PROOT Tutorials – Session 11

PROOF, GRID, AliEn

Fons Rademakers

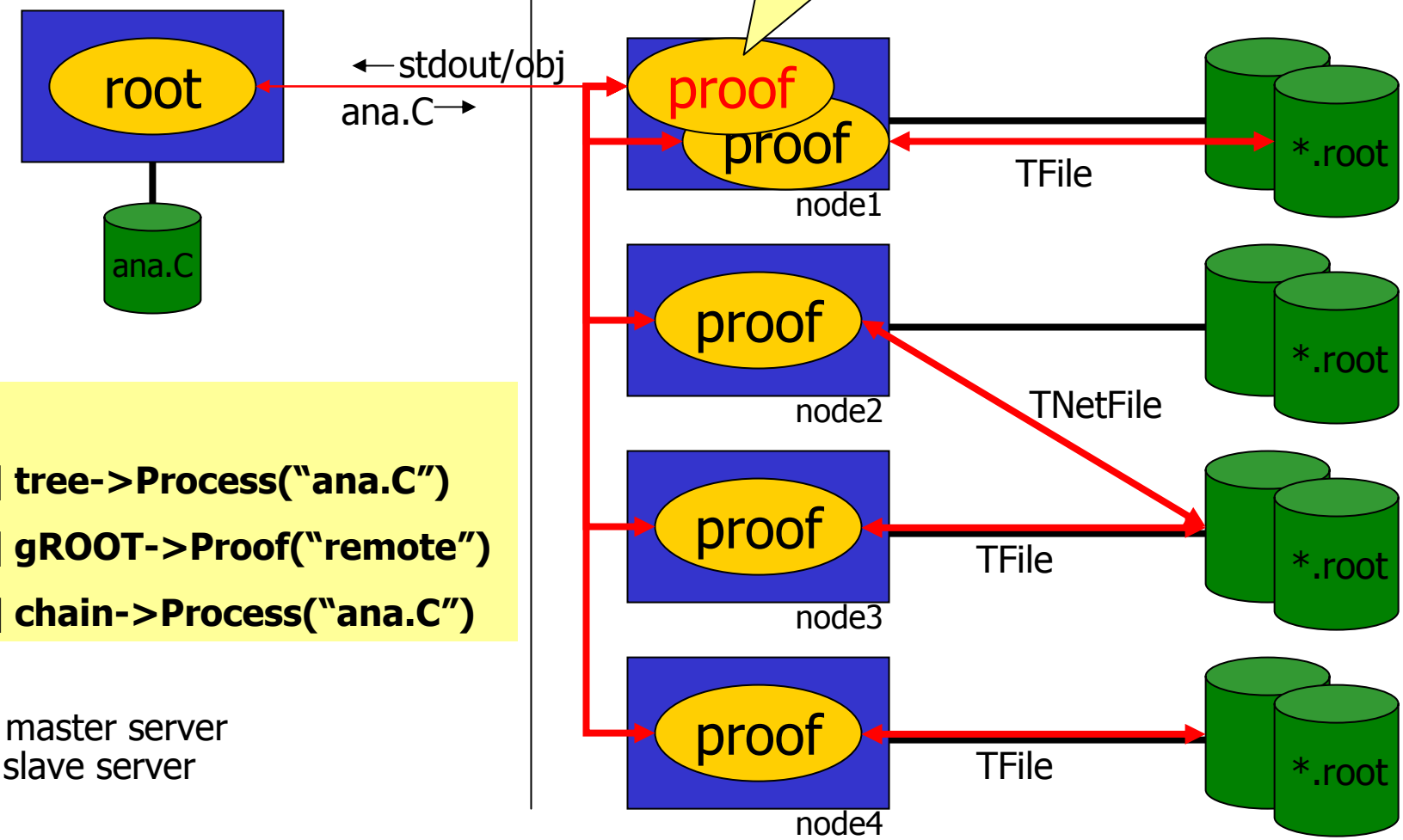
Bring the KB to the PB not the PB to the KB

