A. Casajus, R. Graciani, A. Tsaregorodtsev

What is DIRAC?

“Distributed Infrastructure with Remote Agent Control”

A software **Framework**

to **Manage**

Distributed computing **Activities**

for a **User Community**.

DIRAC

- Motivation
- The project
- Design choices
- What **DIRAC** can do for **you**
- Summary

Motivation

- 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 10s of institutions
- However, other application domains are quickly approaching these scales
- 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 and solutions

- 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
- The complexity of managing the VO workload resulted in specific software layer on top of the standard grid middleware. Among the LHC experiments
  - AliEn in Alice
  - PanDA in Atlas
  - GlideIn WMS, Phedex in CMS
  - DIRAC in LHCb

Distributed Computing

- Distributed Computing is about **splitting up** a Computing Task into parts that run on **multiple computers** communicating over a network.
- The main goal of a Distributed Computing system is to **connect users to resources** in a transparent, open, and scalable way.
- A Distributed Computing system must be **fault tolerant** and deal with **heterogeneous environments** and unpredictable resource or network failures.

User Communities

- User Communities are groups of individuals with a **common interest** that may want to **share** their resources for **mutual benefit**:
  - LHCb:
    - 450 physicists from 15 countries
    - Study CP Violation and rare B decays.

The larger the community, the larger the benefit that each individual will get.

DIRAC PROJECT

The DIRAC Project

- **DIRAC** has been developed inside the LHCb Collaboration to easy the handling of all their distributed computing activities:
  - Simulated Data production,
  - Detector and Simulated Data distribution, replication,…
  - Data processing: reconstruction, stripping,…
  - User Analysis

  Integrating in a single system Grid and non-Grid resources (i.e. the Online Filter Farm, or Sites not willing to install gLite) and activities of different nature, requirements and priorities.
- **DIRAC** code and LHCb specific extensions are supported today by the following team:

  A.Tsaregorodtsev**, **A.Casajus**, **R.Graciani**, Z.Mathe, **S.Paterson**, V.Romanovsky, **M.Sapunov**, **A.C.Smith**, **V. Fernandez

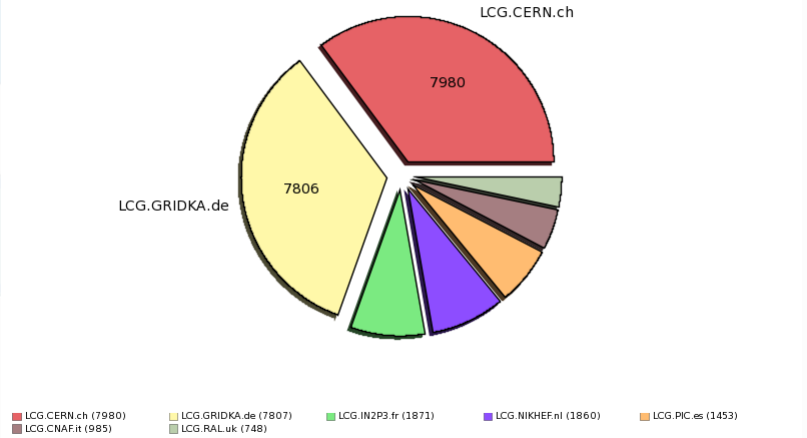
A bit of History

- Initial **DIRAC** version dates back to 2003, and is based on the following design principles:
  1. Distributed system based on collaborating proactive Agents and passive Servers.
  2. Late binding of resource to payload. Pull scheduling.
  3. Central TaskQueue + TaskQueue Optimizers.
  - Most of the resources where non-Grid.
  - Only Monte Carlo simulation was considered.
  - Security was not a concern.
  - This version was intended to proof the DIRAC design principles.
- Slowly integrate Grid Resources as they become available.
- Allow management of full production by a single operator.

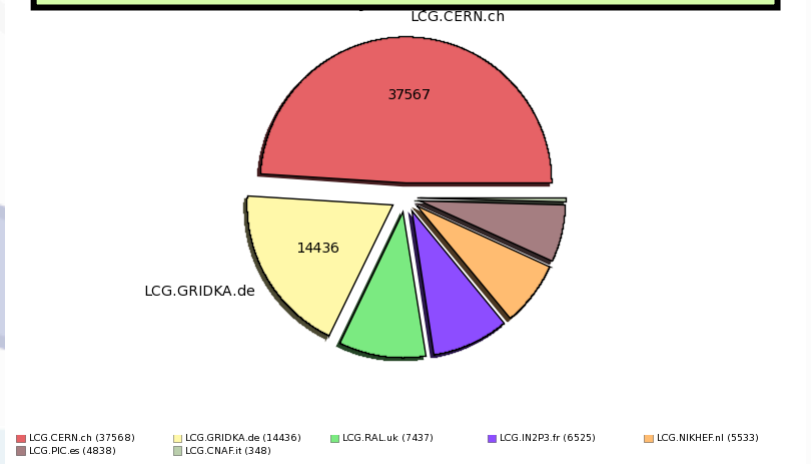
Last DIRAC version (2008-?)

