CRAB: Introduction

CMS Tier3 Workshop
12 August 2010

Eric Vaandering
CMS / Fermilab

Outline

- Overview
- Client/Server
- Usage
- Support
- Troubleshooting

CRAB:

CMS
Remote
Analysis
Builder

CRAB

- CRAB enables submission of CMSSW jobs to all CMS datasets within the data-location driven CMS computing structure
- The aim of CRAB is to hide as much of the complexity of the GRID as possible from the end user
- CRAB provides a user front-end to
  - Find data in and publish data to DBS
  - Split user jobs into manageable pieces
  - Transport user analysis code to the data location for execution (compiled on submitting node)
  - Execute user jobs, check status and retrieve output

Software Components

- CRAB interacts with many different pieces of CMS and Grid software on behalf of the user
  - DBS: What data exists?
  - Phedex: Where is it?
  - BDII/SiteDB: What sites are available, who is the user?
  - Proxies/MyProxy: User authentication
  - Dashboard: Statistics and status of jobs
  - Grid middleware: Job submission, I/O

DBS

- Data Bookkeeping Service
- What datasets exist where. What files they contain and mapping to runs, lumi blocks
- Production creates datasets and registers them with DBS
- Dataset is PrimaryDataset/ProcessedDataset/Tier
  - Primary: Describes the physics channel
  - Processed: Which software was used to process
  - Tier: Kind of information: RAW/SIM/DIGI/RECO/etc
- Homepage for DBS is http://cmsweb.cern.ch/dbs_discovery/

UI

- CRAB (CMS Remote Analysis Builder) is the CMS user front end to the GRID
- The User Interface is the GRID specific software for authentication, job submission and all other GRID interactions
- CRAB works with UI’s from EGEE and from OSG
- FNAL: pre-initialized on cmslpc.fnal.gov
- CERN initialization: `source /afs/cern.ch/cms/LCG/LCG-2/UI/cms_ui_env.(c)sh`
- You won't have to use it directly, CRAB uses it for you

CRAB Server or Client

- Originally CRAB was only standalone. Ran from CLI and interacted with Grid jobs directly
- Recently transitioning most use to CRAB Server
  - “crab” now a client interface to a server which submits and tracks jobs. Server is based on ProdAgent
  - Recommended way to work now
- Standalone still exists. Mostly useful to interact with local schedulers (condor, PBS)
  - Going away at some point. (Will discuss later.)

CRAB Server Overview

- CRABServer
  - Client-server architecture
    - A server is placed between the user and the Grid to perform a set of actions for the user
  - Main advantages
    - Automates (as much as possible) the whole analysis work flow (reduced babysitting)
    - Improves the scalability of the system
- Client-server implementation is transparent to the end users
  - Interface, installation and configuration procedures and usage remain the same as for standalone

GRID certificates

- CMS uses GRID certificates and a dedicated Virtual Organization (VO) management to have better access control for specific tasks/groups
- Your GRID certificate is important, follow all rules, don’t let it expire!
- Can be a pain the first time, but then it's over

Underlying analysis model

User runs interactively on small samples in the local environment
to develop the analysis code and test it

Once ready the user selects a large (whole) sample to submit the very same code to analyze many more events

The results are made available to the user to be analyzed interactively to produce final plots

How to configure CRAB

CRAB is controlled by a configuration file: **crab.cfg**. This file must be in the same directory as the CMSSW .py configuration file for the analysis code.

- crab.cfg is structured into **sections,** each with related settings.

The sections are:

[CRAB]
[CMSSW]
[USER]
[GRID]

- Template crab.cfg files, with comments on all the configuration parameters, are distributed with CRAB. ($CRABDIR/python/crab.cfg and full_crab.cfg)

- Only some parameters are mandatory, others depend on the user's specific needs.

Basic configuration options

[CRAB]

jobtype: defines the kind of job CRAB should run (cmssw)

scheduler: defines which flavor of GRID or batch middleware will be used by CRAB (condor_g, glite, condor, pbs)

use_server: start CRAB in server mode (recommended)

[CMSSW]

datasetpath: identifies the dataset you want to access. Enter just the datasetpath (string format /primary/process/tier) obtained from the DBS discovery page

pset : The name of the CMSSW .py file of your CMSSW job. This file must be in the same directory as the CRAB configuration file (crab.cfg).

total_number_of_lumis: Total number of lumi sections to be processed by CRAB. If set to -1, all lumis of the selected dataset are processed

lumis_per_job : Number of lumis per job. CRAB will create as many jobs as needed to process the total_number_of_lumis

Basic configuration (2)

[USER]

return_data : Defines the way CRAB handles user output. Default of 1 uses the GRID middleware sandbox.

copy_data : Output is staged out to defined SE, default is 0 to not stage out

storage_element : Defines the SE url for stage-out.

storage_path : Path on the selected SE for stage-out.

