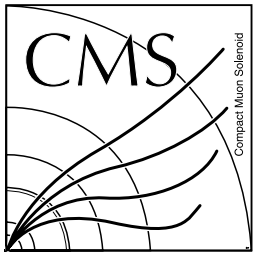
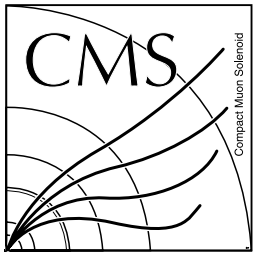


Client/Server Grid applications to manage complex workflows

Filippo Spiga*

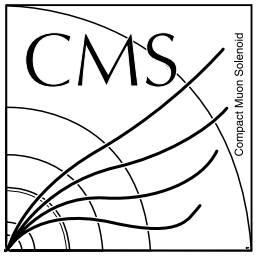
on behalf of CRAB development team

* INFN Milano Bicocca (IT)



Outline

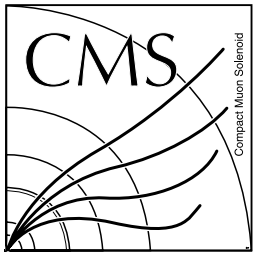
- Science Gateways and Client/Server computing
- Client/server approach for CMS distributed analysis
- Usage statistics and trends
- Considerations, remarks and future works



Science Gateways & Client/Server Computing

- A Science Gateway is a community-developed set of tools, applications, and data collections that are integrated via a portal or a suite of applications.

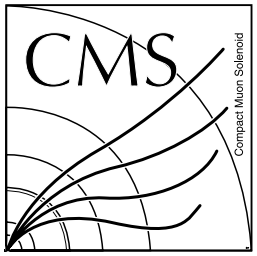
- The Client/Server Computing is a distributed application architecture that partitions tasks or work loads between service providers (*servers*) and service requesters (*clients*).



Science Gateways & Client/Server Computing

- Science gateways and client/server applications …
  - share the main fundamental ideas
  - address the same purposes
  - are realized using different technologies
  - are deployed in different ways
  - interact with users through different interfaces

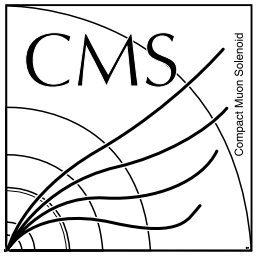
→ The choice between them will be made considering the target community (both users and developers)



Motivations & Advantages

(for users)

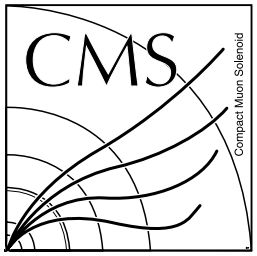
- In the context of HEP computing, the client/server applications can…
  - automate (as much as possible) complex analysis workflows
  - allow more advanced job monitoring with centralized front ends
  - hide configuration complexity on heterogeneous grid environment
  - hide implementation details to the users
  - automatic resubmission in case of failure
  - allow (strong) centralized policies



Motivations & Advantages

(for operators and developers)

- In the context of HEP computing, the client/server applications can…
  - extend/integrate the functionalities of the middleware
  - help to manage centrally updates
  - improve support to users by operators
  - avoid destructive congestions interacting with the grid
  - improve the scalability of the entire “computing model”



