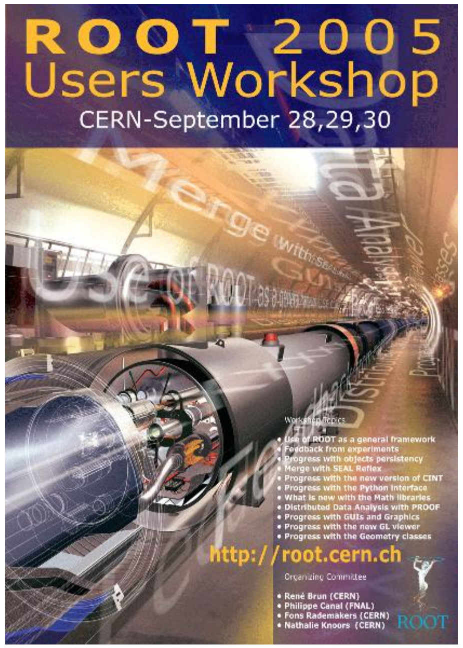

ROOT 2005
Users Workshop
CERN-September 28,29,30
Workshop Topics
• Use of ROOT as a general framework
• Feedback from experiments
• Progress with objects persistency
• Merge with SEAL Reflex
• Progress with the new version of CINT
• Progress with the Python interface
• What is new with the Math libraries
• Distributed Data Analysis with PROOF
• Progress with GUIs and Graphics
• Progress with the new GL viewer
• Progress with the Geometry classes
http://root.cern.ch
Organizing Committee
• René Brun (CERN)
• Philippe Canal (FNAL)
• Fons Rademakers (CERN)
• Nathalie Knoors (CERN)
ROOT

PROOF news

Gerardo Ganis / CERN

People working on the project:
B. Bellenot, M. Biskup, G. Kickinger, A. Peters, F. Rademakers / CERN
M. Ballintijn / MIT, D. Feitchinger / PSI

2005 Intl. ROOT Users Workshop
28 – 30 Sept 2005, CERN, Switzerland

Outline

- PROOF overview
- New features and improvements
- Plans
- Summary

PROOF – Parallel ROOT Facility

- System to export the ROOT analysis model on clusters of computers for interactive analysis of large data sets

- Flexible multi-tier architecture
  - in GRID contexts adapts to cluster of clusters or wide area *virtual clusters*

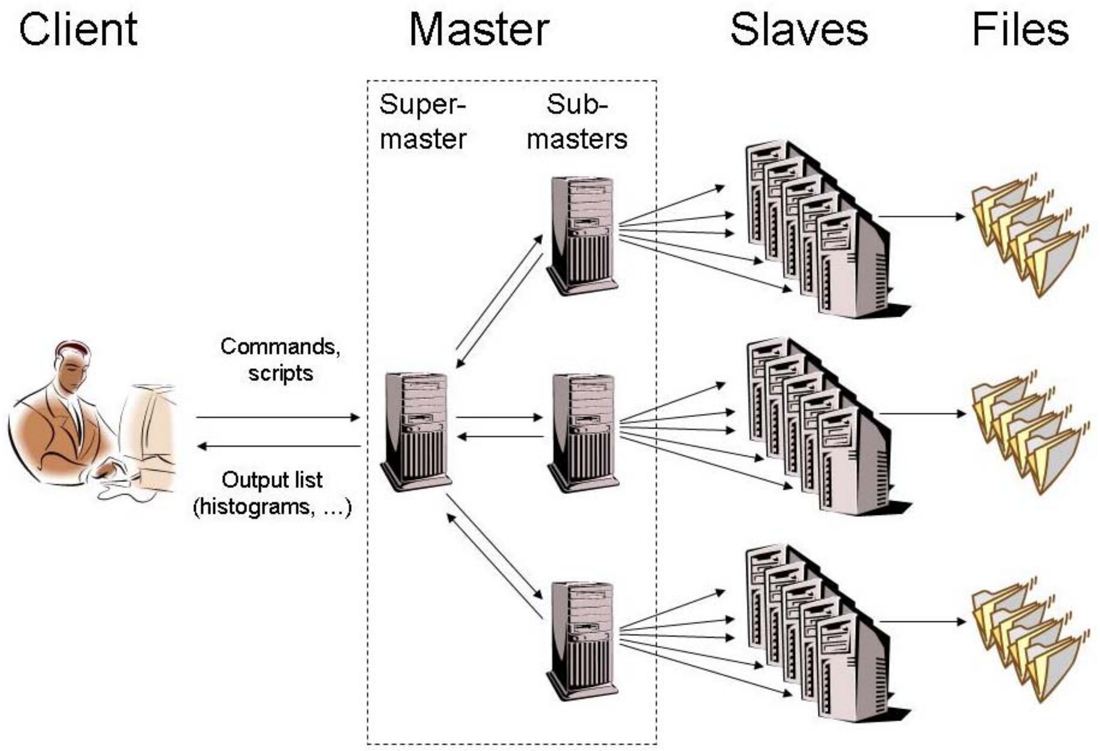
- Exploit inter-independence of entries in a tree or directory to achieve basic parallelism
  - data set split into *packets* assigned to worker nodes on demand

