Distributed Analysis with AliEn & ROOT

A.J.-Peters et.al. for the ALICE Collaboration
Overview

Introduction
- Analysis with GRID Systems

I  AliEn System as a GRID Framework
- Some Design Features
- Performance Tests

II Batch Analysis with AliEn&ROOT
- Task Splitting in AliEn
- Result Merging

III Interactive Analysis with AliEn + ROOT/PROOF
- SuperPROOF Environment
  - PROOF in a distributed environment
  - Packetizing for SuperPROOF clusters
  - Interactive Job Reservation in AliEn
Introduction
How can we analyse data in the future?

- LHC era experiments will produce data in the order of PB/a

- Conventional Mass Storage System and Batch Queue System of individual computing centers can handle very well subsets of this dataset.

- The scale change requires an equivalent scale change from single computing centers to a GRID of computing centers

- Efficient Analysis of large datasets requires:
  - Indexing and preselection of data using meta data tags
  - highly parallel processing with efficient data distribution
Introduction

A step towards a solution: (Ali) ROOT + AliEn

**ROOT** is becoming most popular available physics analysis toolkit.
- Interactive analysis work in familiar C++ style syntax
- Data visualisation, an object-oriented I/O system
- Crucial role for the LCG project ?!
- Is successfully used within the AliROOT framework of the ALICE experiment as an all in one solution

**PROOF** extends workstation based concept of ROOT to the 'parallel ROOT facility'.
- User procedures are kept identical during an analysis session
- Tasks are distributed automatically in the background

**AliEn** as a GRID analysis platform provides two key elements:
- A global filesystem
  - Files are indexed and tagged in a virtual file catalogue and everywhere globally accessible
- A global queuesystem
  - Global job scheduling according to resource requirements
I assume, you all know ROOT.

What is AliEn?
AliEn System as a GRID Framework

- **AliEn** = ALICE Environment
  - AliEn originally designed as distributed computing and file storage facility for the Alice experiment

- under **development since 2001** by:
  - small team of the Alice Offline Group (2-4 FTE)
  - temporary people like summer students etc.

- **Implementation** of main components
  - PERL - Web Service (using SOAP::Lite)
  - C/C++ - API, File Transfer Layer, Globus/GSI

- **Open Source Components**:
  - SOAPlite, gSOAP, MySQL, OpenSSL, OpenLDAP, Apache ...
    - only a small part of the code had to be developed
Basic Guidelines and Concepts

- **Modularity**
  - Service Plugin Architecture
  - SOAP Communication Bus

- **Scalability**
  - Distributed Databases
  - Distributed Core Services

- **User Identity**
  - Concept of “user” + permissions
  - Concept of user assigned resources

- **Virtual Organisations (VO)**

- **Interoperability**
  - Grid2Grid Interfaces

- **File/Queue System Similarity**
  - Central Configuration/VO configuration

- **Security**
Some Design Features of AliEn
AliEn System as a GRID Framework

-Virtual Grid Components-

* Global Queue System
  * Batch Job System
  * Interactive Job System

* Global File System
  * File Catalogue
    * Browsing
    * Access
    * Manipulation
    * Replication
  * Meta Data Catalogue
AliEn System as a GRID Framework

-Simplified System View-

Global File System
Storage Elements
Computing Elements
Core Services
Databases
AliEn System as a GRID Framework

-Simple Layout-

* How do Components communicate?
AliEn System as a GRID Framework

SOAP Service Communication

Core Services
- User Interface
- API Service
- SOAP API
- Authentication Service
- Logging Service
- Information Service

Applications

SE
- Storage Element
  - Filetransfer Daemon

CE
- Computing Element
  - Cluster Monitor

Process Monitors

SOAP Protocol

User Interface
- Transfer Optimizer
- Job Optimizer
- Authentication Service
- Logging Service
- Information Service

API Service
- Transfer Manager
- Job Manager
- Information Service
How do we provide DB access for various DB engines?

How do we preserve modularity?
AliEn System as a GRID Framework

-Core Services DB Access-

- Core Services talk to Core DBs through DB Proxies via RPC calls

- DB Proxies connect to “any” DB via PERL DBI module

- Core DB information exported to outside world via Core SOAP Services
How can we design a scalable file catalogue?

