Introduction

DIRAC Project

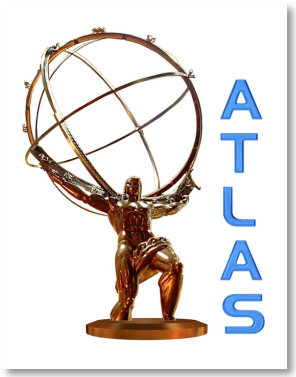
Outline

- What is DIRAC (short explanation)
- Specific issues of large Grid Communities
- DIRAC solution
  - Workload Management
  - Data Management
  - Other systems
  - Extending DIRAC
  - User interfaces
- Tutorial plan

DIRAC is not …

- DIRAC is not a yet another Grid Web Portal
- DIRAC is not a yet another Meta Scheduler
- DIRAC is not a yet another job submission frontend
- DIRAC is not a yet another Application Server
- DIRAC is all of the above but also much more

HEP applications



- HEP experiments collect unprecedented volumes of data to be processed on large amount of geographically distributed computing resources
  - 10s of PBytes of data per year
  - 10s of thousands CPUs in 100s of centers
  - 100s of users from 100s of institutions

However, other application domains are quickly approaching these scales

Large VO issues

- Large user communities (Virtual Organizations) have specific problems
  - Dealing with heterogeneous resources
    - Various computing clusters, grids, etc
  - Dealing with the intracommunity workload management
    - User group quotas and priorities
    - Priorities of different activities
  - Dealing with a variety of applications
    - Massive data productions
    - Individual user applications, etc

General problems

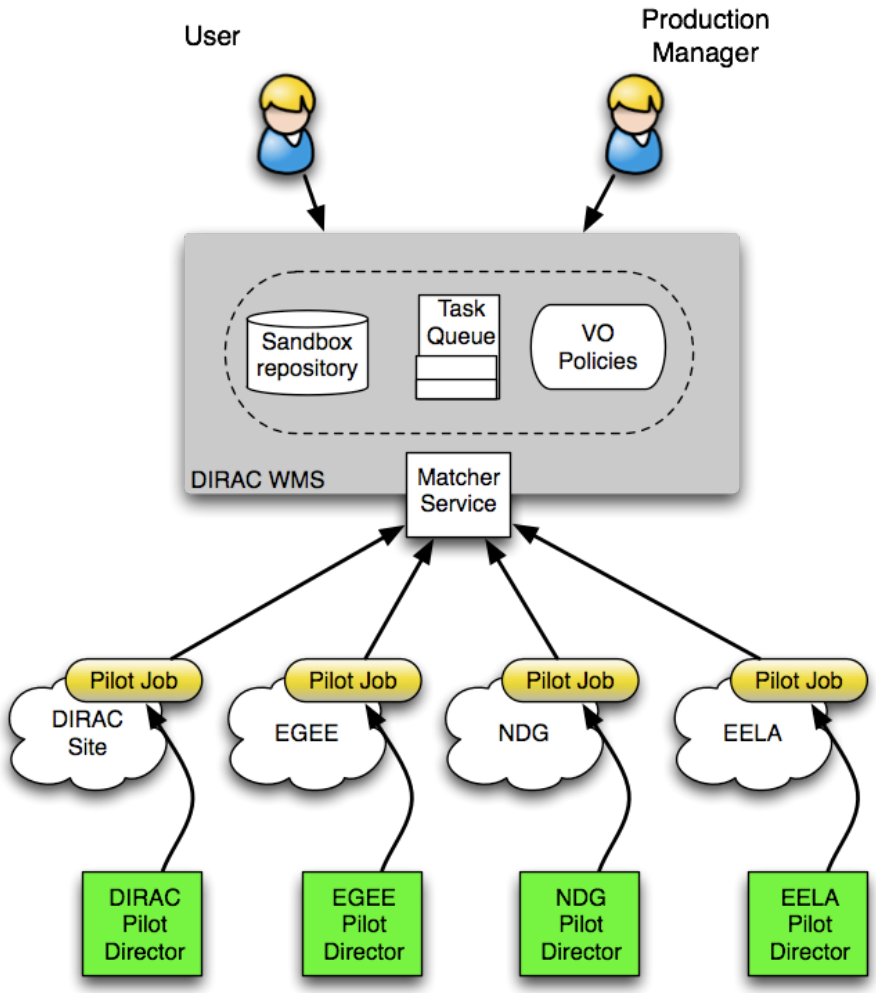
- Overcome deficiencies of the standard grid middleware
  - Inefficiencies, failures
    - Production managers can afford that, users can not
  - Lacking specific functionality
- Alleviate the excessive burden from sites – resource providers – in supporting multiple VOs
  - Avoid complex VO specific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

