# DIRAC – Le système de production et d'analyse de LHCb

*V. Garonne*

# Plan

- DIRAC en 30 secondes
- Les concepts
- L'architecture
- Les choix d'implémentation
- L'interface avec LCG
- Conclusion

http://dirac.cern.ch

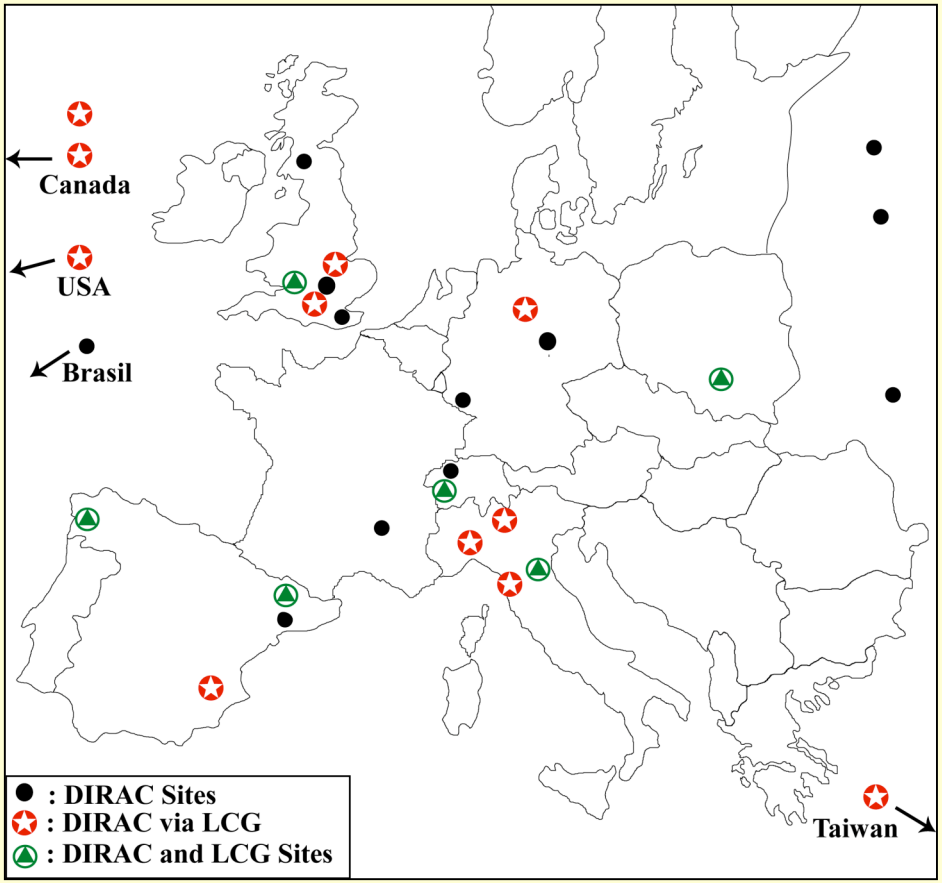
# DIRAC en 30 secondes

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

- Système LHCb de grille pour la production de simulation Monte-Carlo et analyse
- Intègre toutes les ressources de calcul disponibles pour LHCB (site, ressources LCG, simple PC)
- Composer d'un ensemble de service légers et d'agents
- S'exécute de manière autonome une fois installés et configurés sur les sites de production
- Implémenté en Python et utilise XML-RPC comme protocole d'accès au service

