Tools to use heterogeneous Grid schedulers and storage systems

Mattia Cinquilli

INFN and University of Perugia

CMS Collaboration

Perugia

INFN

Istituto Nazionale
di Fisica Nucleare

Outline

- Grid: the advantage of the distribution
- Grid: different implementations
- CMS specific case
- Heterogeneity
- A standard interface:
  - to the middleware: BossLite
  - to the storages: SEAPI
- Results and conclusions

Perugia

INFN

Istituto Nazionale di Fisica Nucleare

Grid

- *A large-scale geographically distributed hardware and software infrastructure composed of heterogeneous networked resources owned and shared by multiple administrative organizations*

- Some advantages:
  - usage of distributed resources
  - responsibilities distributions
  - no central management: local

- Tends to be more loosely coupled, heterogeneous and geographically dispersed

- The ideal Grid has common protocols and standard interfaces...but:
  - different Grid implementations exist, with specific interfaces
  - different local site solutions are adopted and different storages can be supported

Perugia

INFN

Istituto Nazionale di Fisica Nucleare

Grid Implementations

- Various middleware projects have created generic infrastructure to allow various projects to harness a particular associated Grid, or for the purpose of setting up new grids.
    - EGEE, OSG, NorduGrid, ...
- Every implementation has its own characteristic with a specific middleware (software) that connects applications to resources.
- In some cases different Grid can interact with no changes in the middleware software, guaranteeing interoperability to access at resources of different Grid.
- But there is not guarantee about transparency at the application level

CMS Computing model

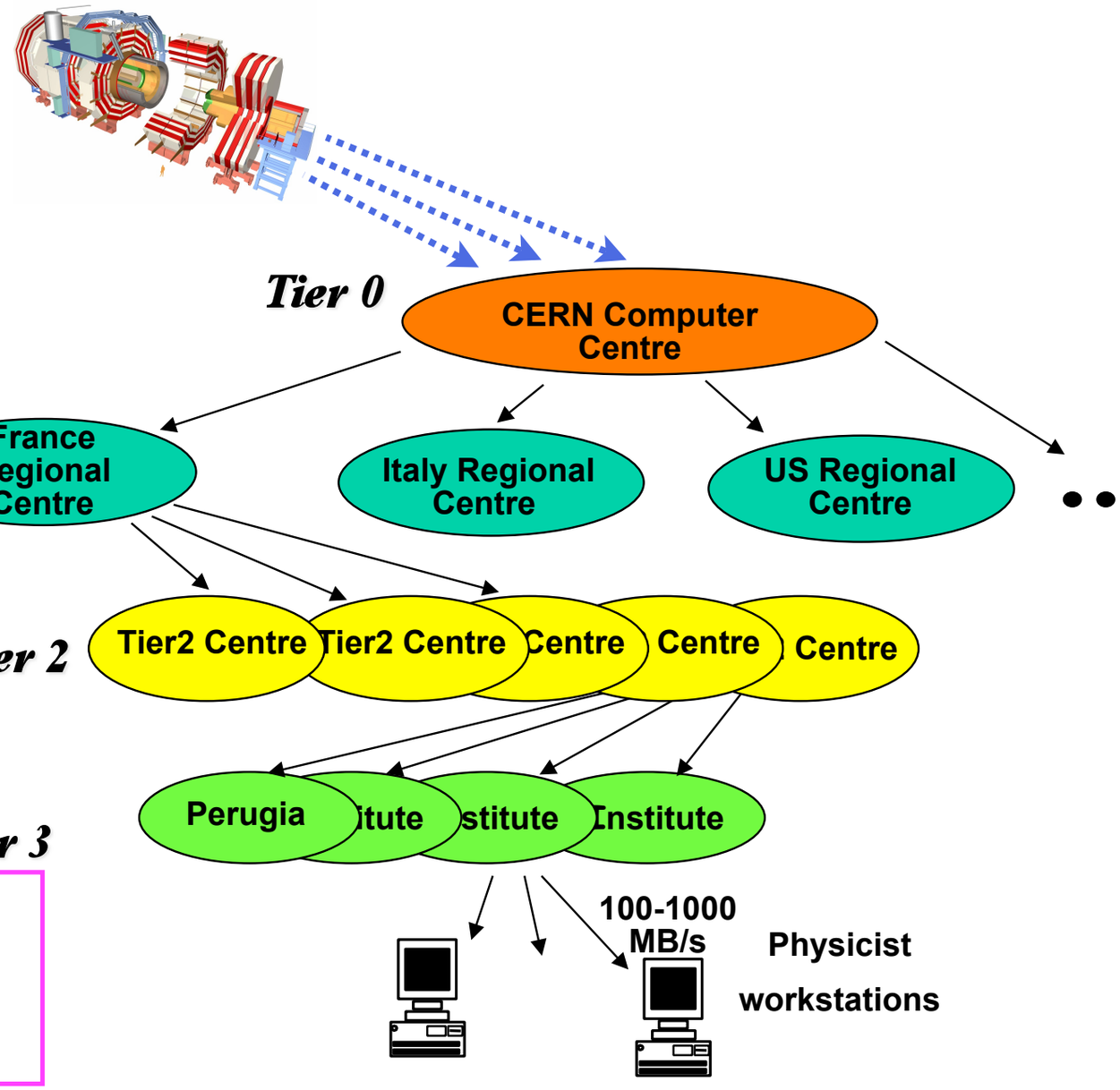
CMS is one of the four experiments integrated into LHC accelerator locate at CERN. It will register 1-2PB of raw data per year. It is formed by a collaboration of ~200 institutes and ~3000 physicist all around the world.

