Future of Grid resource exploitation systems

R. Graciani & A. Tsaregorodtsev

- Introduction
- Large communities
- Distributed Computing Models
- The DIRAC project
- Summary

Introduction

- Wikipedia: **Grid computing** is a term referring to the combination of computer resources from multiple administrative domains to reach a common goal.

- In practice this is implemented using **Middleware** (Condor Toolkit, gLite, UNICORE, ARC,…) tools that provide access to Grid infrastructures.

- But it does not exclude that **HPC** or **Cloud** resources can be integrated when appropriated.

Hardware evolution

- Large number (100-1000) of Cores/GPUs per box:
  - Memory can not follow this grow (price).
  - Requires use of parallel programming techniques to efficiently use the resources.

- Huge disks:
  - I/O does not increase proportionally to capacity.
  - Requires use of parallel distributed file systems.

Future of Grids

- Middleware is moving towards better interoperability.
- Infrastructure is getting fragmented into multiple Grids.
- Special use cases require specialized resources.
- General purpose Grids will not solved all needs.
- Scientific communities are getting global:
  - Computing will be distributed

Large VO issues

- Dealing with heterogeneous resources
  - Various computing clusters, grids, etc

- Dealing with the intra-community policies
  - User groups, quotas and priorities
  - Priorities of different activities

- Dealing with a variety of applications
  - Massive data productions
  - Individual user applications, etc

General issues

- Overcome deficiencies of the standard middleware
  - Inefficiencies, failures
    - Production managers can afford that, users can not
  - Lacking specific functionality
- Alleviate the excessive burden from sites – resource providers – in supporting multiple VOs
  - Avoid complex VO specific configuration on sites
  - Avoid VO specific services on sites

LHC solutions

- The complexity of managing the VO workload resulted in specific software layer on top of the standard grid middleware:
  - ***AliEn*** in Alice
  - ***PanDA*** in Atlas
  - ***GlideIn WMS*** in CMS
  - ***DIRAC*** in LHCb

Distributed Computing Models

- Large scientific projects with important computing needs need to integrate computing resources from different providers.
  - Distributed Computing ⇔ Grid Computing

- Efficient use of resources requires tools for:
  - Planning
  - Manage

  the use of resources.

- Distributed Computing Framework

Evolution

- Capabilities of the resources evolve:
  - Short term: shared usage
  - Long term: SW and HD updates (funding?).
- Requirements of the project evolve:
  - Short term: different phases of the project
  - Long term: research is alive
- Computing Model must be flexible
- Computing Framework must be flexible

Computing Framework (I)

- Workload Management:
  - Handling of computing tasks
  - Locate optimal resource for execution
  - Ensure proper execution
  - Retrieval of results
- Key aspects:
  - Global view of resources and needs (integration of all activities)
  - Provide interoperability by adding a common layer
  - Ready to integrate new domains

Computing Framework (II)

- Data Management:
  - Handing of data to make it available were needed
  - Efficient use of resources (storage, network,..)
  - Flexible access: local, remote
  - Metadata
- Key issues:
  - Dynamic data placing (popularity)
  - Resource management
  - Data Integrity

Computing Framework (III)

- Integrated Systems
  - Close coupling between WMS and DMS
  - Data driven system for automatic data processing
  - Integration of Replica and Metadata catalogs
  - Workflow tools for complex computing tasks