How can we provide global file access?
AliEn System as a GRID Framework
- File Catalogue Components -

- File Catalogue implemented by “distributed” DBs
- DB Access through DB Proxies
- Dedicated Service for Optimization
AliEn Filesystem

Components:
- Storage Systems
- File Transfer Daemons
- File Catalogue Database
- Transfer Services
- Catalogue is based on a database tree structure
- Folders are linked database tables
- Files are table entries with location pointers
Distributed Databases
File Catalogue sharing between VOs
File Access

Global File Transfer Layer in AliEn:
- Scheduled Transfer
  - bbftp (release)
  - gridFTP (under devel/integration)
  - aiod (under devel/integration)

- Interactive Access
  - aiod
Crosslink-Cache-Architecture

Main Cache Cluster
Regional Cache

Client A

File B

aiod – AliEn I/O Daemon

aiod -Server
Client thread

Preload thread

Monitor/Cache Lookup Service

Local Cache

MSS
AliEn Crosslink-Cache-Architecture

Implemented Features

- Data encryption
- SSL certificate based authentication
- Bandwidth regulation
- Variable pre-load cache (\#/size of pages)
- Load Balancing/Connection Redirection
- Cache-Server
  - Support for different cache levels:
    - Main/global (4)
    - Regional (2)
    - Local (1)
    - No cache (0)
  - Variable cache size
  - GUID file identifications

- Transfer routing
- Cache routing
- Cache route lookup
- CMD interface + API Class for server access

ClC Package contains:
- aiod - Server
- aioget - 'get' client
- aioput - 'put' client
- aiolookup - Cache lookup
- guidtool - GUID creation
- libalienio - Client library
Jobs described in JDL (Condor) syntax

Jobs execute registered commands/installed packages

Job Scheduler with passive “Pull” model:
* CEs ask for jobs in regular intervals

Job Optimizer allows “Pseudo-Push” model:
* JDL modifications allow job assignment to a specific resource or a subset of resources

Process Monitor Service reports job information about running jobs to the AliEn /proc filesystem

CE acts as a queue system wrapper
AliEn Job Description Example
Running a ROOT macro on registered data

Executable = "root";
Packages   = "ROOT::3.10.01";
Arguments  = "Command::ROOT -x macro.C"
InputData  = {"/alice/production/peters/*Tree.root"};
InputFile  = {"LF:/alice/user/p/peters/macro.C"};
OutputFile = {"myhisto.root"};

Simple and readable!
### AliEn UI

/proc filesystem
# AliEn UI

**queue system information**

<table>
<thead>
<tr>
<th>AliEn Queue</th>
<th>all</th>
<th>aliprod</th>
<th>[%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running</td>
<td>27</td>
<td>27</td>
<td>100.00</td>
</tr>
<tr>
<td>Assigned</td>
<td>2</td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>Inserting</td>
<td>0</td>
<td>0</td>
<td>100.00</td>
</tr>
<tr>
<td>Queued</td>
<td>10</td>
<td>10</td>
<td>100.00</td>
</tr>
<tr>
<td>Waiting</td>
<td>0</td>
<td>0</td>
<td>100.00</td>
</tr>
<tr>
<td>Error_A</td>
<td>0</td>
<td>0</td>
<td>100.00</td>
</tr>
<tr>
<td>Error_E</td>
<td>0</td>
<td>0</td>
<td>100.00</td>
</tr>
<tr>
<td>Error_S</td>
<td>0</td>
<td>0</td>
<td>100.00</td>
</tr>
<tr>
<td>Error_R</td>
<td>2</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>Expired</td>
<td>46</td>
<td>19</td>
<td>41.30</td>
</tr>
<tr>
<td>Done</td>
<td>4556</td>
<td>4430</td>
<td>89.33</td>
</tr>
<tr>
<td>Idle</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interactive</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job Execution</th>
<th>all</th>
<th>aliprod</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exec. Efficiency</td>
<td>93.04 %</td>
<td>93.55 %</td>
</tr>
<tr>
<td>Assign. Ineff.</td>
<td>0.00 %</td>
<td>0.00 %</td>
</tr>
<tr>
<td>Submission Ineff.</td>
<td>0.00 %</td>
<td>0.00 %</td>
</tr>
<tr>
<td>Execution Ineff.</td>
<td>0.00 %</td>
<td>0.00 %</td>
</tr>
<tr>
<td>Expiration Ineff.</td>
<td>0.82 %</td>
<td>0.43 %</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Present Resource Usage</th>
<th>all</th>
<th>aliprod</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU [GHz]</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>RSize [Mb]</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>VSize [Mb]</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Site Statistic</th>
<th>Done</th>
<th>Run</th>
<th>Queu</th>
<th>Start</th>
<th>Error</th>
<th>Idle</th>
<th>Lact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice::Bar1::PBS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::Bergen::PBS</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::Calcutta::PBS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::Catania::PBS</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
**AliEn UI**