Complex system:

- access to distributed resources through Grid;
- access to local resources through batch systems (local farms and CERN Analysis Facility);
- access to specific CMS services (data management).

High job rate:

- large experimental community;
- huge amount of data produced by the experiment;
- comparable amount of Monte Carlo data samples to be generated and accessed.



CMS has **developed dedicated applications** to interface with the complex system satisfying the requirements without penalizing the performances.

Heterogeneity

- **Middleware**: many implementations exist and in some case it is needed to access resources belonging to different Grid infrastructures.

- **Storage Elements**: choices delegated to local sites; different type of storage elements can have different interfaces and allow different set of operations.

- **Standard interface** needed for the applications, in order to have:
  - optimization in the infrastructure integration
  - easier development
  - chance to test same use cases in different scenarios

Computing infrastructure heterogeneity

- Different infrastructure implementations are used by CMS:
  - Grid: EGEE, OSG, NorduGrid
  - plus local batch system (LSF, PBS, ...).
- Grid middlewares are **interconnected** to guarantee the **interoperability**, but:
  - **different middleware softwares** and **different middleware architectures** have to coexists.
- From the applications point of view it is needed to manage the middleware heterogeneity and to optimize their usage.

BossLite

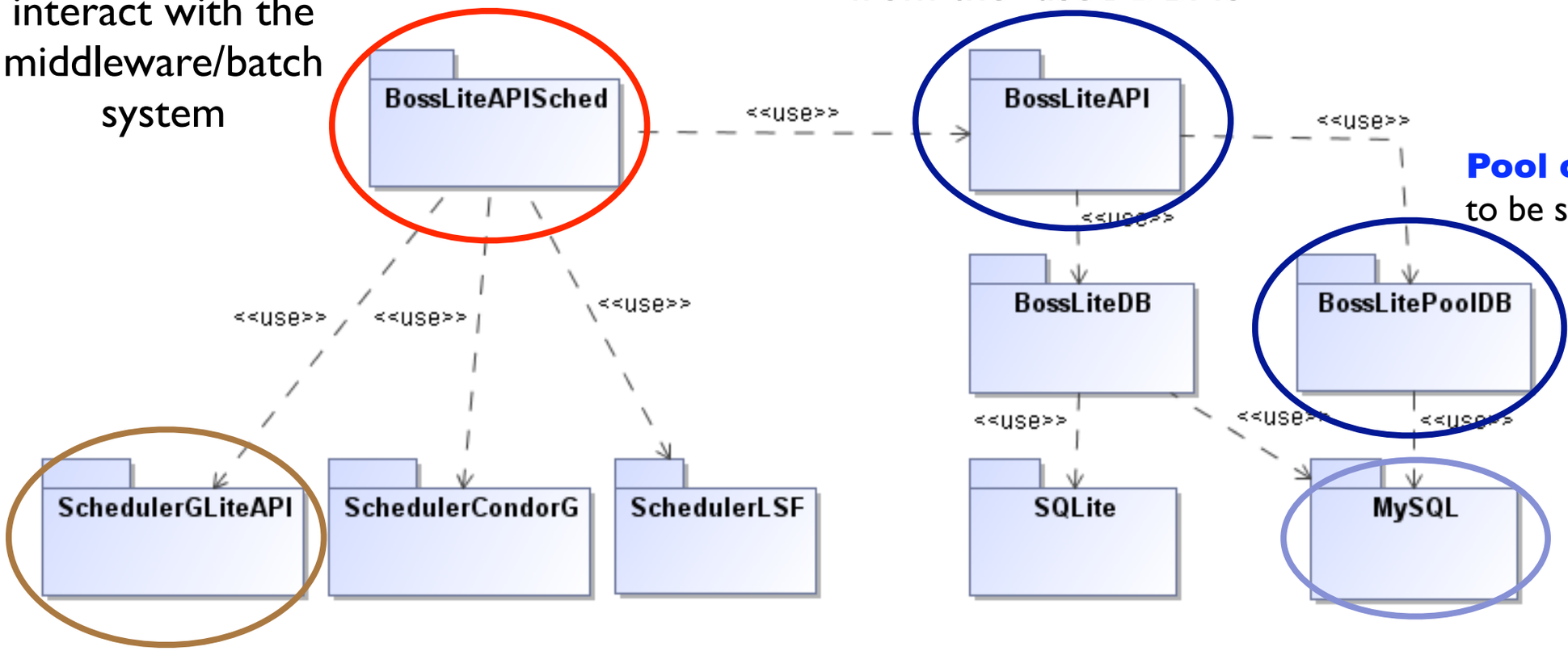
a common Grid/batch interface with logging facilities