DIRAC Grid Solution

- DIRAC is providing a complete grid middleware stack
  - Developed originally for the LHCb experiment with the goal:
    - Integrate all the heterogeneous computing resources available to LHCb
    - Minimize human intervention at LHCb sites
  - Evolved to a generic grid solution to solve user community needs
    - Fault tolerance
    - Quicker turnaround of user jobs
    - Enabling Community policies
    - …

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

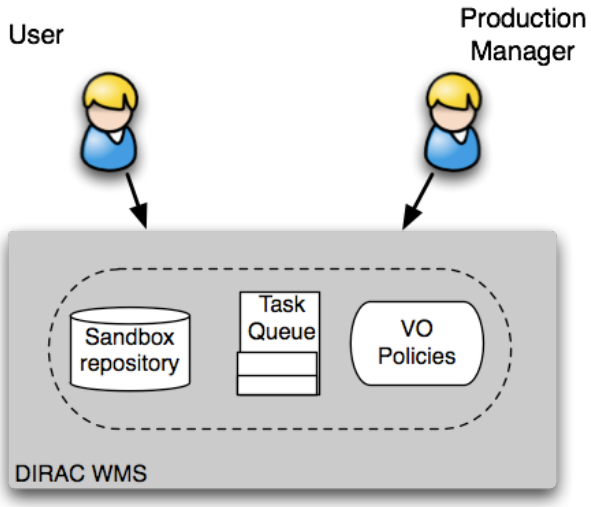


User Job efficiency

- Improved visible efficiency due to pilot agents
  - ~96% efficiency for DIRAC jobs vs 70-90% efficiency for the WLCG jobs
- If some resources are failing, it is just seen as a reduced pool of resources for the users
- An excess of Pilot Jobs over User Jobs just to cover inefficiencies of Computing Resources or Grid middleware
  - it is normal that computing resources are failing but
  - it is not normal that users are suffering from that

WMS: applying VO policies

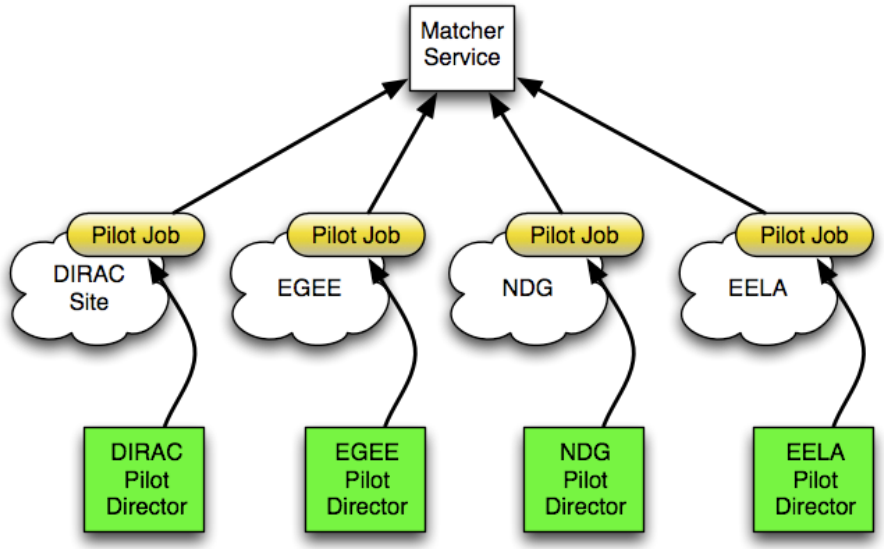
- In DIRAC both User and Production jobs are treated by the same WMS
  - Same Task Queue
- This allows to apply efficiently policies for the whole VO
  - Assigning Job Priorities for different groups and activities
  - Static group priorities are used currently
  - More powerful scheduler can be plugged in
    - demonstrated with MAUI scheduler