The experience with CRAB tool in CMS

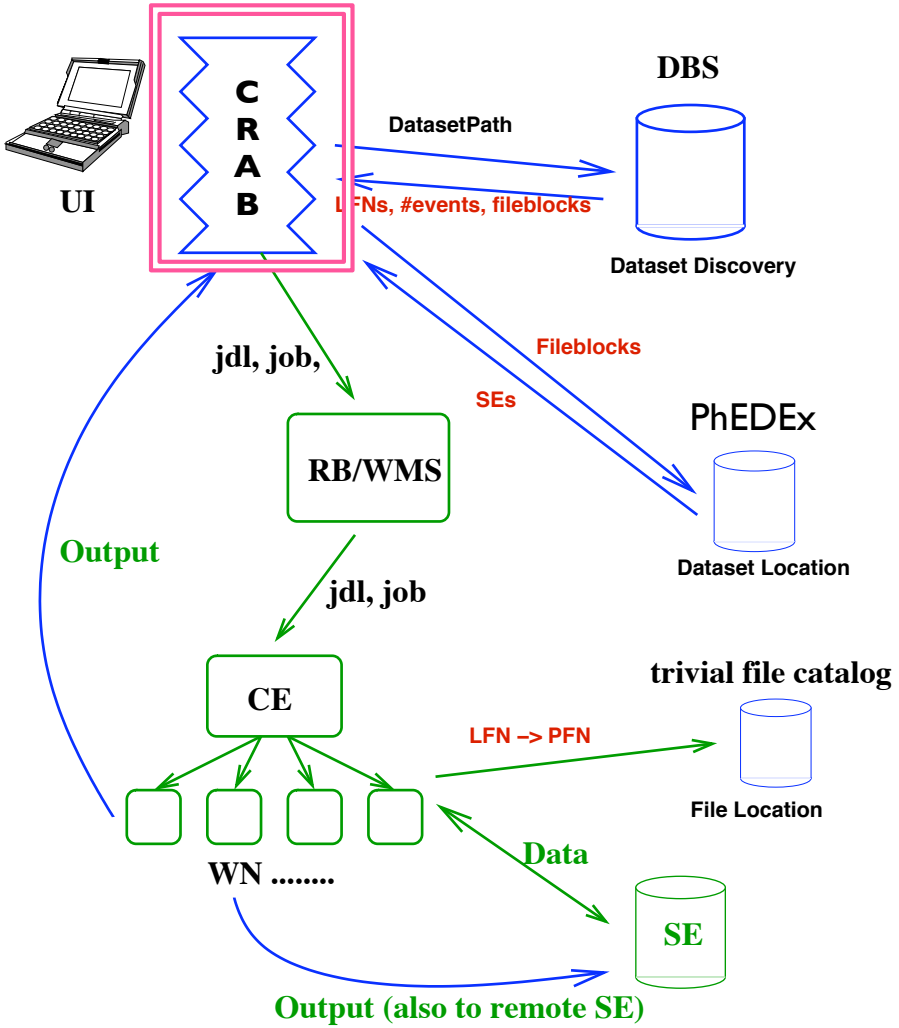
CRAB, *a Python tool intended to create, submit and manage CMS analysis jobs in a Grid environment.*

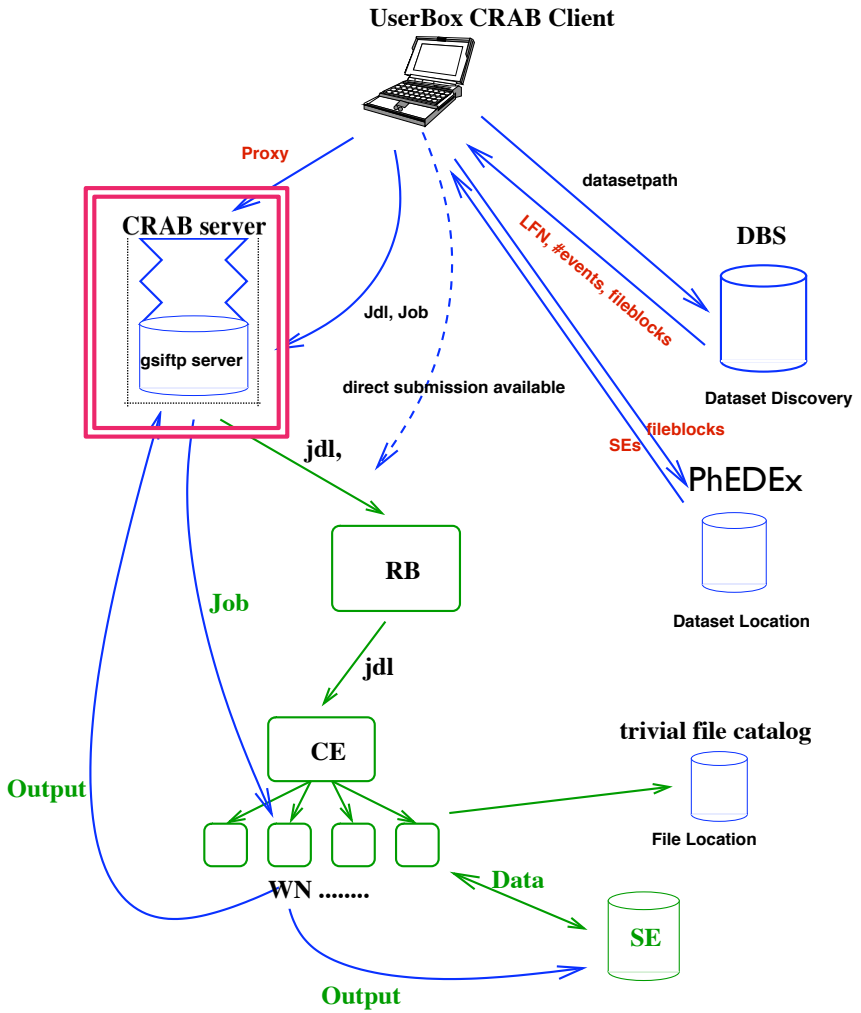
- A single tool addresses different applications/purposes:
  - CMSSW analysis over the Grid
  - private MC production
  - complex workflow automation
- Started as standalone application