- CMS interface to different Grid [WLCG, OSG] and batch systems [LSF, ARC, SGE…]
- Database to track and log information into an entity-related schema
- Information logically remapped into python objects that can be transparently used by the CMS framework and tools
- Requested high efficiency and safe operation in multithreaded environment
- Database interaction through safe sessions and connections
  - Connections pool
  - Focused on connection stability
  - **Thread safe** (to use thread for the speed up in Grid interactions)**.**

BossLite Architecture

Standard interface
allows to transparently interact with the middleware/batch system

Standard interface
to access static and runtime information independently from the used DBMS



Pool of DB connection
to be shared among threads

Database back-end
used for logging and bookkeeping

Plug-in
implements the instructions to interact with a specific middleware/batch system

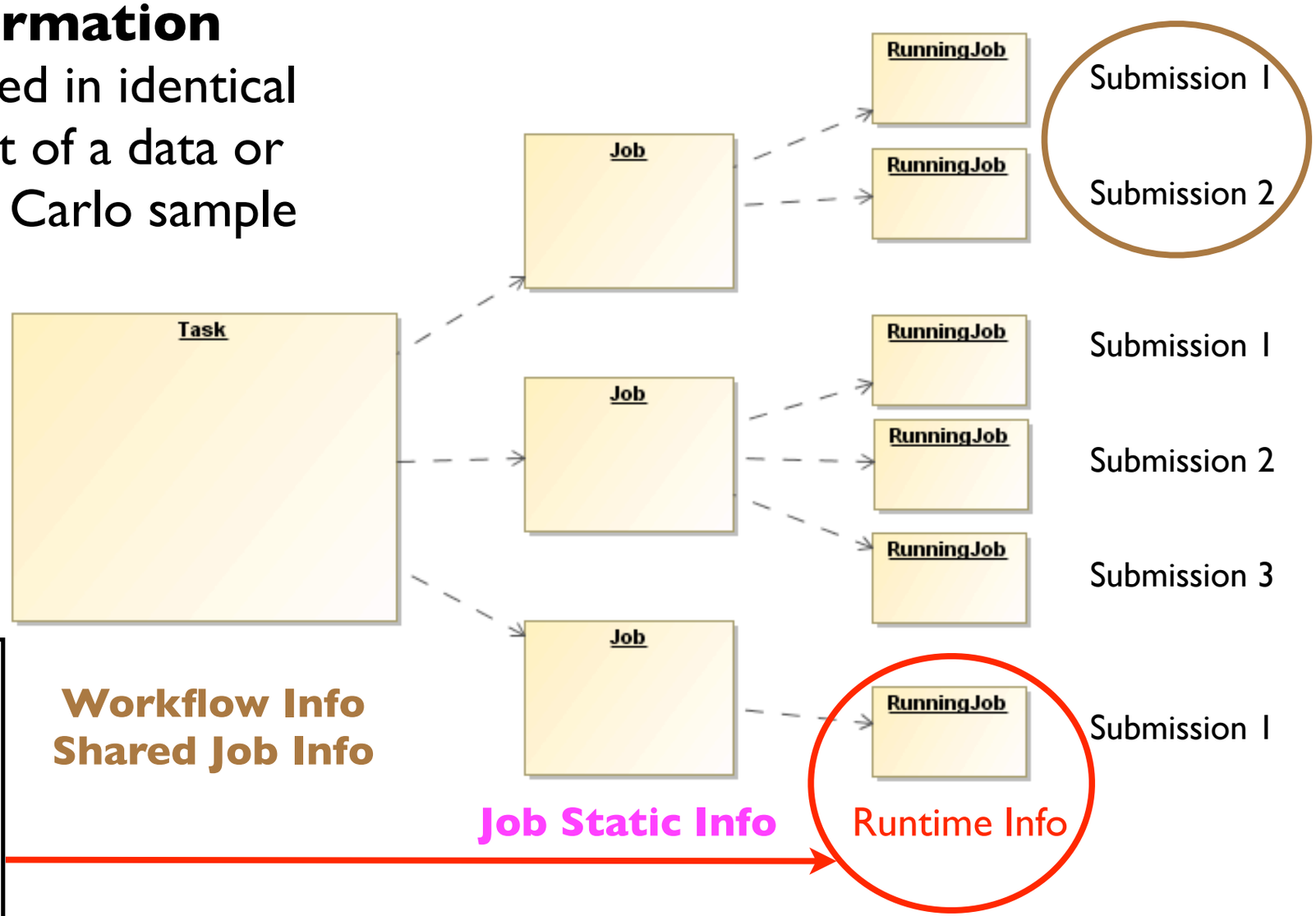
This architecture allows to:
- manage the heterogeneity
- give a transparent access
- optimize middleware/batch interaction
- scale using multithread approach
- guarantee a good extendibility and parallel development