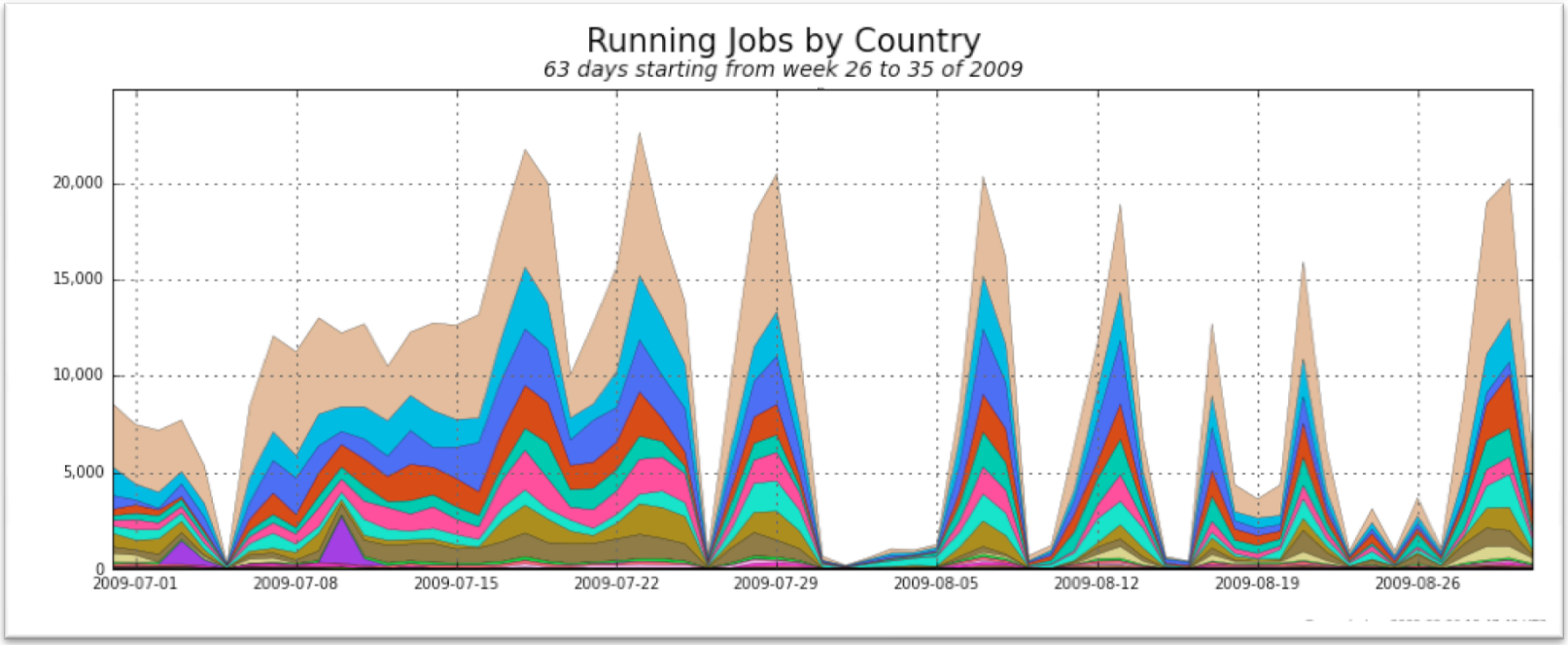
- The VO policies application in the central Task Queue dictates the use of Multiuser Pilot Agents
  - Do not know apriori whose job has the highest priority at the moment of the user job matching
- DIRAC fully supports this mode of operation
  - Multiuser Pilots Jobs submitted with a special "pilot" VOMS role
  - Using glexec on the WNs to track the identity of the payload owner

WMS: using heterogeneous resources

- Including resources in different grids and standalone clusters is simple with Pilot Jobs
  - Needs a specialized Pilot Director per resource type
  - Users just see new sites appearing in the job monitoring
- Resources include:
  - Grids: EGEE, GISELA, NDG
    - OSG is coming
  - Batch clusters, e.g. Torque
    - No special site installation is needed
  - Commercial computing clouds
    - Amazon EC2



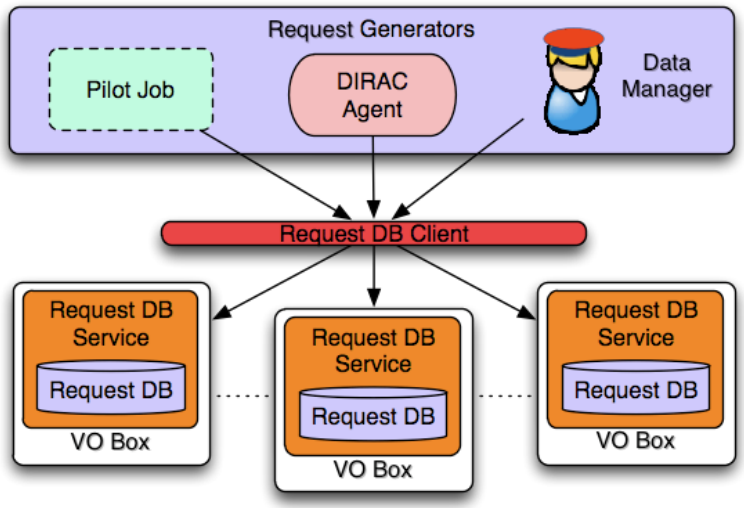
WMS performance



- DIRAC performance measured in the recent production and FEST'09 runs
  - Up to 25K concurrent jobs in ~120 distinct sites
  - One mid-range central server hosting DIRAC services
  - Further optimizations to increase capacity are possible
    - Hardware, database optimizations, service load balancing, etc

Request Management system

- A Request Management System (RMS) to accept and execute asynchronously any kind of operation that can fail
  - Data upload and registration
  - Job status and parameter reports