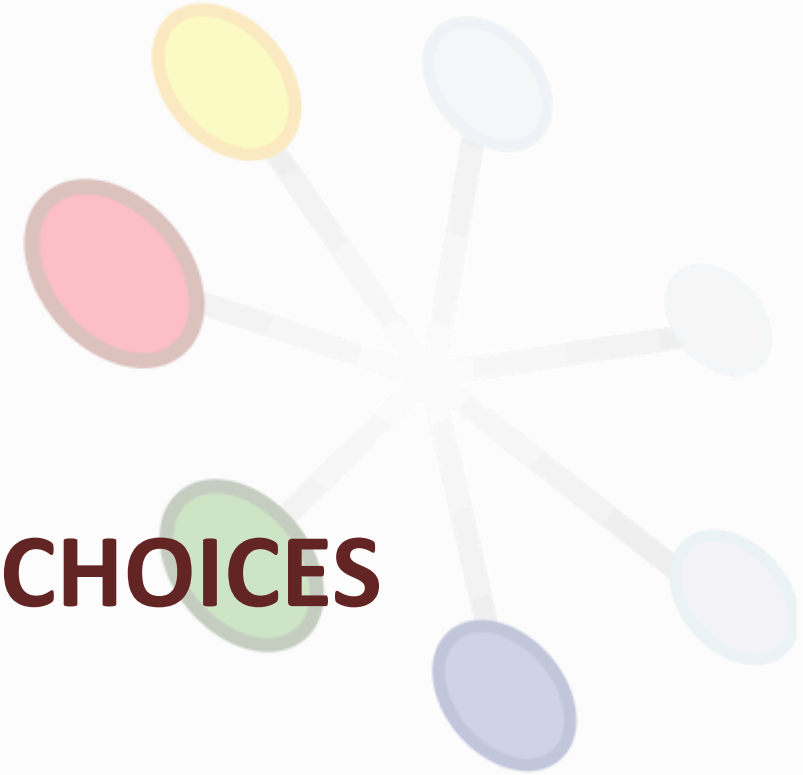
- Improved **Client-Server** protocol integrating:
  - Authorization.
  - Data transfer.
  - Security logging.
- New secured **Web-Portal** integrating:
  - Monitoring.
  - Management.
- **DIRAC Core** including transversal services (Configuration, Logging and Monitoring).
- Improved **Workload Management** with application of central VO policies.
- Improved **Data Management** using Grid tools (SRM, FTS, LFC,…) when appropriated.

2008 LHCb Production Jobs @ Tier1



2008 LHCb User Jobs @ Tier1





DESIGN CHOICES

The DIRAC Software Framework

- **DIRAC** is written in Python as a number of **collaborating Systems**, each one providing the whole framework with a subset of the required functionality.

- **DIRAC Systems** provide their functionality using a varying number of **Servers** and **Agents** that operate in a coordinate manner.

VO centric

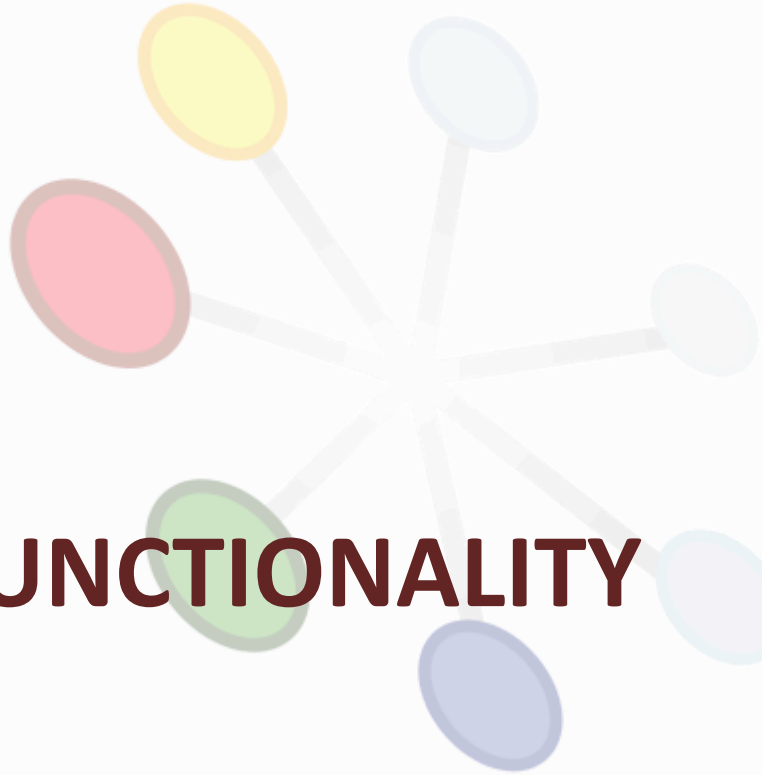
- Give to the community, the VO, a central role in the relation of its users with their computing resources.
  - In the Grid world, the VO is an entity that allows a user community, with common interests, to share distributed computing resources to which they have access.
  - In the real world, the community needs to be able to decide/ know who, when, how, for what, … these resources are used.
  - DIRAC covers this gap, allowing the community to define and enforce policies, quotas, priorities, … for its own users, groups of users as well as for the different activities. All these without any intervention of the resource centers.

