# Analyse Interactive au LAL
# PROOF + DPM

Michel Jouvin
LAL, Orsay
jouvin@lal.in2p3.fr
http://grif.fr

LCG France, Marseille
25 Juin2009

# Agenda

- Les besoins
- L’analyse Atlas
- PROOF
- DPM/Xrootd
- Configuration au LAL et résultats obtenus
- Futur
- Conclusions

# Credits

- Contenu de cette présentation basée sur un travail mené avec le groupe ATLAS/LAL depuis 3 mois
- Ilija Vukotic (ATLAS/LAL) a apporté une aide précieuse pour comprendre les problèmes et proposer les tests
  - Ilija impliqué dans le groupe d'optimisation des structure de fichiers
  - Des résultats spectaculaires (10x) obtenus précédemment dans les T1s

# Les Besoins

- Le LAL (GRIF) veut offrir des ressources T3 à ses utilisateurs pour l'analyse finale
  - Analyse finale = analyse interactive, inadaptée à la grille/batch
  - Lecture (partielle) de beaucoup de fichier + peu de calcul
    - Souvent tracé d'histrogramme
  - Besoin d'une exécution « rapide »
    - Quelques minutes
- Calcul : 1 machine ou 1 cluster interactif
  - Emergence de PROOF comme solution « cluster »
- Les données : éviter la duplication de ce qui est déjà dans la grille
  - Potentiellement volumineux : peu efficace de transférer en permanence
  - Des file systems locaux ou NFS ont une faible scalabilité

# Cas de l'Analyse Atlas

- ROOT-based : plusieurs fichiers de 1+ GB
- Mars : utilisateurs rapportent les mauvaises performances ROOT avec fichier sur la grille
  - 5 MB/s… mais 100% CPU (RFIO ou Xroot)
  - Disk server : 10 Gb/s, UI: 1 Gb/s
- Après analyse, CPU-bounded : décompression des fichiers ROOT
  - ROOT ne peut utiliser qu'un seul coeur
  - Même performance si ROOT installée sur le disk server
  - 110 MB/s sur l'UI si rfcp du fichier
  - CPU consommé par la décompression du fichier
    - Aggravé par l'organisation non optimale de certains fichiers
- Unique solution : PROOF
  - Utilisation de plusieurs cœurs en //
  - Implique l'utilisation de Xroot pour l'accès aux données distantes

# PROOF

- Parallel ROOT
  - Action sur les fichiers d'entrée découpée en unité indépendante exécutée en // sur plusieurs cœurs
  - 1 cœur = 1 worker
    - Pas forcément tous sur la même machine
  - Fait partie du ROOT standard : pas de module additionel
- Implémenté comme un plugin Xrootd
  - Xrootd est le cluster manager (redirecteur)
  - Back-end Xrootd est un plugin exécutant une action sur la portion des fichiers affectée au worker
- Relativement transparent pour l'utilisateur ROOT
  - ROOT reste l'interface
  - Quelques contraintes sur l'application ROOT
    - Définition/utilisation d'un Tselector
  - Plusieurs façon d'utiliser : TProof ou TChain
    - TChain : définition structuré de dataset

# Exemple PROOF

- TProof

```
p=TProof::Open("")
proof->SetParameter("PROOF_lookupOpt", "none")
f=new
TFileInfo("root://grid05.lal.in2p3.fr//dpm/lal.in2p3.fr/home/atlas/atlasscratchdisk/user09/IlijaVukotic/D3PD/user09.IlijaVukotic.D3
PD.data10_7TeV.00153565.physics_MinBias.recon.ESD.f251.Data10.V4.003/user10.FrancoisNiedercorn.data10_7TeV.001535
65.physics_MinBias.recon.ESD.f251.Data10.V4.003.D3PD._01368.root")
dset=new TDSet("test")
dset->Add(f)
p->Process(dset,"/users/atlas/vukotic/proof/FrancSel.C")
```

- TChain

```
ch=new TChain("3po")
ch>Add("root://grid05.lal.in2p3.fr//dpm/lal.in2p3.fr/home/atlas/atlasscratchdisk/user09/IlijaVukotic/D3PD/user09.IlijaVukotic.D
3PD.data10_7TeV.00153565.physics_MinBias.recon.ESD.f251.Data10.V4.003/user10.FrancoisNiedercorn.data10_7TeV.00
153565.physics_MinBias.recon.ESD.f251.Data10.V4.003.D3PD._01368.root");
ch->SetProof()
ch->Process("/users/atlas/vukotic/proof/FrancSel.C")
```