BossLite

Task description

User Workflow Information

Represented as a ***task*** splitted in identical ***jobs*** accessing different part of a data or producing a part of a Monte Carlo sample



Contains runtime information about each submission of the job:

- middleware/batch type (plug-in name)
- middleware/batch id
- status
- destination (ce/wn hostname)
- timestamps
- ...

Structure *flexible* enough to *allow many workflows* managed by the higher software level

Storage Elements

- A Storage Element is an independent service that provides uniform access to storage resources.
    - Scale on quantity of data.
    - Hide the complexities of the storage system.
- Some implementations:
    - dCache: storing and retrieving huge amounts of data, distributed among a large number of heterogenous server nodes, under a single virtual filesystem tree with a variety of access methods.
    - StoRM: light and scalable storage manager service for generic disk based storage system using high performance storage systems based on cluster file system.
    - DPM: the evolution and the replacement for deprecated classical SE adding many advantages to it.
- Various protocols are adopted: GsiFTP, RFIO, SRM, ...
- And many clients can be used to interact with the storage facilities: globus commands, uberftp, rfio, srm, ...

- Need to have a common interface independently from the protocol/client used.

Storage Element API

- **SEAPI** (**S**torage **E**lement **A**pplication **P**rogram **I**nterface) is the tool that has been developed to handle a complete set of operations on many storage facilities by using different protocols and clients.

- It is a simple **framework** which collects the specific requests and performs the operations, translating every single request into the command to be performed and returning detailed results.

- Aim of the development:

  - to give a complete **transparence of usage** by a common interface that links each operation that can be performed, **independently to the used protocol**;

  - the **interaction between different storages** when executing operations that include two storage resources (copy operation);

  - to **handle** with accuracy different kind of **errors** and **failures**, depending on the storage, the protocol and the operation being performed.