Modular nature

- In order to achieve scalability and flexibility, a highly modular design was decided:
  - A Core set of components and services provide DIRAC with coherence and integrity.
  - Functionality is achieved by means of collaborating systems:
    - Some can be shared by different communities.
    - Can run on separated hardware.
  - Components are “glued” together using the DISET communication framework.
  - A given community can add or modify a component to fully adapt to its own needs.

Pull scheduling

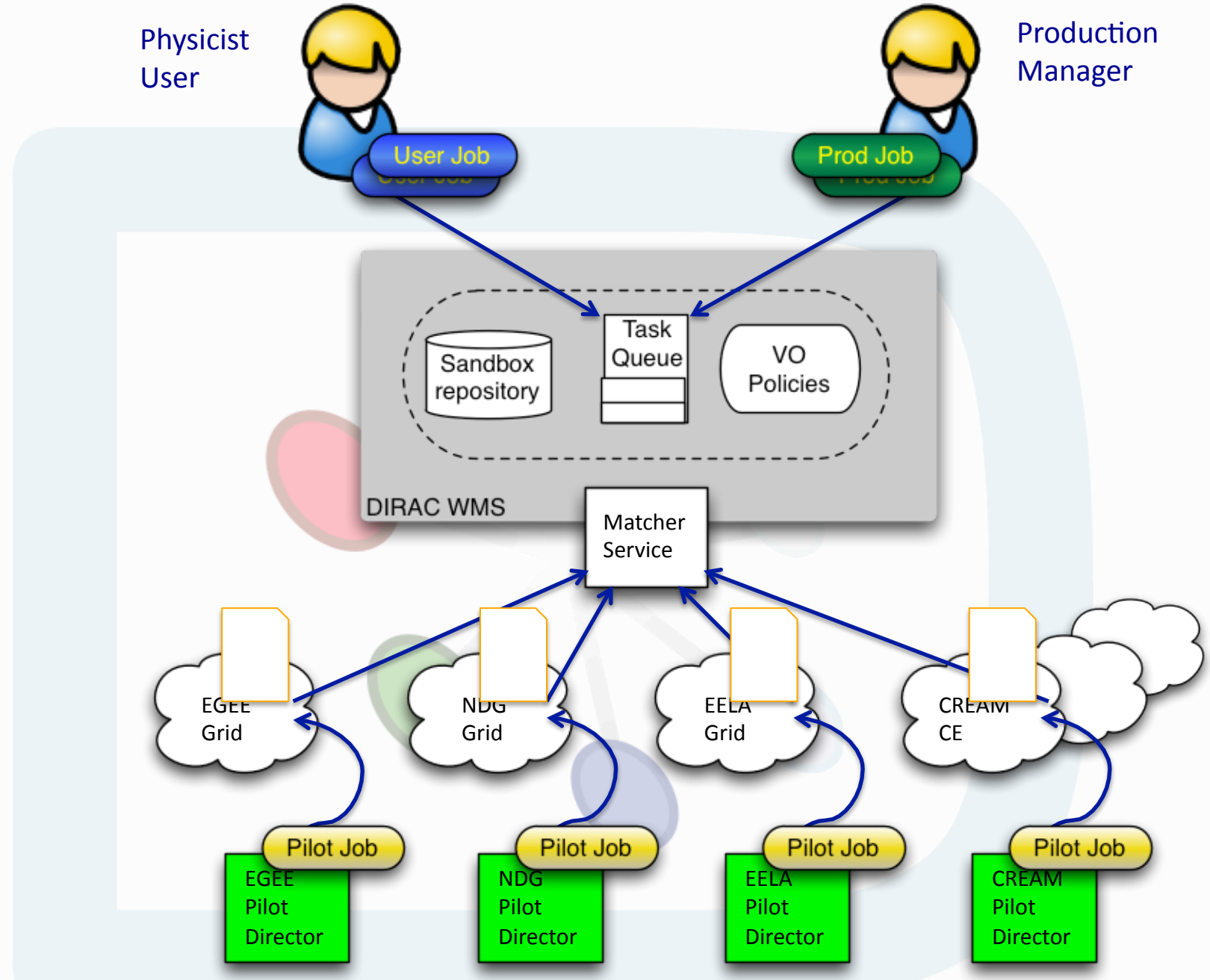
- To extract optimal performance out of the ever changing underlying resources, DIRAC implements pull scheduling with late binding of payload to resource:
  - Knowing the exact state of all available resources at any time is extremely expensive given that:
    - There are intrinsic delays/errors in the status information.
    - Resources are being accessed at the same time by other local and remote schedulers.
  - Thus, instead of assigning a payload to the best matching resource now and “push” the payload through to get it executed, DIRAC pushes empty resource reservation pilots that, once safely running at the final resource, will “pull” the best matching payload.



