WMS and Computing Resources

A. Tsaregorodtsev,
CPPM-IN2P3-CNRS, Marseille,
DIRAC User Workshop, Warsaw, 29-31 May 2017

DIRAC
THE INTERWARE

Plan

- WMS
- Computing Resources
- VMDIRAC
- WMS Web applications

Parametric jobs

- Parametric jobs enhancement
  - Possibility to define multiple sets of parameters, for example:

```
[
   Executable = "/bin/echo";
   Arguments = "Job number %s of %(FirstName)s %(LastName)s";
   Parameters.FirstName = {"Zoltan","Chris","Federico"};
   Parameters.LastName = {"Mathe","Haen","Stagni"};
   Parameters = 3;
   ParameterStart = 1;
   ParameterStep = 1;
]
```

  - As before parameter sequences can contain:
    - List of strings
    - List of lists of strings
    - Numbers calculated by a formula

Parametric jobs cont'd

- In the Job API:

```
def setParameterSequence( self, name, parameterList, addToWorkflow = False ):
  """ Function to define a sequence of values for parametric jobs.

  :param str name: sequence parameter name
  :param parameterList: list of parameter values
  :type parameterList: python:list
  :param bool addToWorkflow: flag to add parameter to the workflow on the fly, if str, then
                             use as the workflow parameter
  :return:
  """
```

  - addToWorkflow argument allows to add the parameter value to the job XML description on the fly at the execution time
    - Single jobDescription.xml file (single Input Sandbox)
    - Needed for bulk submission of jobs in the Transformation System

Bulk submission

- Transformation System enhancement
  - Transformation Agent can be configured to submit a bunch of jobs in one operation:
    - Controlled by **BulkSubmission** configuration flag
  - The operation is not transactional
    - No strict guarantee that in case of failure no jobs will be actually submitted
  - Transactional version of the bulk submission is in the works:
    - Bulk submission of jobs in the status that must be confirmed by the client
    - Client checks that all the requested jobs are created
    - Finally, enables all the created jobs in one transaction
  - The transactional bulk submission uses the mechanism of asynchronous job submission:
    - Master parametric job is submitted
    - Actual jobs are created in an executor
    - Client can interrogate the list of created jobs by giving the master job ID

MultiProcessor Jobs

- Jobs can specify the following parameters:

```
[
   Executable = “/bin/echo”;
   Arguments = “MultiProcessor job”;
   WholeNode = yes;
   …
]
[
   Executable = “/bin/echo”;
   Arguments = “MultiProcessor job”;
   NumberOfProcessors = 6;
   …
]
```

- NumberOfProcessors and WholeNode parameters will be transformed into corresponding tags for the matching mechanism

Running on MultiProcessor resources

- Different approaches for running jobs on computing resources with multiple processors
  - Starting multiple pilots per processor (e.g. in the current VMDIRAC, Vac/VCycle)
  - Starting multiple job agents inside one pilot *(see Andrew's talk)*
  - Using single pilot with a job agent configured to use PoolComputingElement

PoolComputingElement

- PoolComputingElement is used inside the JobAgent started by a pilot
- PoolComputingElement is managing a small "batch system" on the worker node
  - Allows to run several jobs in parallel per processor keeping track of occupied resources
  - Each job can be run in an individual "Sudo" envelope
- Multiple strategies can be realized in the PoolComputingElement
  - Only multiple single-processor jobs
  - Only WholeNode jobs
  - A mixture of jobs with different requirements (realized now)
  - Different job scheduling strategies are to be evaluated
    - Strategy plugin mechanism should be introduced

MultiProcessor resources

- Computing Elements or Cloud VMs can specify WholeNode and NumberOfProcessor properties in their configuration
  - Those properties are discovered by a pilot at run time and presented to the Matcher
  - WholeNode can be a "RequiredTag" in order to force matching only jobs with this requirement
- The matching is done only on the basis of the configuration data discovered by the pilot
  - No special MultiProcessorSiteDirector is needed any more
  - Need more study on how to account the CPU time for multiprocessor jobs

SiteDirector

- SiteDirector is the DIRAC Pilot Factory
  - Sending pilots to suitable sites according to the status of the TaskQueues
  - Up to now was using the SiteMask to keep track of the banned/allowed sites
- Using RequestStatus System general mechanism to get the computing resources status information
  - Using the status of Sites and ComputingElements
  - Not using the status of the Queues
    - will be added later as soon as RSS will be able to deal with those
- The old SiteMask mechanism can still be used
  - For those installations that do not enable RSS for the status information

