LHCb view on Baseline Services

A.Tsaregorodtsev,
CPPM, Marseille
Ph.Charpentier
CERN

Baseline Services WG, 4 March 2005, CERN

Outline

- DIRAC in a nutshell
- Design goals
- Architecture and components
- Implementation technologies
- Agent based deployment
- Conclusion

http://dirac.cern.ch

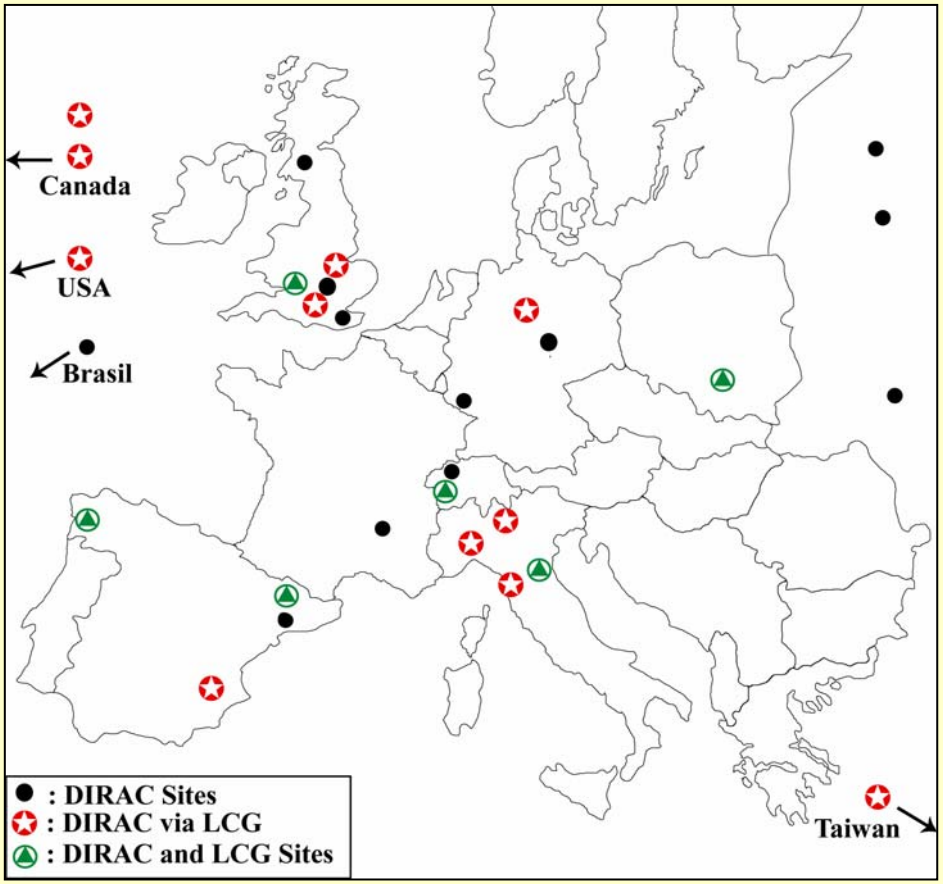
DIRAC in a nutshell

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 scale of resource usage

- Deployed on 20 “DIRAC”, and 40 “LCG” sites
- Effectively saturated LCG and all available computing resources during the 2004 Data Challenge
- Supported up to 4500 simultaneous jobs across 60 sites
- Produced, transferred, and replicated 80 TB of data, plus meta-data
- Consumed over 425 CPU years during last 4 months