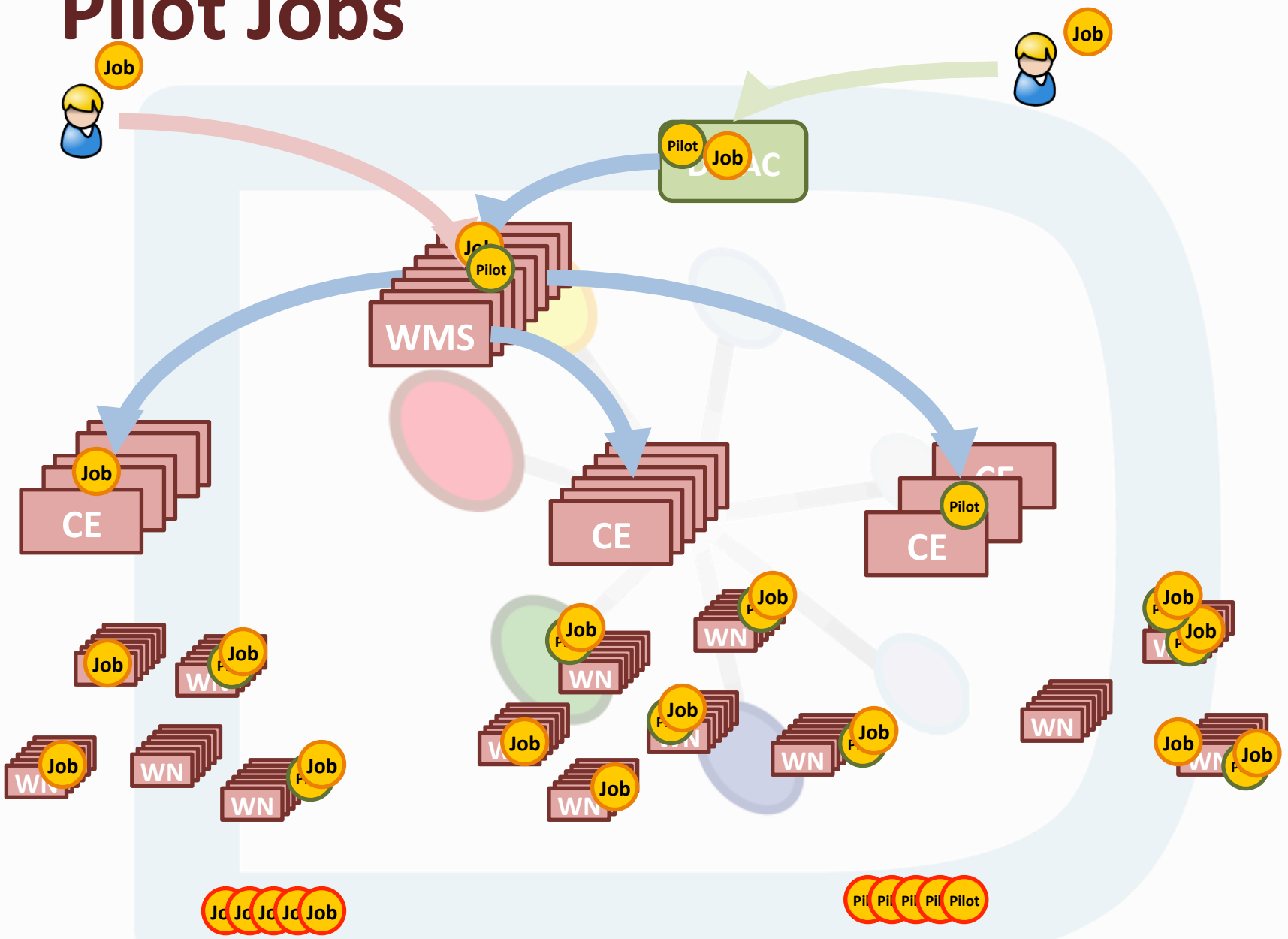
DIRAC FUNCTIONALITY

What can DIRAC do for you?

- **Workload Management** using **Pilot Jobs**

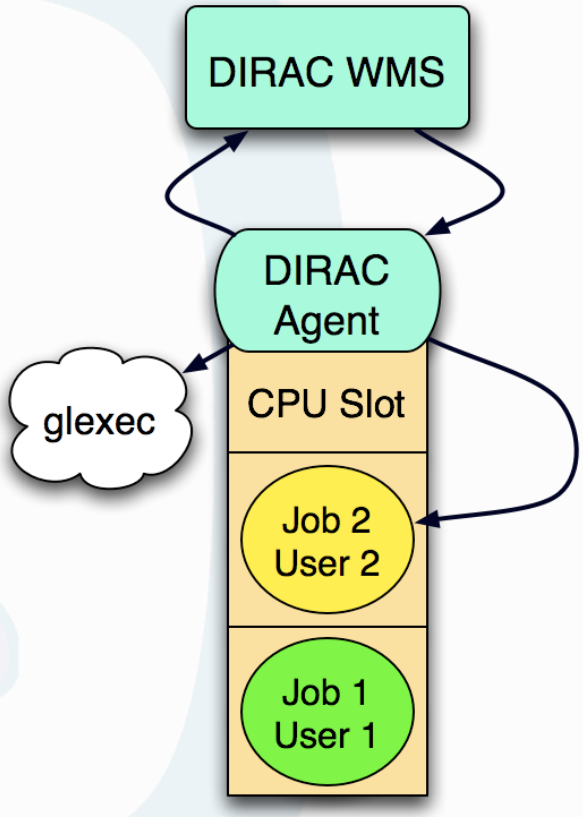

Physicist
User
Production
Manager
User Job
Prod Job
Task
Queue
Sandbox
repository
VO
Policies
DIRAC WMS
Matcher
Service
EGEE
Grid
NDG
Grid
EELA
Grid
CREAM
CE
Pilot Job
EGEE
Pilot
Director
Pilot Job
NDG
Pilot
Director
Pilot Job
EELA
Pilot
Director
Pilot Job
CREAM
Pilot
Director

