    AliAnalysisGrid *alienHandler = CreateAlienHandler();
    ```
  - create the analysis manager: `AliAnalysisManager *mgr = new AliAnalysisManager("testAnalysis");`
  - and connect the plugin to it:
    ```
    mgr->SetGridHandler(alienHandler);
    ```
- Include other actions needed by the analysis manager, like creating input/output containers and connecting them, tweaking the manager behaviour, ...
- launch finally the analysis on AliEn:
  ```
  mgr->StartAnalysis("grid");
  ```

The AliEn plug-in

GRID analysis via plugin



A
coll

Setting the run-mode

The plugin can be used in different modes, in case limiting its action to a subset of operations:

`plugin->SetRunMode(const char *mode)`

- `full`: generate files, copy in grid, submit, merge
- `offline`: generate files, user can change them
- `submit`: copy files in grid, submit, merge
- `terminate`: merge available results
- `test`: generate files + a small dataset, run locally as a remote job
  - `plugin->SetNtestFiles(Int_t nfiles)` – default 1

The recommended usage is to first run the plugin in `test` mode, for validation, only then to run it in `full` mode.

Describing the input data

- `plugin->SetGridDataDir(const char *datadir)`
  - Put here the alien path before run numbers
  - See pcalimonitor.cern.ch for relevant data paths
- `plugin->SetDataPattern(const char *pattern)`
  - Use uniquely identifying patterns
  - Plugin supports making datasets on ESD, ESD tags or AOD
- `plugin->SetRunRange(Int_t min, Int_t max)`
  - Sets the run range to be analyzed
  - Enumeration of run numbers allowed
  - For existing data collections, use `AddDataFile()`
- `plugin->SetRunPrefix(000)`
  - To be used for real data

Describing the output

- `plugin->SetGridOutputDir(const char *dir)`
  - Can be absolute AliEn FC path (/alien/cern.ch/...) or relative to work directory (no slashes)
- `plugin->SetOutputFiles(file1 file2 ...);`
  - Allows a selection of files among the analysis outputs
- `plugin->SetDefaultOutputs()`
  - Enables all outputs of the tasks connected to the analysis manager
- `plugin-SetOutputArchive(log_archive.zip:stderr,stdoutroot_`
  - Will save the standard output/error in a zip and all root files in another zip replicated in 2 storage elements
  - Note: If archiving the output you may want to omit declaring the output files

Other settings

- `plugin->SetROOTVersion(const char *rootver)`
- `plugin->SetAliRootVersion(const char *alirootver)`
  Change root and AliRoot versions when needed, according to the output of `packages` AliEn command.
- Using par files
  - `plugin->EnablePackage(package.par)`
- Using other external libraries available in AliEn
  - `plugin->AddExternalPackage("fastjet::v2.4.0")`
- Compiling single source files
  - `plugin->SetAnalysisSource(mySource.cxx)`
  - But files have to be uploaded to AliEn from current directory
  - `plugin->SetAdditionalLibs(libJETAN.so mySource.cxx mySource.h)`
    - Extra libraries to be loaded (besides AF ones) have to be enumerated in the same method.

Optional settings

- Number of files per job
  - `plugin->SetSplitMaxInputFileNumber(Int_t n)`
- Number of runs per master job
  - `plugin->SetNrunsPerMaster(Int_t n)`
- Number of files to merge in a chunk
  - `plugin->SetMaxMergeFiles(Int_t n)`
- Resubmit threshold
  - `plugin->SetMasterResubmitThreshold(Int_t percentage)`
- Process a single run per job and output to a single directory
  - `plugin->SetOutputSingleFolder(const char *folder)`

References

- AliEn Website with further documentation:

  http://alien2.cern.ch

- ALICE Analysis User Guide:
  http://project-arda-dev.web.cern.ch/project-arda-dev/alice/apiservice/AA-UserGuide-0.0m.pdf
- The AliEn Analysis Plugin: http://aliceinfo.cern.ch/Offline/Activities/Analysis/AnalysisFramework/AlienPlugin.html

Enjoy your AliEn analysis!

Inspecting the certificates

- To view the content of a .pem certificate: `openssl x509 -in usercert.pem -text -noout`
- To view only its validity: `openssl x509 -in usercert.pem -dates -noout`
- To view the content of a .p12 certificate: `openssl pkcs12 -info -nodes -in certificate.p12`

back

The `full` AliEn plugin chain

1. connect to AliEn
2. validate declared input data
3. create XML collections loading them in the AliEn user working directory
4. stream analysis manager and tasks to the file `analysis.root` in the AliEn WD
5. generate the analysis macro (e.g. `AnalysisPt.C` and copy it in the AliEn WD
6. generate the executable script that launches the analysis macro
7. generate the JDL for the batch analysis and copy it to AliEn
8. submit the AliEn job and start an alien shell
9. merge all output files that were produced and registered in the AliEn output directory
10. call the `Terminate()` method

AliEn plugin run-mode actions

- `full`: 1 to 10
- `test`: 1 to 6 plus local stuff
- `offline`: 1 to 7
- `submit`: 8 to 10
- `terminate`: 9 to 10

The recommended usage is to first run the plugin in `test` mode, for validation, only then to run it in `full` mode.