# LHCb use of batch systems

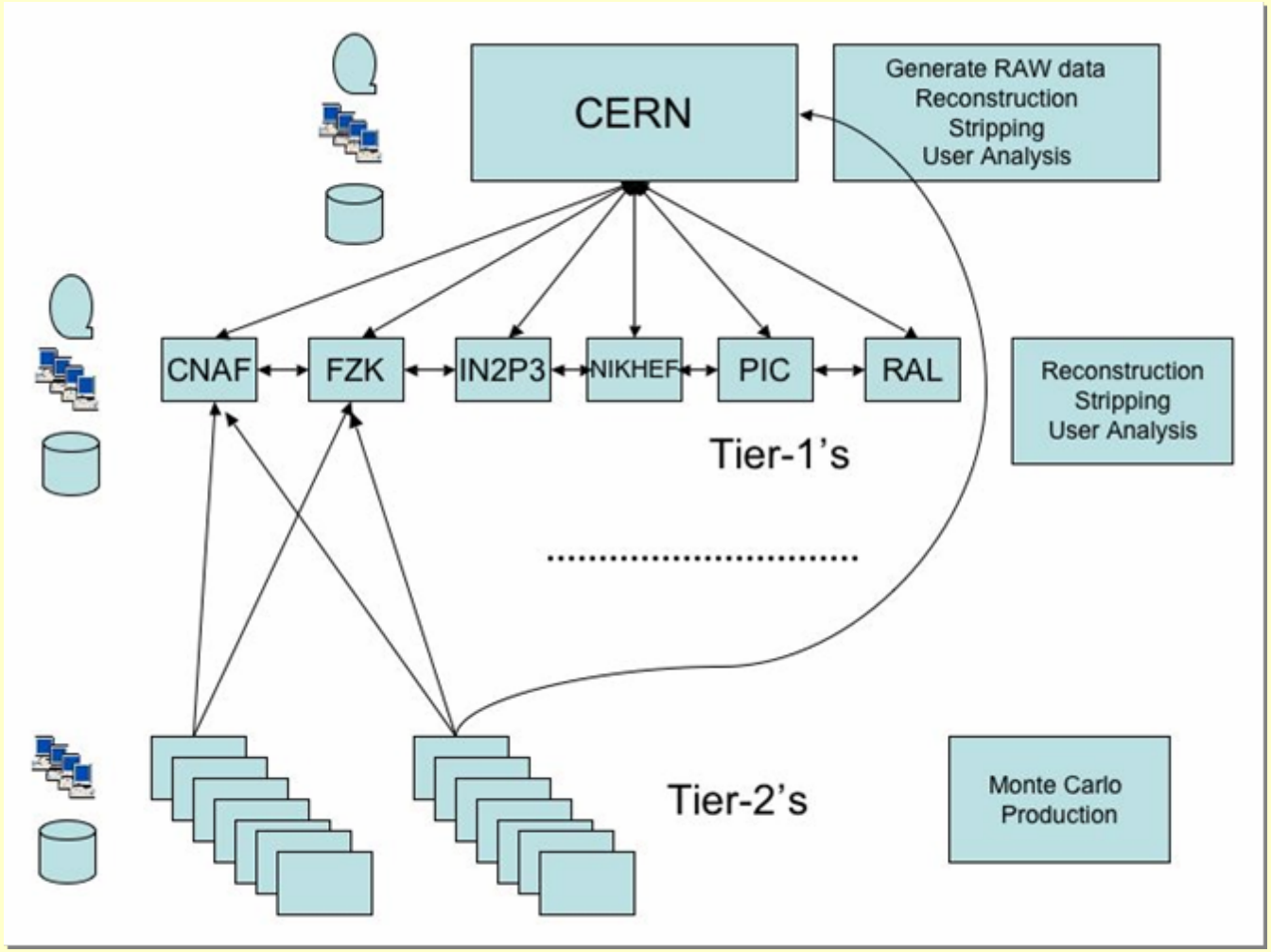
*A.Tsaregorodtsev,*
*CPPM, Marseille*


*HEPiX 2006 , 4 April 2006, Rome*

# Outline

- LHCb Computing Model
- DIRAC production and analysis system
- Pilot agent paradigm
- Application to the user analysis
- Conclusion

