CRAB: status and perspective

Stefano Lacaprara

INFN Legnaro National Laboratory

DM/WM Workshop, CERN 22-Feb-2007

Goal and Status

- CRAB CMS Remote Analysis Builder
  user oriented tool for grid submission and handling of analysis jobs

Goal of the project

provide an user friendly interface for end user interaction with the grid for CMS, including interaction with data management, middleware, remote computing element, basic monitoring functionalities, etc. . . hiding as much as possible the grid complexities to final user.

Status

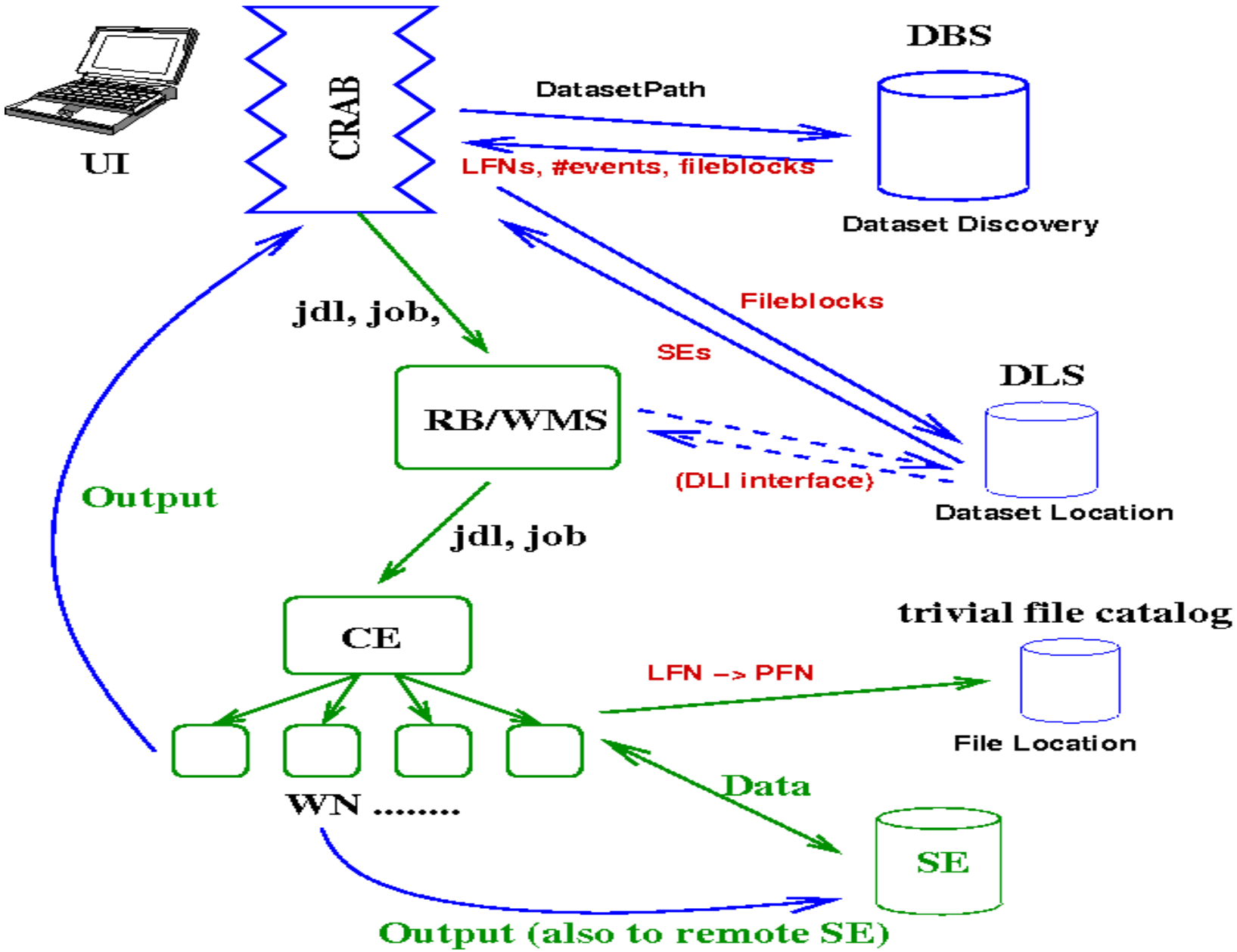
In production. Used since more than two years by end users.
The only computinig tool in CMS used by generic physicist.