DIRAC project

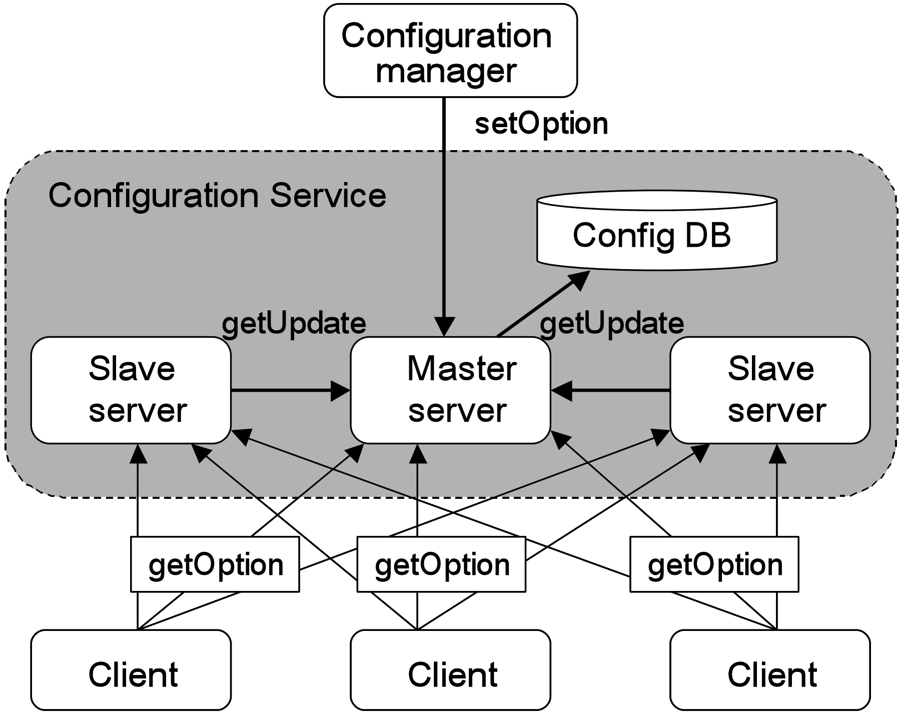
- DIRAC is the framework develop by LHCb to implement its distributed computing model.
- Has become an Open Source project with an increasing number of interested communities (not only in HEP)
- An Agreement is being signed by initial developing institutions and new contributors will be welcome.

General Framework

- Provide means to build functional systems based on:
  - Services
  - Agents
  - Databases
  - Clients
- With DISET custom client/service protocol
  - X509, GSI security standards
  - Fine grained authorization rules
- And distributed configuration