  → it is evolved to a client/server application (in production since July 2007)

Standalone vs Client-Server

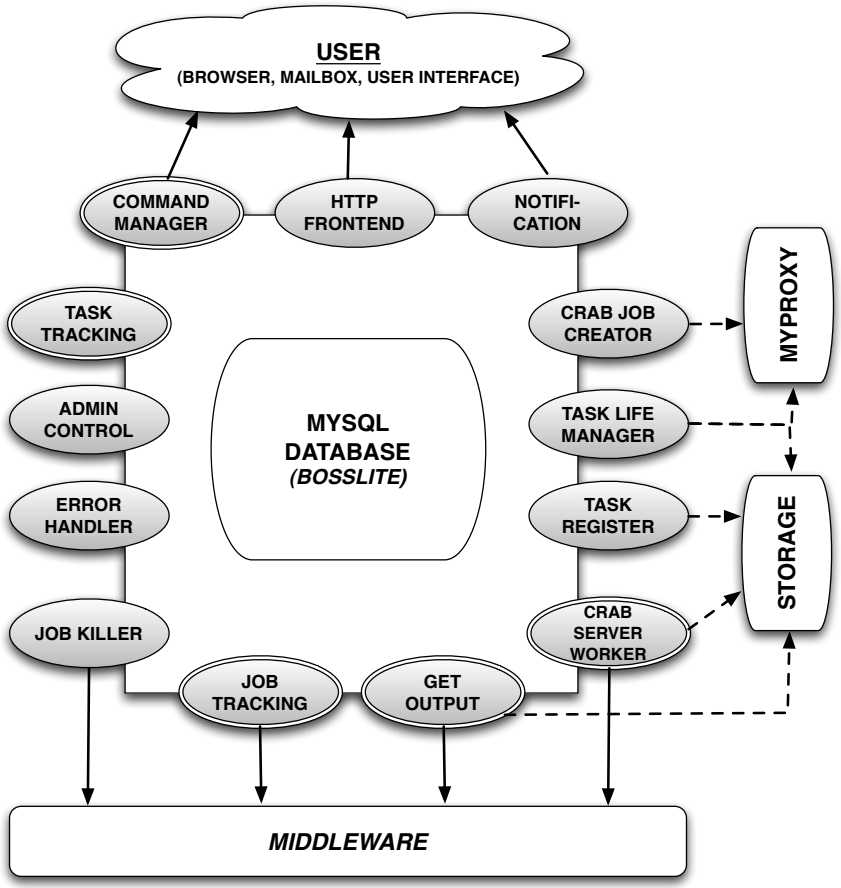
(workflow point of view)