queue system site statistic

<table>
<thead>
<tr>
<th>Site</th>
<th>Done</th>
<th>Run</th>
<th>Queu</th>
<th>Start</th>
<th>Error</th>
<th>Idle</th>
<th>Idact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice::Bari::PBS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::Bergen::PBS</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::Calcutta::PBS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::Catania::PBS</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::CCIN2P3::BQS</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::CERN::FAKE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::CERN::LXSHARE</td>
<td>1896</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::CERN::Oplapro</td>
<td>55</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::CERN::PCEPALICE45</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::CNAF::PBS</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::Cyfronet::PBS</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::FZK::PBS</td>
<td>1132</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::GSI::LSF</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::Houston::PBS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::ITEP::PBS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::JINR::PBS</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::LBL::LSF</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::LCG::Torino</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::OSG::PBS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::Padova::LSF</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::Prague::PBS</td>
<td>568</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Alice::SUBATECH::PBS</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alice::Torino::PBS</td>
<td>187</td>
<td>26</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
AliEn Monitoring – MonaLisa
AliEn Performance Test at SC2003
Submission of 1000 production jobs (run time 5-20 minutes/job)
AliEn Performance Test at SC2003
AliEn Performance Test for Alice Data Challenge Preparation

- new VO for Alice Data Challenge

10 API instances

MySQL DB

Performance of File Registration:
170 Files/s ⇔ 1000 DB inserts/s
Jointventure of AliEn + ROOT

- AliEn Services + Catalogue are accessible via `TAlien(TGrid)` class and global `<gGrid>` variable in ROOT
- `TAlien` uses a SOAP based AliEn C++ API

Examples:

- `TGrid::Connect("alien://aliendb1:15000/?direct","");`
  // inititate gGrid with AliEn plugin (API server at aliendb1, port 15000)
- `gGrid->mkdir("/alice/acat03");`
  // create directory in virtual file catalogue
const char *GetGrid() const { return fGrid; }
const char *GetHost() const { return fHost; }
Int_t GetPort() const { return fPort; }
virtual Bool_t IsConnected() const { return fPort == -1 ? kFALSE : kTRUE; }

virtual const char *GetUser() = 0;

//--- file access
virtual GridAccessPath_t *GetAccessPath(const char *lfn, Bool_t mode = kFALSE, const char *msn = 0, const char *path = 0);
virtual char *GetFilePath(const char *msn, const char *path = 0);

//--- file access posix interface
virtual GridFileHandle_t GridOpen(const char *lfn, Int_t flags, Uint_t mode = 0);
virtual Int_t GridClose(GridFileHandle_t handle);
virtual GridResultHandle_t GridRead(GridFileHandle_t handle, void *buffer, Long_t size, Long64_t offset);
virtual Int_t GridWrite(GridFileHandle_t handle, void *buffer, Long_t size, Long64_t offset);
virtual Int_t GridFstat(GridFileHandle_t handle, gridstat_t *statbuf);
virtual Int_t GridSync(GridFileHandle_t handle);
virtual Int_t GridFchmod(GridFileHandle_t handle, Uint_t mode);
virtual Int_t GridChown(GridFileHandle_t handle, Uint_t owner, Uint_t group);
virtual Int_t GridLink(const char *source, const char *target);
virtual Int_t GridSymLink(const char *source, const char *target);
virtual Int_t GridLstat(const char *lfn, gridstat_t *statbuf);
virtual Int_t GridLstat(const char *lfn, gridstat_t *statbuf);
virtual GridFileHandle_t GridOpendir(const char *dir);
virtual const char *GridFileEntry_t *GetFileEntry(GridFileHandle_t handle);
virtual GridFileHandle_t GridGetPhysicalFileNames(const char *lfn);

//--- file catalog browsing
virtual GridResultHandle_t OpenDir(const char *ldn);
virtual GridResult_t *ReadResult(GridResultHandle_t hResult);
virtual void CloseResult(GridResultHandle_t hResult);
virtual void ResetResult(GridResultHandle_t hResult);