Pilot Jobs

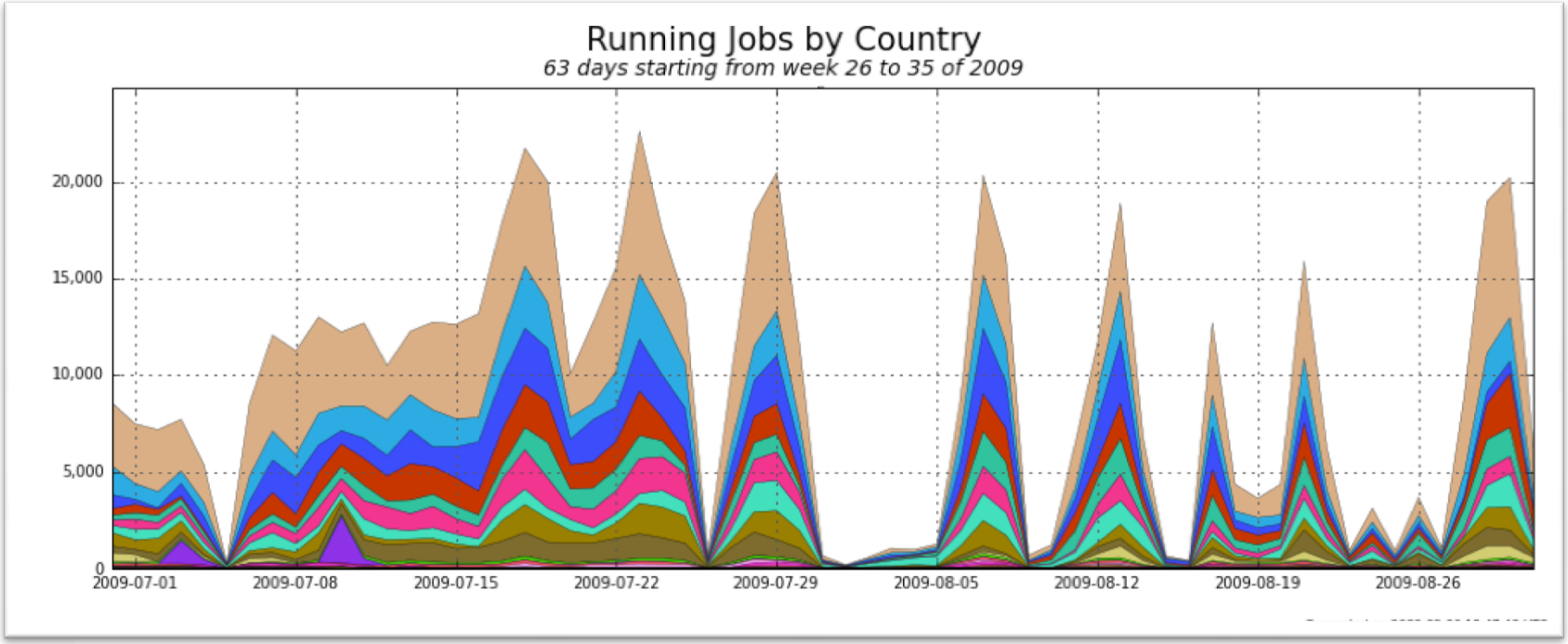

Job
Job
Pilot
Job
AC
Pilot
WMS
Job
CE
CE
Pilot
CE
Job
Job
Job
WN
Job
Job
Job
WN
Job
Job
WN
Job
WN
Job
WN
Job
Job
Job
Job
Job
Job
Pilot