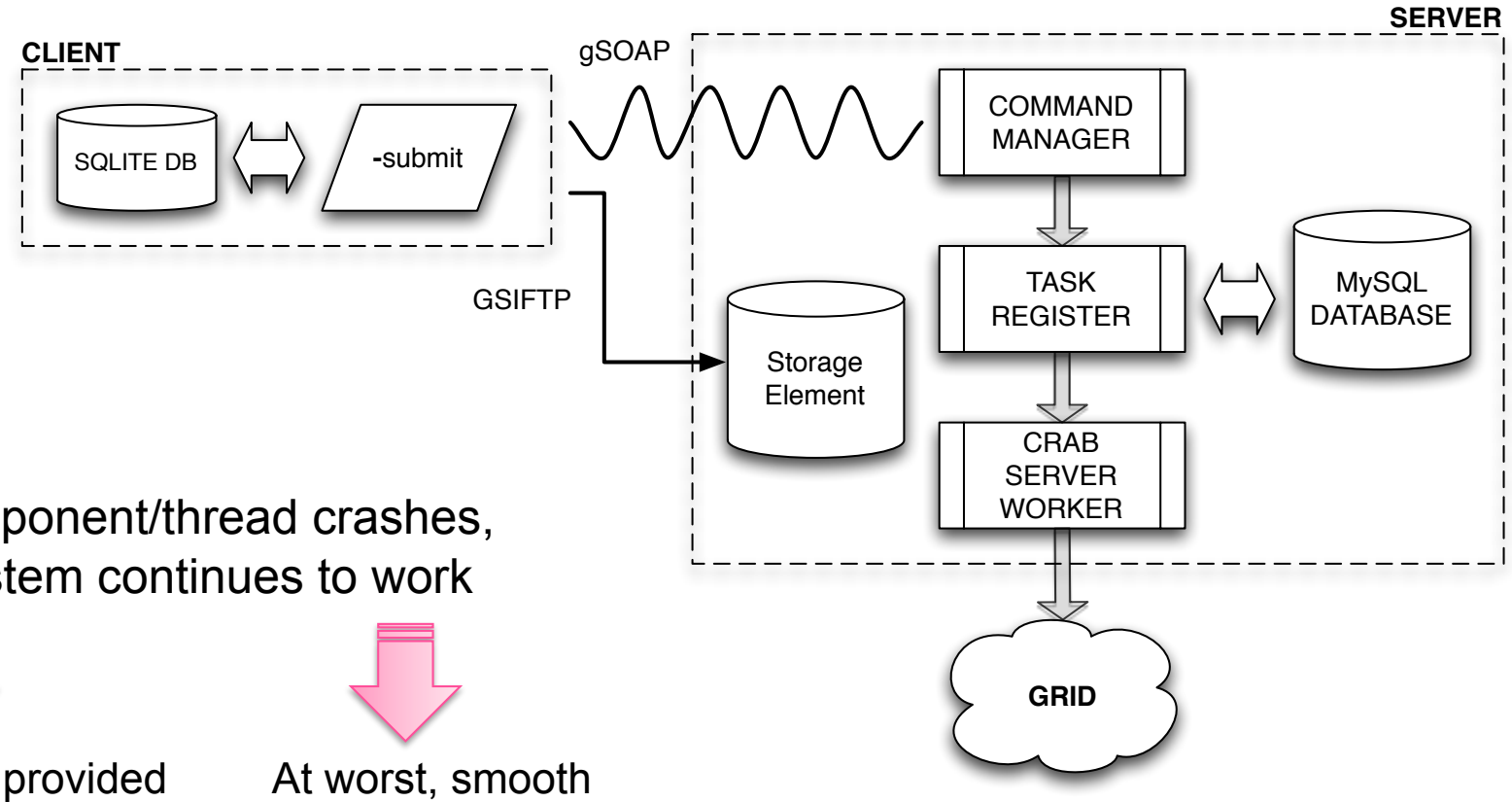


Server architecture

- Agent-based model
- Publish&Subscribe message service
- Most components are multi-threading
- Transparent support to multiple storage systems (SEAPI) and multiple middlewares (BossLite)



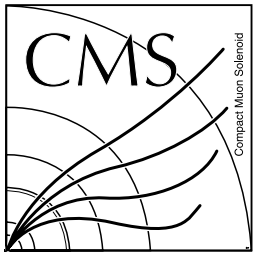
Components interaction example
(submission)