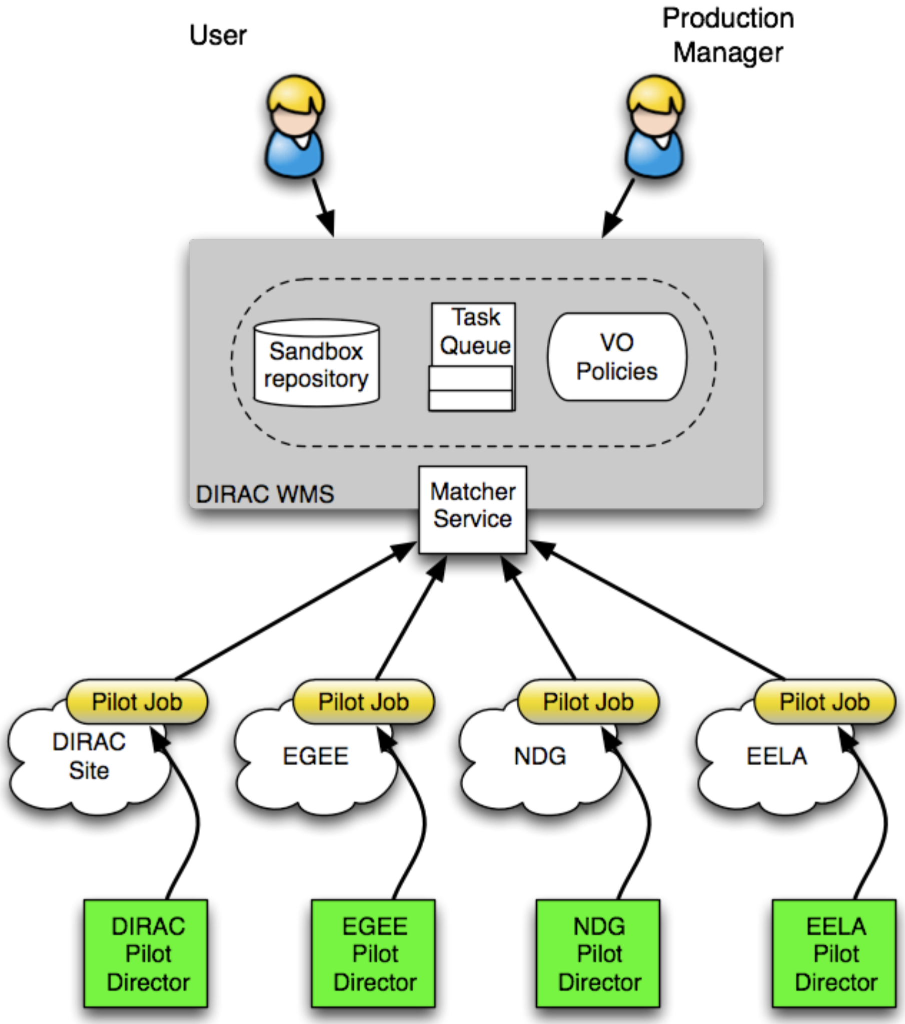
DIRAC base services

- Redundant Configuration
  - Provides service discovery and setup parameters for all the components
- Full featured proxy management
  - Proxy storage and renewal
  - Support for multiuser pilot jobs
- System Logging
  - Collect essential error messages
- Monitoring
  - Monitor the service and agents behavior
- Accounting
  - Collect statistical data of resource usage



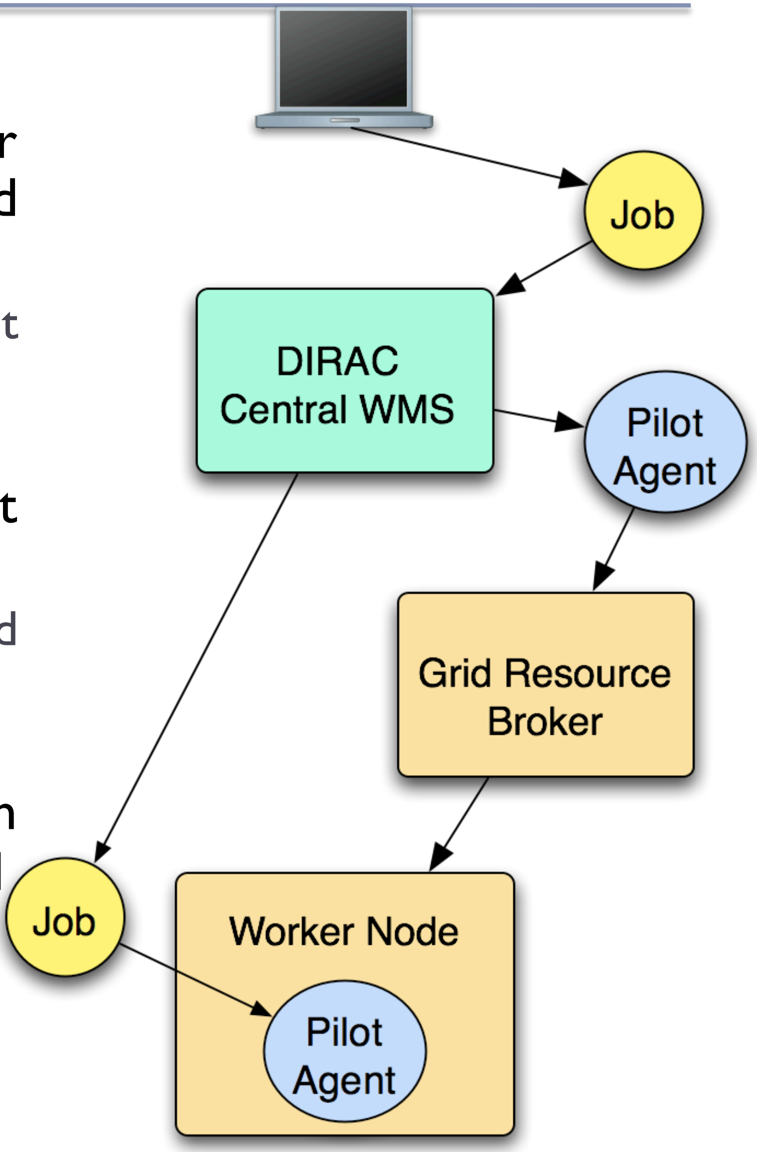
DIRAC WMS

- Jobs are submitted to the DIRAC Central Task Queue with credentials of their owner (VOMS proxy)
- Pilot Jobs are submitted by specific Directors to a Grid WMS with credentials of a user with a special Pilot role
- The Pilot Job fetches the user job and the job owner's proxy
- The User Job is executed with its owner's proxy used to access SE, catalogs, etc