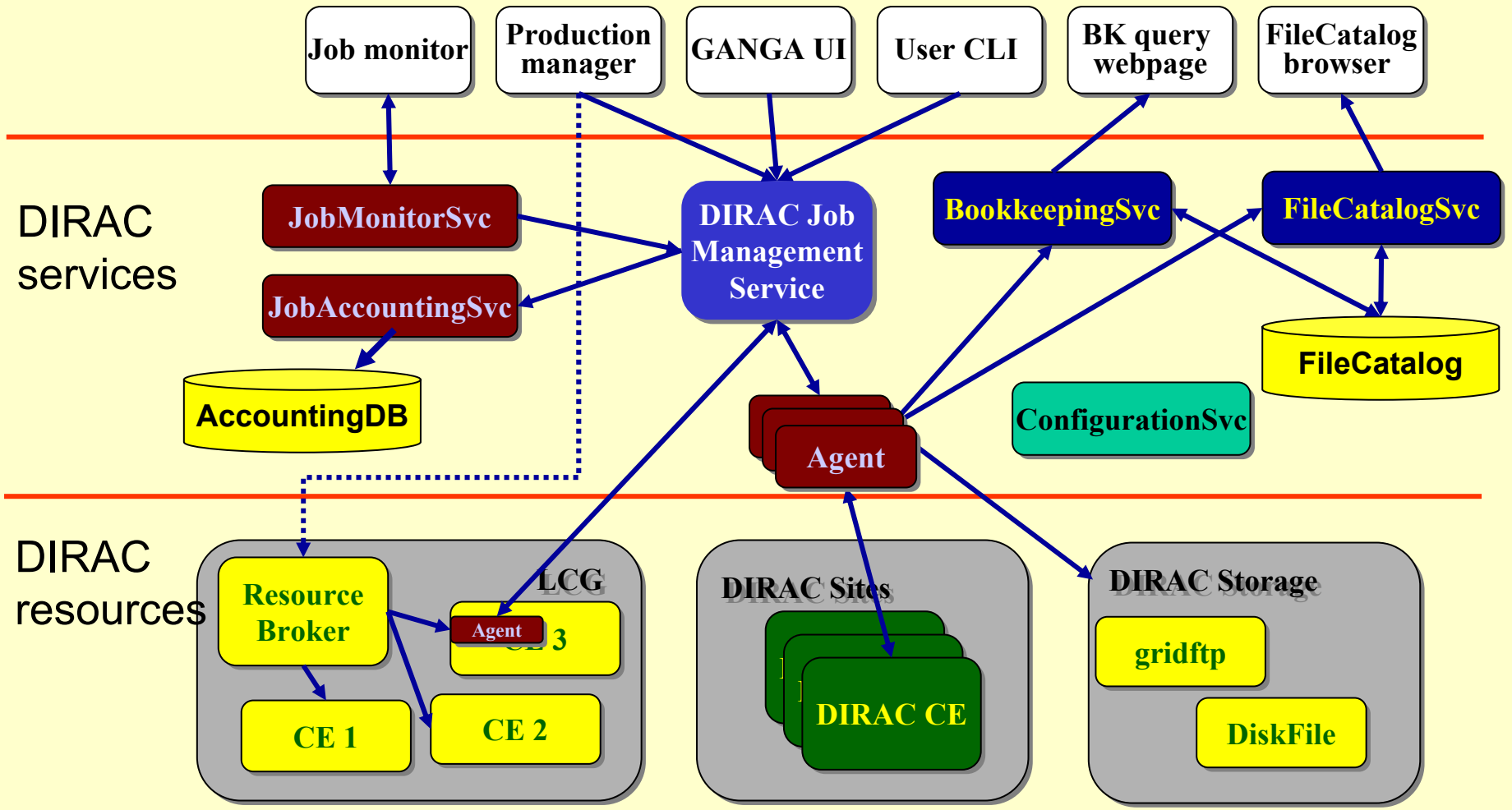


DIRAC design goals

- 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

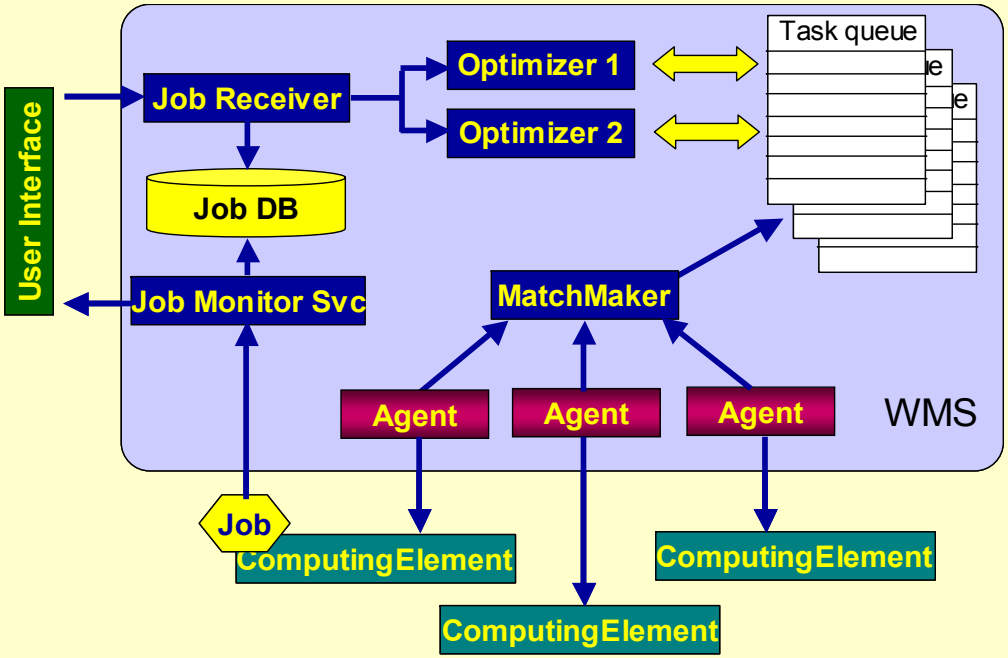
DIRAC architecture

DIRAC Services and Resources

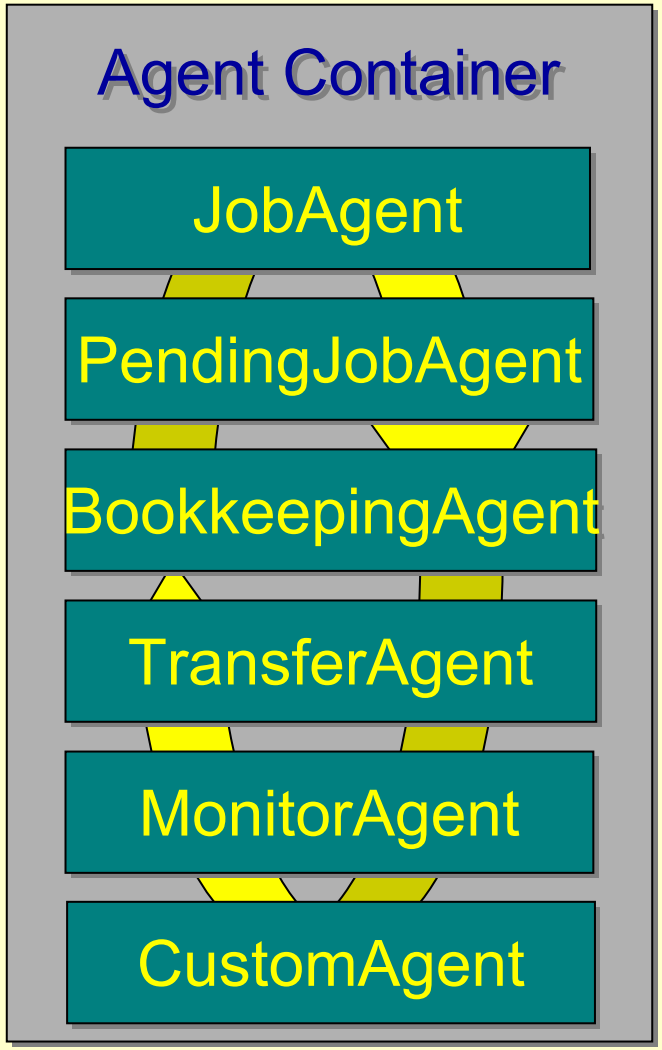


DIRAC workload management

- 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



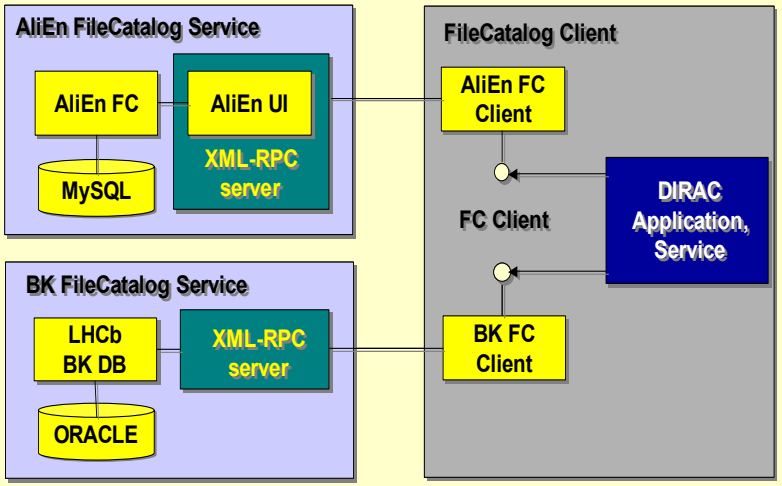
DIRAC: Agent modular design



- Agent is a container of pluggable modules
  - Modules can be added dynamically
- Several agents can run on the same site