Further optimizations

- 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

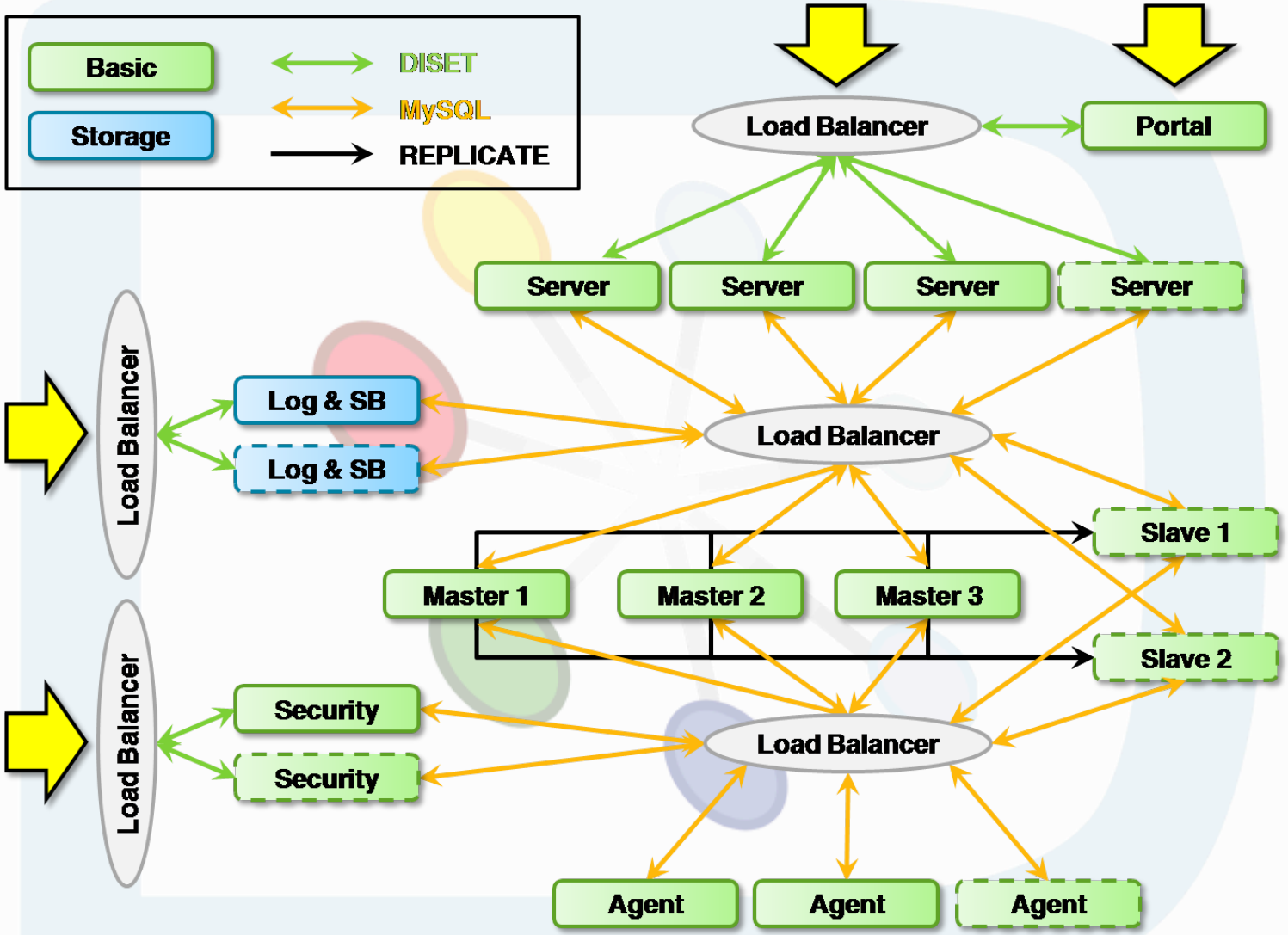


WMS performance



- 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

Scalable & Redundant



What can DIRAC do for you?

- **Workload Management** using **Pilot Jobs** with:
  - Optimized pull scheduling with **late binding** of resources to payloads.
  - Central **TaskQueue** to allow enforcement of community-wide policies.
  - Overlay layer to heterogeneous computing resources providing **homogeneous access**.
  - Treatment of Input|Output sandbox and data.
  - Single system for **Production** and **User** payloads.
  - Fully integrated with **Data Management**.

What can DIRAC do for you?

- **Data Management** with **Transfer Agents**:
  - Consistent handling of data:
    - **Upload**
    - **Replication**
    - **Removal**
  - Use the same tool from UI and payloads, with timeouts, retries, failover destinations,…
  - Support for synchronous and asynchronous operations.
  - Dedicated treatment for relevant channels (FTS).
  - Provide payloads with input data with:
    - Optimized scheduling based on input data location.
    - Uniform access to input data.
    - Redundant and failover mechanism for data upload.

What can DIRAC do for you?

- Flexible **Payload** definition:
  - User defined scripts or executables.
  - Abstract payload **Workflow** objects:
    - Workflow = Σ Steps
    - Step = Σ Modules
- **Production** management:
  - Formulated using payload workflow **Templates**.
  - Based on:
    - Varying input parameters.
    - Varying input data.