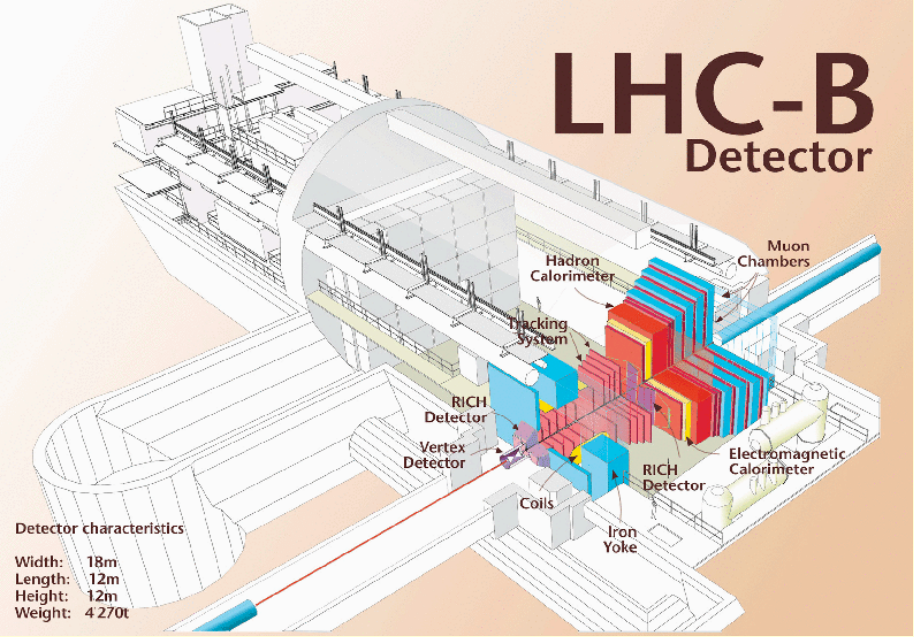
# Plate-forme de Production

- Déployée sur 20 sites conventionnels et sur 40 sites “LCG”
- Toutes les ressources LCG et autres saturés effectivement pendant le Data Challenge 2004
- Le système a supporté 5500 jobs simultanés sur 60 sites
- 80 TO de données produites, transférées, et répliquées
- 425 CPU années consommées pendant 4 mois



# Contexte

- Expérience LHCb
  - Détecteur de physique des particules au CERN
  - Générera des données à un taux de 40 MO/s à partir de 2007
    - soit 3.4 TO/jour
  - 500 physiciens
  - 100 instituts/sites
  - simulations déjà en exécution




L'ordre de grandeur

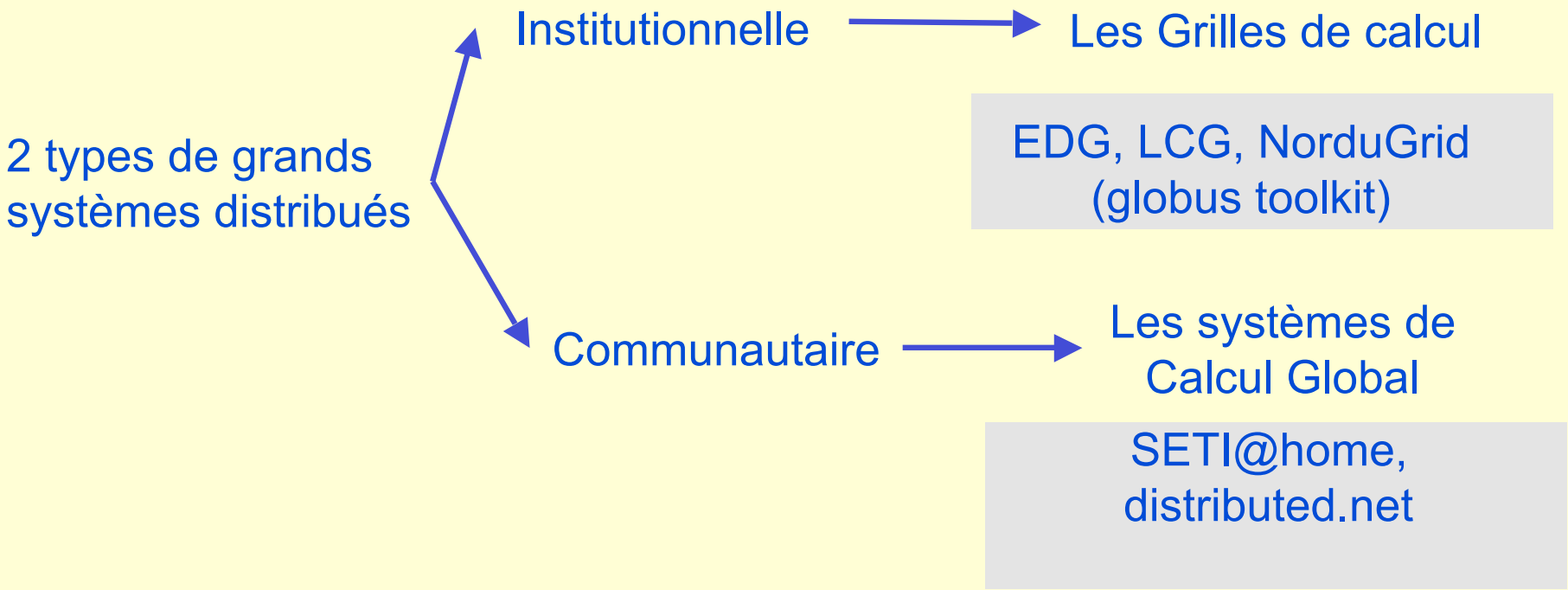
100,000 tâches en attente

30,000 tâches en exécution

100 sites

# Les solutions

- Classification pragmatique des solutions:



# Comparaison

| Grid computing | Global Computing |
|---|---|
| Sites de calculs, clusters | Simple PC |
| Stable | Volatilité |
| Générique | « ad hoc » application |
| Sécurité | Aucune sécurité |
| Gros grain | Petit grain |
| Toutes V.O | Une seule organisation |
| Technologie client-serveur | Technologie d'Internet |
| Intrusif | Léger |
| Goulots d'étranglements | Grande scalabilité |
| PUSH | PULL |

# Les concepts dans DIRAC

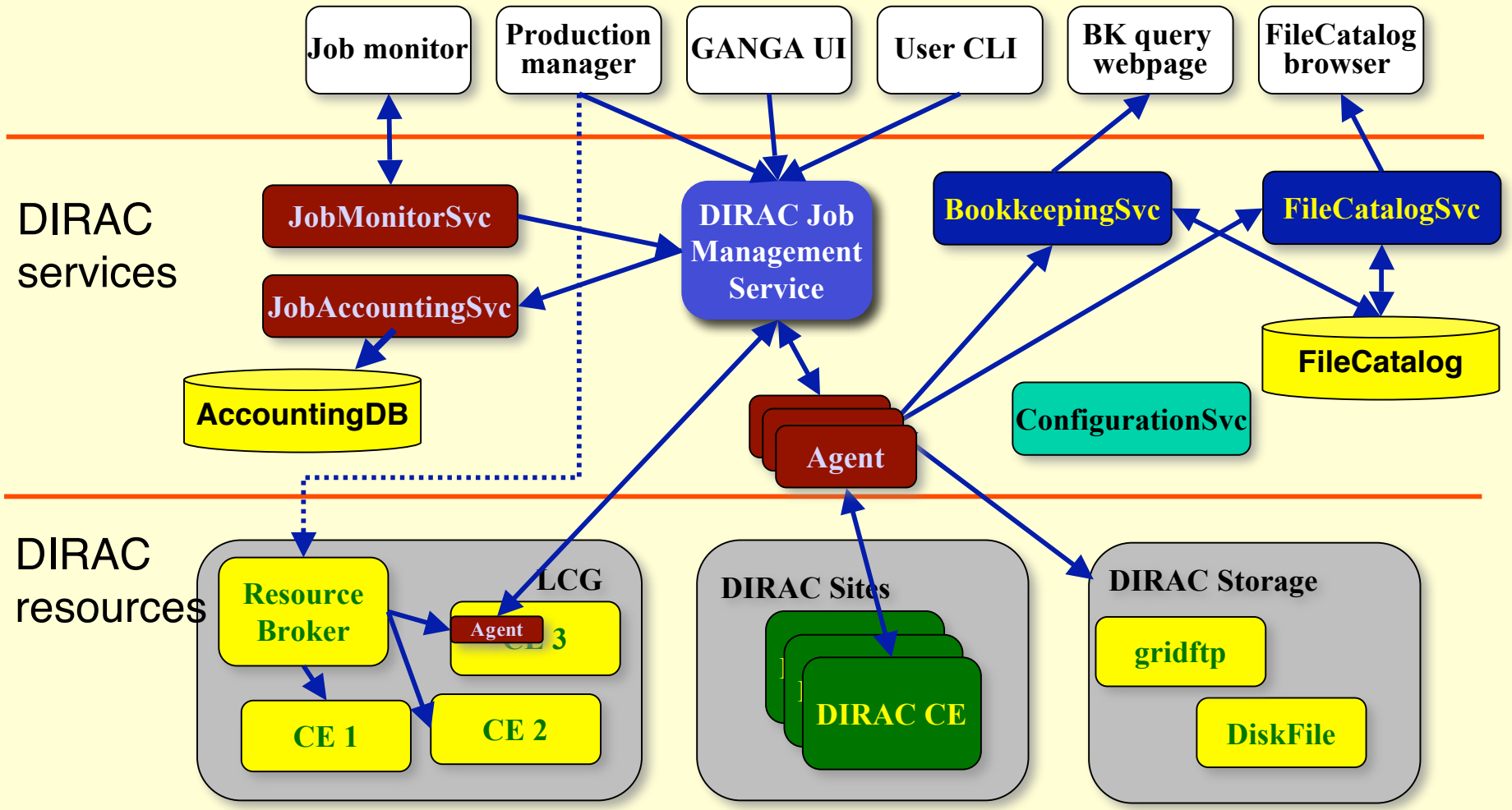
- Un système qui fusionne les deux approches
- Solution « COMMUNAUTAIRE »
- Souple, extensible, robuste et léger
  - Doit être facile a déployer sur différentes plate-forme
  - Non-intrusif
    - Pas besoin de privilèges root, pas de machine dédiées sur les sites participants
    - Doit être facile a configurer, maintenir et utiliser
- Favoriser la réutilisation de l'existant ou de tiers composants
- Haut niveau d'adaptabilité
  - Il y`a toujours des ressources disponibles en dehors du domaine LCGn
    - Sites exclus de LCG, PC, …
  - Nous devons consommer toutes ressources disponibles
- Conception modulaire  chaque niveau
  - Facilité d'ajout de nouvelle fonctionnalité