Parallel ROOT Facility

- The PROOF system allows:
  - Parallel analysis of trees in a set of files
  - Parallel analysis of objects in a set of files
  - Parallel execution of scripts

  on clusters of heterogeneous machines
- Its design goals are:
  - Transparency, scalability, adaptability
- Prototype developed in 1997 as proof of concept, full version nearing completion now

Parallel Script Ex n



$ root

root [0] tree->Process("ana.C")

root [1] gROOT->Proof("remote")

root [2] chain->Process("ana.C")

proof = master server
proof = slave server

Data Access Strategies

- Each slave get assigned, as much as possible, packets representing data in local files
- If no (more) local data, get remote data via rootd and rfio (needs good LAN, like GB eth)
- In case of SAN/NAS just use round robin strategy

PROOF Transparency

- Make working on PROOF as similar as working on your local machine
- Return to the client all objects created on the PROOF slaves
  - The master server will try to add "partial" objects coming from the different slaves before sending them to the client

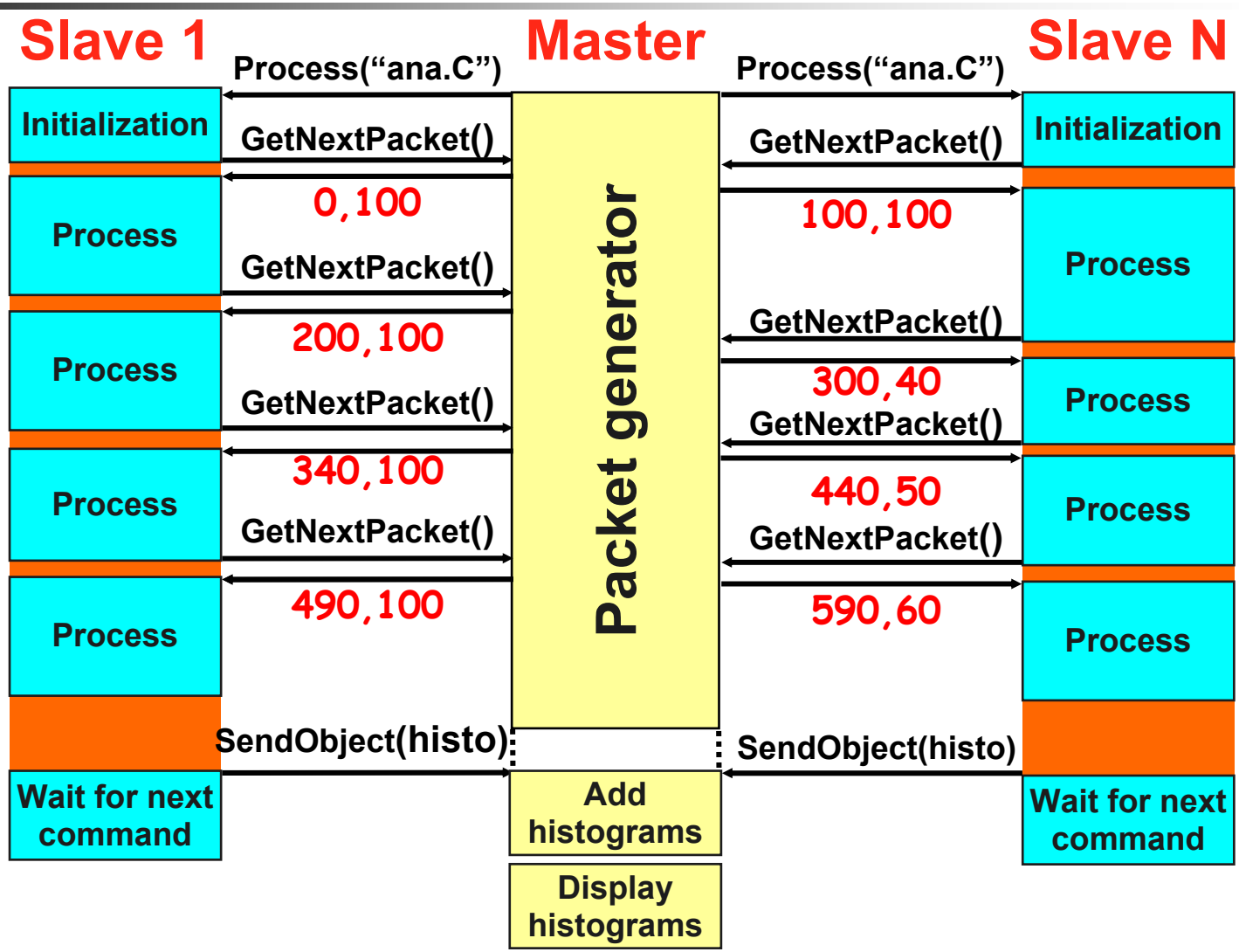
PROOF Scalability

- Scalability in parallel systems is determined by the amount of communication overhead (Amdahl's law)
- Varying the packet size allows one to tune the system. The larger the packets the less communications is needed, the better the scalability
  - Disadvantage: less adaptive to varying conditions on slaves

PROOF Adaptability

- Adaptability means to be able to adapt to varying conditions (load, disk activity) on slaves
- By using a "pull" architecture the slaves determine their own processing rate and allows the master to control the amount of work to hand out
  - Disadvantage: too fine grain packet size tuning hurts scalability