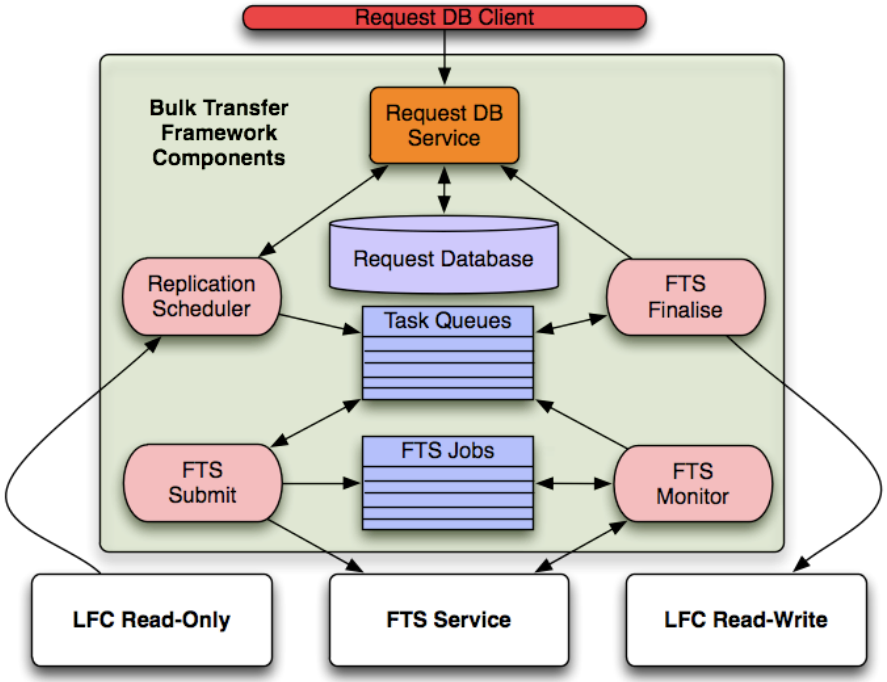
- Request are collected by RMS instances on VO-boxes at 7 Tier-1 sites
  - Extra redundancy in VO-box availability
- Requests are forwarded to the central Request Database
  - For keeping track of the pending requests
  - For efficient bulk request execution

Data Management tools

- Storage Element services
  - SRM Storage Elements
  - DIRAC Storage Elements
  - Transparent access for the user
    - Replication, access from the jobs
- File Catalogs
  - LFC
  - DIRAC File Catalog
    - Compact, high performance
    - Includes user defined metadata
  - Uniform access
    - both catalogs can be used simultaneously

Data Management System

- All the Data Distribution operations
  - Pit to CERN transfers
  - T0-T1 transfers
  - T1-T1 transfers
- Based on the Request and Production Management Systems
  - Automatic transfer scheduling
  - Full monitoring of ongoing operations
- Using FTS for bulk data transfers
  - Full failure recovery
- Comprehensive checks of data integrity in SEs and File Catalogs



DIRAC development environment

- Python is the main development language
  - Fast prototyping/development cycle
  - Platform independence
- MySQL database for the main services
  - 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