Perugia

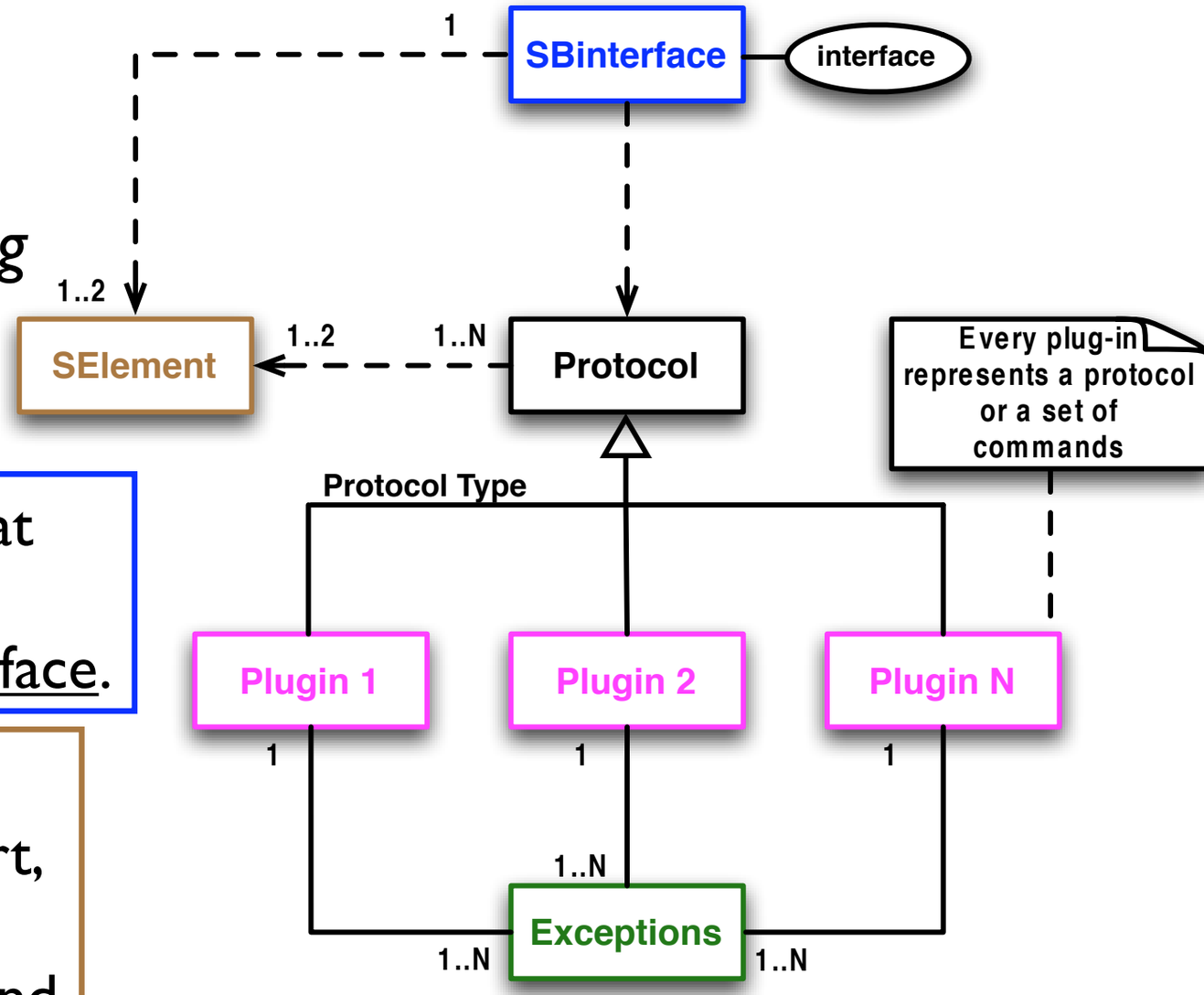
INFN

Istituto Nazionale di Fisica Nucleare

SEAPI

Implementation and architecture

To have an easy **extendible** and **maintainable** tool and to face with the heterogeneity, *modular programming* has been used, through ***plug-ins***.



interface: generic entity that load plug-ins at run time and translates requests; it is an intermediate which provides a standard interface.