# DIRAC architecture

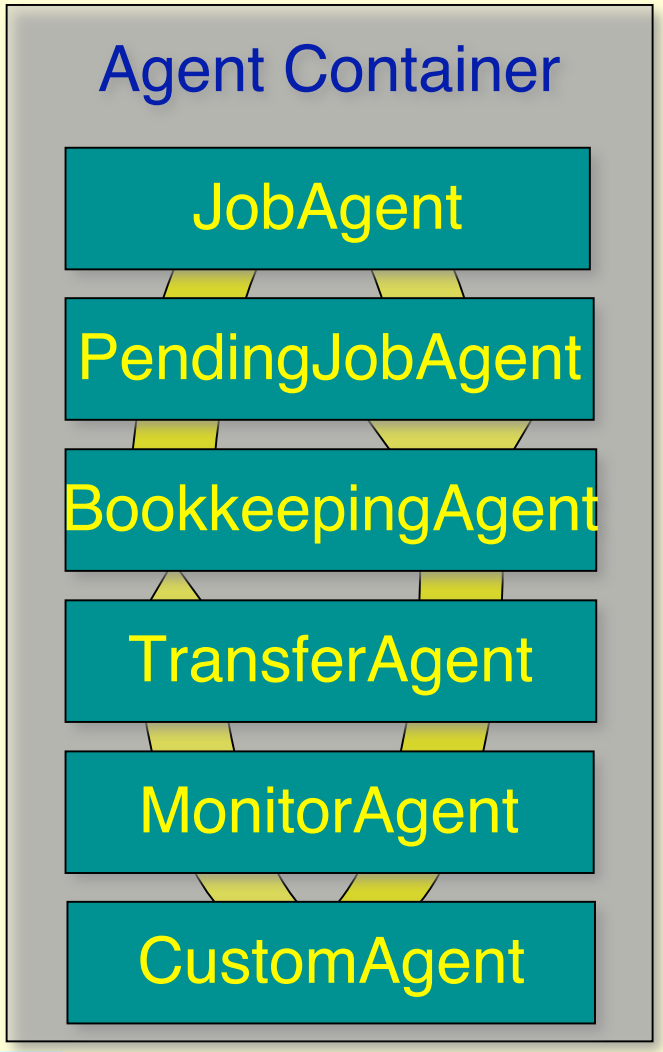
# DIRAC architecture

- Architecture Orientée Service (SOA)
  - Inspire par le concept de “grid services” OGSA/OGSI
  - Suivant l’architecture LCG/ARDA RTAG
- Inspiration ARDA
  - Architecture ouverte avec des interfaces bien définis
  - Permettant de remplacer les services
  - Fournir du choix et de la compétition