# DPM/Xrootd

- Xrootd est un des protoles d'accès supportés par DPM
  - Comme gsiftp, rfio, https
- Utilise Xrootd standard avec un plugin pour le back-end permettant de lire/écrire dans DPM
  - Architecture similaire à PROOF
  - Même solution mise en œuvre pour Hadoop…
- Plugin DPM pour Xrootd : DPM-xrootd
  - Implémente toutes les méthodes Xroot
  - Dépendance importante à la version de Xroot
    - Raison de la difficulté de mise à jour après la version initiale suite à un changement majeur d'architecture Xroot
    - Architecture et API maintenant stables
- Performance en transfert devrait être équivalente
  - Performance des open peut être différentes car très liée au back-end

# Configuration LAL

- Utiliser les machines interactives Atlas/LAL pour construire un cluster PROOF
  - 2 machines 8 cœurs, intégrés à l'environnement LAL
- DPM 1.7.4 + DPM/Xrootd plugin 2.1.4
  - En cours de release
  - Version de Sept. 09 (pas de changement notable depuis)
- ROOT version 5.26/27
  - En avance sur ce qui est utilisé par Atlas/LHCb
  - Amélioration importante des performances I/O
    - TTreeCache
    - http://indico.cern.ch/getFile.py/access?contribId=22&sessionId=1&resId=1&materialId=slides&confId=92416 (R. Brun, DM Jamboree)
- PROOF : 2 configurations
  - PROOF-lite : cluster créé dynamiquement par ROOT avec tous les cœurs de la machine
  - Cluster PROOF avec configuration standard (1 nœud)

# Installation ROOT/PROOF

- Reconstruction de ROOT au LAL depuis les sources
  - Interface RFIO/DPM au lieu de RFIO/Castor
  - Un peu long mais aucune difficulté
  - Facilement installable dans un espace partageable
- Installation PROOF = installation ROOT
- Configuration PROOF
  - Aucune si PROOF-lite mais pas efficace si plusieurs utilisateurs sur la même machine
  - Configuration très simple (1 fichier) si serveur configuré
    - Exemple pour 1 machine

```
### Load the XrdProofdProtocol to serve PROOF sessions
if exec xrootd
xrd.protocol xproofd:1093 libXrdProofd.so
fi
xpd.workdir /scratch/proof/sandbox
```

    - Pour plusieurs machines, ajouter la liste des workers et mettre le même fichier sur toutes les machines

# Résultats Obtenus

- Support ROOT très réactif... et très intéressé par cette config
- Problème de synchronisation avec Ilija : pas pu obtenir les chiffres pour cette présentation
  - Update après le workshop
- Conclusions préliminaires après discussions avec Ilija
  - Bonne scalabilité avec le nombre de cœurs (8)
    - ~100% pour chaque cœur
    - Temps total ~ proportionnel aux nombres de cœur
    - ~40 MB/s par session ROOT
  - Requiert la dernière génération de fichiers ROOT
    - Sinon amélioration limitée liée à la structuration très peu optimale : beaucoup de petits accès random

# Problèmes Rencontrés

- Methode xdr_locate non supportée par plugin DPM/Xrootd
  - Non utilisé par ROOT : permet de valider l'existence et l'accessibilité des fichiers à envoyer aux workers
  - Bug du plugin
- Initialement : fix dans ROOT trunk
  - Desactivation optionelle du locate
  - Impact si fichier inaccessible : message pas très pertinent
- Fix obtenu début juin pour début DPM/Xrootd
  - Implémentation de la méthode manquante
  - Release avec DPM 1.7.4

# Futur

- Mise en place d'un cluster de plusieurs machines
  - Machines interactives des groupes Atlas/LHCb (5)
- Benchmarking avec différents tests utilisateurs
  - Principalement Atlas au LAL mais ALICE IRFU/IPNO
  - Chaque cas semble particulier : très dépendant de la structure du fichier ROOT
- Comparaison performance avec d'autres configuration Xrootd
  - IRFU : Xrootd « standard » + GPFS
  - LAF
- Extension à LHCb ou autres groupes intéressés
  - Machines interactives standard du LAL : tout utilisateur y a potentiellement accès, intégré aux fichiers standards
  - LHCb moins intéressé par PROOF : parallelisme intégré dans Gaudi

# Conclusions

- Analyse ROOT : CPU-bounded
  - Nécessité de paralléliser le traitement : PROOF
- PROOF requiert Xrootd comme méthode d'accès aux données
- Dupliquer les données depuis la grille est couteux (HW) et peu performant
  - Utiliser les données directement depuis la grille
  - DPM/Xrootd permet à Xrootd d'utiliser les fichiers stockés dans DPM sans duplication ni staging
- Bonne performance et scalabilité avec 8 cœurs démontrées par les premiers tests Atlas
- Mise en œuvre PROOF très simple dans un T2
  - Permet de tirer parti des données déjà disponibles
  - Simplifie la gestion de la compétition entre utilisateurs par rapport à LAF nationale
- La performance dépend aussi du caching ROOT...

# Liens Utiles

- Stratégie de caching dans ROOT et évolutions à venir
  - http://indico.cern.ch/getFile.py/access?contribId=22&sessionId=1&resId=1&materialId=slides&confId=92416