If a component/thread crashes, the system continues to work

Other services provided by the server will be not compromised

At worst, smooth degradation of QoS



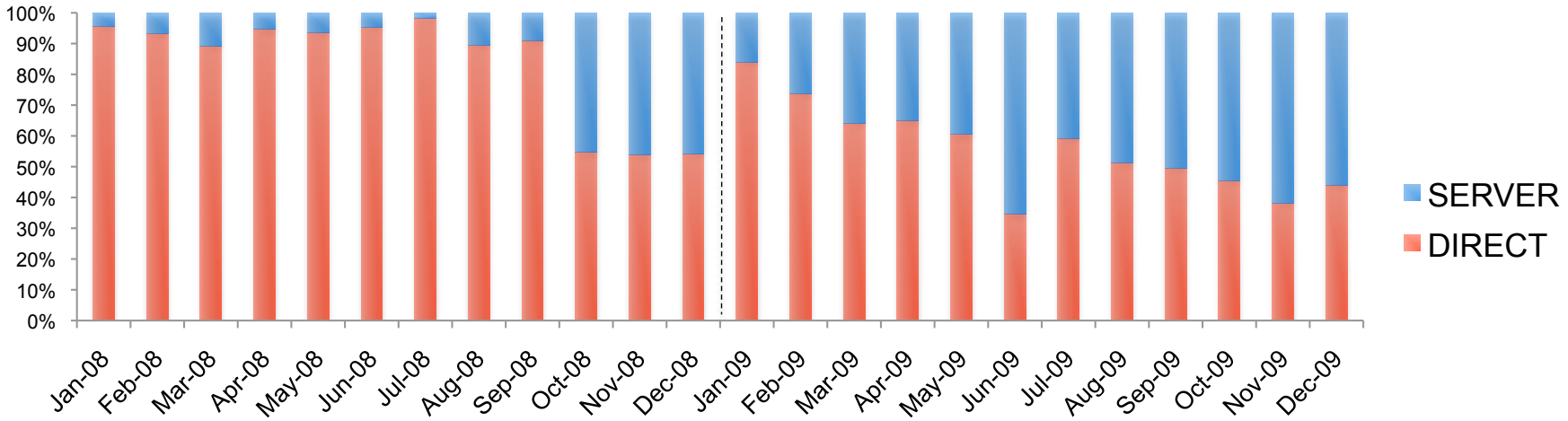
CRAB deployment on Grid infrastructure

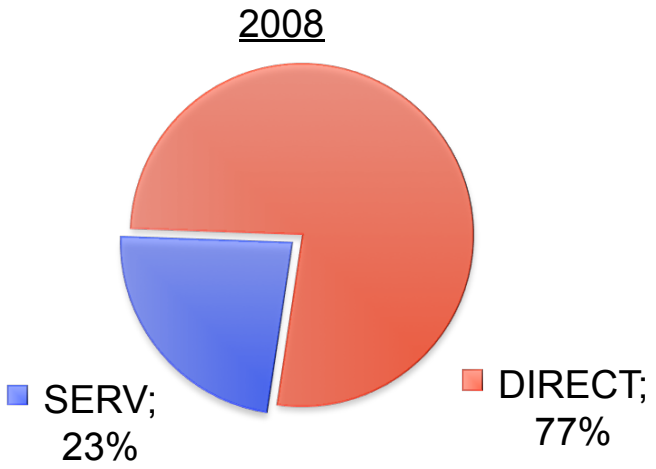
- CRAB deployment strategy for CMS
  - enforce fault-tolerance
  - avoid single point congestion
  - gLite is the main middleware used
  - periodic challenges are performed to test and stress the infrastructure
- Different CRAB-AS in production now
  - located at CERN (CH), UCSD (US), BARI (IT)
  - each physic groups can deploy it's own unofficial server
- Analysis Operators' team (AnaOps) manages and monitors the services

