MPI Service

Vanessa Hamar

CC-IN2P3

Overview

- MPI Applications
- MPI DIRAC Extension
- MPI Pilot Roles
- MPI Ring Life Cycle
- Distributed File System
- Application Server
- Conclusions
- Future work

MPI Applications

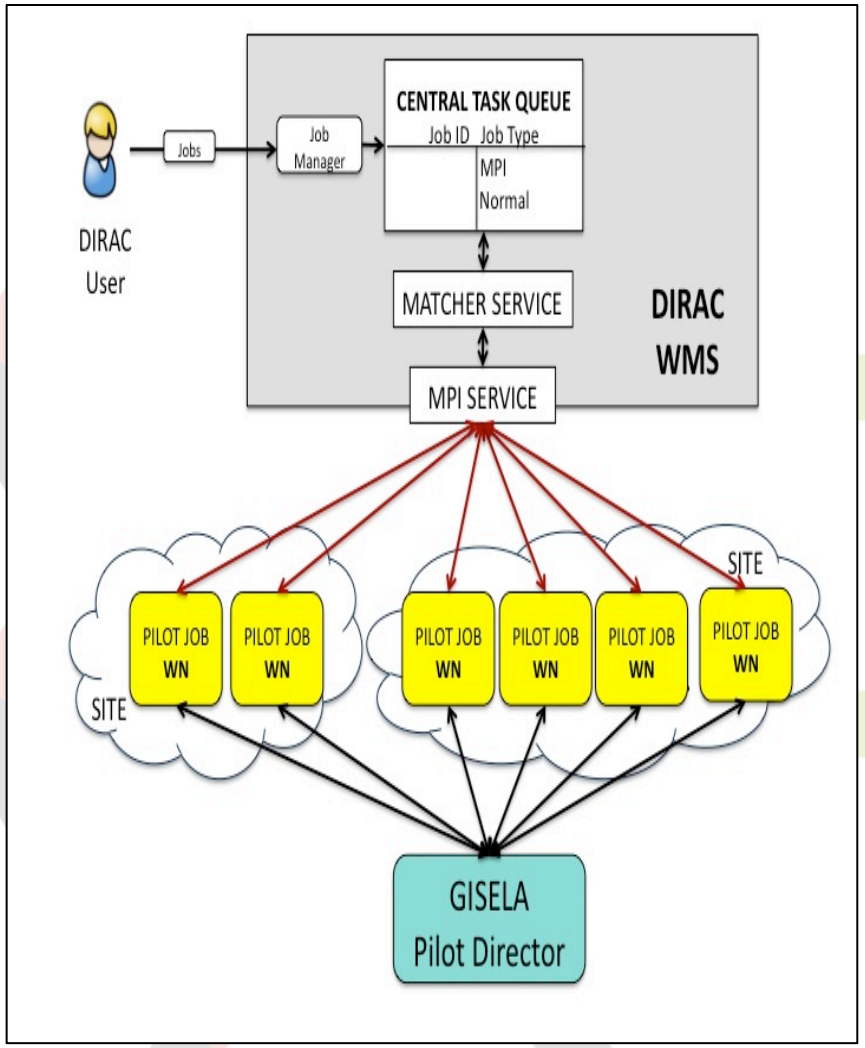
- In case of GISELA framework project, approximately 30% of application are parallel applications, special attention was made when at the moment of application's execution some problems where encountered, between them:
  - Most of the sites were not supporting MPI jobs.
  - MPI installation procedure is not standard, some sites are sharing a filesytem between the Worker Nodes (WNs) others allows Secure Shell (SSH) connections without passwords.

MPI Applications

- Even if, a site, than fulfill all requirements is selected to run a MPI job, this is no guarantee that the job will be executed successfully.
- For the users, this situation represents a complicated scenario to select in which sites MPI jobs could be executed.
- In the other hand, DIRAC introduced the now widely used concept of Pilot Agents. This allows to build efficient Workload Management Systems (WMS) that are resilient to failures in the ever changing Grid environment.

MPI DIRAC Extension

- This situation raised the idea to have MPI pilots in DIRAC, capable to create a suitable environment in the sites to execute MPI jobs, even if the sites are not providing MPI support.