SiteDirector

- SiteDirector interrogates ComputingElement services to get their status information
  - Avoiding overloading of local batch queues
- It can be undesirable to interrogate some ComputingElement services
  - E.g. ARC Computing Element status is obtained from the BDII index
- ComputingElement can define **QueryCEFlag** configuration option to instruct the SiteDirector not to interrogate the service
  - In this case the estimation of the ComputingElement status is done based on the information in the PilotAgentsDB
    - This is a less precise information but still can be better for certain CEs

Pilots framework

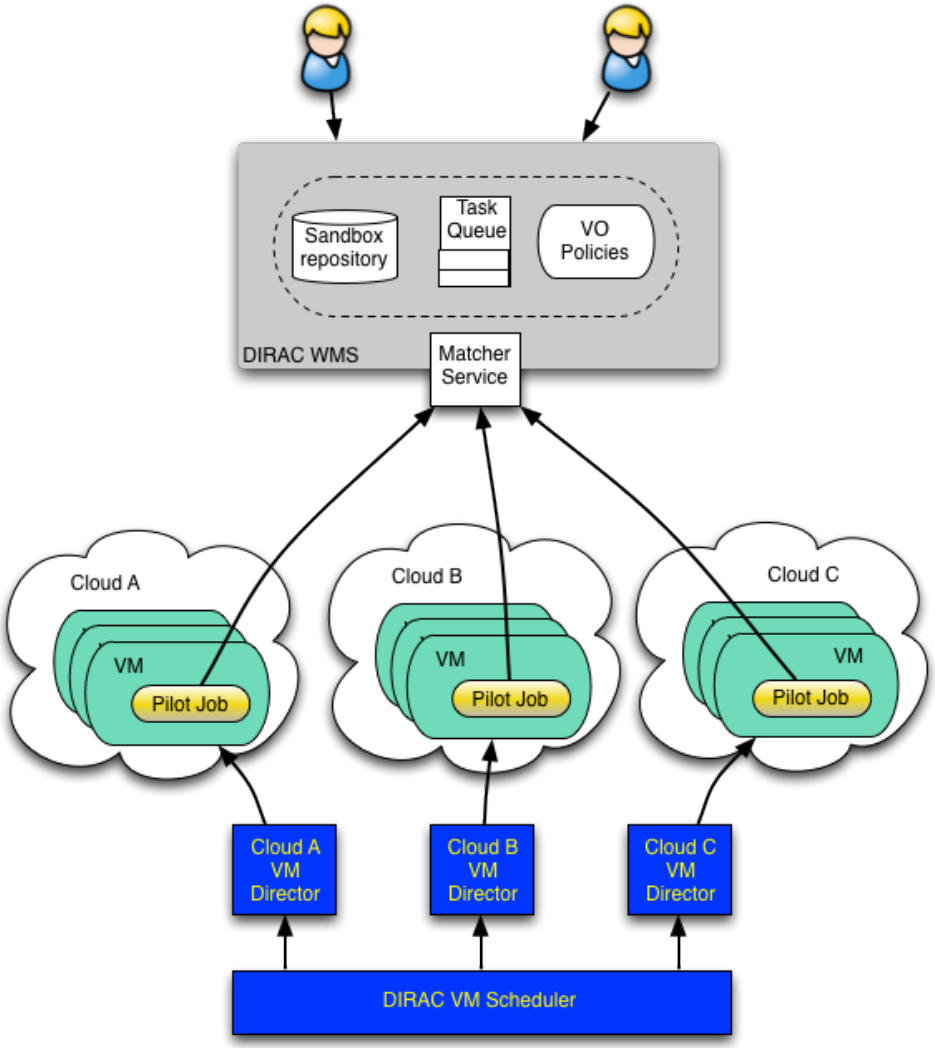
- The pilots are now used in multiple contexts
  - Grids, Clouds, Boinc
- Pilots are starting in the DIRAC agnostic environment
  - Bringing DIRAC software as part of the initialization process
- That is why the pilot framework is being separated into an independent subproject ***Pilot*** (*see Andrew's talk*)
  - https://github.com/DIRACGrid/Pilot
  - Not in production yet
- This is the place where also the bootstrapping codes for virtualized environments are kept
  - Now only for the Vac/Vcycle environments
  - Must be also suitable for the VMDIRAC bootstrapping