Workflow For Tree Analysis – Pull Architecture



PROOF Error Handling

- Handling death of PROOF servers
  - Death of master
    - Fatal, need to reconnect
  - Death of slave
    - Master can resubmit packets of death slave to other slaves
- Handling of ctrl-c
  - OOB message is send to master, and forwarded to slaves, causing soft/hard interrupt

PROOF Authentication

- PROOF supports secure and un-secure authentication mechanisms
- Same as for rootd
  - UsrPwd
  - SRP
  - Kerberos
  - Globus
  - SSH
  - UidGid

Architecture and Implementation

TSelector – The algorithms

- Basic ROOT TSelector

```
// Abbreviated version
class TSelector : public TObject {
Protected:
    TList *fInput;
    TList *fOutput;
public
    void Init(TTree*);
    void Begin(TTree*);
    void SlaveBegin(TTree *);
    Bool_t Process(int entry);
    void SlaveTerminate();
    void Terminate();
};
```

TDSet – The data

- Specify a collection of TTrees or files with objects

```
root[0] TDSet *d = new TDSet("TTree", "tracks", "/");
OR
root[0] TDSet *d = new TDSet("TEvent", "", "/objs");
root[1] d->Add("root://rcrs4001/a.root");
…
root[10] d->Print("a");
root[11] d->Process("mySelector.C", nentries, first);
```

- Returned by DB or File Catalog query etc.
- Use logical filenames ("lfn:…")

Sandbox – The Environment

- Each slave runs in its own sandbox
  - Identical, but independent
- Multiple file spaces in a PROOF setup
  - Shared via NFS, AFS, shared nothing
- File transfers are minimized
  - Cache
  - Packages

Sandbox – The Cache

- Minimize the number of File transfers
  - One Cache per file space
- Locking to guarantee consistency
- File identity and integrity ensured using
  - MD5 digest
  - Time stamps
- Transparent via TProof::Sendfile()