PROOF – Design goals

- Transparency
  - Same syntax as in normal ROOT session
  - Input objects copied from client
  - Output objects merged, returned to client
- Scalability and Adaptability
  - Handle heterogeneous server performances
  - Vary packet size according to number of worker nodes and relative performance
  - Multi-Level-Master addresses cases of wide area clusters

PROOF – Multi-tier Architecture

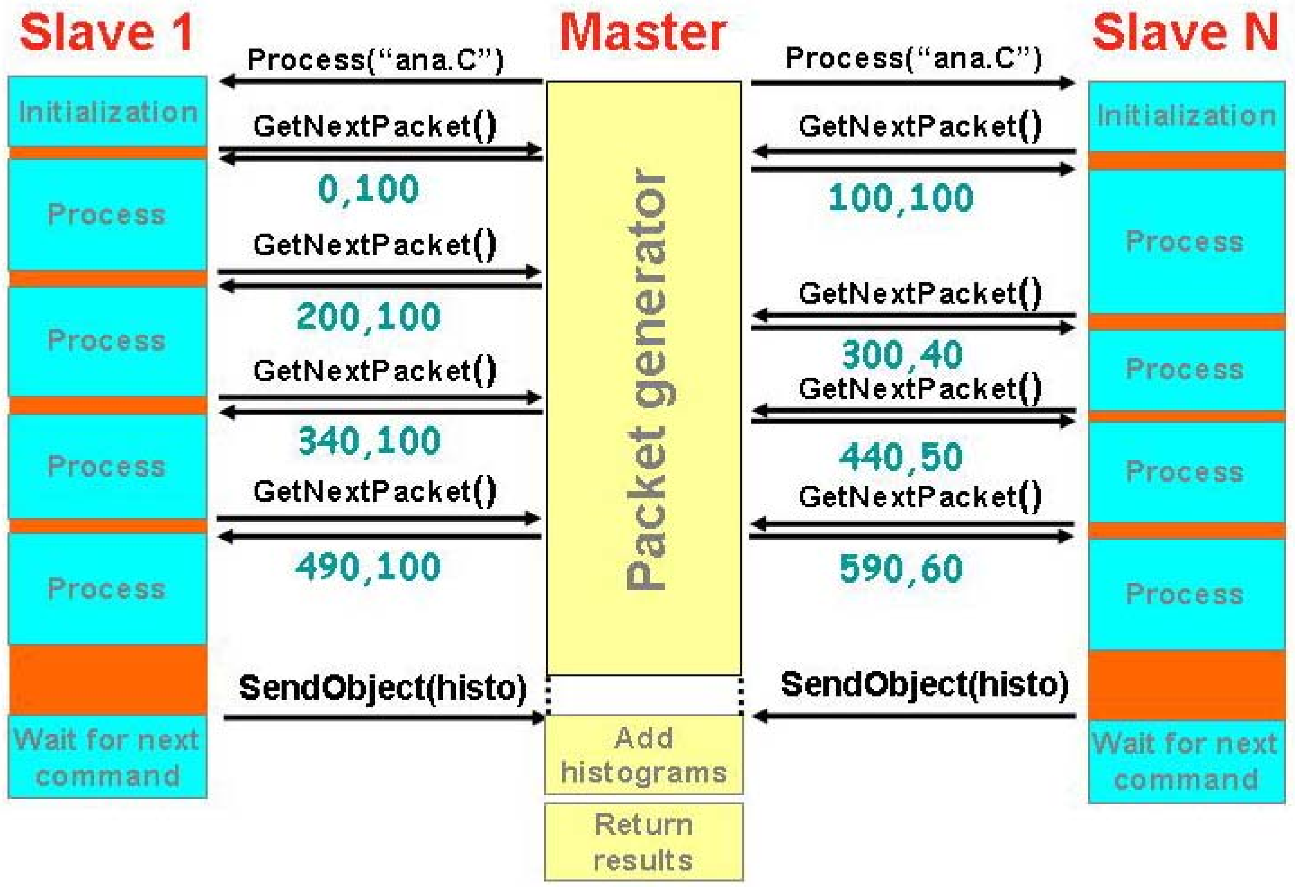
- Optimize for data locality
- If not possible, remote data via (x)rootd, rfiod, dCache …



less important —good connection ?→ VERY important

PROOF – Workflow

- Pull architecture
  - *dynamic load balancing* naturally achieved



PROOF – User Sandbox

- User’s have their own sandbox on each worker node

- File transfers minimized
  - cache packages, selector
  - File integrity: MD5 checksums, timestamps