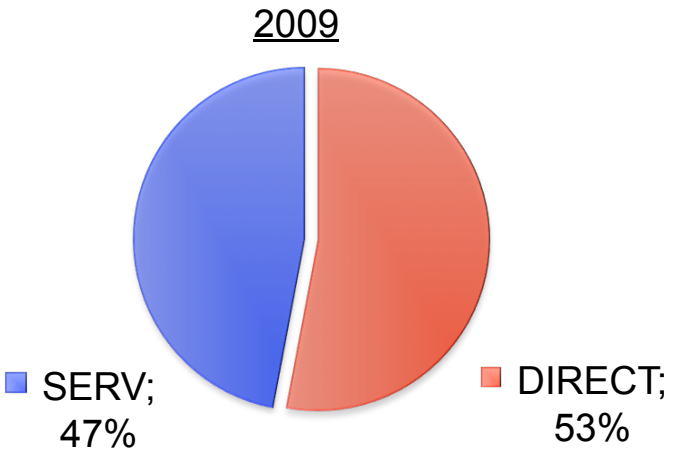
Statistics: starting point

- Source of statistics: CMS Dashboard
- Focus on comparison between standalone and client/ server approaches
  - trend analysis from Jan 2008 to Feb 2010
- Statistic is not “perfect”
  - sometimes testing jobs submitted by CRAB developers are counted as analysis jobs…
- Only analysis jobs on T2 level were considered

CRAB Standalone vs Server
(2008 – 2009)