  - Equipped with different set of modules as defined in their configuration
- Data management is based on using specialized agents running on the DIRAC sites

File Catalogs

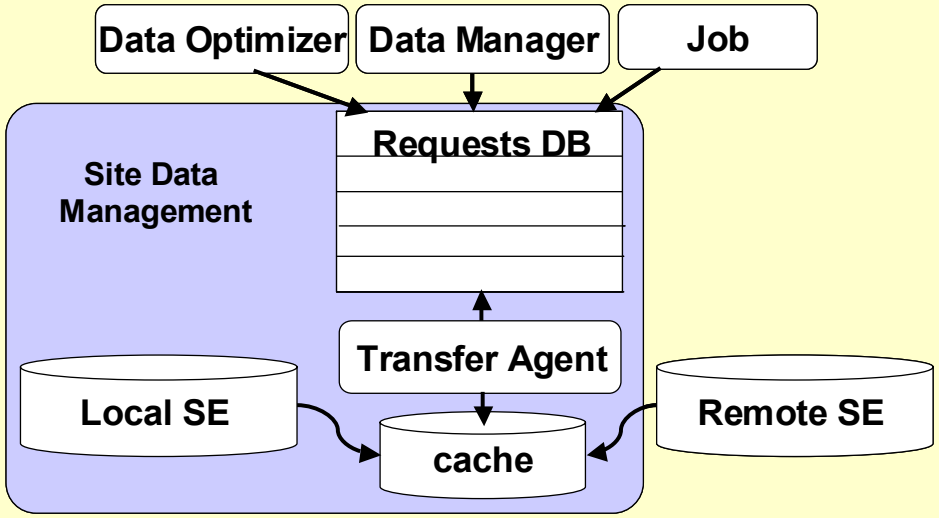
- DIRAC incorporated 2 different File Catalogs
  - Replica tables in the LHCb Bookkeeping Database
  - File Catalog borrowed from the AliEn project



- Both catalogs have identical client API's
  - Can be used interchangeably
  - This was done for redundancy and for gaining experience
- Other catalogs will be interfaced in the same way
  - LFC
  - FiReMan

Data management tools

- DIRAC Storage Element is a combination of a standard server and a description of its access in the Configuration Service
  - Pluggable transport modules: gridftp,bbftp,sftp,ftp,http, …
- DIRAC ReplicaManager interface (API and CLI)
  - get(), put(), replicate(), register(), etc
- Reliable file transfer
  - Request DB keeps outstanding transfer requests
  - A dedicated agent takes file transfer request and retries it until it is successfully completed
  - Using WMS for data transfer monitoring