MPI DIRAC Extension

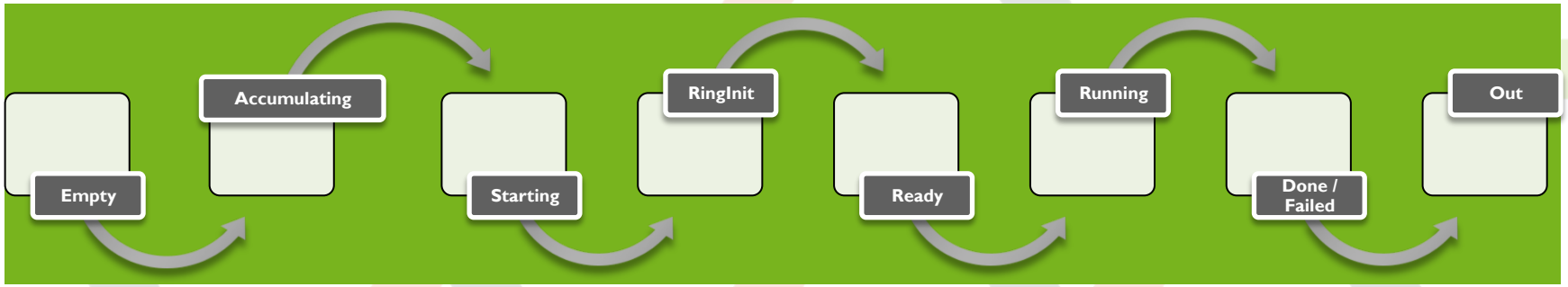
- MPI DIRAC extension components were designed as described below:
  - **MPI service** responsible of orchestrate MPI pilot jobs behavior, establish connection with DIRAC Matcher Service and MPI Database and wait for MPI Agent connections.
  - **MPI pilot job** is responsible to start MPI Agent in the WN and decide when liberate the resource.
  - **MPI Agent** is in charge to establish communication with MPI service, send the information about the WN environment; create, start, finalize MPI rings daemons and environments according information received from MPI service.
  - **MPI Job database** stores the information about MPI rings created and the user job associated.

MPI Pilot Roles

- In DIRAC, in each MPI pilot job cycle, it can take a role between two possibilities:
  - **Master**: The first MPI pilot who make the request to MPI service and the following facts are true:
    - Exist in the task queue a MPI user job in status “Waiting”
    - It is not other ring in accumulating state deployed in the site.
  - **Slave**: are all the MPI pilots deployed in the same site than make a request to MPI Service will have the role slave until the number of CPUs required for the job is fulfilled.

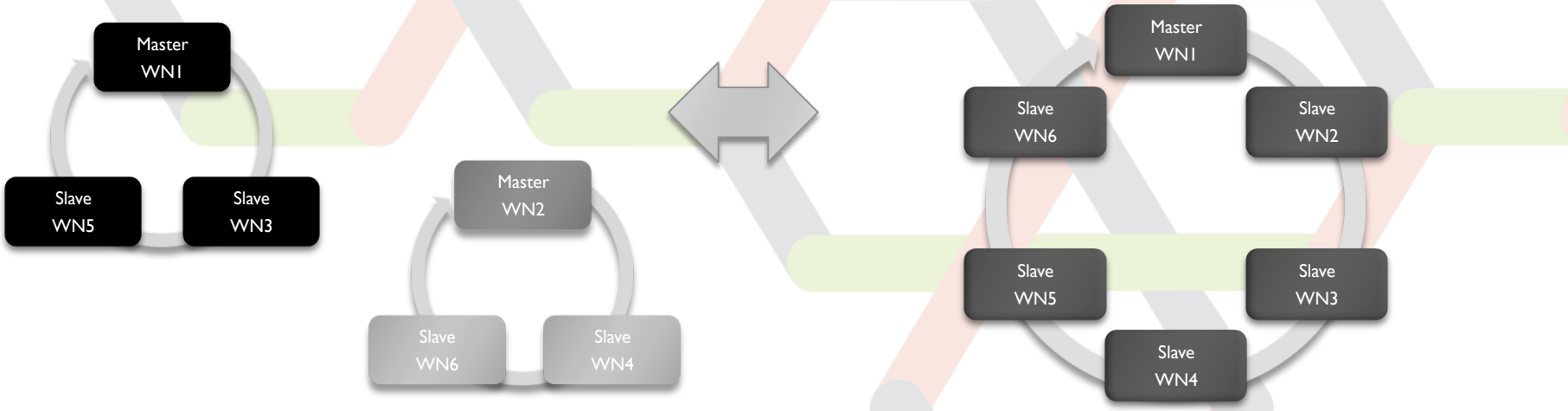
MPI Ring Life Cycle

- During a MPI Ring cycle in DIRAC different statuses are passed throw:



MPI Ring Life Cycle

- A MPI ring will be destroyed when:
  - MPI job is executed successfully.
  - In the case a problem occurs during the job execution.
  - The number of slaves is not reached in a given time.
- If one of the two last cases happen, the MPI user job is rescheduled.
- After the ring is out, MPI pilot jobs will check if execution time available is enough to start a new cycle or if it will be stopped and the WN is liberated.