# DIRAC Services and Resources



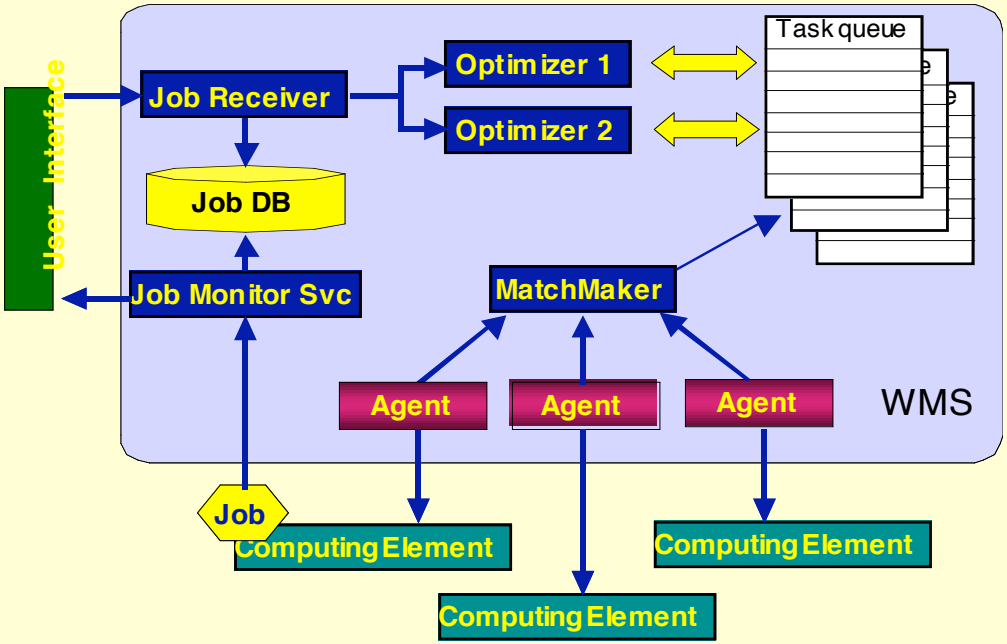
# DIRAC: Agent modular design




- Agent est un conteneur de modules “pluggable”
  - Pouvant être ajoutées dynamiquement
- Plusieurs agent peuvent tournés sur un même site
  - Équipés avec un ensemble de modules différents prédéfinis dans leurs configuration
- La gestion des données est basée sur des agents spécialisés s’exécutant sur les sites DIRAC