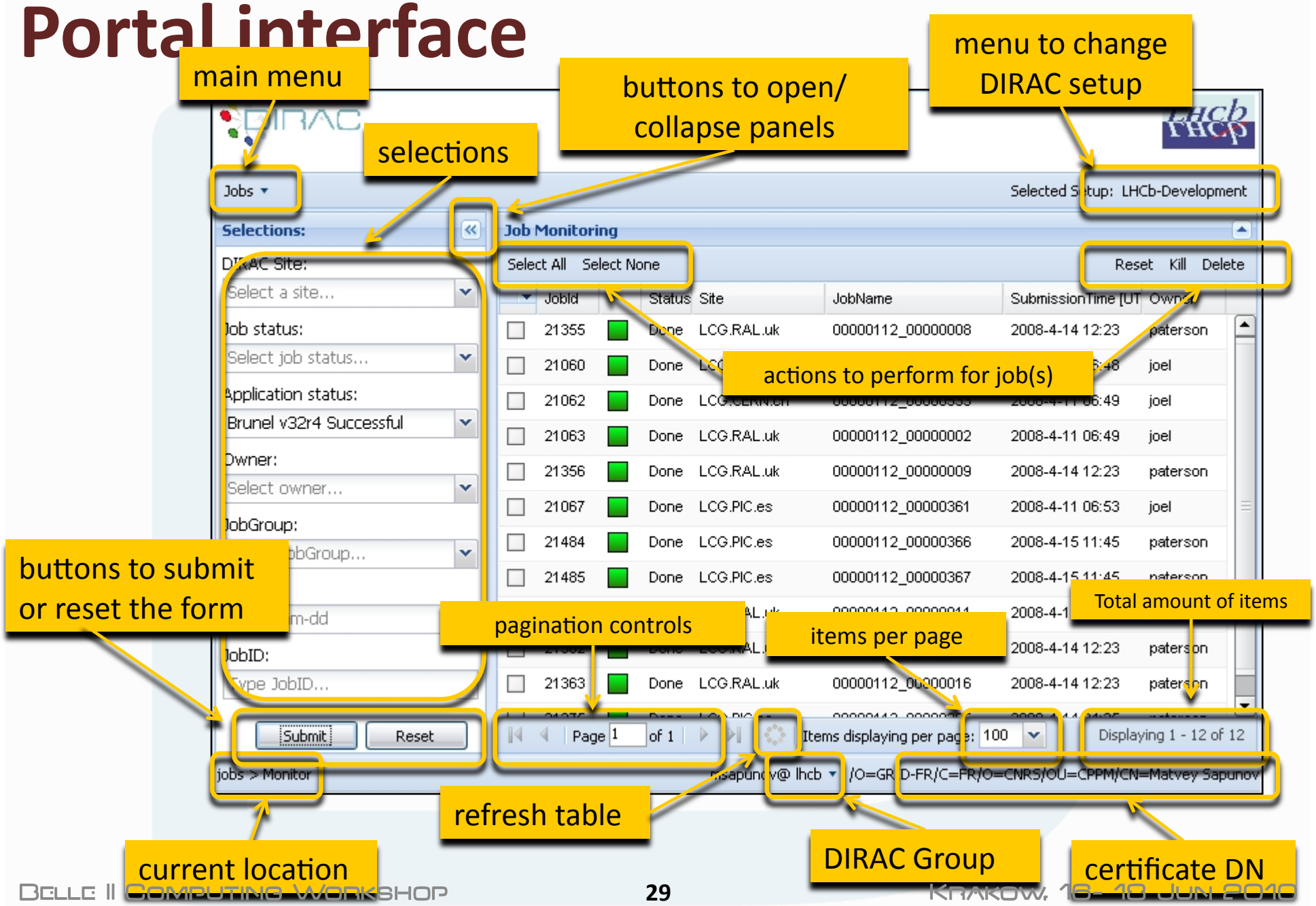
And all this with?

- Transparent **Integration** of Grid and non-Grid **resources**.
- Full security (authentication + authorization) based on **X509** proxies and certificates.
- Centralized System- and Security-**Logging**.
- Dedicated **Monitoring** and **Accounting** tools for all activities.
- And **Web Portal** for interactivity.