# LHCb Computing Model



# DIRAC overview

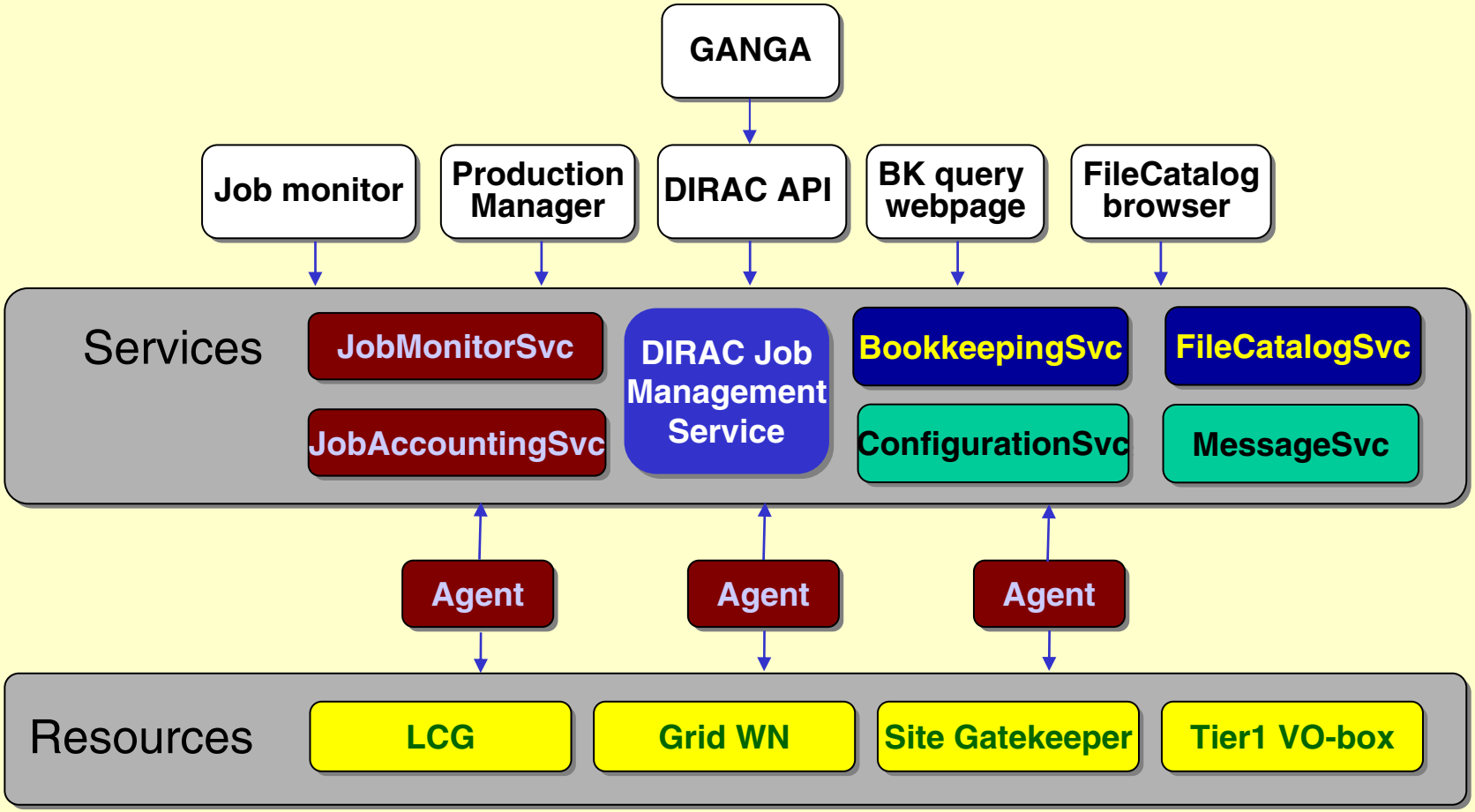
*DIRAC* – *D*istributed *I*nfrastructure with *R*emote *A*gent *C*ontrol

- LHCb grid system for the Monte-Carlo simulation data production and analysis
- Integrates computing resources available at LHCb production sites as well as on the LCG grid
- Composed of a set of light-weight services and a network of distributed agents to deliver workload to computing resources
- Runs autonomously once installed and configured on production sites
- Implemented in Python, using XML-RPC service access protocol

# DIRAC design goals

- Light implementation
  - Must be easy to deploy on various platforms
  - Non-intrusive
    - No root privileges, no dedicated machines on sites
  - Must be easy to configure, maintain and operate
- Using standard components and third party developments as much as possible
- High level of adaptability
  - There will be always resources outside LCG*n* domain
    - Sites that can not afford LCG, desktops, …
  - We have to use them all in a consistent way
- Modular design at each level
  - Adding easily new functionality