Pilot Jobs in a nutshell

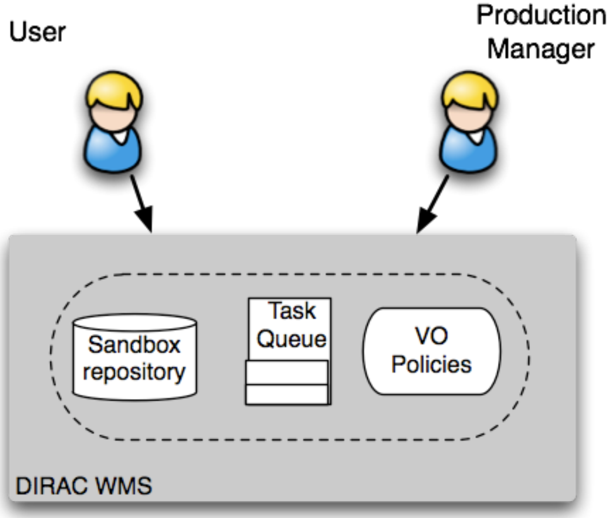
- Pilot agents are deployed on the Worker Nodes as regular jobs using the standard grid scheduling mechanism
  - Form a distributed Workload Management system
  - Reserve the resource for immediate use
- Once started on the WN, the pilot agent performs some checks of the environment
  - Measures the CPU benchmark, disk and memory space
  - Installs the application software
- If the WN is OK the user job is ***pulled*** from the central DIRAC Task Queue and executed
  - Terminate gracefully if no work is available



Applying VO policies

- In DIRAC both User and Production jobs are treated by the same WMS

- This allows to apply efficiently policies for the whole VO
  - Assigning Shares for different groups and activities

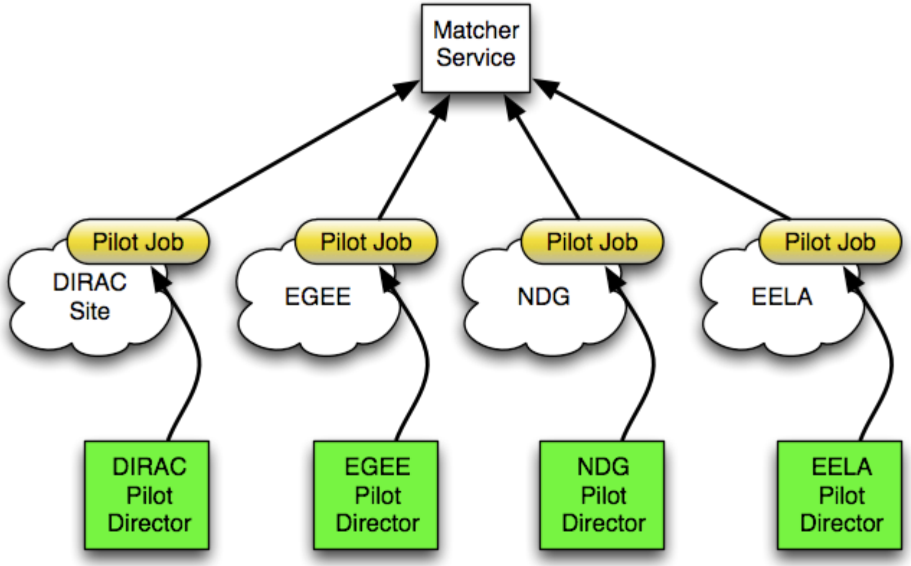


- The VO policies application in the central Task Queue dictates the use of Multiuser Pilot Agents
  - Do not know apriori whose job has the highest priority at the moment of the user job matching