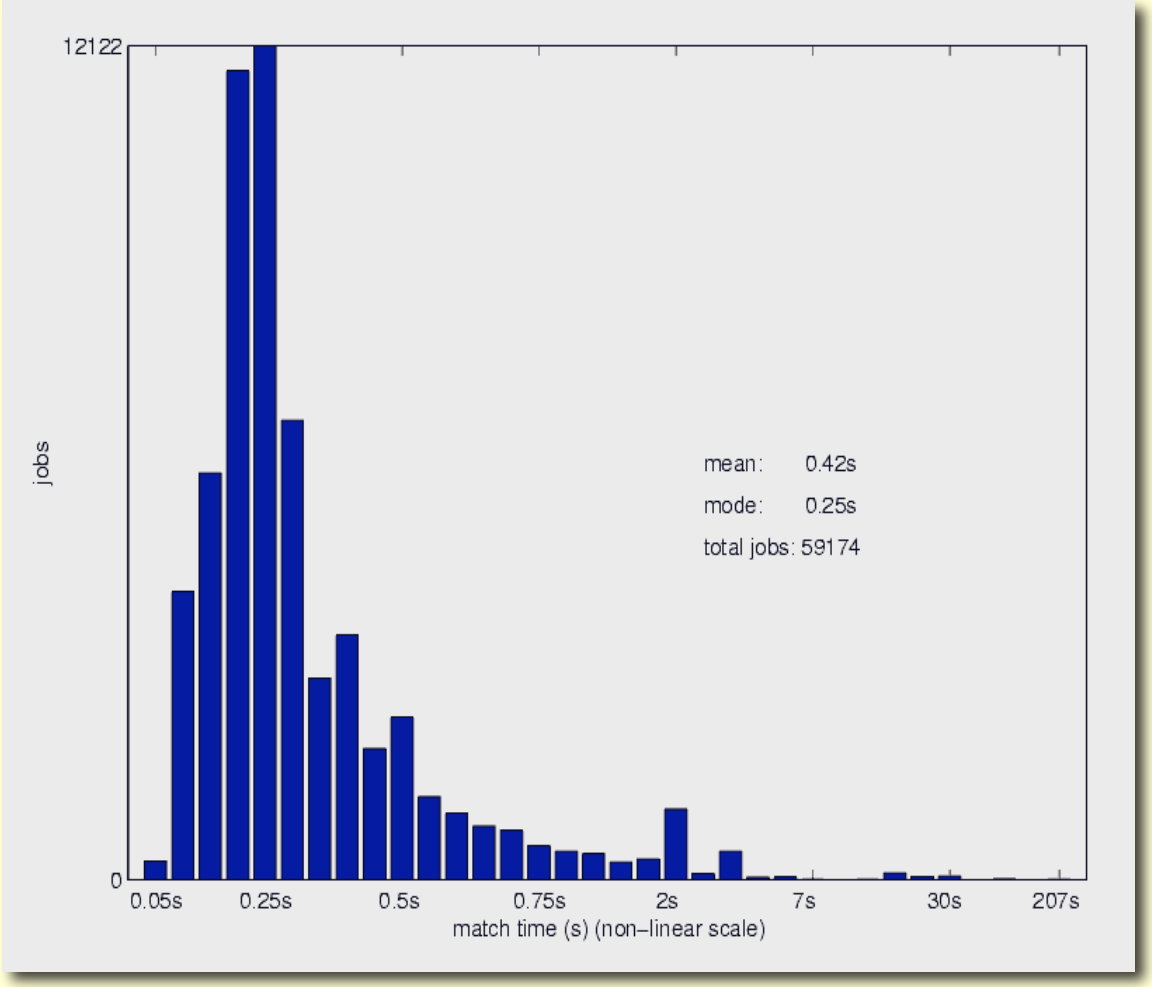
# DIRAC workload management

- Applique un paradigme d'ordonnancement "***PULL***"
- Les Agents demande des jobs quand la ressource correspondantes est disponible
- Utilise le langage Condor ClassAd et le "Match maker" pour trouver les jobs adaptés aux ressources
- Les Agents contrôlent l'exécution des jobs sur le site
- Les Jobs reportent les états et environnement a un service central de monitorage



# Matching efficacité

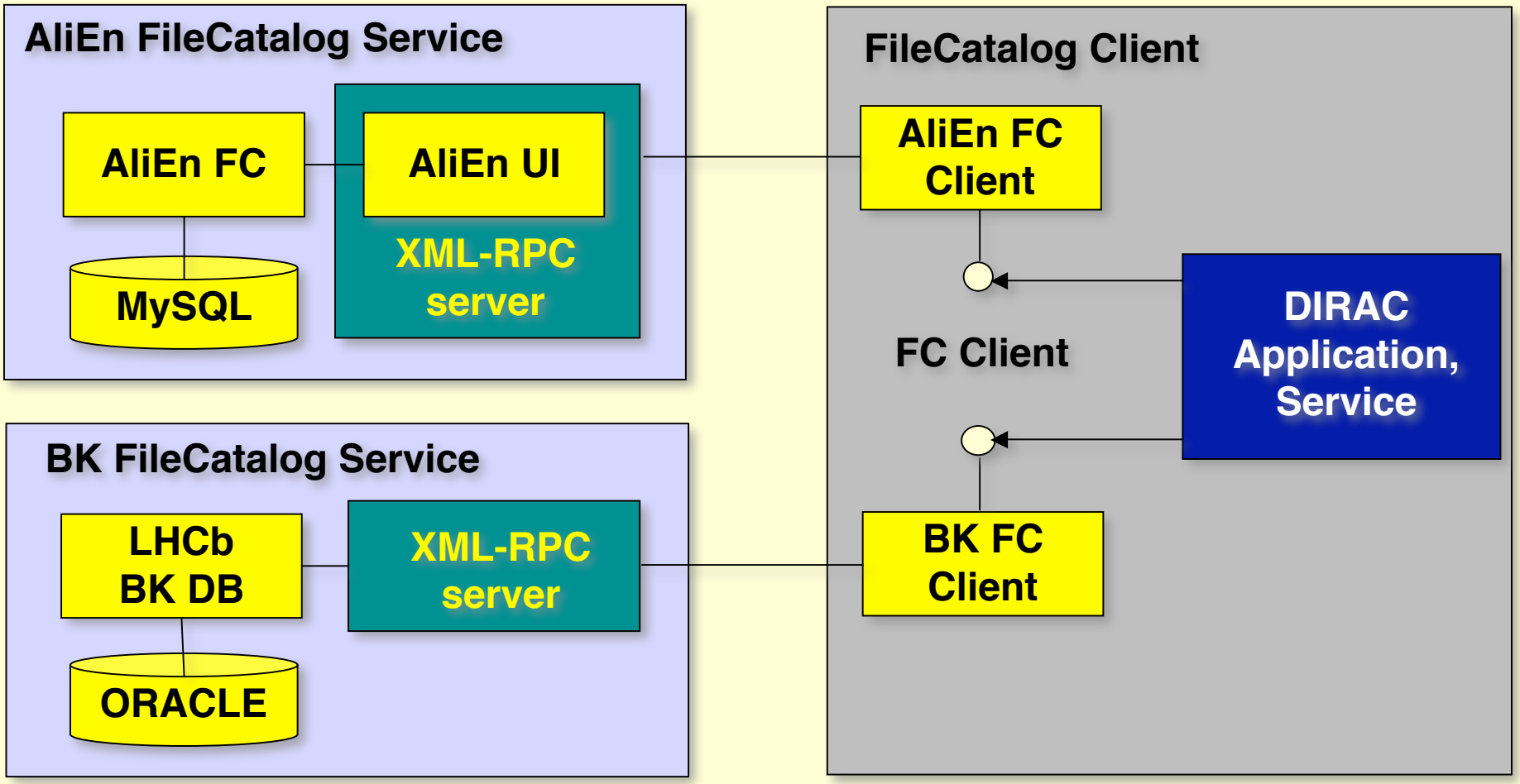
- En moyenne 420ms pour 60,000 jobs
  - Utilise Condor ClassAds et Match maker
- Les jobs en attentes sont groupés par catégories
- Les correspondances job-ressource sont effectuées par catégories
- Typiquement 1,000 a 20,000 jobs en attentes