# DIRAC Services, Agents and Resources

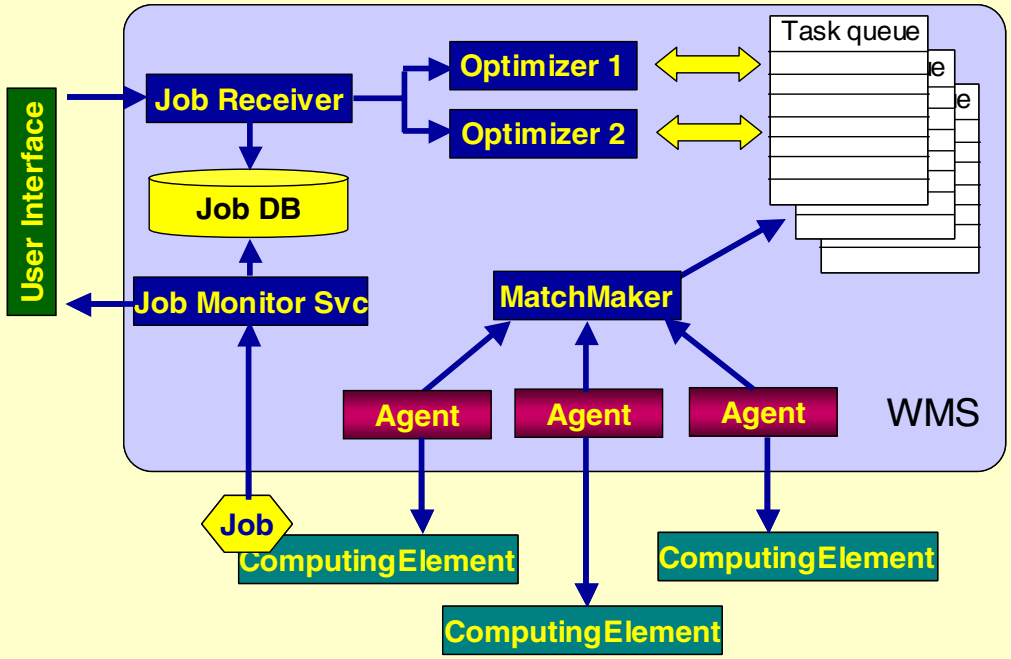


# DIRAC Services

- DIRAC Services are permanent processes deployed centrally or running at the VO-boxes and accepting incoming connections from clients (UI, jobs, agents)
- Reliable and redundant deployment
  - Running with watchdog process for automatic restart on failure or reboot
  - Critical services have mirrors for extra redundancy and load balancing
- Secure service framework:
  - XML-RPC protocol for client/service communication with GSI authentication and fine grained authorization based on user identity, groups and roles

# DIRAC workload management

- Realizes ***PULL*** scheduling paradigm
- Agents are requesting jobs whenever the corresponding resource is free
- Using Condor ClassAd and Matchmaker for finding jobs suitable to the resource profile
- Agents are steering job execution on site
- Jobs are reporting their state and environment to central Job Monitoring service



# WMS Service

- DIRAC Workload Management System is itself composed of a set of central services, pilot agents and job wrappers
- The central Task Queue allows to apply easily the VO policies by prioritization of the user jobs
  - Using the accounting information and user identities, groups and roles (VOMS)
- The job scheduling happens in the last moment
  - With Pilot agents the job goes to a resource for immediate execution
- Sites are not required to manage user shares/priorities
  - Single long queue with guaranteed LHCb site quota will be enough

# DIRAC Agents

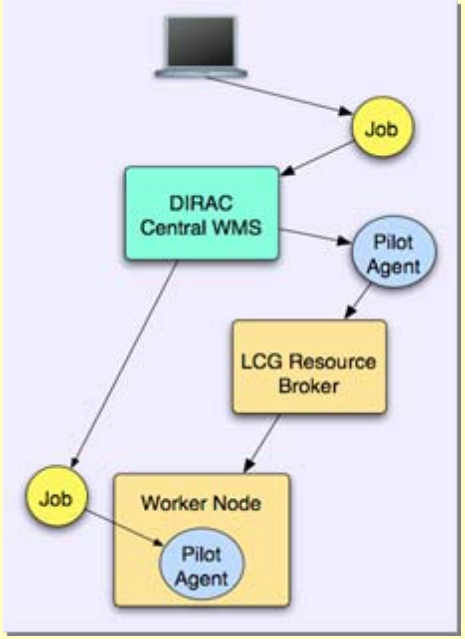
- Light easy to deploy software components running close to a computing resource to accomplish specific tasks
  - Written in Python, need only the interpreter for deployment
  - Modular easily configurable for specific needs
  - Running in user space
  - Using only outbound connections
- Agents based on the same software framework are used in different contexts
  - Agents for centralized operations at CERN
    - E.g. Transfer Agents used in the SC3 Data Transfer phase
    - Production system agents
  - Agents at the LHCb VO-boxes
  - Pilot Agents deployed as LCG jobs

# Pilot agents

- Pilot agents are deployed on the Worker Nodes as regular jobs using the standard LCG scheduling mechanism
  - Form a distributed Workload Management system
- 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 retrieved from the central DIRAC Task Queue and executed
- In the end of execution some operations can be requested to be done asynchronously on the VO-box to accomplish the job

# Distributed Analysis

- The Pilot Agent paradigm was extended recently to the Distributed Analysis activity
- The advantages of this approach for users are:
  - Inefficiencies of the LCG grid are completely hidden from the users
  - Fine optimizations of the job turnaround
    - It also reduces the load on the LCG WMS