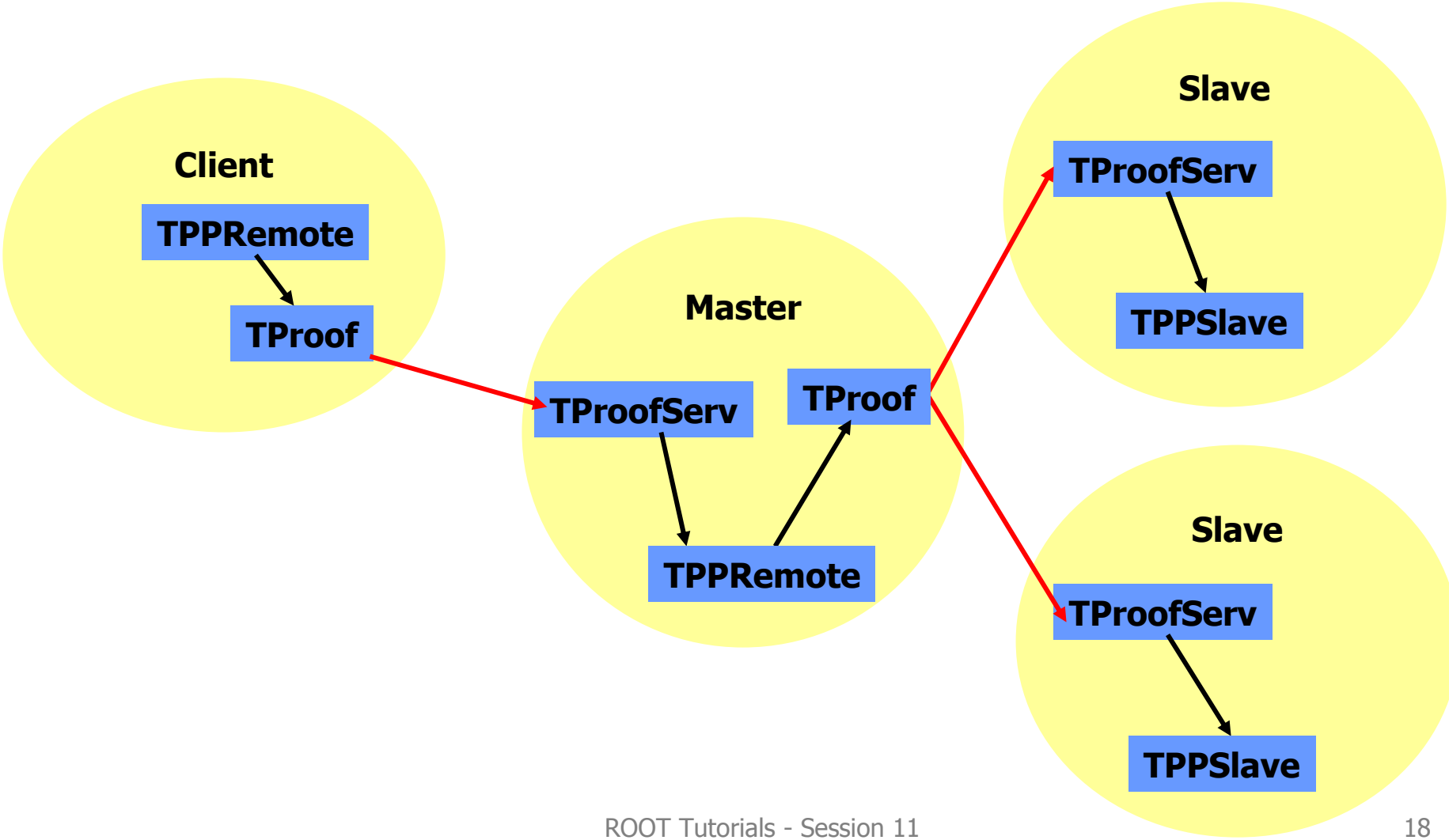
Sandbox – Package Manager

- Provide a collection of files in the sandbox
- Binary or Source packages
- PAR files: PROOF ARchive. Like Java jar
  - Tar file, ROOT-INF directory
  - BUILD.sh
  - SETUP.C, per slave setting
- API to manage and activate packages