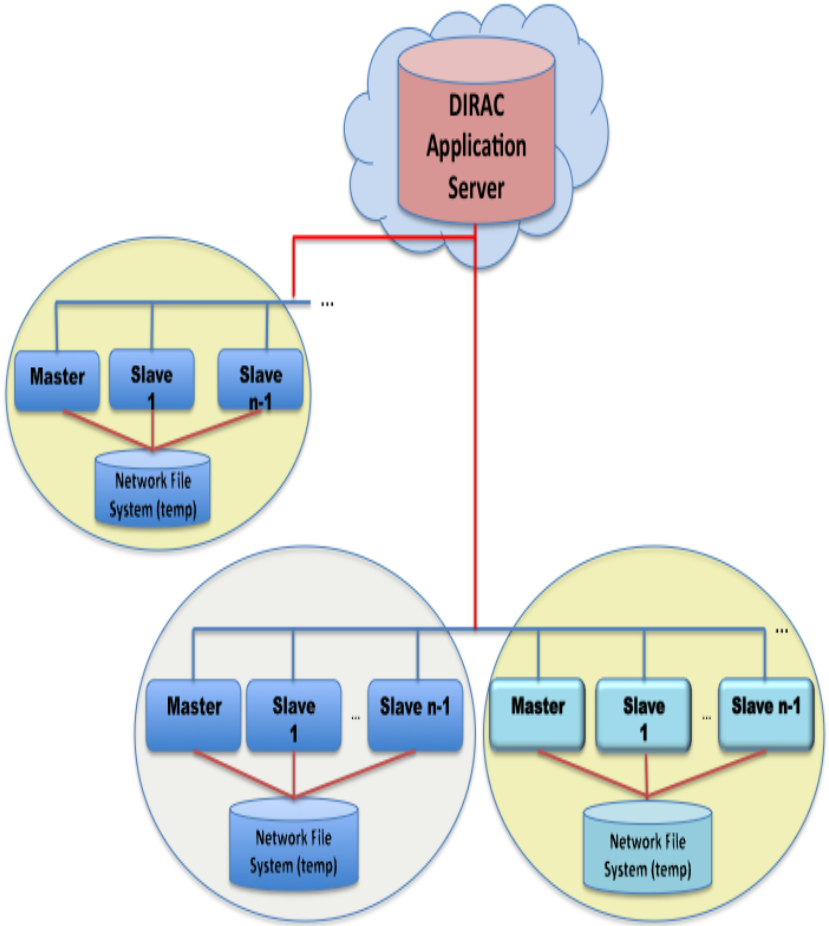


Distributed File System

- **Cooperative Computing Tools (CCTOOLS)** is a package composed by a set of tools:
  - **Chirp** a distributed filesystem designed for exporting access to file data for use in Grid computation**.**
  - **Parrot** is a tool than allows talking with chirp protocol and others protocols like http, grow, ftp, irods, hdfs, cenrvmfs; in any place without root or special privileges.
- Other advantage is the use of ACLs in each directory and the use of Globus as authentication method. Root or access privileges are not required.

Application Server

- In order to test this solution, the steps done are listed below:
  - A Chirp server was installed to distribute the applications between all the WNs, populating this with the user application.
  - ACLs were configured to use Globus ad authentication method
  - DIRAC pilot jobs were modified in order to start on the fly a chirp server in the Master Pilot, and to be used for all the pilots and environment variables were modified to include CCTOOLS path from DIRAC application server .
  - User jdl was modified to start the application including parrot commands.
  - It was not necessary to make any change in MPI Service to run this new solution.



Conclusions

- A different approach of how to execute MPI applications in the Grid was encountered.
- Extend DIRAC to support another kind of jobs in an easy way is an advantage for DIRAC middleware.
- This work open a new door in the way to the use of resources in the Grid, it can be improved and extended.

Future Work

- For future work, using DIRAC to run MPI applications approach includes:
- Support of other MPI flavors in the MPI Agents,
- Gridify more applications in order to find problems and improve the code.
- With the apparition of Cloud Computing, DIRAC Workload Management System is going to be enhanced, also, it is going to be possible for the MPI pilot jobs use more than one CPU improving the times of MPI ring creation.
- Evaluate other ways of distribute applications as for example, Parrot is supporting actually CernVM-FS, this combination allows to use CernVM-FS as server of DIRAC applications and use Parrot tools to mount the application in the WNs.