[GRID]

This is where scheduler specific information resides no matter what scheduler you use.

se_white_list : List of SE where the job can run.

se_black_list : List of SE to ban.

ce_white_list : List of CE to allow.

ce_black_list : List of CE to ban.

Datasetpath

- Can check that your selected dataset exists at least at one site using the DBS discovery page
- During creation, CRAB contacts DBS to collect all information needed for job splitting and job preparation. (Number and location of files, blocks, events.)
- If needed, sites can be excluded or selected for submission (parameters in [EDG] section of crab.cfg):
  - Primary selection depends on SE name given on DBS discovery page:
    - se_black_list: exclude sites
    - se_white_list: selects sites

```
[GRID]

se_white_list = unl.edu
```

  - Secondary selection depends on CE name given during CRAB data discovery (see submission):
    - ce_black_list: excludes CEs of sites
    - ce_white_list: selects CEs of sites

```
[GRID]

se_white_list = fnal.gov
ce_white_list = cmslcgce2.fnal.gov
```

  - “se_white_list = US_T2” selects all SEs with US_T2_* in SiteDB

Output handling

- CRAB has two ways of handling output:
  - The output sandbox
  - Copy files to a dedicated storage element (SE)
- Like the input sandbox, the output sandbox is limited in size (stand-alone only):
  - Input Sandbox: 10 MB
  - Output Sandbox: 50 MB
    - TRUNCATED if it exceeds 50 MB ➞ corrupt files
- Rule of thumb:
  - If you would like to get event files back, please use a storage element

Data Publication

- CRAB allows you to publish the results of your work and share with others through DBS
- Requirements
  - SE that allows user copied data
    - Must be able to place into CMS's LFN structure
  - A private DBS server (several centrally maintained)
    - Need to know dbs_url
  - Must know at submission time that you will want to publish data
- Full instructions are at

  https://twiki.cern.ch/twiki/bin/view/CMS/SWGuideCrabForPublication

How to run CRAB

The basic CRAB workflow is organized into 4 steps:

- Job Creation
- Job Submission
- Check job status
- Output retrieval

Job Creation: **crab –create**

At this level CRAB interacts with the DBS system, organizes the jobs of the task according to the user's job splitting parameters, packs the users specific code(/lib /module /data), prepares the script to configure the remote environment, and (using BOSS) prepares the jdl file to communicate with the RB

It also creates the working directory which is organized in 4 subdirectories named:

/job : CRAB specific stuff
/log : CRAB log file location
/res : default results destination
/share : CRAB and scheduler specific

How to run CRAB (2)

Job submission: **crab –submit**

The submission uses the previously created CRAB project to submit the jobs.
Before the real submission, CRAB always checks for available resources preventing the submission of unmatched jobs. By default all created jobs are submitted.

Job status: **crab –status**

This command checks the status of all jobs in the CRAB project.
For each job CRAB prints on the screen the job id, scheduler status, site hosting the jobs, cmssw exit code, & job exit code. The output gives also a summary with a list of job IDs sorted by status categories.By default the status of all jobs is checked.

Job output : **crab –getoutput**

This command retrieves the output of all jobs of a CRAB project which have status “Done”. By default the retrieved output files are copied in the “res” sub-dir of the CRAB workingdir. Included are the standard output and error of the jobs (CMSSW stdout and stderr) and the output files specified in crab.cfg.

Even if your job fails, run **crab -getoutput**. Otherwise your output clogs up a server.

How to run CRAB (3)

For submit, status and getoutput the user can overwrite the default behavior selecting individual jobs by:

- All : default behavior
- 1,2,3 : individual jobs
- 1-3 : job range

The latter two can be combined using commas (e.g. 1,2-4)

Using a specific directory for a CRAB project each command must be executed as:

crab –<command> -c <directory>

Installation

- A fully functional installation of CRAB requires the GLite middleware.
  - To support EEGE sites (European T1/T2)
  - Sometimes not easy, but it is not too bad
- OSG client only can work for certain stand-alone configs or for communication with the server
  - Not supported, not really recommended

Additional CRAB features

- ✓ crab -kill

  Kill jobs which have been submitted to the grid.