# Catalogue de fichier

- Les catalogues de fichiers stockent des informations a propos de la la localisation physique des replicas(data set)
- DIRAC utilise 3 catalogues de fichiers différents
  - LHCb Bookkeeping Database
  - AliEn project
  - LCG
- Toutes les catalogues ont une interface identique
  - Peuvent être interchangeable
  - Permets de la redondances, de tester des composants de de gagner de l'expérience

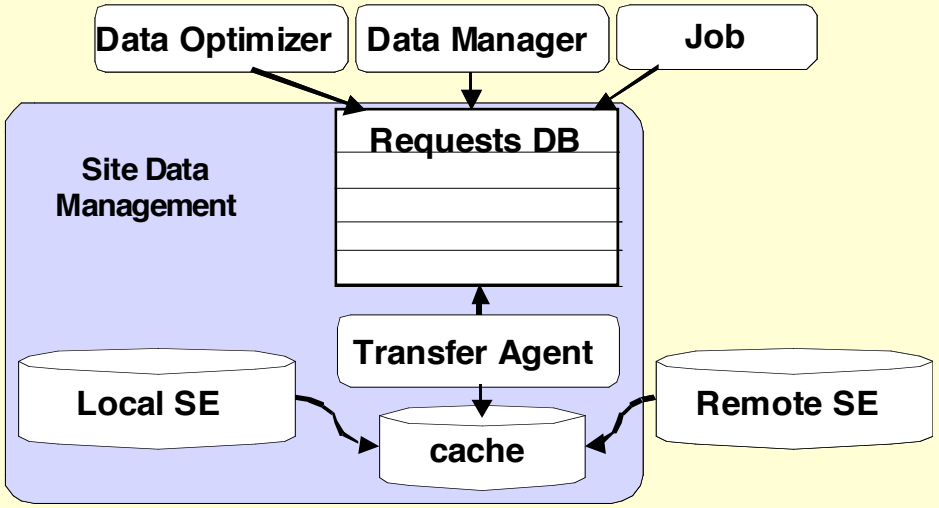
# File catalogs



# Outils de gestion de données

- DIRAC Storage Element est une combination de serveur standard et d'une description de son acces
  - Modules de transport "Pluggable": gridftp,bbftp,sftp,ftp,http, …
- Interface DIRAC ReplicaManager (API and CLI)
  - get(), put(), replicate(), register(), etc