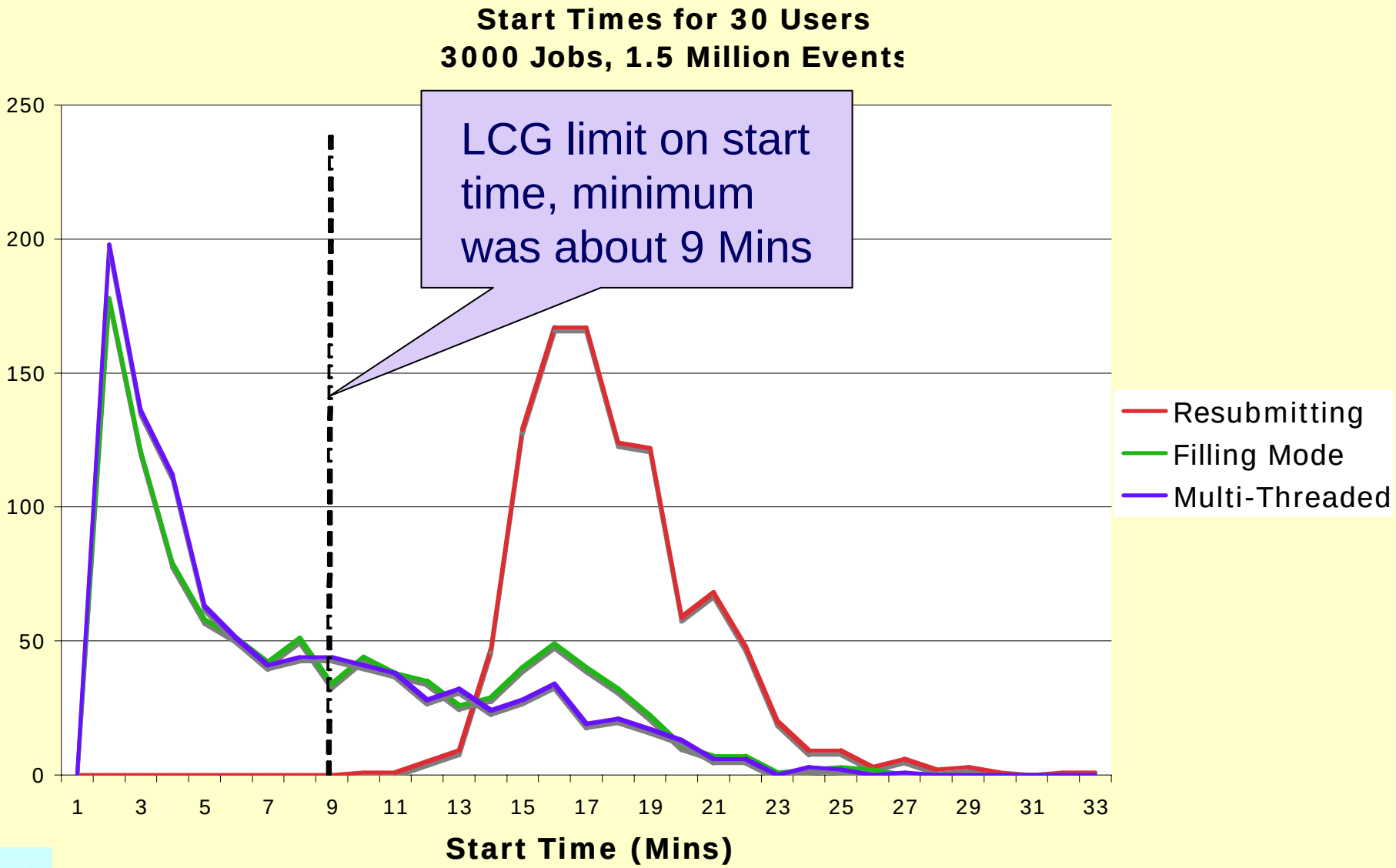

- The system was demonstrated to serve dozens of simultaneous users with about 2Hz submission rate
  - The limitation is mainly in the capacity of LCG RB to schedule this number of jobs

# DIRAC WMS Pilot Agent Strategies

- The combination of pilot agents running right on the WNs with the central Task Queue allows fine optimization of the workload on the VO level
  - The WN reserved by the pilot agent is a first class resource - there is no more uncertainly due to delays in in the local batch queue

- DIRAC Modes of submission
  - 'Resubmission'
    - Pilot Agent submission to LCG with monitoring
    - Multiple Pilot Agents may be sent in case of LCG failures
  - 'Filling Mode'
    - Pilot Agents may request several jobs from the same user, one after the other
  - 'Multi-Threaded'
    - Same as 'Filling' Mode above except two jobs can be run in parallel on the Worker Node