storage: represents a Storage Element including information about it (hostname, port, protocol, ...); it is used by the interface to understand which plug-in is needed to load and by the plugin to execute the command.

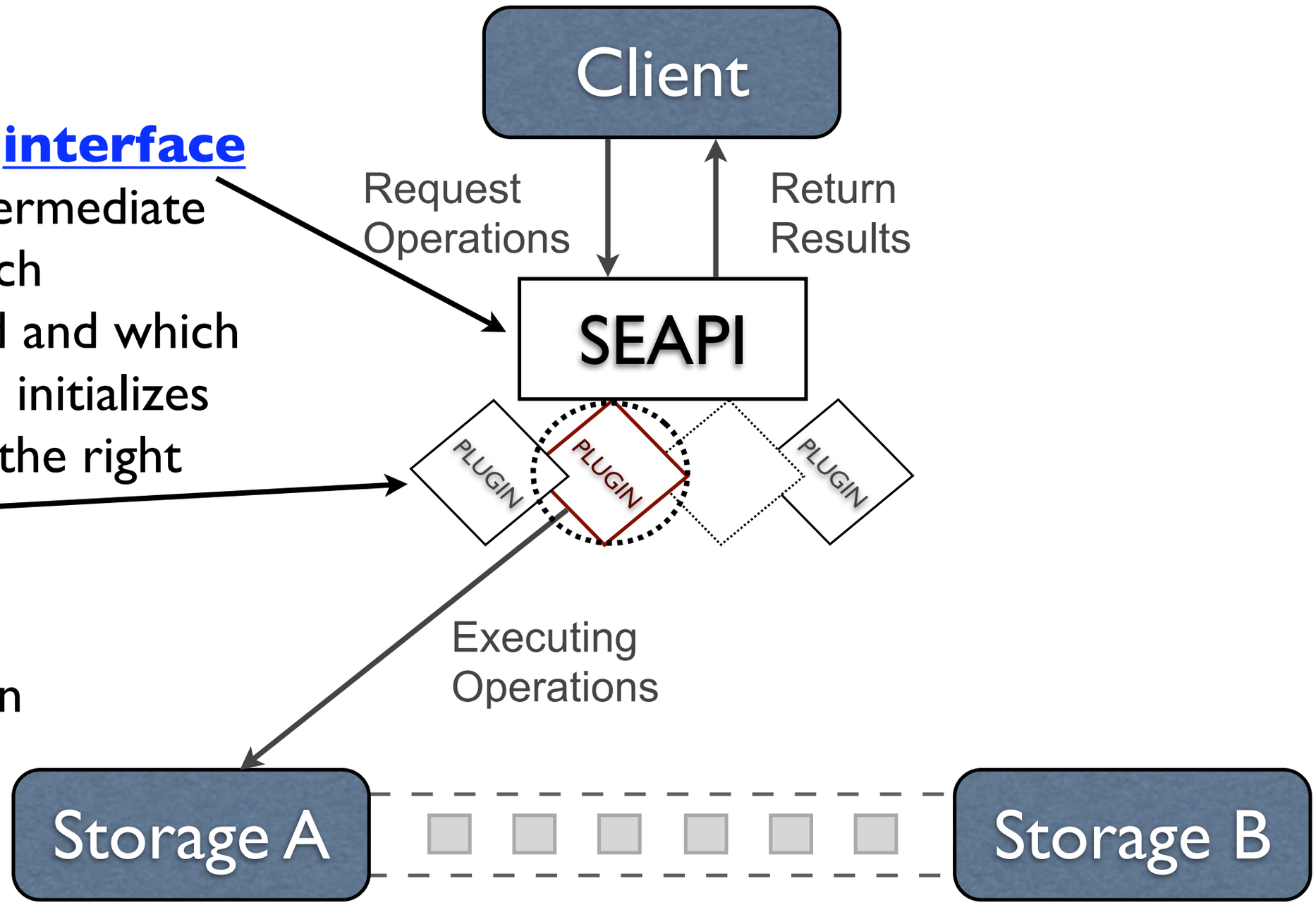
plug-ins: each plug-in represents a different protocol and contains the right set of the protocol instructions and options.