DIRAC Framework

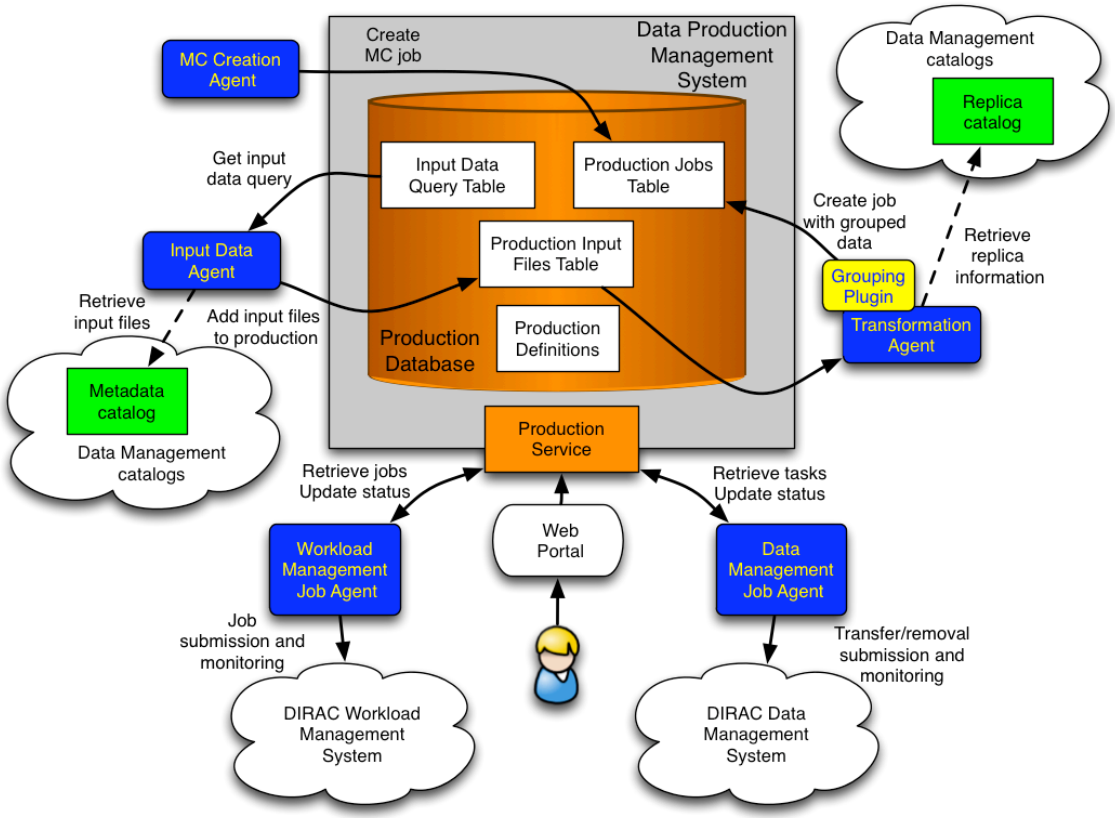
- Services oriented architecture
  - DIRAC systems consist of services, light distributed agents and client tools
- All the communications between the distributed components are secure
  - DISET custom client/service protocol
    - Control and data communications
  - X509, GSI security standards
  - Fine grained authorization rules
    - Per individual user FQAN
    - Per service interface method
    - Per job

DIRAC extensions

- High level LHCb systems are built in the same DIRAC framework
  - Collaborating services and agents
  - Web based monitoring and controls
  - Detailed authorization rules

LHCb Production Management System

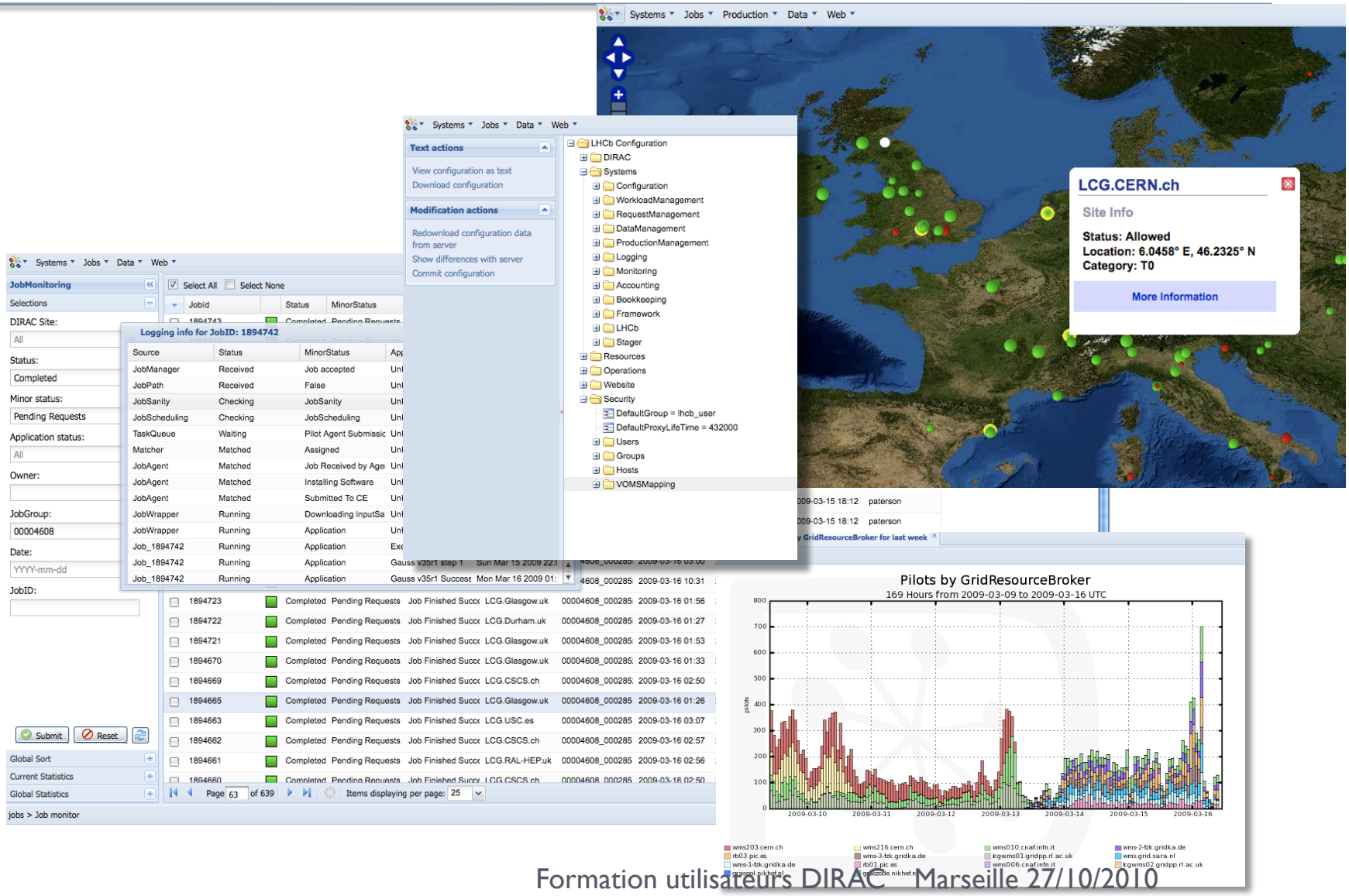
- Example extension
  - LHCb specific parts implemented as plugins
- Production Management built on top of the DIRAC WMS and DMS
  - Data requests formulated by users are processed and monitored using Web based tools
  - Automatic data reconstruction jobs creation and submission according to predefined scenarios
  - Interfaced to the LHCb Bookkeeping Database