Underlying analysis model

Batch-like, simple analysis model

- User runs interactively on small samples somewhere in order to develop his code and test it.
- Once happy, select large sample(s) and submit the very same code to analyze xillions of events
- The results is made avaialble to user to be analyzed interactively
- iterate ...
- Analysis can be done in step, saving the intermediate ones and iterate over the latest ones.

Workflow


UI
CRAB
DBS
DatasetPath
LFNs, #events, fileblocks
Dataset Discovery
jdl, job,
Fileblocks
SEs
DLS
RB/WMS
(DLI interface)
Dataset Location
Output
jdl, job
CE
trivial file catalog
LFN –> PFN
File Location
WN .....
Data
SE
Output (also to remote SE)

Requirements

To run successfully a job, CRAB requires:

- CMSSW is installed at remote site (CE) and tag properly published
- Site is configured with Trivial File Catalog and Site Local Configuration (to be sourced by CRAB wrapper on WN to setup local CMS environment)
- Data already available at site SE and properly published on DM system (DBS/DLS). Data driven model: CMS baseline does not foresee any *on-demand* data movement triggered by jobs.
- Pool/Root compilant access to data on SE from WN (posix, rfio, dcap, ...). The actual protocol is defined in TFC.

CRAB Features

- Trivial installation (tar-ball + run a configuration script): can be done also for a group of users (eg at CERN)
- User setup a configuration file `crab.cfg` (input dataset, how many events, splitting, where to put output, which RB, and much more, see tomorrow tutorial)
- Interact with user working area (SCRAM, CMSSW), including modification of User ParameterSet (.cfg), packing of user private code and libraries, etc ...
- Interact with DM to get infos about dataset and location. Using these informations a proper job splitting is done, taking into account user requirements and data distribution.
- Interact with grid: setup of `jdl`, job wrapper, actual submission, status check, output retrieval, kill, etc ...
- Interact with Dashboard for monitoring purpose.
- Interact with user, providing a very simple interface which hides (as much as possible) the complexities underneath.

CRAB Features (II)

- Uses BOSS for interaction with schedulers and internal logging & bookkeeping;
- Support for EDG submission (RB), G-Lite (WMS) including bulk submission;
- Support for Condor-G submission (OSG to OSG sites);
- Inter-operability with LCG - OSG Fully supported submission, via RB (WMS), to OSG sites (as well as EGEE ones, of course). Fully transparent to final user (not to CRAB developer)
- Support also for limited MC generation (i.e. jobs with no input dataset), asked by many users.
- Output handling including return to UI or copy to SE (see next slides).

Interaction with DataManagement

- At UI Level
  - Query to DBS: get detailed info about dataset (# files, LFNs, # events, etc . . .)
  - Query to DLS to get list of SEs. This can be done also at RB (WMS) level, using the DLI interface of DLS: not done yet.
- At WN level
  - LFN to PFN resolution, using local TFC
  - No interaction with non local services
  - Local to local and global to global
- Output handling (see next)