error manager & exceptions: entities that manage errors coming from invalid requests (eg: protocol mismatch) or from the command execution (eg: looking at the operation results)

SEAPI: how it works

Common and generic **interface** which works as an intermediate and depending on which operation is requested and which protocols are involved initializes and dynamically loads the right **plug-in**

The selected **plug-in** performs the operation specifying default and user options.



Managing credentials inside the plug-ins (proxy, kerberos)

Perugia

INFN

Istituto Nazionale di Fisica Nucleare

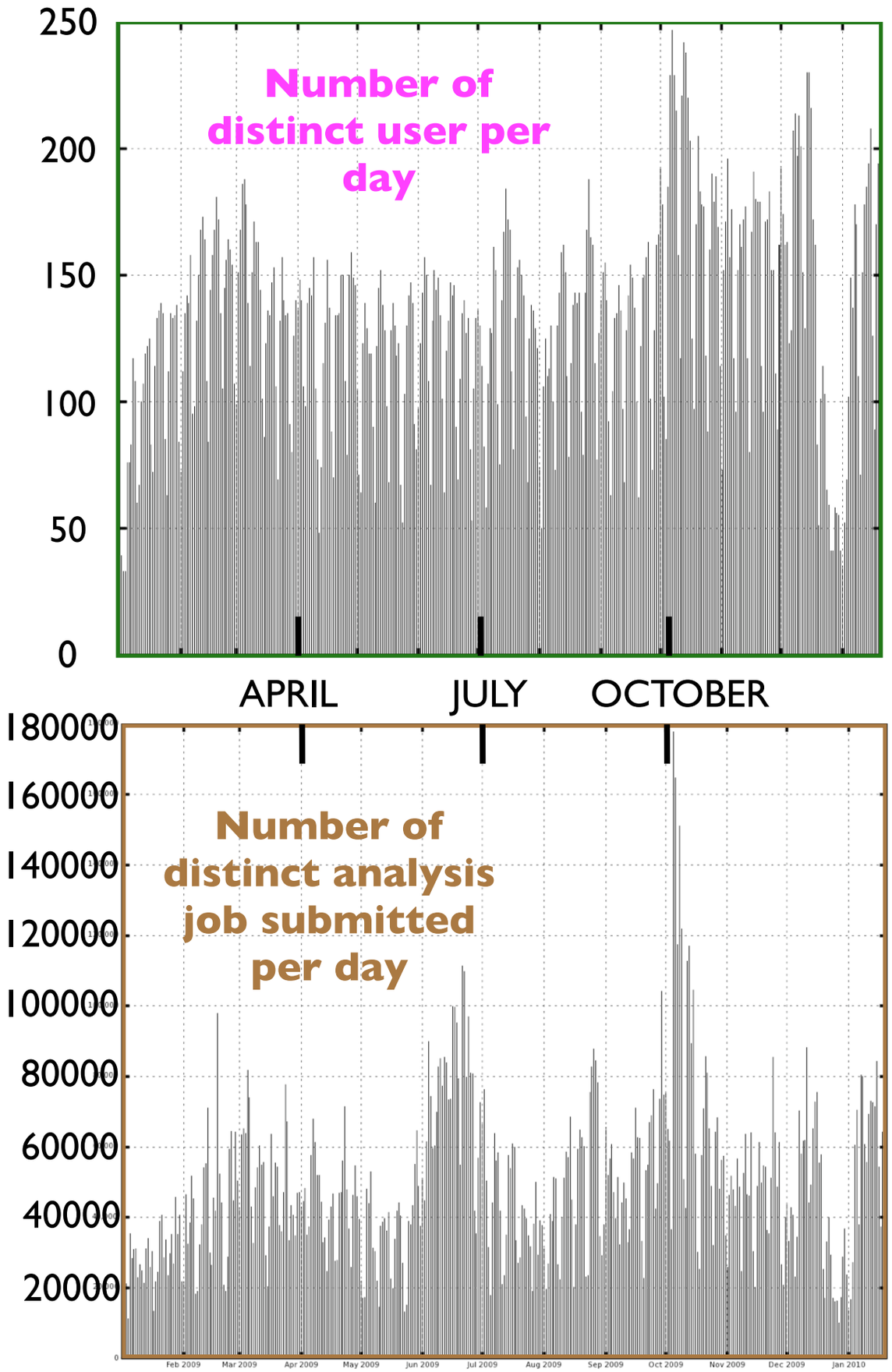
Results

- BossLite

  the standard framework used by CMS user analysis tool, private and central production tool to interact with all the middleware/ batch systems

- SEAPI

  the standard interface used by CMS user analysis tool and private production in every interaction with storages (remote stage out, sand box transfers, ...).