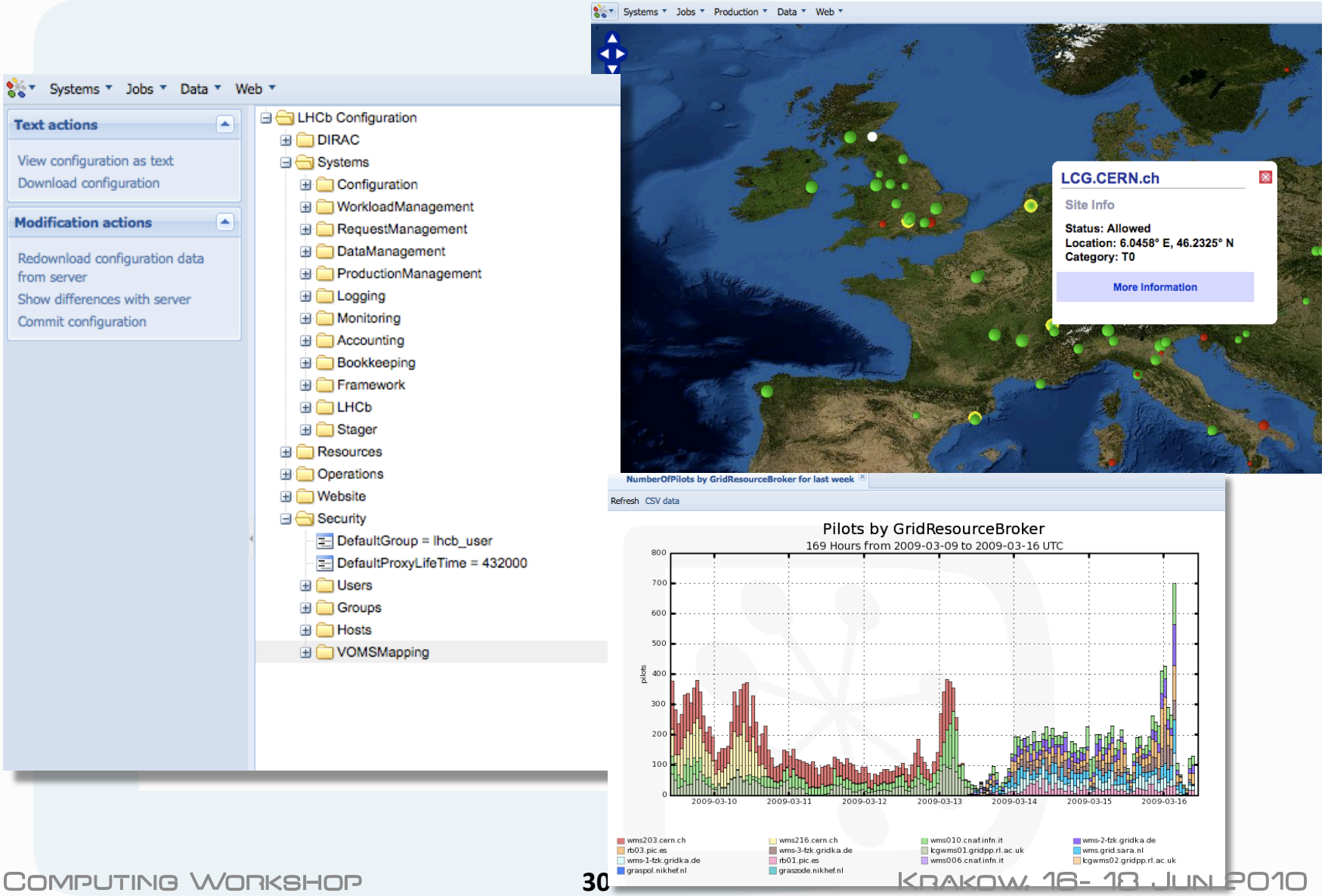
DIRAC Web Portal

- Provides a user **friendly** interface with DIRAC.
- DIRAC users are organized in groups, with **different privileges**:
  - The Web Portal reacts to the user's group
    - Really different profiles: production manager, administrator, analysis user…
  - Apply **authentication** and **authorization** rules to user requests
- **Secure** interface based on grid certificates
- Not only information display, it is a full **interactive web application**:
  - Take advantage of modern web technologies
  - Mimic a desktop application

Portal interface



More examples



Summary (What can DIRAC do for you?)

- **Simplify Management** of your computing Activities.
- **Minimize** your **Manpower**.
- **Easy User** transition.
- **Help** porting your **Application**.
- **Interactive Operation** for Users and Managers.
- **Full Control** and traceability.
- **Unify Access** to your resources.
- **Reduce** dedicated **Hardware**.

Everything with full Grid Standard Security.

What is coming next?

- Recoding of **Workload Management** with:
  - Better integration with DIRAC payload Workflow description.
  - Better handling of multi VO installations.
  - Remote Job Repository.
  - Improved handling of job transitions.
- Improve **multi-VO** support.
- Interactive **DIRAC python prompt**.
- **Submission** of "ad hoc" payload Workflows through Web Portal.
- DIRAC **File Catalogue** and Meta Data Catalogue (LFC is an LHCb choice)
- Whatever might come from interactions with other VOs.
  - We are waiting for you

Summary

- **DIRAC** is the software framework **develop in LHCb** (a large VO) to manage all its distributed computing activities.
- **DIRAC** is a flexible and configurable framework that can accommodate **other user communities**.
- Using DIRAC will **reduce the effort** to port your application to the Grid and will allow smooth **integration with non-Grid resources** (ie, super computers).
- DIRAC team is willing to **support** other user communities and **collaborate** with them to cover their use cases.
- We are waiting for your **comments** and **questions**.

Glossary

- **DIRAC**:

  Distributed **I**nfrastructure with **R**emote **A**gent **C**ontrol

  DIRAC is a software **Framework** to **Manage Distributed** computing **Activities** for a **User Community**.

- **DIRAC component**:

  Minimal functional entity in DIRAC:

  - **Server**: Passive DIRAC component, permanently running, waiting for queries or requests that are answered or inserted using a persistent backend.

    `JobRecieber`: gets new payload from the user and inserts it in the DIRAC Job DataBase.

  - **Agent**: Active DIRAC component, running either permanently in a loop or executing a limited number of iterations, that interacts with the persistent backend, either directly or via a server, and take actions.

    `JobAgent`: matches a pending payload and executes it.

  - **Script**: command line DIRAC component that allow the user to interact with DIRAC, each script, typically, exposes one function of DIRAC.

    ```
    dirac-wms-job-submit
    ```

  - **API**: python API DIRAC component that exposes a limited part of the DIRAC functionality that allow the user to achieve complex functionality.

    ```
    from DIRAC.Interfaces.API.Dirac  import Dirac
    Dirac = Dirac()
    ```

Glossary

- **DIRAC System**:

  Group of DIRAC components, and associated python modules, that offers a distinct functionality to DIRAC. No DIRAC system is functional by itself. DIRAC Systems are classified:

  - **Core**: minimal working set: Configuration, Logging, Monitoring
  - **Framework**: transversal components: Notification, Plotting, ProxyManager, SecurityLogging , Accounting, RequestManagement, Stager.
  - **General Purpose**: provide functionality to the user: WorkloadManagment, DataManagement, ProductionManagement.
  - **VO Specific**: extensions f or a given VO: LHCb, Bookkeeping

- **DIRAC System Instance**:

  The instantiation of the components of a given system using the same backend. Depending on the system there might be several instances of the Servers and Agents to achieve scalability, redundancy, fault tolerance.

- **DIRAC Setup**:

  A fully functional collection of DIRAC System Instances that is access by the end user.

  A given System Instance might be shared in different Setups.

  - **LHCb-Production**
  - **LHCb-Development**
  - **DIRAC-demo**

When talking about DIRAC we typically refer to a DIRAC Setup.