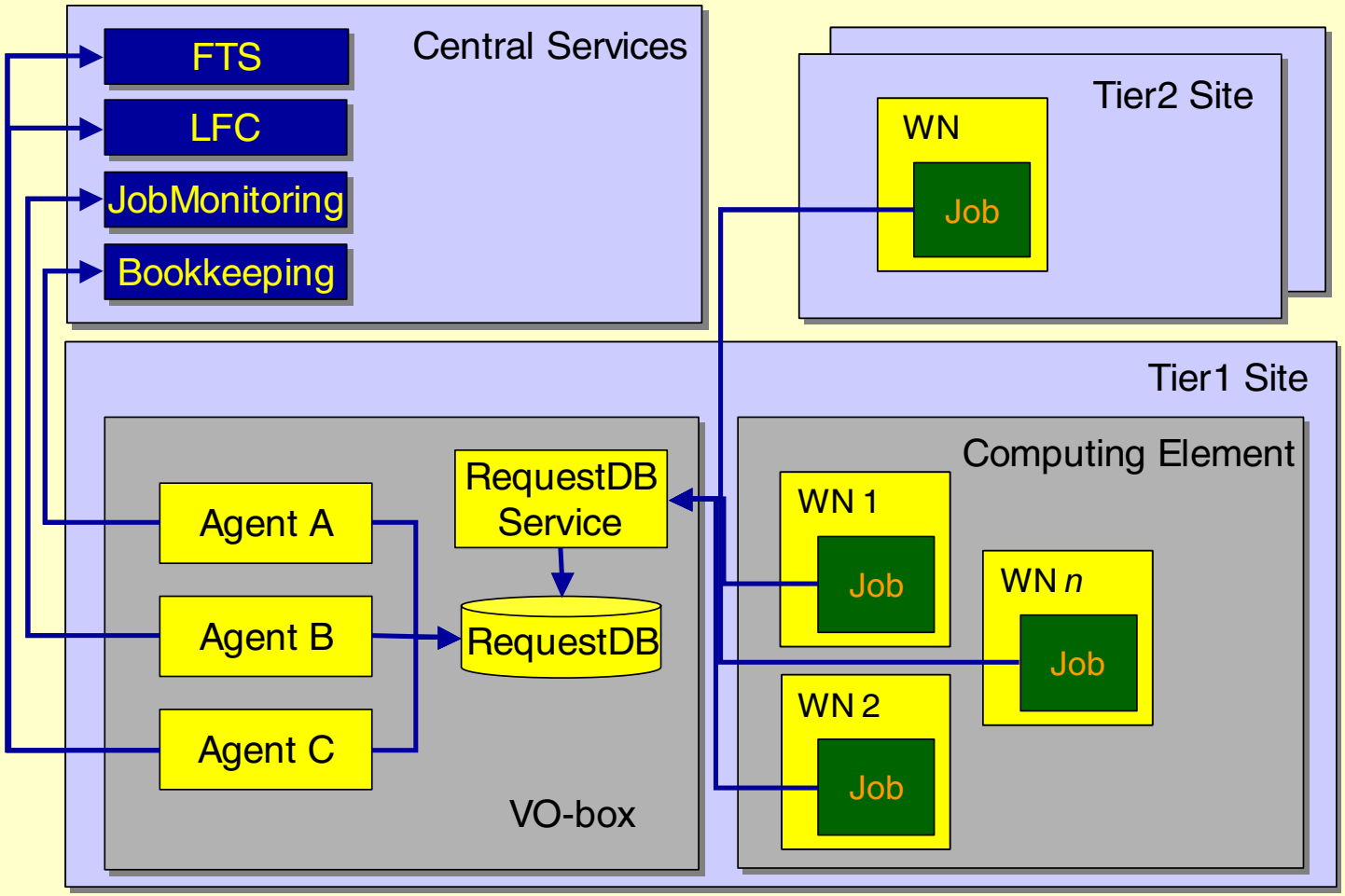
# Start Times for 10 Experiments, 30 Users



