WMS and Resources

A. Tsaregorodtsev,
CPPM-IN2P3-CNRS, Marseille,
22 May, DIRAC User Workshop,
CC/IN2P3, Lyon
Plan

- Pilot based WMS
- Pilots as resources federators
- Resources types that can be connected
- User perspective
- Increasing the scale with bulk operations
- Conclusions
Pilot jobs are submitted to computing resources by specialized Pilot Directors.

After the start, Pilots check the execution environment and form the resource description:
- OS, capacity, disk space, software, etc.

The resources description is presented to the Matcher service, which chooses the most appropriate user job from the Task Queue.

The user job description is delivered to the pilot, which prepares its execution environment and executes the user application.

In the end, the pilot is uploading the results and output data to a predefined destination.
One evident advantage is that the users’ payload is starting in an already verified environment

- In early days of the grid and even now users saw an important decreasing of their jobs failure rate

The environment checks can be tailored for specific needs of a particular community by customizing the pilot operations
Pilot based WMS

- Site resources providers does not need to distinguish individual users
  - One user identity represents the whole community to the sites
  - Simplifies site management but needs special trust relation between the site and the community
- Sites does not need to organize local resources to meet the community requirements
  - E.g. special queues per community groups with special fair sharing
- Adding new sites to the pool of DIRAC managed resources is considerably simpler
  - DIRAC does not require special services to be deployed on sites
    - There are exceptions (see the HPC case below)
Pilot based WMS

- User jobs submitted to the system are not passed immediately to a selected site but wait in the central repository – Task Queue
  - Very efficient job matching to the site properties (see below)
  - Possibility to apply community policies by dynamically adjusting the job priorities
    - Similar mechanism to batch systems fair sharing mechanism
    - Job priorities can be adjusted using community specific plugins
    - Standard plugins include static group shares
    - Job priorities of users in the same group are dynamically adjusted based on the recent history of the consumed resources as provided by the Accounting service
What makes job matching efficient

- Single central Task Queue serving requests from the army of pilots (>100K simultaneous jobs demonstrated)
- User jobs are classified in groups with identical requirements which reduces by orders of magnitude the number of entities to match
- Very efficient matching mechanism based on the MySQL databases engine
  - No JDL matching operation
- Typical $O(10^{-1})$ sec matching time
  - Even with $O(10^5)$ jobs in the waiting queue
Pilots can be deployed on a computing resource in different ways

- Sending to Computing Elements (CREAM, HTCondor, ARC or SSH +BatchSystem) by the central Pilot Factory
- Instantiating VMs in Cloud resources on request from a central CloudDirector (VMDIRAC). Pilots are downloaded in the bootstrapping process and started with the provided configuration parameters
- Starting VMs sporadically “from vacuum” (Vac/Vcycle). Pilots are downloaded and started similarly to the previous case

Pilots are started in a DIRAC free environment. The pilot codes as well as tools for performing the bootstrapping of the pilot environment are collected in a new independent package Pilot 3.0 (see presentation by Wojciech Jan Krzemien)

- Release v6r20 WMS can work with Pilots 3.0 although not enabled by default
DIRAC was initially developed with the focus on accessing conventional Grid computing resources
- WLCG grid resources for the LHCb Collaboration
- It fully supports multiple grid middlewares and infrastructures
  - EGI, WLCG, OSG, NorduGRID, etc
- Other types of grids can be supported
  - As long we have customers needing that

**Standalone clusters**
- Access through SSH/GSISSH tunnel
- Batch systems supported: LSF, BQS, SGE, PBS/Torque, Condor, OAR, SLURM
  - Used to access HPC centers

**BOINC Volunteer resources**
- Running pilots on volunteer machines
- Separation of secure and unsecure parts, plugins for results validation

*More on opportunistic resources in a presentation tomorrow*
VMDIRAC extension

- VMDIRAC extension developed for Belle MC production system
  - Dynamic VM spawning taking Amazon EC2 spot prices and Task Queue state into account

- Now VMDIRAC is a general purpose service for VMs life cycle management
  - Creation
  - Monitoring
  - Discarding
Cloud endpoint plugins to interact with particular cloud provides

Cloud endpoint abstraction

- Implementations (IHEP, Beijing)
  - Apache-libcloud
    - Catch-all library, but not really...
  - Rocci
    - Using command line interface
    - Allow connections with GSI proxies
  - EC2
    - Boto python API
  - Occi
    - Using OCCI REST interface directly

More implementations are in the works

- OpenNebula, Google, Azur, IBM, …
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
  - Can be configured to halt the VM with different policies

- **VM Scheduler** orchestrates spawning and halting virtual machines depending on the Task Queue status, Accounting history
  - Necessary for fair sharing of cloud resources
  - Work in progress
User interfaces

- For the users all the internal WMS/pilots machinery is completely hidden. They see all the DIRAC operated computing resources as single large batch system
  - Command line (COMDIRAC style):

    ```
    [atsareg:-/work/test/DiracTest] $ dsub /bin/echo "Hello world"
    946
    [atsareg:-/work/test/DiracTest] $ dstat
    JobID  Owner  JobName  OwnerGroup  JobGroup  Site  Status  MinorStatus  SubmissionTime
    946  atsareg  Unknown  dirac_tutorial  00000000  LCG.OBSPM.fr  Matched  Job Received by Agent 2018-05-21 23:54:48
    [atsareg:-/work/test/DiracTest] $ doutput 946
    [atsareg:-/work/test/DiracTest] $ ls -l 946
    total 8
    -rw-r--r-- 1 atsareg staff 12 May 22 01:54 std.out
    ```
  - DIRAC API (see Developer hands-on session)
  - Web Portal
## Jobs in the Web Portal