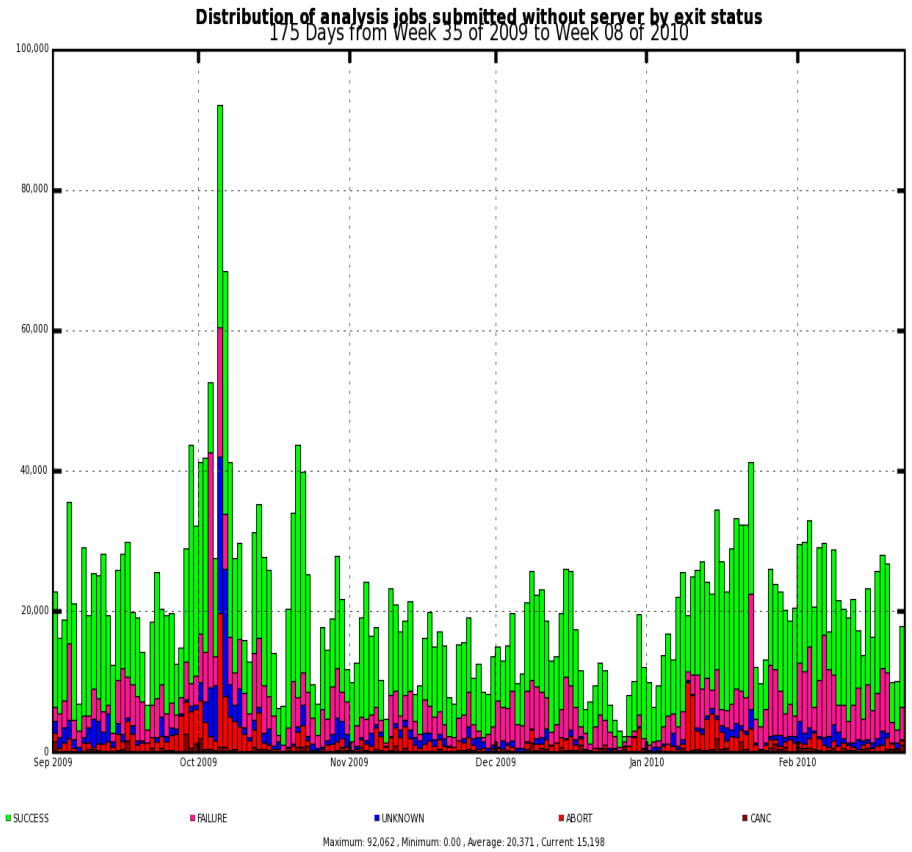


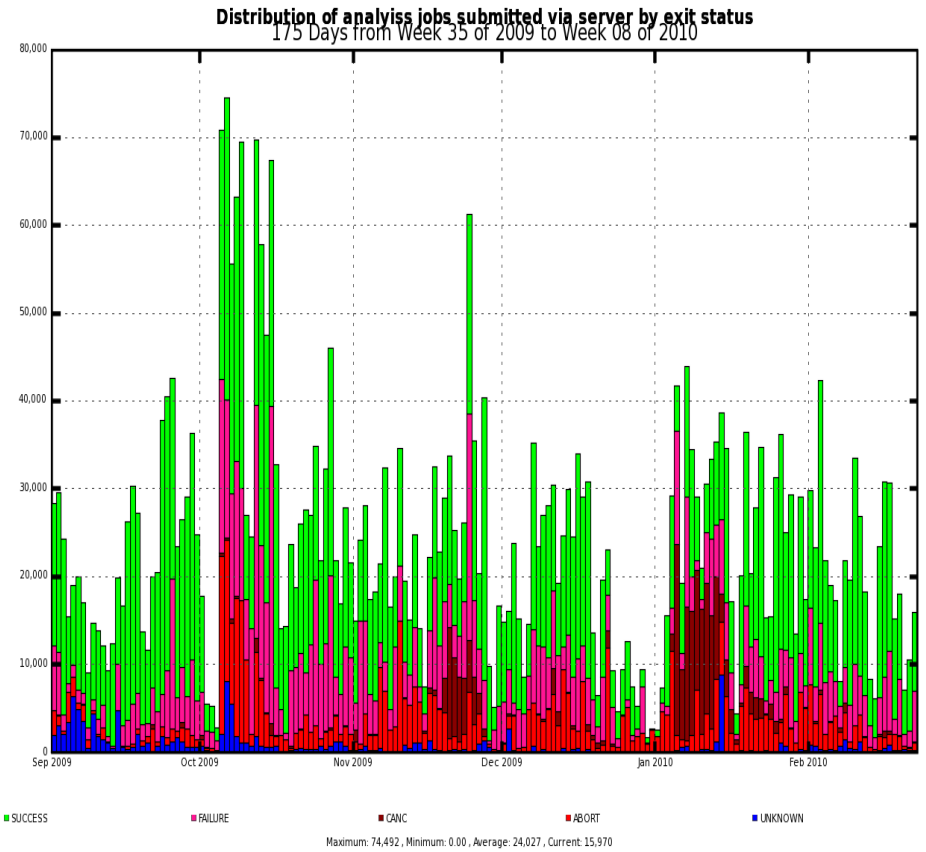


Source: CMS Dashboard, queried @ 27/03/2010

CRAB Standalone vs Server

(last 6 months, Sep'09 - Feb'10)