- ✓ crab -resubmit and -forceResubmit

  Resubmit exactly the same job (which is either abort or killed)

- ✓ crab –printId

  Find the GRID id of specific jobs (can be useful for debugging purposes)

- ✓ crab -postMortem

  Find more info about aborted jobs which may help in debugging problems.

Current Development

- CRAB must constantly keep up with changes to CMSSW and Grid middleware. Unless you have a very good reason, try to stay current with releases
- CRAB 2.x (current version) is now in maintenance mode. Only important bug fixes are being made
- CRAB3 is in the planning stages. CRAB3 will be “server only” and will be based on WMAgent, the new CMS framework for running all production and analysis jobs
  - No more direct submission, including condor/pbs
  - Allows us to consolidate limited development effort

Debugging CRAB

- GRID analysis requires many different systems to work hand-in-hand (Sites, CEs, SEs, dCache, Castor, DBS, BDII, (and all the other acronyms) ...). Many unforeseen problems can arise at the boundaries between those many systems.

- CRAB provides a common user interface and tries to notify the user of all problems. This is hard and CRAB is not always successful.

- Due to the complexity of GRID analysis, CRAB is not always the source of the problem (rarely actually). Most of the time, systems don’t behave properly or don’t play nicely together

- Backup slides:
  - give hints on how to avoid problems beforehand
  - list common problems and errors
  - try to help debugging problems and errors

How to get CRAB support

Best source for user support is the CRAB feedback hypernews:
https://hypernews.cern.ch/HyperNews/CMS/get/crabFeedback

All CRAB questions and suggestions can be posted to this forum,
CRAB developers try to solve the problems and give solutions.
User suggestions can help improve the tool.

A troubleshooting guide is at
https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookGridJobDiagnosisTemplate
(still under construction)

Questions not directly related to CRAB (GRID related problems, CMSSW specific problems, etc...) should be referred to other hypernews forums

Additional Documentation:
https://twiki.cern.ch/twiki/bin/viewauth/CMS/SWGuideCrab

Backup slides

Debugging Overview

- We'll follow the four main CRAB steps:
  - create
  - submit
  - status
  - getoutput
- But a common source of problems is completely avoidable: submitting CMSSW code with bugs/misconfigurations, so

Always test your code locally first!!

Job length

- Choose your job-splitting carefully:
  - Don’t write too much output (<= 2 GB)
  - Don’t run too long (8 - 24 hours jobs, 8 hours is optimal)
- Estimate your running time from your local tests and help GRID sites to put your job in appropriate queues

```
[EDG]

## cpu time and wall_clock_time(=real time) in minutes. Written into the jdl file
#max_cpu_time = 60
#max_wall_clock_time = 60
```

This is not always required but helps avoid problems in some cases

Output handling

- Exceeding the output sandbox size is a common source of user problems
- The output sandbox is TRUNCATED when exceeding 50 MB resulting in corrupted files
- Rule of thumb:
  - If you would like to get CMSSW ROOT files back, please use a storage element
  - If you only need histograms or ntuples back, sandbox is probably OK

SE Interaction

- The GRID is based on distributed facilities accessible to a very large number of users and uses proxies of certificates to authenticate its users against services (CEs, Ses)
  - The proxy is mapped against a local account which is different from service to service
- When you use a storage element within CRAB, it is not actually your user account which transfers the file but your proxy and the account mapped to your proxy on your SE
- So it is important that the target directory on your SE (owned by your account) is group-writable so that the mapped account of your proxy can write to it
- Examples:

```
CASTOR: rfchmod +775 /castor/cern.ch/user/u/username/subdir
dCache: chmod +775 /pnfs/cms/WAX/resilient/username/subdir
```

Creation: job splitting

- Datasets are grouped by:
  - DataSet → Block → File
- CRAB job splitting obeys block borders, meaning that a single job of a project cannot span two blocks
  - Sometimes leads to more jobs than requested
- Blocks of a single DataSet can be located at different sites
- If using white_list and black_list statements, jobs for only parts of the DataSet can be created.