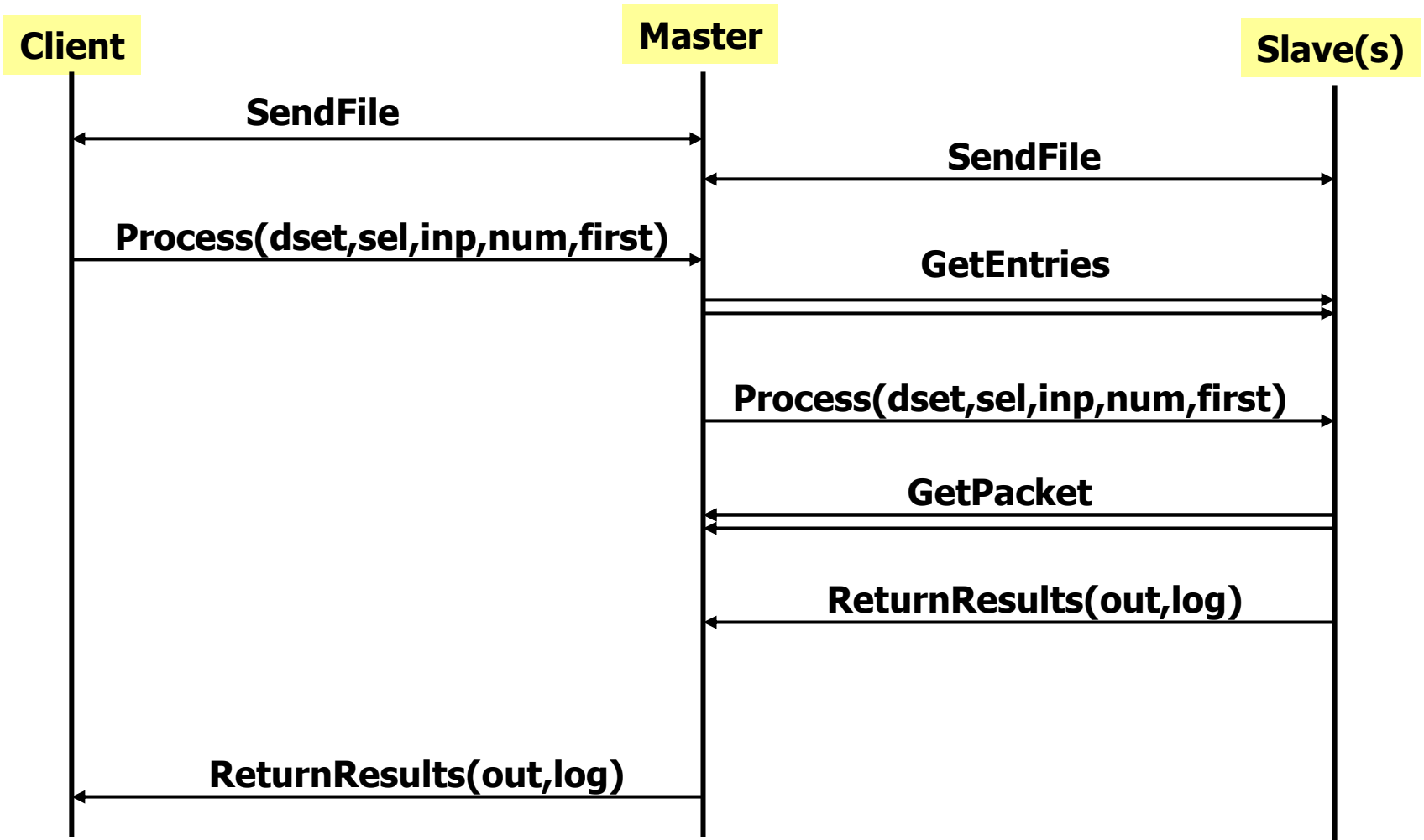
Implementation Highlights

- TProofPlayer class hierarchy
  - Basic API to process events in PROOF
  - Implement event loop
  - Implement proxy for remote execution
- TEventIter
  - Access to TTree or TObject derived collection
  - Cache file, directory, tree

TProofPlayer



Simplified Message Flow



Dynamic Histogram Binning

- Implemented using THLimitsFinder class
- Avoid synchronization between slaves
- Keep score-board in master
  - Use histogram name as key
  - First slave posts limits
  - Master determines best bin size
  - Others use these values