Computing Resources

- No major new developments were done recently for Computing Resources
  - Several minor fixes
  - For Cloud resources see below
- Currently available Computing Resources
  - CREAM
  - ARC
  - HTCondor
  - SSH/GSISSH with the following backends:
    - Single or multiple hosts
    - Batch systems: Torque, Condor, GE, LSF, OAR, SLURM

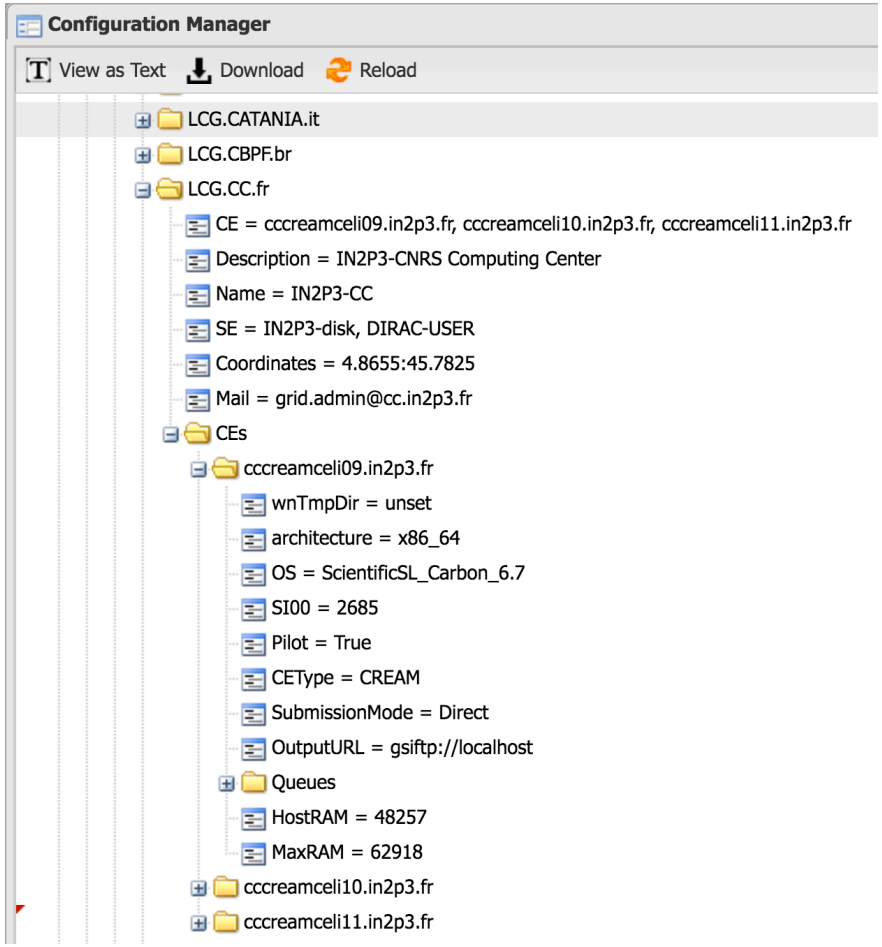
Clouds

- VM scheduler
  - Dynamic VM spawning taking Task Queue state into account
  - Discarding VMs automatically when no more needed
- The DIRAC VM scheduler by means of dedicated VM Directors is interfaced to
  - OCCI compliant clouds:
    - OpenStack, OpenNebula
  - Apache-libcloud API compliant clouds
  - Amazon EC2