# VO-box

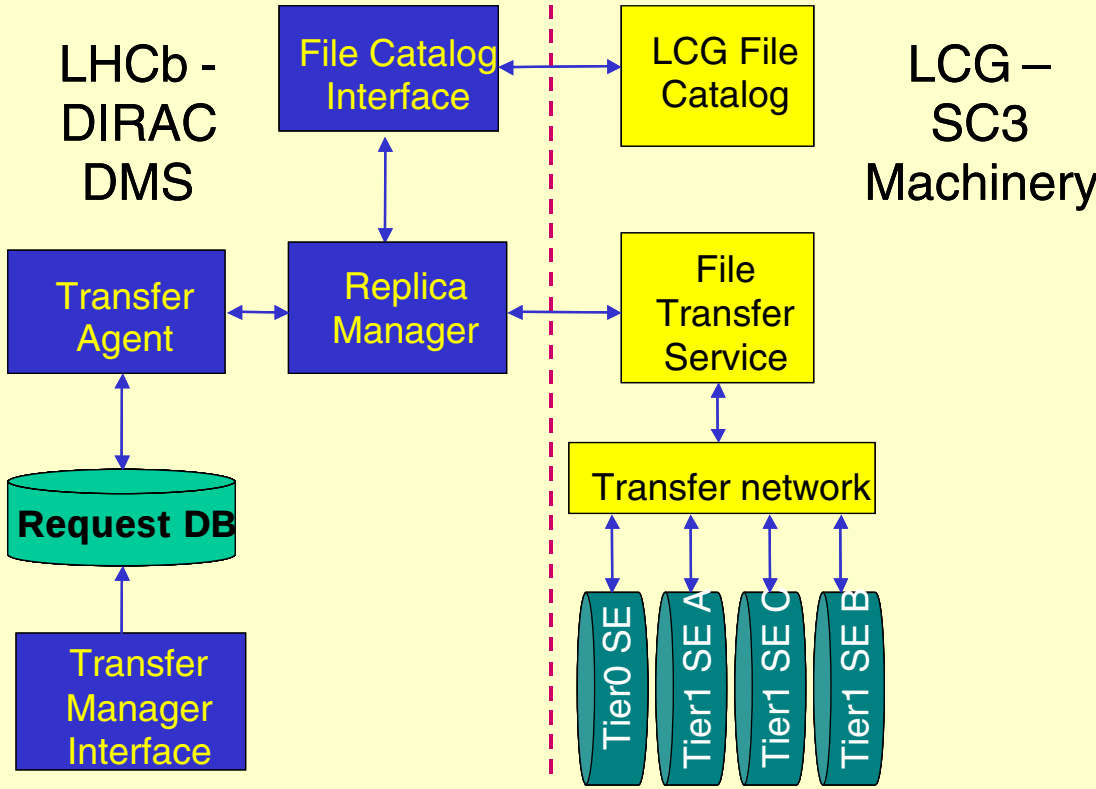
- VO-boxes are dedicated hosts at the Tier1 centers running specific LHCb services for
  - Reliability due to retrying failed operations
  - Efficiency due to early release of WNs and delegating data moving operations from jobs to the VO-box agents
- Agents on VO-boxes execute requests for various operations from local jobs:
  - Data Transfer requests
  - Bookkeeping, Status message requests