Merge API

- Collect output lists in master server
- Objects are identified by name
- Combine partial results
- Member function: Merge(TCollection *)
  - Executed via CINT, no inheritance required
- Standard implementation for Histograms
- Otherwise return the individual objects

Setting Up PROOF

Setting Up PROOF

- Install ROOT system
- For automatic execution of daemons add proofd and rootd to /etc/inetd.conf (or in /etc/xinetd.d) and /etc/services (not mandatory, servers can be started by users)
  - The rootd (1094) and proofd (1093) port numbers have been officially assigned by IANA
- Setup proof.conf file describing cluster
- Setup authentication files (globally, users can override)

PROOF Configuration File

```
# PROOF config file. It has a very simple format:
#
# node <hostname> [image=<imagename>]
# slave <hostname> [perf=<perfindex>]
#                  [image=<imagename>] [port=<portnumber>]
#                  [srp | krb5]
# user <username> on <hostname>

node csc02   image=nfs

slave csc03  image=nfs
slave csc04  image=nfs
slave csc05  image=nfs
slave csc06  image=nfs
slave csc07  image=nfs
slave csc08  image=nfs
slave csc09  image=nfs
slave csc10  image=nfs
```

The AliEn GRID

AliEn a Lightweight GRID

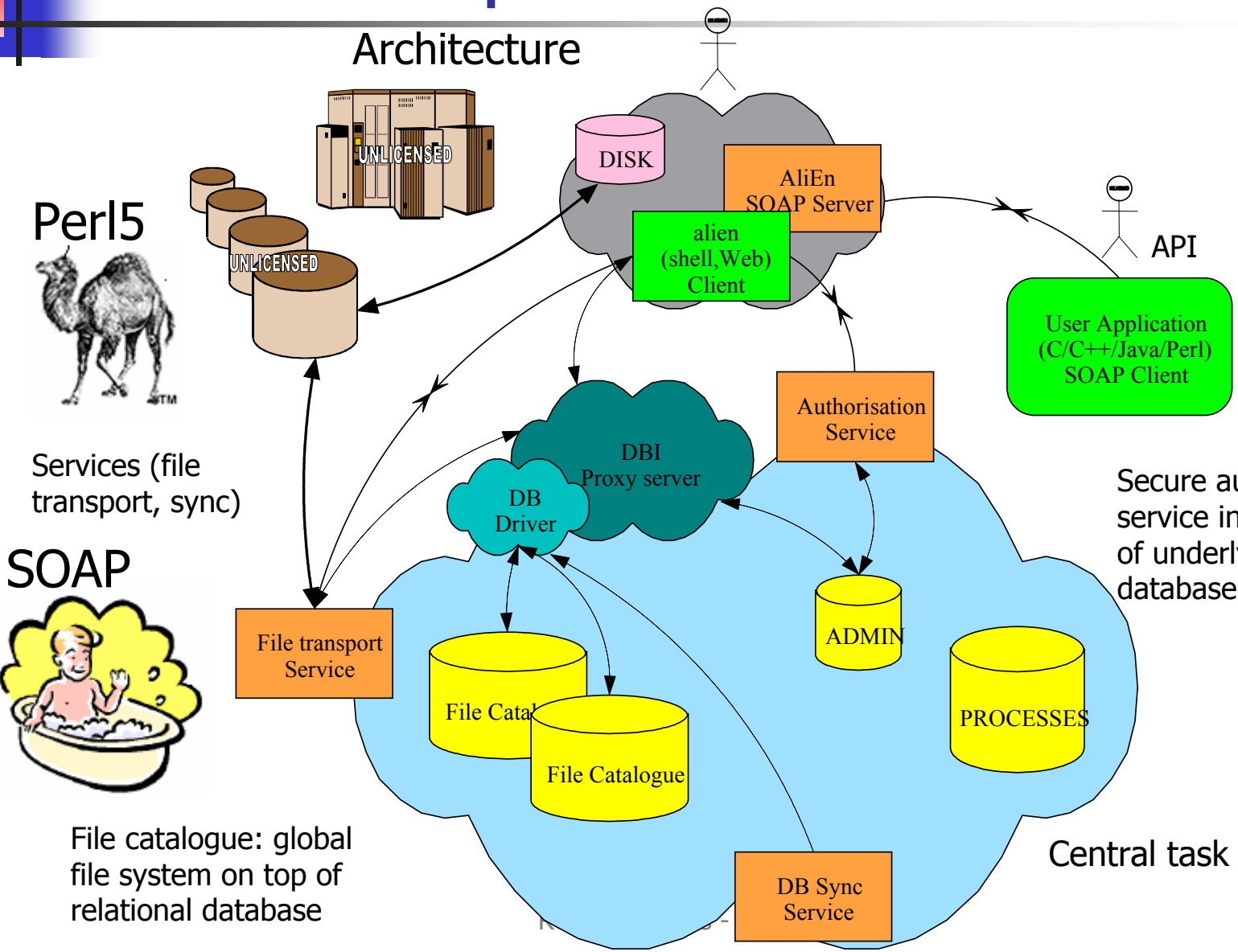
- AliEn (http://alien.cern.ch) is a lightweight alternative to full blown GRID based on standard components (SOAP, Web services)
  - Distributed file catalogue as a global file system on a RDBMS
  - TAG catalogue, as extension
  - Secure authentication
  - Central queue manager ("pull" vs "push" model)
  - Monitoring infrastructure
  - C/C++/perl API
  - Automatic software installation with AliKit