DIRAC: User perspective

- Easy client installation for various platforms (Linux, MacOS)
  - Includes security components
- Familiar usage patterns
  - JDL notation for job description
    - Simplified with respect to the « standard » JDL
  - Command line tools
    - à la gLite UI commands
    - e.g. dirac-wms-job-submit
- Extensive Python API for all the tasks
  - Job creation and manipulation, results retrieval
    - Possibility to use complex workflow templates
  - Data operations, catalog inspection
  - Used by GANGA user front-end
- DIRAC Web Portal

Web Portal: example interfaces



Summary

- DIRAC has most of the features of a "standard" Grid middleware stack
- Occasional users will not see much difference in functionality compared to other middlewares
  - E.g. gLite middleware, Ganga frontend
  - Better efficiency and turnaround for intensive work
- Power users will see extra support:
  - Massive job execution
  - Data operations
- Community administrators get tools to apply community policies
  - User and group priorities, quotas
- Site administrators can easily include their resources
  - Easy addition of new resources without bulky installation
  - Easy user management with only one "VO user"
- The DIRAC project is in full development
  - More new exciting features to come – stay tuned !
  - Your contributions are welcome
  - Quick bug fixes and feature request implementation

DIRAC Tutorial plan

- Getting Started
  - Getting ready the DIRAC software
  - Getting ready user credentials
- Job execution mechanics
  - Basic job operations explained
- Job manipulation tools
  - Submission, monitoring, getting results
- Data management tools
  - Upload, download, replication, access in the jobs
  - Metadata management
- Web Portal tools
  - Other info available on the Web Portal