//--- PROOF interface
virtual TGridResult *CreateGridResult(GridResultHandle_t handle);
virtual char *GridResultPath(const char *lfn, Bool_t mode = kFALSE, const char *msn = 0, const char *path = 0);

//--- file catalog management
virtual Bool_t IsConnected() const { return fPort == -1 ? kFALSE : kTRUE; }
virtual GridFileHandle_t GridOpen(const char *lfn, Int_t flags, Uint_t mode = 0);
virtual Int_t GridClose(GridFileHandle_t handle);
virtual GridResultHandle_t GridRead(GridFileHandle_t handle, void *buffer, Long_t size, Long64_t offset);
virtual Int_t GridWrite(GridFileHandle_t handle, void *buffer, Long_t size, Long64_t offset);
virtual Int_t GridFstat(GridFileHandle_t handle, gridstat_t *statbuf);
virtual Int_t GridSync(GridFileHandle_t handle);
virtual Int_t GridFchmod(GridFileHandle_t handle, Uint_t mode);
virtual Int_t GridChown(GridFileHandle_t handle, Uint_t owner, Uint_t group);
virtual Int_t GridLink(const char *source, const char *target);
virtual Int_t GridSymLink(const char *source, const char *target);
virtual Int_t GridLstat(const char *lfn, gridstat_t *statbuf);
virtual Int_t GridLstat(const char *lfn, gridstat_t *statbuf);
virtual GridFileHandle_t GridOpendir(const char *dir);
virtual const char *GridFileEntry_t *GetFileEntry(GridFileHandle_t handle);
virtual GridFileHandle_t GridGetPhysicalFileNames(const char *lfn);

//--- file catalog queries
virtual GridResultHandle_t Find(const char *path, const char *file, const char *conditions = 0);
virtual GridResultHandle_t FindEx(const char *path, const char *file, const char *conditions = 0);
virtual Int_t Mkdir(const char *ldn, Bool_t recursive = kFALSE);
virtual Int_t Rmdir(const char *dir, Bool_t recursive = kFALSE);
virtual Int_t M(const char *lfn, Bool_t recursive = kFALSE);
virtual Int_t C(const char *sourceLn, const char *targetLn) = 0;
virtual Int_t Mv(const char *sourceLn, const char *targetLn) = 0;
virtual Int_t Chmod(const char *lfn, Uint_t mode) = 0;
virtual Int_t Chown(const char *lfn, const char *owner, const char *group) = 0;
virtual Int_t AddFile(const char *newLn, const char *PFN, Int_t size = -1, const char *msn = 0, char *guid = 0);
virtual Int_t AddFileMirror(const char *lfn, const char *PFN, const char *msn = 0);
Task Splitting in AliEn

1 **Job Spawning:**
   - a production job is spawned \(<n>\) times by the Job Optimizer

2 **File based Splitting:**
   - a task is split into one job for each input data file

3 **Storage Location Splitting:**
   - a task is split into one job per storage system

4a/b **File size/number based Splitting:**
   - a task is split into one job for a group of input data files with a maximum input data size or maximum number of files

5 **Combinations of 3+4a & 3+4b & 3+4a+4b**
Batch Analysis AliEn + ROOT

• Task splitting is executed by the AliEn JobOptimizer

• From the ROOT prompt
  → Input Data query
  → Registration of Input Files and macros
  → Submission of single job requests,
    which are spawned afterwards
  → Status Check
  → JDL creation
  → Interactive result file merging
  → batch result file merging
Batch Analysis AliEn + ROOT

Since the task splitting is defined in the JDL, we can commit any kind of job from the ROOT prompt using the appropriate JDL syntax:

\[
g\text{Grid}->\text{SubmitJob(“<jdlfile> [parameter1] [parameter2] .... ”);}
\]

F.e. for a production job with 100 spawned subjobs:

\[
g\text{Grid}->\text{SubmitJob(“/alice/jdl/AliRootProd.jdl myproductiondir 100”);}\]

For analysis specify SE and file size based job splitting and put as input data a query with the specific data f.e. \texttt{ana.jdl}:

\[
\begin{array}{l}
\text{InputData=“/alice/production/aliprod/prod001/*galice.root”;}\\
\text{Split=”SE”;}\\
\text{MaxInputFileSize=”100000000”;}\\
\text{MaxInputFileNumber=”10”;}
\end{array}
\]
**Batch Analysis – Result Merging**

To merge automatically histogram files of submitted analysis job, task splitting with storage location and a maximum number of files can be used, to merge

1\textsuperscript{st} in each site locally the results and
2\textsuperscript{nd} the premerged results of all sites.