Other Services

- Configuration Service
  - Provides configuration information for various system components (services,agents,jobs)
- Bookkeeping (Metadata+Provenance) database
  - Stores data provenance information
- Monitoring and Accounting service
  - A set of services to monitor job states and progress and to accumulate statistics of resource usage

Implementation details

XML-RPC protocol

- Standard, simple, available out of the box in the standard Python library
  - Both server and client
  - Using expat XML parser
- Server
  - Very simple socket based
  - Multithreaded
  - Supports up to 40 Hz requests rates
- Client
  - Dynamically built service proxy
- Did not feel a need for something more complex
  - SOAP, WSDL, …

Instant Messaging in DIRAC

- Jabber/XMPP IM
  - asynchronous, buffered, reliable messaging framework
  - connection based
    - Authenticate once
    - “tunnel” back to client – bi-directional connection with just outbound connectivity (no firewall problems, works with NAT)
- Used in DIRAC to
  - send control instructions to components (services, agents, jobs)
    - XML-RPC over Jabber
    - Broadcast or to specific destination
  - monitor the state of components
  - interactivity channel with running jobs

Dynamically deployed agents

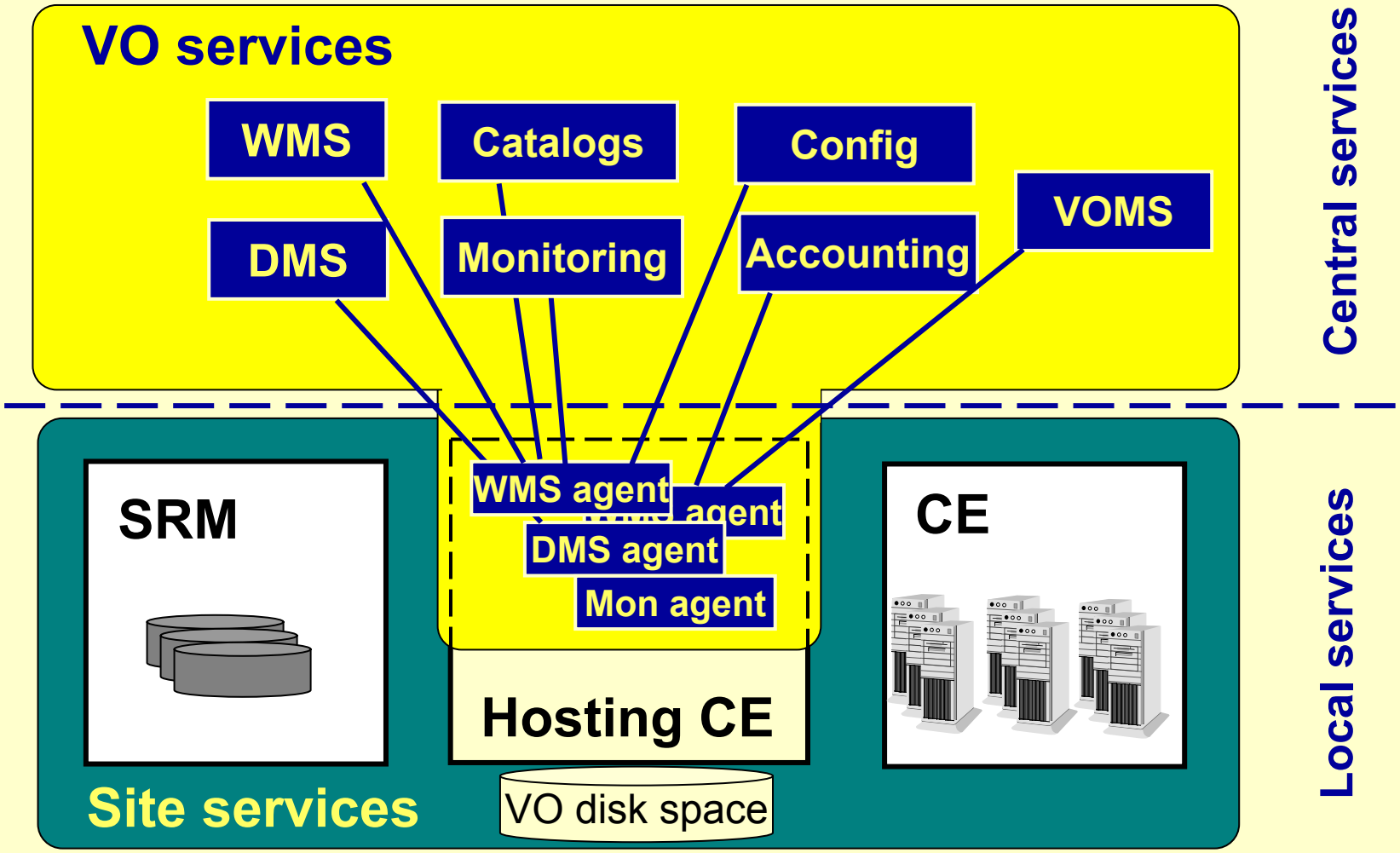
How to involve the resources where the DIRAC agents are not yet installed or can not be installed ?