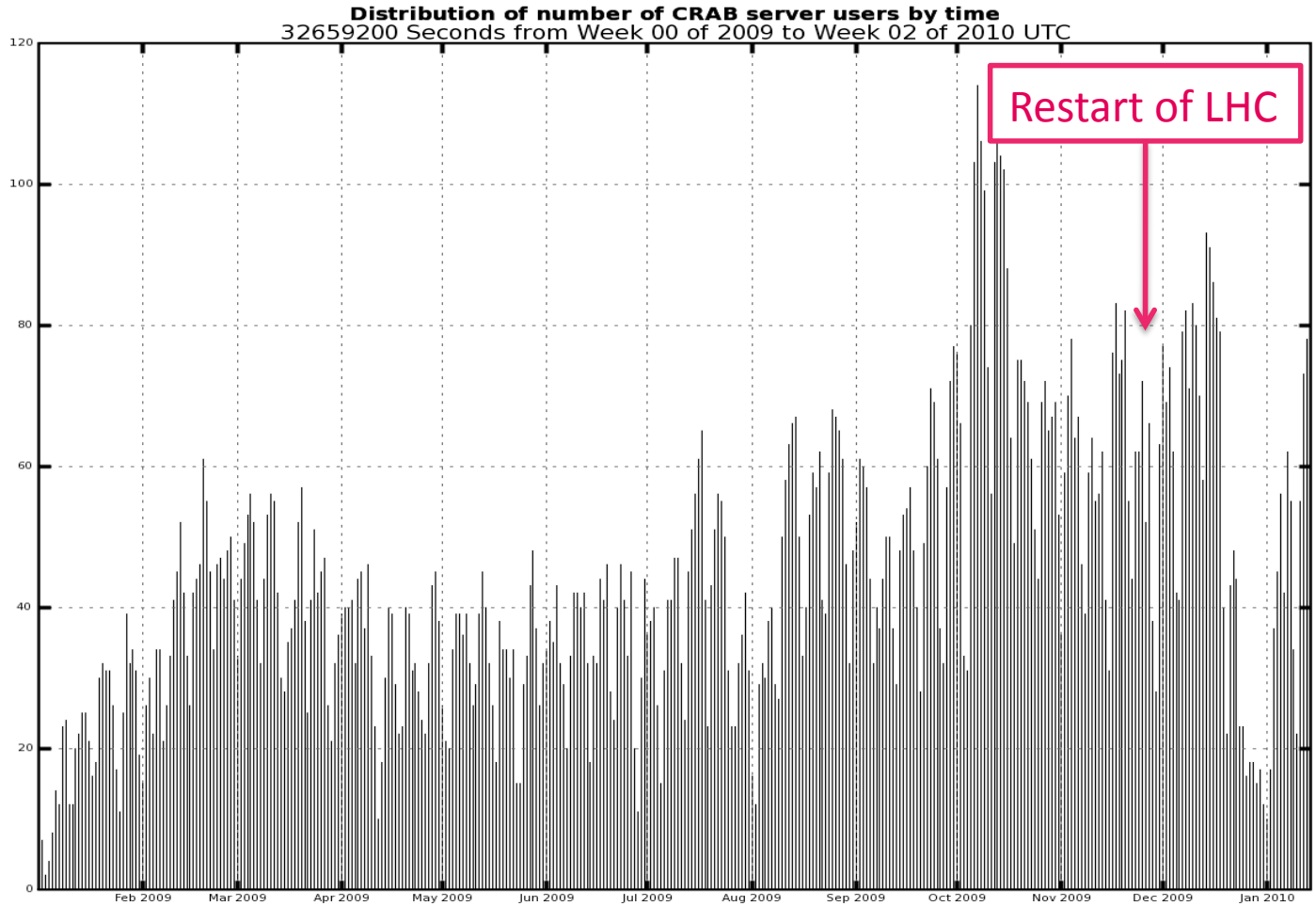
→ 54% of submissions were made through the server

Source: CMS Dashboard, queried @ 27/03/2010

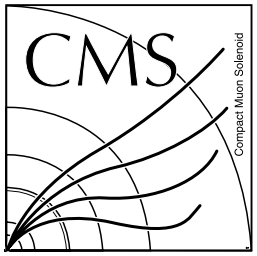
CRAB Standalone vs Server
(2009)

Number of distinct CRAB users:
150~200/day

Max distinct users peak reached:
~250

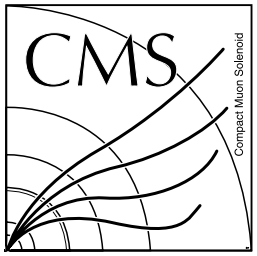


Source: CMS Dashboard, queried @ ~02/2010



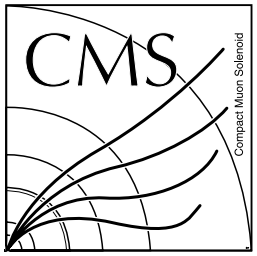
Considerations & Remarks

- Architectural aspects of the software are the key factors to reach success
  - good performance achieved
  - good stability and reliability
- Inside CMS community, the adoption of client/server approach is increasing
  - good feedback by users
  - good quality of service reached
- The only (relevant but known) problems are related to data movement
  - stage-out remains a critical operation across the Grid, influenced by several factors



Improvements & Future works

- Consolidate the actual architecture
  - change the interaction with gLite middleware (CLI instead of API)
  - better strategies for stage-out operations
  - maintain the best interoperability for all middlewares supported
  - reduce the number of jobs aborted
  - improvements for intelligent resubmission
- New framework for all CMS computing: WMCore
  - one application targets the online (T0), the official production (T1) and the user analyses (T2)
  - communications based on RESTful web services
  - too young to go to production (expected for…)



THANK YOU FOR YOUR ATTENTION

Questions?

Special thanks to Daniele Spiga (CRAB), Fabio Farina (CRAB), Mattia Cinquilli (CRAB) and Julia Andreeva (CMS Dashboard)

The CRAB history

(backup slide)