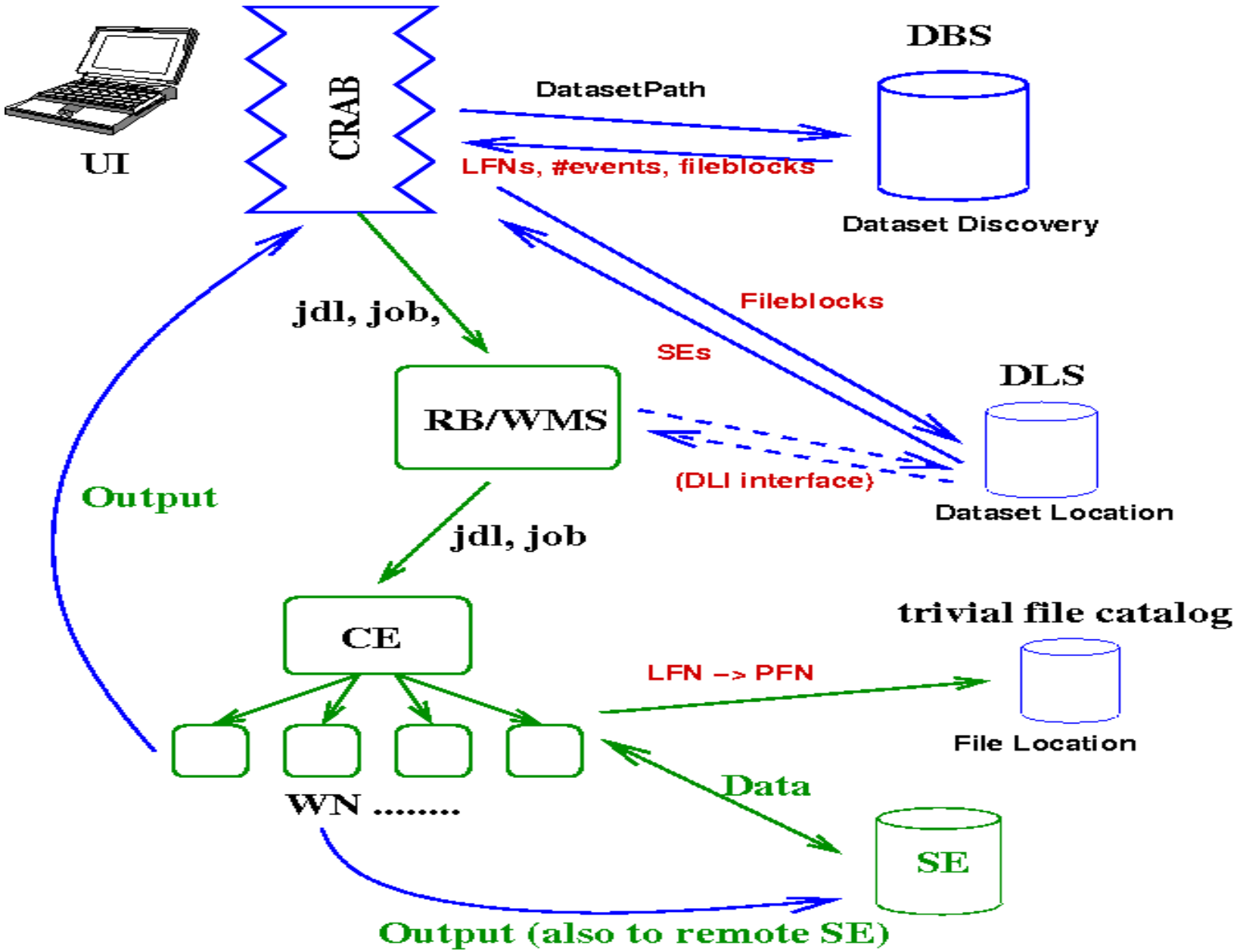
Output handling

- Option 0
  - Get back the output to UI via output sandbox
  - Handy, very simple
  - Can produce heavy load on RB: limited in size
- Option 1
  - Save output to SE (typically "close" to user);
  - Close means users has posix-like access to it (eg can open via `root`)
  - No (almost) limit on size;
  - More complex: user has to define (and thus know) a SE close to him
  - No reliable file transfer mechanism available: must do "by hand", via multiple tries
  - Ownership issue: files are owned by `cms003` not as `slacapra`
- Output publication (next)