- Workload management with resource reservation
  - Sending agent as a regular job
  - Turning a WN into a virtual LHCb production site
- This strategy was applied for DC04 production on LCG:
  - Effectively using LCG services to deploy DIRAC infrastructure on the LCG resources
- Efficiency:
  - >90 % success rates for DIRAC jobs on LCG
  - While 60% success rates of LCG jobs
    - No harm for the DIRAC production system
  - One person ran the LHCb DC04 production in LCG

Agent based deployment

- Most of the DIRAC subsystems are a combination of central services and distributed agents
- Mobile agents as easily deployable software components can form the basis of the new paradigm for the grid middleware deployment
  - The application software is deployed by a special kind of jobs / agent
  - The distributed grid middleware components (agents) can be deployed in the same way

Deployment architecture



Hosting CE

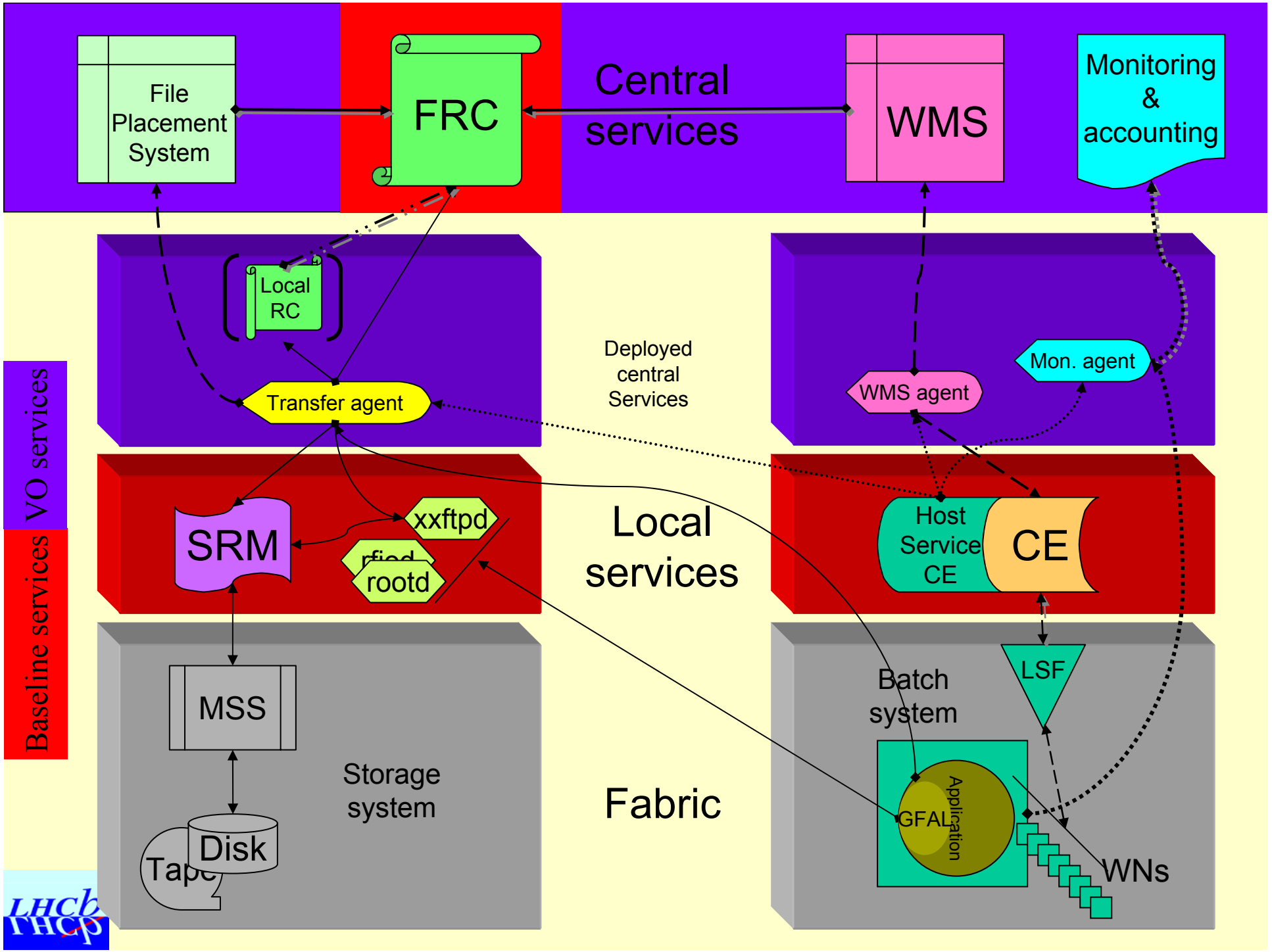
- Standard CE with specific configuration
- Restricted access – for VO admins only
- Out/Inbound connectivity
- Visible both to external WAN and to the Intranet
- Provides VO dedicated disk space, database access:
  - VO software
  - Requests database
- No wall clock time limit for jobs
- No *root* privileges for jobs

Natural distribution of responsibilities

- Site is responsible for providing resources
  - Site administrators should be responsible only for deploying software providing access to resources