# LHCb VO-box architecture

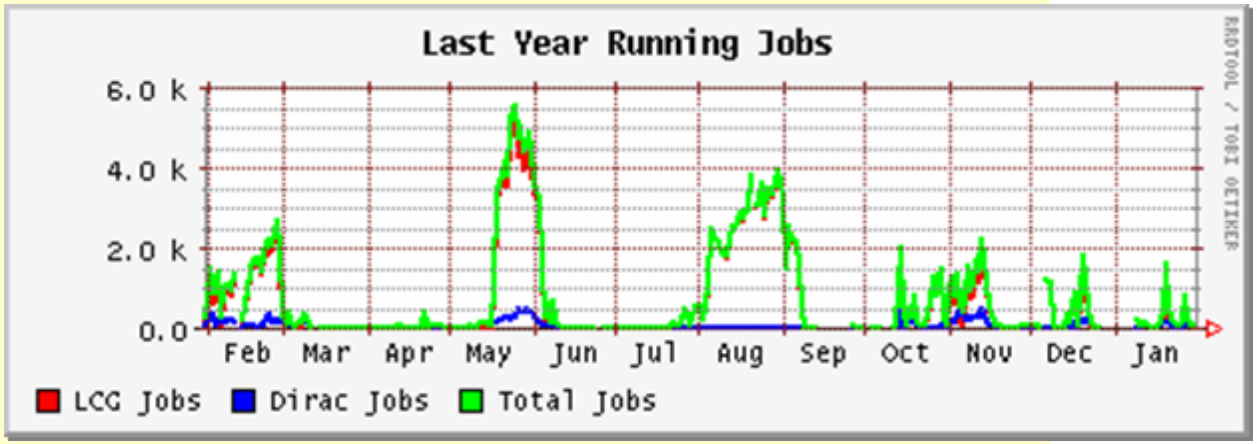


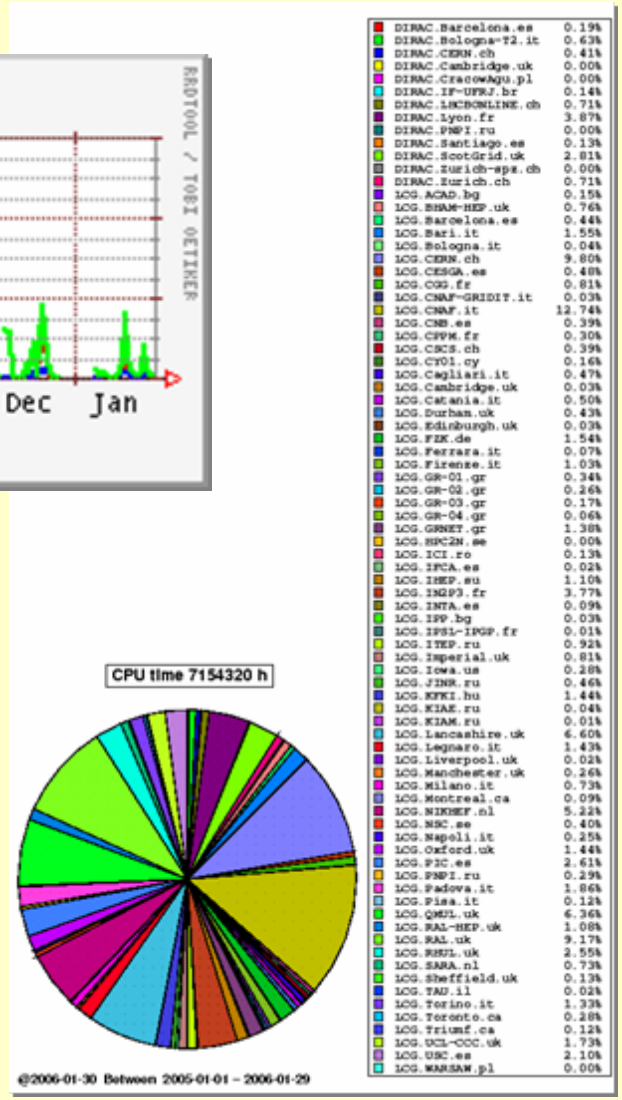
# Transfer Agent example

- Request DB is populated with data transfer/replication requests from Data Manager or jobs
- Transfer Agent
  - checks the validity of request and passes to the FTS service
  - uses third party transfer in case of FTS channel unavailability
  - retries transfers in case of failures
  - registers the new replicas in the catalog



# DIRAC production performance



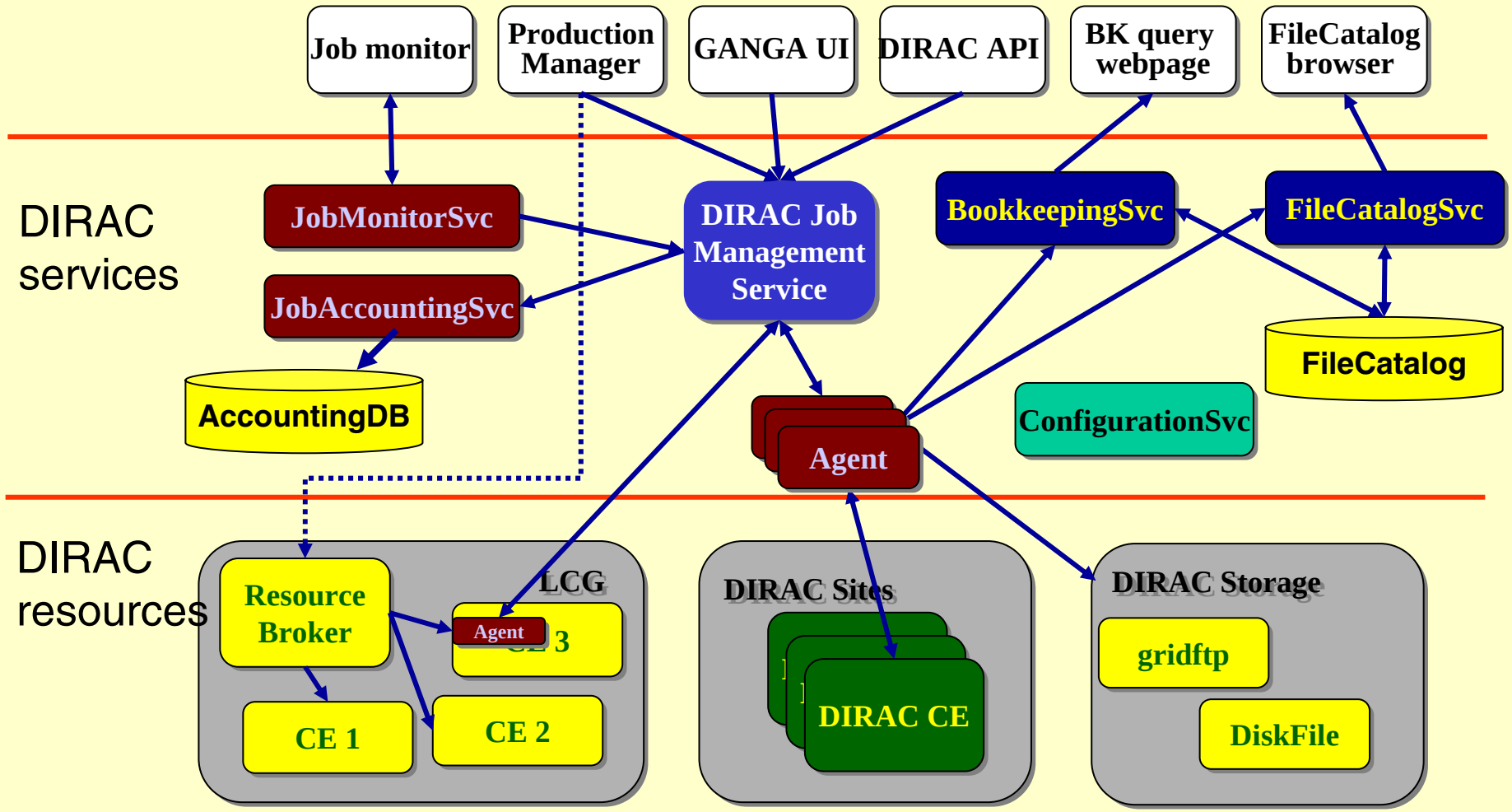

- Up to over 5000 simultaneous production jobs
  - The throughput is only limited by the capacity available on LCG
- ~80 distinct sites accessed through LCG or through DIRAC directly