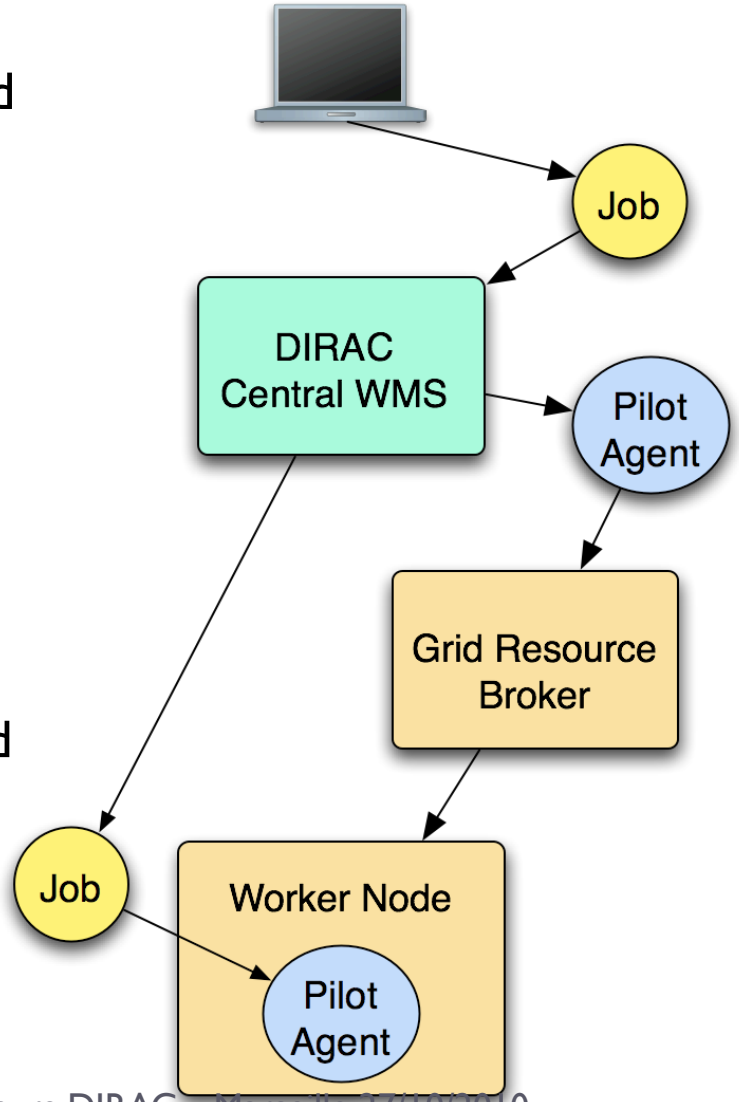
DIRAC Tutorial setup

- DIRAC installation at dirac.in2p3.fr
  - Running in a virtual machine at CC
  - Started as installation for the VO "formation"
  - Intended to be a permanent service for the users of the NGI EGEE/France
    - Now only VO "formation" fully supported
    - Multiple VO support is coming
- Resources
  - gLite Sites and SEs available to the VO "formation"
  - One DIRAC SE (DIRAC-USER)