    - Computing resources – CE (Condor-C ?)
    - Storage resources – SE (SRM ?)
- Managing the load on the grid is the responsibility of the VO
  - VO administrators should be responsible for deploying VO services components - even those running on sites

Services breakdown

- Inspired from the Dirac experience
- Deployment by RCs of basic services:
  - Fabric
  - Basic CE for WM and SRM for DM (on top of MSS and/or DMP)
  - File access services (rootd / xrootd / rfio…)
- Deployment by VOs of higher level services
  - Using deployment facilities made available on site
- Additional baseline services
  - File catalogue(s): must be robbust and performant
  - Other services: a la carte


File Placement System
FRC
Central services
WMS
Monitoring & accounting
Local RC
Transfer agent
Deployed central Services
WMS agent
Mon. agent
VO services
Baseline services
SRM
xxftpd
rootd
Local services
Host Service CE
CE
MSS
Storage system
Batch system
LSF
Application
GFAL
Disk
Tape
Fabric
WNs
LHCb

Conclusions

- DIRAC provides an efficient MC production system
- Data reprocessing with DIRAC has started
- Using DIRAC for analysis is still to be demonstrated
- DIRAC needs further developments and integration of third party components
- Relations with other middleware developments are still to be defined following thorough assessments
- We believe that DIRAC experience can be useful for the LCG and other grid middleware developments and deployment