The 2\textsuperscript{nd} job depends on the 1\textsuperscript{st}, in the JDL one has to specify:

```
JobPrecessor={<jobId 1\textsuperscript{st} >}
```
Basic requirements:
- Analysis data has to be stored as objects derived from TObject in ROOT trees
- proofds have to load extension libraries for user-specific objects to access the data members
- Analysis code has to be inserted in the automatically generated selector macro for the object to be analyzed:

```cpp
<classobject>->MakeSelector();
```

Recipe:
- store your objects in trees
- use the selector macro for analysis code
Interactive Analysis with AliEn + ROOT/PROOF

- Proof allows **interactive Analysis** on local clusters with a **static master/slave configuration**

Classical Proof Setup:

- in the GRID Environment, **resources are distributed and temporary**:  
  - no guaranteed incoming access to every possible slave server  
  - resources are dynamic and not static available
Interactive Analysis with AliEn + ROOT/PROOF

AliEn Grid Proof Setup:

- Guaranteed site access through multiplexing TcpRouters
**Site Access**
- TcpRouting Service for connection forwarding to 'idle' jobs (proofds)
  - retrieves information about local proofds from the queue system
  - does blind packet forwarding
  - established connections change status of proofd to 'interactive'

**Master Access**
- access through temporary TcpRouter to connect to hidden port
- access is gained like in a booking system for a reserved time

**Authentication**
- Proof master authenticates with a crypted secret password to site daemons-crypted password is distributed via the TcpRouter service
- Proof client authenticates with a different temporary password retrieved with the AliEn API
Interactive Analysis with AliEn + ROOT/PROOF

Work Distribution

• interactive analysis with Proof steered by a data packetizer

• in a local cluster:
  → cluster wide accessible data can be processed by all slaves
  → packet takeover by all slaves!

• in a grid environment:
  → site wide accessible data can be processed by all slaves
  → packet takeover by all slaves within one site!
Interactive Analysis with AliEn + ROOT/PROOF

Daemon Population + Reservation

- daemons can be **started manually** with a special **interactive JDL tag** on a dedicated site

- daemon assignment through dedicated Proof service:
  - corresponding to **accessability of input data**
  - **availability** of daemons and resources
  - **limited duration**
  - identified by **session ID**

- **Proof populator service** in preparation:
  - **dynamic** increase and decrease of proofd **population**
    - depending on available resources
    - depending on static population configuration per site
    - depending on booking requests
Interactive Analysis with AliEn + ROOT/PROOF

Sample Session: Connect/Query
Interactive Analysis with AliEn + ROOT/PROOF

Sample Session: proofd information/request
Sample Session: connection to assigned proofds
Sample Session: data processing
Sample Session: data processing with feedback histogram
Interactive Analysis with AliEn + ROOT/PROOF

Sample Session: data processing
Unification of Batch & Interactive Analysis with AliEn + ROOT/PROOF

**current implementation:**

- datasets are represented by objects of the type **TDSet in ROOT**
- a **GRID data query** assigns data files to TDSet Objects
- the “**process**” method **initiates** the **interactive processing** on the assigned GRID proof cluster

**to come:**

- the same “**process**” method **initiates** the **batch processing** of the same data set and automatic merging of results.

**ALICE will test the analysis facilities during the physics data challenge in early 2004.**
Evolution from Queue Model to Job Agents

- no distinction between interactive and batch mode
- faster response and optimized scheduling
- advance reservations
- avoid queue drainer (black hole) scenario
- Job Agents on user workstations
  (advertised as lower quality or low I/O service)
Outlook & Summary

• AliEn provides already now most of the required features to a GRID system for LHC experiments
• UI is getting very similar to unix standards (/proc, df ...)
• AliEn modularity (distributed databases and services) allow a good scalability of the system
• AliEn will serve as a basis for the next service oriented GRID prototype at CERN (ARDA recommendation)
• AliEn was already used running 30,000 jobs in productions for the ALICE PPR report
• AliEn will be tested extensively during the upcoming ALICE data challenges

Conclusions

• we are close having one solution for data analysis:
  ▶ complete embedding of GRID functionality in ROOT
  ▶ equal procedures for batch and interactive analysis