**Job Monitor**

<table>
<thead>
<tr>
<th>Site</th>
<th>Status</th>
<th>Application Status</th>
<th>Site</th>
<th>Job Type</th>
<th>Last Update [UTC]</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCG.GRIF.fr</td>
<td>000...</td>
<td></td>
<td>LCG.GRIF.fr</td>
<td></td>
<td>2018-05-20 13:21:16</td>
</tr>
<tr>
<td>LCG.GRIF.fr</td>
<td>000...</td>
<td></td>
<td>LCG.GRIF.fr</td>
<td></td>
<td>2018-05-20 13:17:02</td>
</tr>
<tr>
<td>LCG.GRIF.fr</td>
<td>000...</td>
<td></td>
<td>LCG.GRIF.fr</td>
<td></td>
<td>2018-05-20 13:21:17</td>
</tr>
<tr>
<td>LCG.GRIF.fr</td>
<td>000...</td>
<td></td>
<td>LCG.GRIF.fr</td>
<td></td>
<td>2018-05-20 13:21:40</td>
</tr>
<tr>
<td>LCG.GRIF.fr</td>
<td>000...</td>
<td></td>
<td>LCG.GRIF.fr</td>
<td></td>
<td>2018-05-20 13:18:31</td>
</tr>
<tr>
<td>LCG.GRIF.fr</td>
<td>000...</td>
<td></td>
<td>LCG.GRIF.fr</td>
<td></td>
<td>2018-05-20 13:19:00</td>
</tr>
<tr>
<td>LCG.GRIF.fr</td>
<td>000...</td>
<td></td>
<td>LCG.GRIF.fr</td>
<td></td>
<td>2018-05-20 13:20:23</td>
</tr>
<tr>
<td>LCG.GRIF.fr</td>
<td>000...</td>
<td></td>
<td>LCG.GRIF.fr</td>
<td></td>
<td>2018-05-20 13:21:31</td>
</tr>
<tr>
<td>LCG.GRIF.fr</td>
<td>000...</td>
<td></td>
<td>LCG.GRIF.fr</td>
<td></td>
<td>2018-05-20 13:26:24</td>
</tr>
<tr>
<td>LCG.GRIF.fr</td>
<td>000...</td>
<td></td>
<td>LCG.GRIF.fr</td>
<td></td>
<td>2018-05-20 13:20:49</td>
</tr>
<tr>
<td>LCG.GRIF.fr</td>
<td>000...</td>
<td></td>
<td>LCG.GRIF.fr</td>
<td></td>
<td>2018-05-20 13:25:34</td>
</tr>
<tr>
<td>LCG.GRIF.fr</td>
<td>000...</td>
<td></td>
<td>LCG.GRIF.fr</td>
<td></td>
<td>2018-05-20 13:20:29</td>
</tr>
<tr>
<td>LCG.GRIF.fr</td>
<td>000...</td>
<td></td>
<td>LCG.GRIF.fr</td>
<td></td>
<td>2018-05-20 13:26:02</td>
</tr>
<tr>
<td>LCG.GRIF.fr</td>
<td>000...</td>
<td></td>
<td>LCG.GRIF.fr</td>
<td></td>
<td>2018-05-20 13:25:37</td>
</tr>
</tbody>
</table>

**Job Launchpad**

- **Executable**: `/bin/ls`
- **JobName**: `DIRAC_atsareg_352816`
- **Arguments**: `-ltrA`
- **Output Sandbox**: `std.out, std.err`

**Proxy Status**: Valid

**Input Sandbox**

**LFN**: [Browse]
Bulk job operations

- Submit many jobs with one command all belonging to one JobGroup
  - Jobs executing similar payloads (parameter sweeping)
    - Sharing input sandbox
  - Examine the status of jobs in a given group
  - Get outputs of jobs in a given group

- Bulk submission and group operations can be the basis for small data production systems for relatively small communities

- For full-fledged data-driven workflows one should consider using the Transformation System (see presentation by Luisa Arrabito)
Transactional bulk submission

- Bulk submission is available since several years, however its use was limited because of the no guarantee that all the requested jobs are successfully submitted.

- Failures in actual submission of part of the jobs can damage significantly production systems
  - Difficult to keep the production bookkeeping consistent
  - Some jobs can be submitted without the Production System knowing about them

- Bulk submission with a high guarantee of the success of the overall operation was needed
Transactional bulk submission

- Pure transaction in the database sense cannot be implemented
  - Several services and multiple database queries are involved in the operation

- The procedure was defined as the following
  - Bulk job submission is initiated by the client
  - Bulk job description is received and all the necessary operations to enter the jobs into the JobDB are performed
  - Jobs are entered with status Submitting which does not allow their further processing
  - The client receives the list of newly created job IDs and checks that the number of jobs is consistent
  - Finally, the client confirms the bulk operation sending back the list of jobs. This results in a single transaction in the database changing all the jobs’ status to Received

- If some failure happens at any step, the client will get an error and the jobs in the worst case will stay in Submitting status. Those jobs will be eventually removed by the JobCleaningAgent
Conclusions

- Pilot based WMS has proven to be efficient in the HEP experiments
- It is now available for the users of dedicated and multi-VO DIRAC services
- A large variety of computing resources can be federated due to the pilot mechanism
- Users can build personal and community production systems with the help of bulk job operations provided by the DIRAC WMS