Creation: TarBall creation

- CRAB ships the user's code to the site for execution. This tarball contains:
  - all libraries of the CMSSW user project directory
  - all files in all data directories in the src directory of the user's CMSSW project directory
- To not exceed CRAB's internal limitation of 10MB input sandbox:
  - Don't have too many unused libraries in your local project
  - Don't have any unnecessary files in any data directories, especially don't execute CRAB from within a data directory

Submission

- During submission job requirements are
  - Installed CMSSW software version
  - List of SEs hosting requested DataSet
  - Additional CE requirements according to ce_black/white_lists
    - Production status of site (passing SAM tests)
- If compatibility check fails, you can check the software installation status at:
  - http://cmsdoc.cern.ch/cms/ccs/wm/www/Crab/cmsC
  - http://home.fnal.gov/~burt/all_cmssoft.html
    - OSG-Only: http://home.fnal.gov/~burt/osg_cmssoft.html

Submission

- If CRAB cannot find any compatible resources:

```
crab. Matched Sites :['cmslcgce.fnal.gov', 'cmslcgce2.fnal.gov']
crab. Found 2 compatible site(s) for job 1
crab. Matched Sites :['ce102.cern.ch', 'ce106.cern.ch', 'ce107.cern.ch', 'ce113.cern.ch',
'ce119.cern.ch', 'ce121.cern.ch', 'ce120.cern.ch', 'ce114.cern.ch', 'ce123.cern.ch', 'ce108.cern.ch',
'ce118.cern.ch', 'ce122.cern.ch']
crab. Found 12 compatible site(s) for job 3
```

- the site(s) might fail tests which continuously check the status of the site (called Site Availability Monitoring, SAM). If failing a defined number of tests, the site is

```
crab. No compatible site found, will not submit job X
```

  - https://lcg-sam.cern.ch:8443/sam/sam.py

- If the job cannot be submitted at all because no compatible resources are found and the expected sites from the data discovery fail SAM tests, wait till the problems get fixed and submit again after a while.

Status check

```
crab. crab (version 1.5.2) running on Mon Jun 11 17:51:55 2007


crab. Working options:
  scheduler           edg
  job type            CMSSW
  working directory
/afs/cern.ch/user/s/spiga/scratch0/Tutorial/CMSSW_1_3_1/src/Demo/MyTrackAnalyzer/test/crab_0_070611_17401
4/


crab. Checking the status of all jobs: please wait
Chain    STATUS             E_HOST                                  EXE_EXIT_CODE JOB_EXIT_STATUS
-------------------------------------------------------------------------------------------------
1        Running            cmslcgce2.fnal.gov
2        Running            cmslcgce.fnal.gov
[.........................................]
9        Running            cmslcgce.fnal.gov
10       Running            cmslcgce.fnal.gov


>>>>>>>>> 10 Total Jobs


>>>>>>>>> 10 Jobs Running
          List of jobs: 1,2,3,4,5,6,7,8,9,10
crab. Log-file is
/afs/cern.ch/user/s/spiga/scratch0/Tutorial/CMSSW_1_3_1/src/Demo/MyTrackAnalyzer/test/crab_0_070611_17401
4/log/crab.log
```

Important information for debugging: site the job ran on (E_HOST)

Status check

- Important to know: The difference between job status “aborted” and “done”:
  - “done” jobs actually ran at the site while “aborted” jobs indicate a problem with the GRID infrastructure (mainly RB).
- The cause of job aborts can be investigated using

```
crab -postMortem <job>
```

- producing an ascii file with the output of

```
edg-job-get-logging-info -v 2 <https grid-id>
```

- summarizing the cause of the problem
- When looking for help in case of aborted jobs, it’s usually a good idea to provide the postMortem output for debugging purposes

Getoutput

- One error which might occur is related to the RB:

```
Error caught Unable to retrieve the status for:
https://gdrb03.cern.ch:9000/DYZIjEKTNRiB44am8d7RBQ
    edg_wll_JobStatus: Connection refused: edg_wll_ssl_connect(): server closed the
connection, probably due to overload
    crab. Unable to retrieve the status for:
https://gdrb03.cern.ch:9000/DYZIjEKTNRiB44am8d7RBQ
    edg_wll_JobStatus: Connection refused: edg_wll_ssl_connect(): server closed the
connection, probably due to overload
```

- One possible reason is a full disk on the RB. This problem might go away by itself. If not report it and deactivate / change the RB configuration