# Conclusions

- The Overlay Network paradigm employed by the DIRAC system proved to be efficient in integrating heterogeneous resources in a single reliable system for simulation data production
- The system is now extended to deal with the Distributed Analysis tasks
  - Workload management on the user level is effective
  - Real users (~30) are starting to use the system
- The LHCb Data Challenge 2006 in June
  - Test LHCb Computing Model before data taking
  - An ultimate test of the DIRAC system

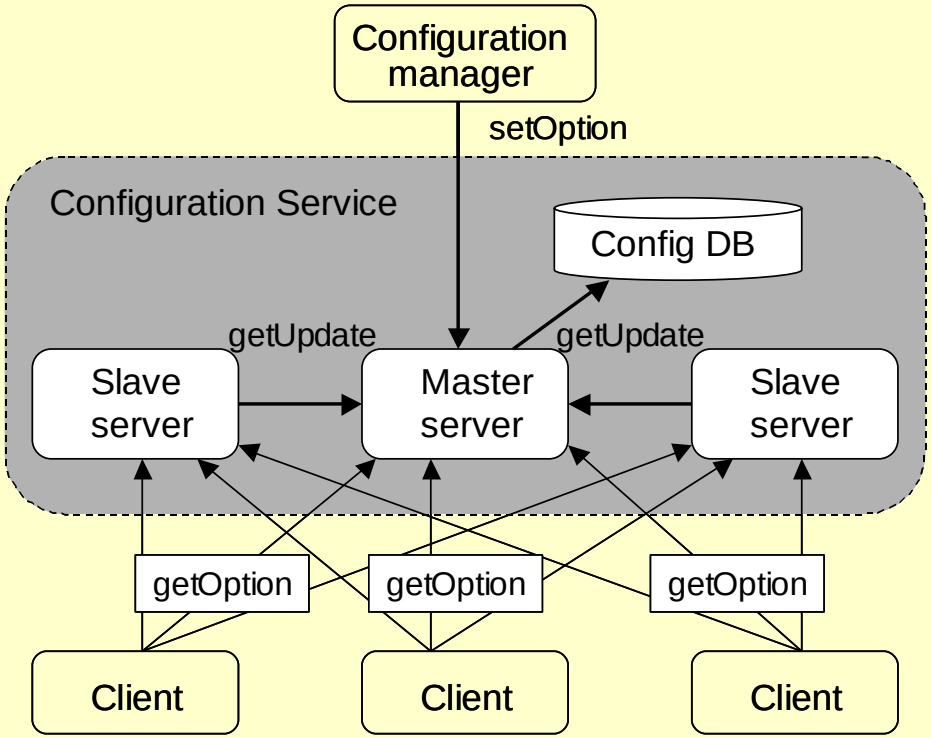
# Back-up slides

# DIRAC Services and Resources



# Configuration service

- Master server at CERN is the only one allowing write access
- Redundant system with multiple read-only slave servers running at sites on VO-boxes for load balancing and high availability
- Automatic slave updates from the master information
- Watchdog to restart the server in case of failures



# Other Services

- Job monitoring service
  - Getting job heartbeats and status reports
  - Service the job status to clients ( users )
    - Web and scripting interfaces
- Bookkeeping service
  - Receiving, storing and serving job provenance information
- Accounting service
  - Receives accounting information for each job
  - Generates reports per time period, specific productions or user groups
  - Provides information for taking policy decisions

# DIRAC

- DIRAC is a distributed data production and analysis system for the LHCb experiment
  - Includes workload and data management components
  - Was developed originally for the MC data production tasks
  - The goal was:
    - integrate all the heterogeneous computing resources available to LHCb
    - Minimize human intervention at LHCb sites
  - The resulting design led to an architecture based on a set of services and a network of light distributed agents

# File Catalog Service

- LFC is the main File Catalog
  - Chosen after trying out several options
  - Good performance after optimization done
  - One global catalog with several read-only mirrors for redundancy and load balancing
- Similar client API as for other DIRAC “File Catalog” services
  - Seamless file registration in several catalogs
  - E.g. Processing DB receiving data to be processed automatically

# DIRAC performance

- Performance in the 2005 RTTC production
  - Over 5000 simultanuous jobs
    - Limited by the available resources
  - Far from the critical load on the DIRAC servers