The Core GRID Functionality !!

- AliEn is routinely used in production for Alice PPR

AliEn Components

Architecture



Perl5

Services (file transport, sync)

SOAP

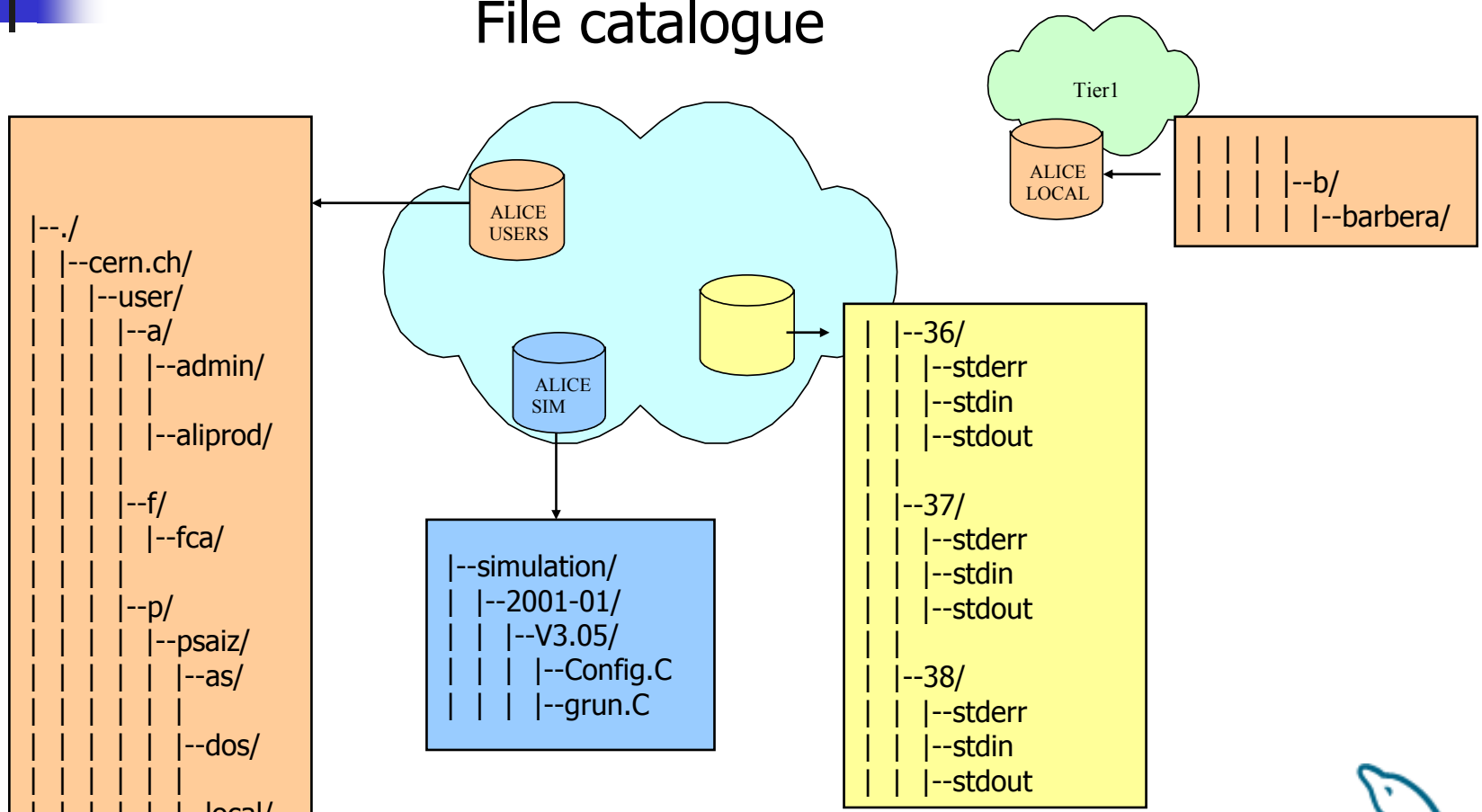
Secure authentication service independent of underlying database