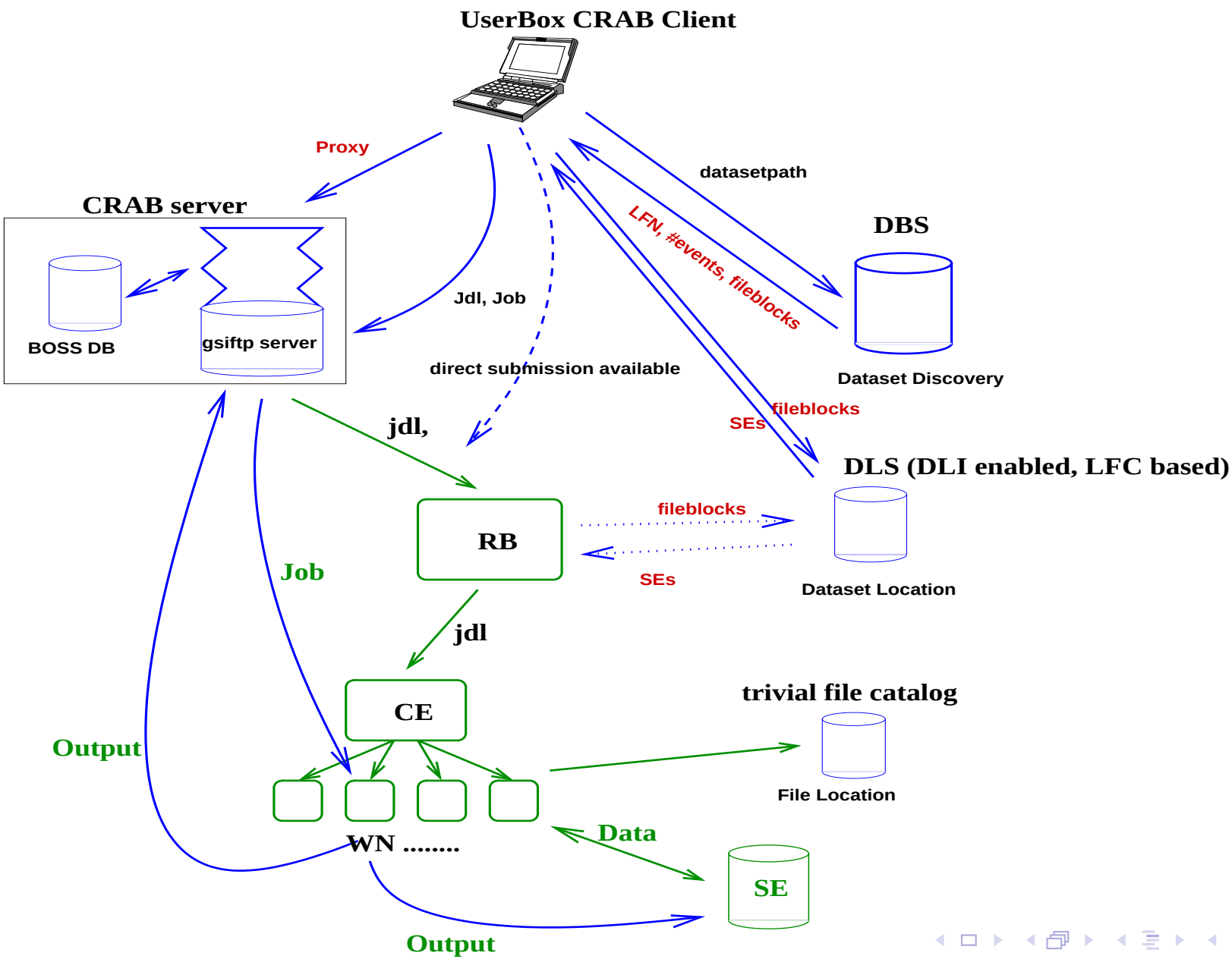
User output publication

- What? Only EDM output, not private one (eg histos, ascii, private trees, . . .)
- Where? Logical and pysical
  - Logical Local scope DBS/DLS (user level or group level). Can also go to global scope if wanted.
  - Difference is only is visibility, not information content
  - Need to enforce a user namespace for dataset names `UserStefanoLacapraraHiggsGoldSample/CMSSW_2_3_0/DST`
  - Physical AFAIU from CMS computing model, on "his" T2 (T3?) SE
  - Enforce namespace as well
- When? First produce output and collect needed infos.
- decide afterwards if (and where) to publish
  - Use cases: run on X sample with improved jet algo
  - 1^{st} try: bug found on code. 2^{nd} try: bad parameter used in cfg.
  - 3^{rd} try: good. Publish after physics check.