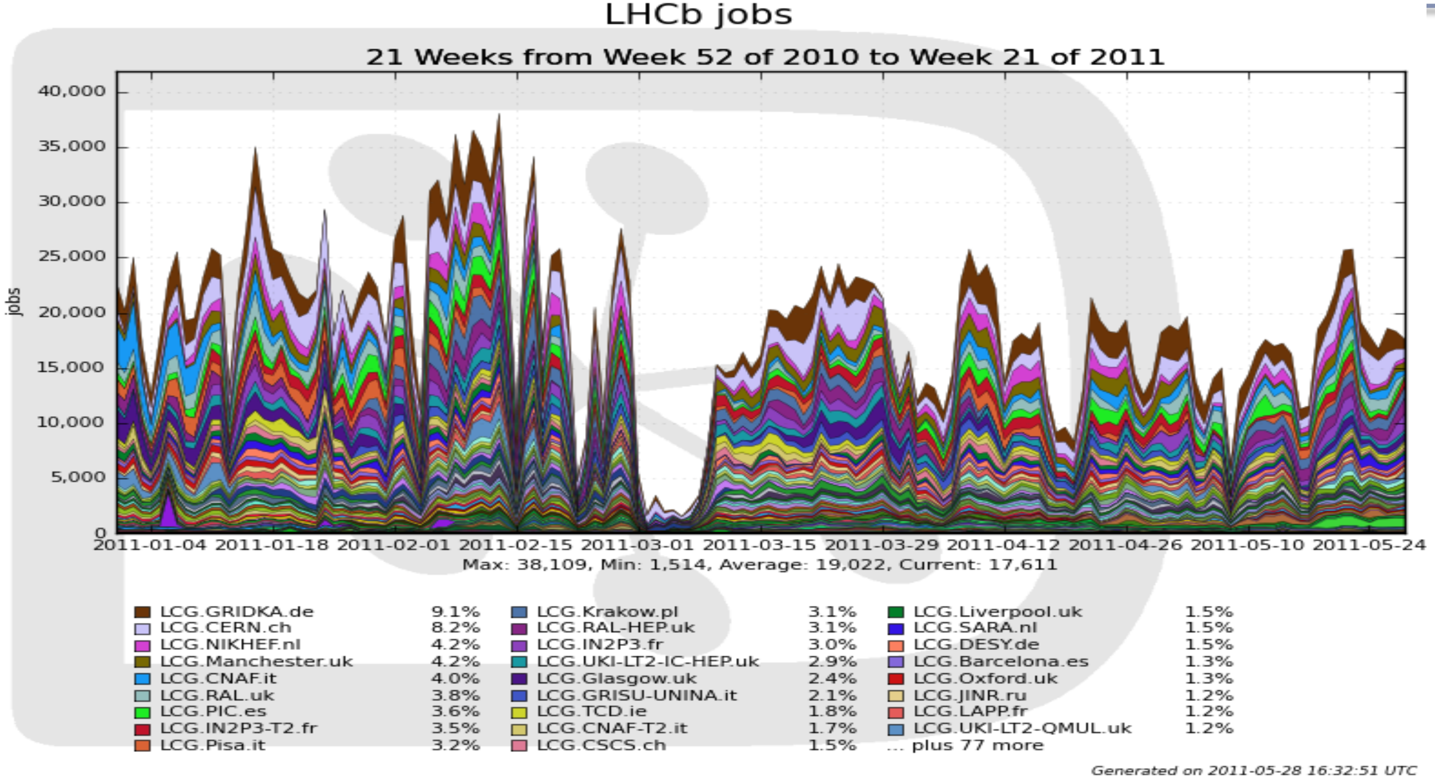
- DIRAC fully supports this mode of operation
  - Multiuser Pilots Jobs submitted with a special “pilot” VOMS role
  - Can use glexec on the WNs to track the identity of the payload owner

Heterogeneous resources

- Including resources in different grids and standalone clusters or clouds is simple with Pilot Jobs
  - Needs a specialized Pilot Director per resource type
  - Demonstrated with NDG, EELA and OSG grid sites
  - Demonstrated with Amazon
  - Users just see new sites appearing in the job monitoring



WMS performance



- DIRAC performance in production
  - Up to 35K concurrent jobs in ~120 distinct sites
  - 5 mid-range central servers hosting DIRAC services
  - Further optimizations to increase capacity are possible
    - Hardware, database optimizations, service load balancing, etc

Development environment

- Python is the main development language
  - Fast prototyping/development cycle
  - Platform independence
- MySQL database for the most components
  - ORACLE database backend for the LHCb Metadata Catalog
- Modular architecture allowing an easy customization for the needs of a particular community
  - Simple framework for building custom services and agents

Summary

- Moving from a dedicated Grid to systems built from grids, clouds, dedicated clusters,…
- Large communities have complex requirements
- Versatile tools to implement distributed systems
- DIRAC provides a framework to build such systems
- Flexibility must be built in