File catalogue: global file system on top of relational database

Central task queue

AliEn Components

File catalogue



Files, commands (job specification) as well as job input and output, tags and even binary package tar files are stored in the catalogue

PROOF and the GRID

PROOF Grid Interface

- PROOF can use a Grid Resource Broker to detect which nodes in a cluster can be used in the parallel session
- PROOF can use Grid File Catalogue and Replication Manager to map LFN's to PFN's
- PROOF daemons can be started by Grid job scheduler
- PROOF can use Grid Monitoring Services
- Access via abstract Grid interface

TGrid Class – Abstract Interface to AliEn

```
class TGrid : public TObject {
public:
   virtual Int_t        AddFile(const char *lfn, const char *pfn) = 0;
   virtual Int_t        DeleteFile(const char *lfn) = 0;
   virtual TGridResult *GetPhysicalFileNames(const char *lfn) = 0;
   virtual Int_t        AddAttribute(const char *lfn,
                                     const char *attrname,
                                     const char *attrval) = 0;
   virtual Int_t        DeleteAttribute(const char *lfn,
                                        const char *attrname) = 0;
   virtual TGridResult *GetAttributes(const char *lfn) = 0;
   virtual void         Close(Option_t *option="") = 0;

   virtual TGridResult *Query(const char *query) = 0;

   static TGrid *Connect(const char *grid, const char *uid = 0,
                         const char *pw = 0);

   ClassDef(TGrid,0)  // ABC defining interface to GRID services
};
```

Running PROOF Using AliEn

```
TGrid *alien = TGrid::Connect("alien");

TGridResult *res;
res = alien->Query("lfn:///alice/simulation/2001-04/V0.6*.root");

TDSet *treeset = new TDSet("TTree", "AOD");
treeset->Add(res);

gROOT->Proof(res);   // use files in result set to find remote nodes
treeset->Process("myselector.C");

// plot/save objects produced in myselector.C
. . .
```

Future

- Ongoing development
- Event lists
- Friend Trees
- Scalability to O(100) nodes
- Multi site PROOF sessions
- Move from AliEn to ARDA

Demo!

- The H1 example analysis code
  - Use output list for histograms
  - Move fitting to client
- 13 fold H1 example dataset
  - 52 files
  - 3.5 Gbyte
  - 3.6 Million Events
- 13 pchret23 machines