CRAB Server


UI
CRAB
DBS
DatasetPath
LFNs, #events, fileblocks
Dataset Discovery
jdl, job,
Fileblocks
SEs
RB/WMS
DLS
(DLI interface)
Dataset Location
Output
jdl, job
CE
trivial file catalog
LFN –> PFN
File Location
Data
WN .....
SE
Output (also to remote SE)

CRAB Server


UserBox CRAB Client
Proxy
datasetpath
CRAB server
DBS
LFN, #events, fileblocks
Jdl, Job
BOSS DB
gsiftp server
direct submission available
Dataset Discovery
fileblocks
SEs
jdl,
DLS (DLI enabled, LFC based)
RB
fileblocks
Job
SEs
Dataset Location
jdl
CE
trivial file catalog
Output
File Location
WN .....
Data
SE
Output

Goal

- Automatize as much as possible the interaction with the grid, including submission, resubmission, error handling, output retrieval, etc ...
- Try to reduce the unnecessary human load.
- Try to improve scalability of the whole system;
- Provide a "service" to physicist to take care of task once defined by user
- Move all possible action to server side, reducing to a minimum those on client side
- The direct submission to the grid (CRAB-standalone) will be supported. Live switching from standalone to server as well.

Generalities

- Re use as much as possible the existing code and tools
- Maintain at all cost the simple CRAB interface for user
- The server adopt the general PA architecture: Components are implemented as independent agents, communication is performed through an asynchronous and persistent message service
- Plus a grid-Ftp server

Workflow

- `crab -create` as in standalone, interact with DM, with user working area and set up the jobs;
- `crab -submit` submit task to server and not to the grid. The server will take responsability of the task and its completion. This include shipping from client to server of payload and user proxy;
- `crab -status` ask to server the status of the job. The server check the status of the task quering L&B asyncrhonously;
- `crab -getoutput` retrieve to client the output already retrieved to server from the grid;

`gridftp server` allows for all comunication from client to server (apart from proxy). It is also used to store input and output sandboxes: with glite WMS w/o going through WMS itself.

Components

DropBoxGuardian Check the dropBox for new CRAB user task to be managed.

ProxyTarAssociator Associate the task to the right user proxy. The Proxy is shipped to server with a LCG compliant mechanism (developed initially by ARDA group) which ensure securety.

CrabWorker submit jobs to the grid in the CRAB style. It fully reuse the "submitter" components of CRAB, so support EDG/glite/glite-bulk/condor_g submissions.

JobTraking Track every single job (via BOSS) reuse PA component

ErrorHandler Perform a basic error handling reuse PA component

JobSubmitter Resubmit single job if needed reuse PA component

TaskTracking Keep general informations about all tasks under execution.

Notification Notify the user when the task is ready via e-mail.

Issues

- Where to deploy Servers? T2
- Server Disk Space management
- Scalability: we already know (from JR experience) that we can manage of the order of tens kjobs per server, probably more with glite and bulk

Status and Plan

Detailed plan at
`https://twiki.cern.ch/twiki/bin/view/CMS/CRABPlan`

- Feb. : Support for DBS2 and migration to SL4 (almost done)
- Feb. : prototype 0 with basic functionalities for CRAB Server to be used by beta tester (done)
- Mar. : publication on DBS of user output
- Apr. : add more functionalities (kill jobs, more advanced error handling, server disk space management, . . .) to CRAB Server and increase number of users O(10)
- May. : scalability test of CRAB server
- Jun. : CRAB Server available to users

Summary

- CRAB is up and alive;
- Used with success for more than 2 years;
- Used also to access (using offcial DM tools) MTCC data as well as Tracker real data;
- Next major step is publication of user output to DM system;
- Firt prototype of CRAB Server setup and under test;

User support

In spite of dedicated effort and good result, it is still a weak point. All possible effort must be done to guarantee training and support. The CMS Computing system must include physicist!

CRAB team and contributors

Stefano Lacaprara, Daniele Spiga, Federica Fanzago, Marco Corvo, Oliver Gutsche, Fabio Farina, Carlos Kavka, Mattia Cinquilli, Alessandra Fanfani, William Bacchi, Alvise Dorigo, Giuseppe Codispoti (boss)