Backup slides

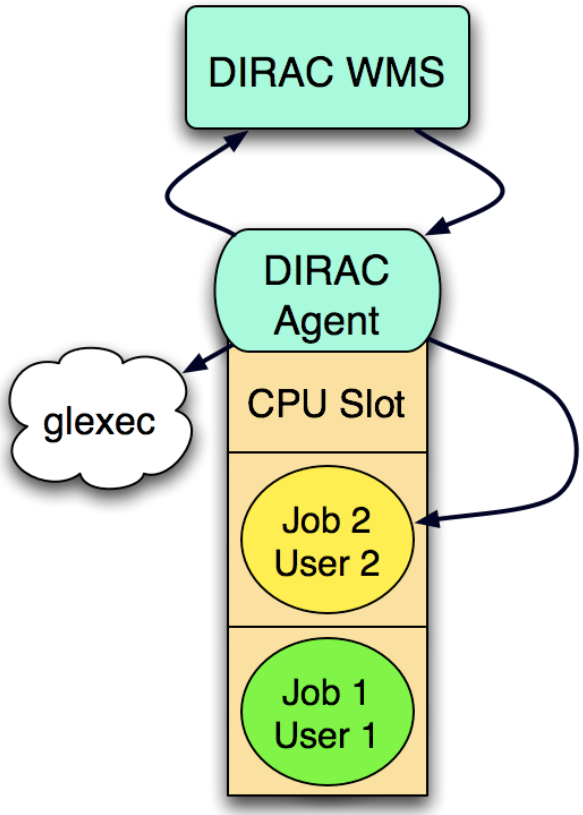
Pilot Jobs in a nutshell

- 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



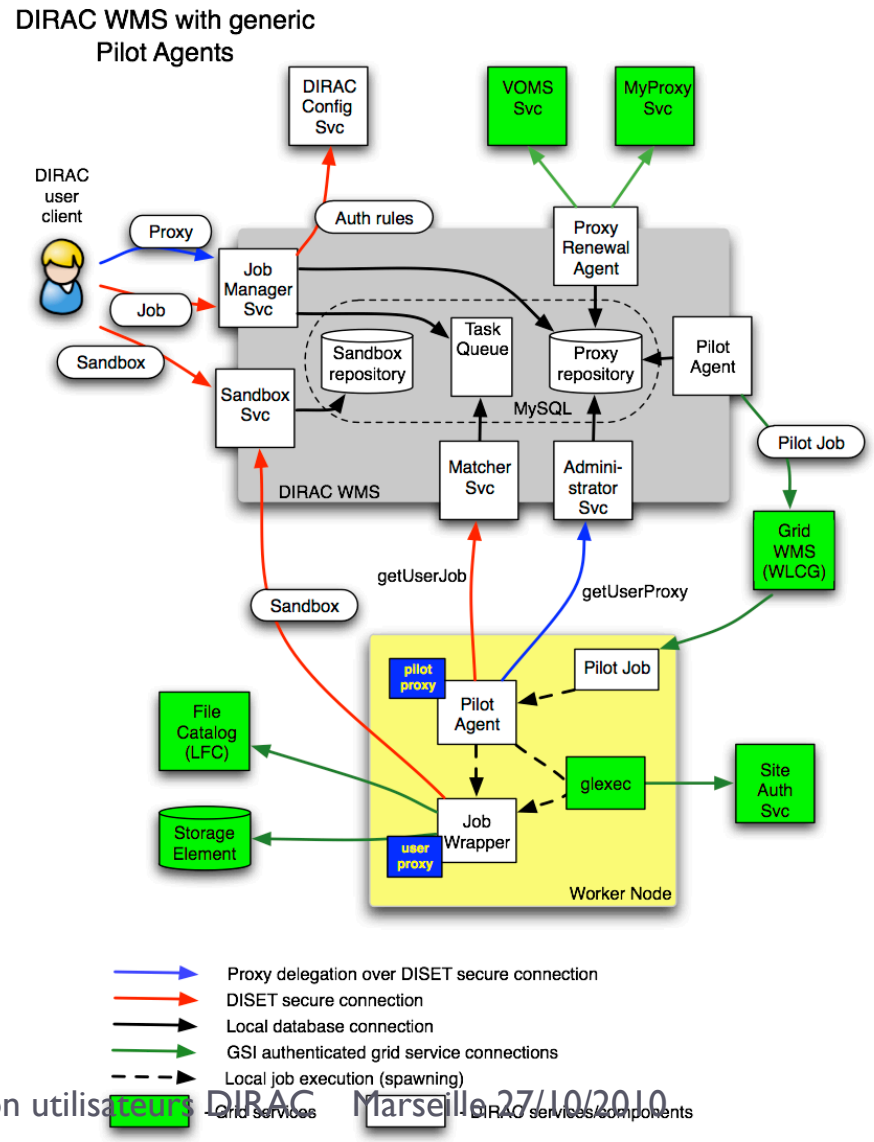
Workload optimization

- Pilot Agents work in an optimized 'Filling Mode'
  - Multiple jobs can run in the same CPU slot
  - Significant performance gains for short, high priority tasks
  - Also reduces load on LCG since fewer pilots are submitted
  - Needs reliable tools to estimate remaining time in the queue
- Considering also agents in a "preemption" mode
  - Low priority task can be preempted by a high priority tasks
    - Low priority, e.g. MC, jobs behave as resource reservation for analysis jobs



Security issues of the model

- The VO WMS must be as secure as the basic grid middleware
  - User job submissions using grid security standards: GSI
  - Secure proxy storage in the WMS repository
- The VO WMS takes over the user proxy renewal
  - Limited user proxy
  - Limited number of proxy retrievals per pilot
- Sites still retain the full right to control which individuals are accessing their resources
  - SCAS/glexec facility to authorize user workload execution on the worker node

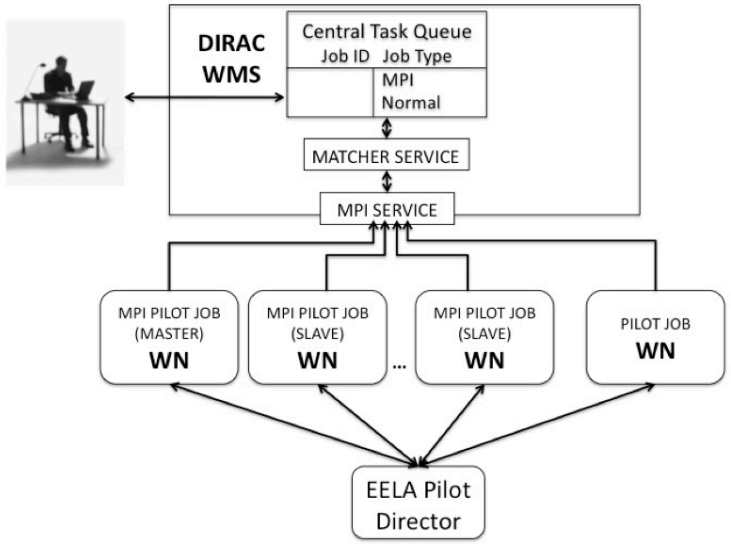


Advantages for site resources providers

- No need for a variety of local batch queues per VO
  - One long queue per VO would be sufficient
  - 24-48 hours queue is a reasonable compromise
    - Site maintenance requirements
  - Reduced number of grid jobs
- No need for specific VO configuration and accounting on sites
  - Priorities for various VO groups, activities
  - User level accounting is optional
- In the whole it can lower the site entry threshold
  - Especially useful for newcomer sites

Support for MPI Jobs

- MPI Service developed for applications in the EELA Grid
  - Astrophysics, BioMed, Seismology applications
  - No special MPI support on sites
    - MPI software installed by Pilot Jobs
  - MPI ring usage optimization
    - Ring reuse for multiple jobs
      - Lower load on the gLite WMS
    - Variable ring sizes for different jobs



Disclaimer

- This is not an introduction to Computing Grids
- DIRAC developers has a lot of experience with HEP applications but less with other domains
  - Do not hesitate to ask questions

Configuration Service

- This is the back bone of the whole system
  - Provides service discovery and setup parameters for all the DIRAC components
- Multiply redundant for high availability
- Contains only static information
  - Unlike R-GMA or BDII