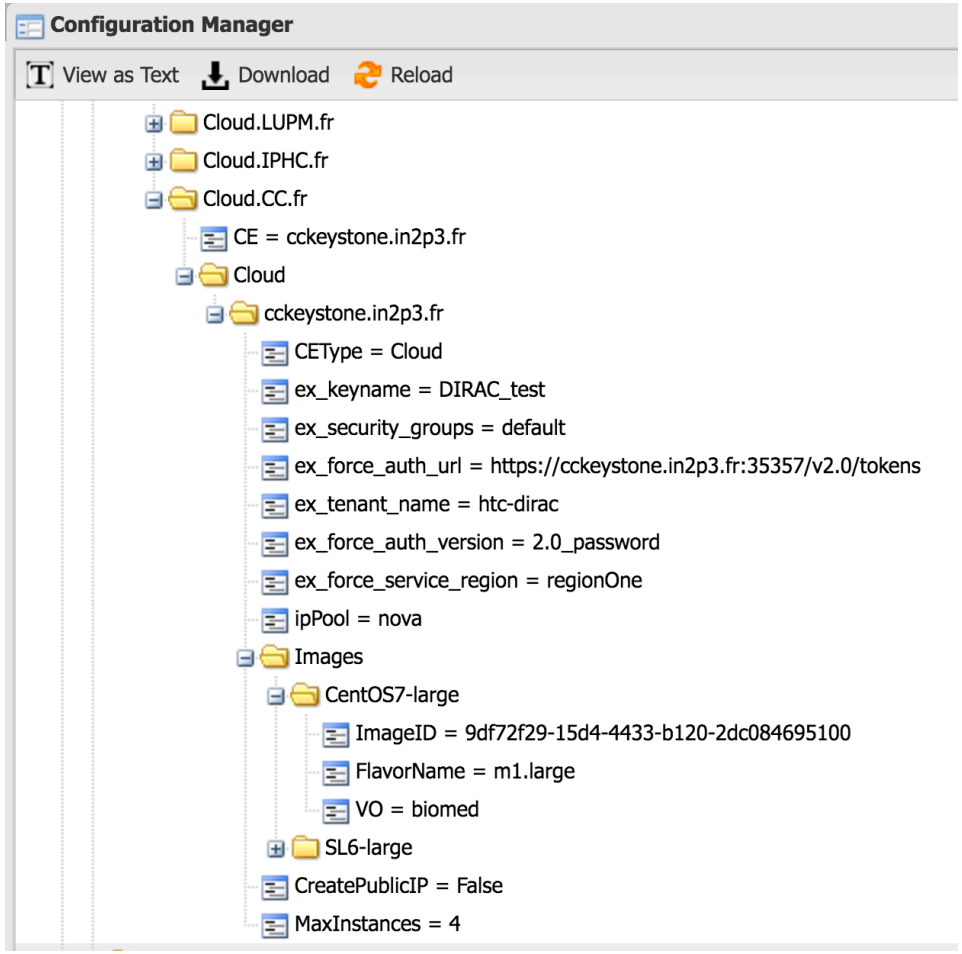


Cloud resources configuration

Grid sites



Cloud sites



VMDIRAC 2

- VM submission
  - Cloud endpoint abstraction
  - Implementation(*IHEP, Beijing*)
    - Apache-libcloud
      - Catch-all library, but not really…
    - Rocci
      - Using command line interface
      - Allow connections with GSI proxies
    - EC2
      - Boto python API
  - Other implementation with direct access to the REST service interfaces is in the works
  - CloudDirector similar to SiteDirector
- ToDo
  - Cloud endpoint testing/monitoring tools for site debugging
    - Follow the endpoint interface evolution

VMDIRAC 2

- VM contextualization
  - Standard minimal images
    - No DIRAC proper images, no image maintenance costs, but …
  - Cloudinit mechanism only
  - Using a passwordless certificate passed as user data
  - Using bootstrapping scripts similar to LHCb Vac/Vcycle
    - Using pilot 2.0
  - On the fly installation of DIRAC, CVMFS, …
    - Takes time, can be improved with custom images
  - Starting as many pilots as they are cores (single core jobs)

Managing VM life cycle

- VM Monitor Agent is launched in parallel with the pilot process during the VM bootstrapping
  - This is a watchdog for activities on the VM
  - Sends heartbeats and VM status information to the central VM Manager service
  - Can receive instructions from the central service as a response to the heartbeat
    - E.g., halt, drain and other commands
  - Monitors the VM status
    - CPU load
    - Pilots status via log files
  - Can be configured to halt the VM with different policies
    - Strict life time, à la batch system
    - Zero CPU load
    - No active payloads

VMDIRAC 2

- VM web application
  - Basic VM monitoring and controls
- Needed developments
  - Enhanced monitoring, accounting
    - No Google tools !
  - VM manipulation by administrators
    - Start, halt, other instructions to the VMMonitor agent
  - Possibility to connect to VM to debug problems
    - Web terminal console
    - On the fly public IP assignment

Web Portal

- No major developments for the WMS related web pages
- Developed a link between the File Catalog application and the Launchpad application
  - On request of the Eiscat 3D community
    - In the EiscatDIRAC extension
    - Will be imported into the core DIRAC
  - Selected files are appearing in the Launchpad job description as InputData with a simple button press

DIRAC
THE INTERWARE

Conclusions

- Stable functioning of the WMS subsystem
- Several pressing developments
  - Transactional bulk job submission
  - Managing multi-processor payloads
  - Cloud subsystem (Pilot,VMDIRAC)
- Many opportunities for contributions

DIRAC
THE INTERWARE

Back-up slides

VM management