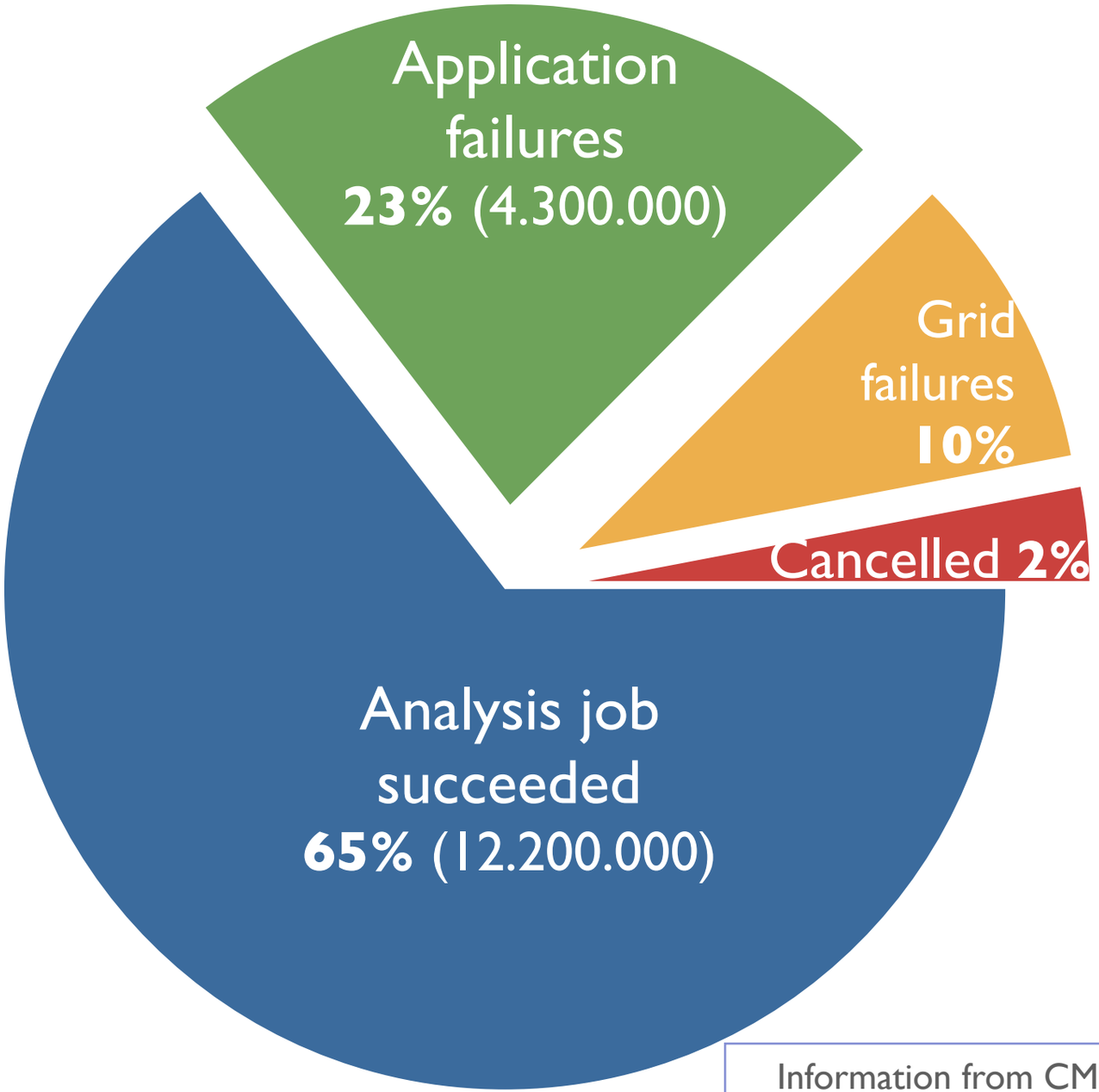


Information from CMS official monitoring (Gen. 2009 - Gen 2010)

Results

Analysis job submitted in 2009:
18.900.000

→ Average:
~52.000 job/day



Application failures

mainly due to:

- remote stage out failures
- errors in user code
- technical problems at local site
- tool failures

Grid failures

mainly due to:

- middleware failures
- user interaction not optimized

Information from CMS official monitoring
(Gen. 2009 - Gen 2010)

Conclusions

- BossLite and SEAPI have been deployed in CMS Workload Management (analysis and production) tools since the start of 2008.
- The most crucial design metrics have been demonstrated:
  - during the **daily usage by applications in production**
  - during specialized **CMS challenges** to verify the readiness of the applications and the infrastructure itself.
- The developed tools have optimized the integration of the middleware with high level applications.
- Future:
  - SEAPI optimization for the remote stage out of user results
  - Introduction of new plug-ins and update of already existing plug-ins.

Perugia

INFN

Istituto Nazionale
di Fisica Nucleare

THE END

- BACK UP SLIDES

Grid Characteristics

- **Heterogeneity**: a grid hosts both software and hardware resources that can be very varied ranging from data, files, software components or networks...
- **Multiple administrations**: each organization may establish different security and administrative policies under which their owned resources can be accessed and used.
- **Consistent access**: a grid must be built with standard services, protocols and interfaces thus hiding the heterogeneity of the resources while allowing its scalability.
- **Pervasive access**: the grid must grant access to available resources by adapting to a dynamic environment in which resource failure is commonplace.
    - The grid must tailor its behavior as to extract the maximum performance from the available re-sources

Grid distribution of responsibilities

- Grid is **distribution**: not just of resources but also about responsibilities and management.
- **No central management**: computational and storage resources are implemented and managed by the local sites.
  - Advantages: reduced costs, delegated responsibilities, improved load balancing...
- Depending on local choice and requirements, **different Grid sites can adopt different solutions.**
- This takes to a not predictable heterogeneity that even the Grid cannot hide to the application/user level:
  - interaction between application and resources is not connected by the Grid itself as for Data Repository (Storage Element).

Perugia

INFN

Istituto Nazionale di Fisica Nucleare

Compact Muon Solenoid

- CMS is one of two large general-purpose particle physics detectors integrated into the proton-proton Large Hadron Collider (LHC) at CERN (European Organization for Nuclear Research).
  - It has 15 millions of channels; through them data will be taken at a rate of 20 MHz/s and then selected by an on-line selection system (trigger) that will reduce the frequency to 100 Hz (writing data frequency):
    - this means 100 MB/s and ~1 PB data per year.
  - It is formed by a collaboration of ~200 institutes and more then 2000 physicist all around the world.
- To satisfy the usage of distributed resources and to manage the big quantity of data, CMS has defined a computing model based on the Grid.

Perugia

INFN

Istituto Nazionale di Fisica Nucleare

BossLite interface to gLite

- Bulk submission, bulk match-making, bulk status query
    - Faster, more efficient
- Access through WMProxy python API
    - Needed to allow the association between BossLite jobs and their Grid Identifiers (not trivial through the CLI)
    - No parsing of "human readable" streams
    - But the use of the API is complex and the UI tools (e.g. UI configuration, input sandbox transfers, ...) are lost
    - Exposed to python compatibility issues
- Access to LB information through API
    - Easy and fast check of job status
    - Easy to extract many more useful information at runtime: destination queue, status reason, scheduling timestamps…