- Transfert de fichier efficace
  - Une DB de requêtes garde les requêtes de transfert
  - Un agent dédié prends les requêtes de transfert et essaye jusqu'au succès
  - Utilise le WMS pour monitorer le transfert des données



# Autres Services

- Service Configuration
  - Fournit les informations de configurations pour divers composants du système (service,agents,jobs)
- Bookkeeping (Metadata+Provenance) database
  - Stockent les informations de l'origines des données
- Service Monitoring et Accounting
  - Un ensemble de services qui monitorent l'état des job, leurs évolutions, l'état des services et accumulent des statistiques sur l'utilisation des ressources

# Choix d'implémentation technologiques

# Protocole XML-RPC

- Standard, simple, disponible dans la distribution standard python
  - Pour le serveur et le client
  - Utilise le parser XML expat
- Serveur
  - Base sur les sockets
  - Multithreaded
  - Support un taux de requêtes de 40 Hz
- Client
  - Construit dynamiquement un proxy d'accès au service
- Pour l'instant pas besoin de plus complexe
  - SOAP, WSDL, …

# Instant Messaging sur la grille

- Plusieurs acteurs: Agents, Clients et Services
- Agents et Clients changent de locations
- Accès limité sur les réseaux
- Besoin d'une communication duplex
- Idée: Utilise la messagerie instantanée asynchrone, bufférisé et sure – Jabber/XMPP IM

# Instant Messaging (2)

- *"Chat Rooms"* fournit un mécanisme de broadcast ad hoc, et une liste dynamique des jobs "actifs", des services, des agents et clients.
- *Information/Query* mécanisme pouvant être utilisé pour avoir une interface RPC
- *Presence* utilisable pour surveiller le statut de composant
- *Connection*:
  - "tunnel" avec NAT et/ou derrière un firewall
  - Une seule authentification
- Humains peut interagir avec des composants en utilisant des clients standard

# Recouvrement sur pannes

- Si une panne survient:
- Back-up réguliers des databases
- Journal de toutes les opérations
- Exécuter plus d'une seul instance d'un même service
  - Configuration service tourne au CERN et a Oxford
- Fiabilité des services s'exécutant:
  - Runit

# Runit service management tool

*http://smarden.org/runit/*

- Runit packag
  - Re- démarrage Automatique sur panne
  - Créations des logins avec estampilles
  - Gère la taille des logs
  - Interface Simple interface pour contrôler les services
  - Exécute les services en mode daemon
  - S'utilise au niveau simple utilisateur
- Gestion de service léger

# Interfacing to LCG

# Dynamically deployed agents

*Comment impliquer des ressources ou les agents DIRAC ne sont pas encore installés ou ne peuvent pas être installés ?*

- Workload management avec réservation de ressource
  - Envoie un agent comme un job “classique”
  - Une fois sur le nœud, il le transforme en site de production virtuel
  - Déploie le soft à la voléé(pacman+cmt)
  - Cette stratégies a été utiliser pour la production avec LCG
  - Utilisation des services LCG pour déployer l’infrastructure DIRAC
- Efficacite:
  - >90 % de succès pour les jobs DIRAC
  - Contre 60% de succès avec les jobs LCG
  - Une seule personne exécute la production LHCb DC04 sur LCG

# Conclusions

- Une architecture orientée service est essentielle pour construire une système “ad hoc” répondant au besoins d’une organisation
- Le concept d’agent léger et facile à personnaliser se sont révèlés être très utiles
- DIRAC a saturé et utilisé toutes les ressources disponibles jusqu’a maintenant
  - Les limites ne sont pas encore atteintes par fautes de ressources !!!

# Authors

- DIRAC development team

  TSAREGORODTSEV Andrei, GARONNE Vincent, STOKES-REES Ian, GRACIANI-DIAZ Ricardo, SANCHEZ-GARCIA Manuel, CLOSIER Joel, FRANK Markus , KUZNETSOV Gennady, CHARPENTIER Philippe

- Production site managers

  BLOUW Johan , BROOK Nicholas, EGEDE Ulrik, GANDELMAN Miriam , KOROLKO Ivan , PATRICK Glen , PICKFORD Andrew , ROMANOVSKI Vladimir , SABORIDO-SILVA Juan , SOROKO Alexander , TOBIN Mark , VAGNONI Vincenzo, WITEK Mariusz , BERNET Roland