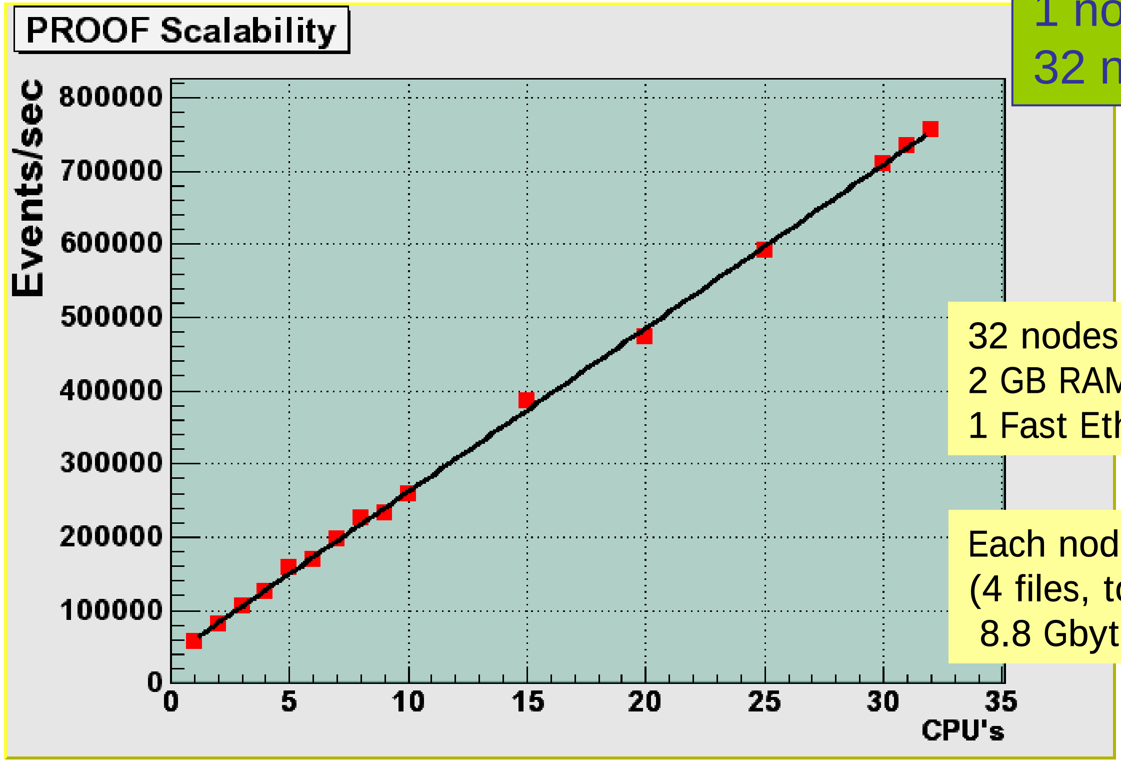
- Package manager to upload files or packages
  - binary or source
  - PAR (PROOF Archive, like Java jar)
    - provides ROOT-INF directory, BUILD.sh, SETUP.C to control setup in each worker
  - **`TProof`** API to handle all this

PROOF – Additional issues

- Error handling
  - Death of master: fatal
  - Death of worker: redirect its work to other workers
  - Ctrl-C interrupt: send as OOB message (C->M, M->W)

- Authentication
  - Supported: password-based protocols, globus/GSI, Krb5, fast identification
  - Credential forwarding
  - Mixed configuration possible
    - e.g. pwd (C->M), fast ID internally (M->W)

PROOF – Scalability



8.8GB, 128 files
1 node: 325 s
32 nodes in parallel: 12 s

32 nodes: dual Itanium II 1 GHz CPU's,
2 GB RAM, 2x75 GB 15K SCSI disk,
1 Fast Eth, 1 GB Eth nic (not used)

Each node has one copy of the data set
(4 files, total of 277 MB), 32 nodes:
8.8 Gbyte in 128 files, 9 million events

PROOF – data analysis

More about this in M.Biskup's talk

Normal ROOT

- **TChain**: collection of **TTree**
- **TSelector**: frame for **Begin()**, **Process()**, **Terminate()**

```
TChain a("h42");
{// Define the data set
   a.Add("root://oplapro62.cern.ch//tmp/dstarmb.root");
   a.Add("root://oplapro62.cern.ch//tmp/dstarp1a.root");
   a.Add("root://oplapro62.cern.ch//tmp/dstarp1b.root");
   a.Add("root://oplapro62.cern.ch//tmp/dstarp2.root");
   // Process the selector
   a.Process("h1analysis.C");
}
```

Local processing

PROOF

- Same chain, same selector

```
{// Open PROOF
   TProof *proof = new TProof("master")
   // Process the selector
   a.Process("h1analysis.C");
}
```

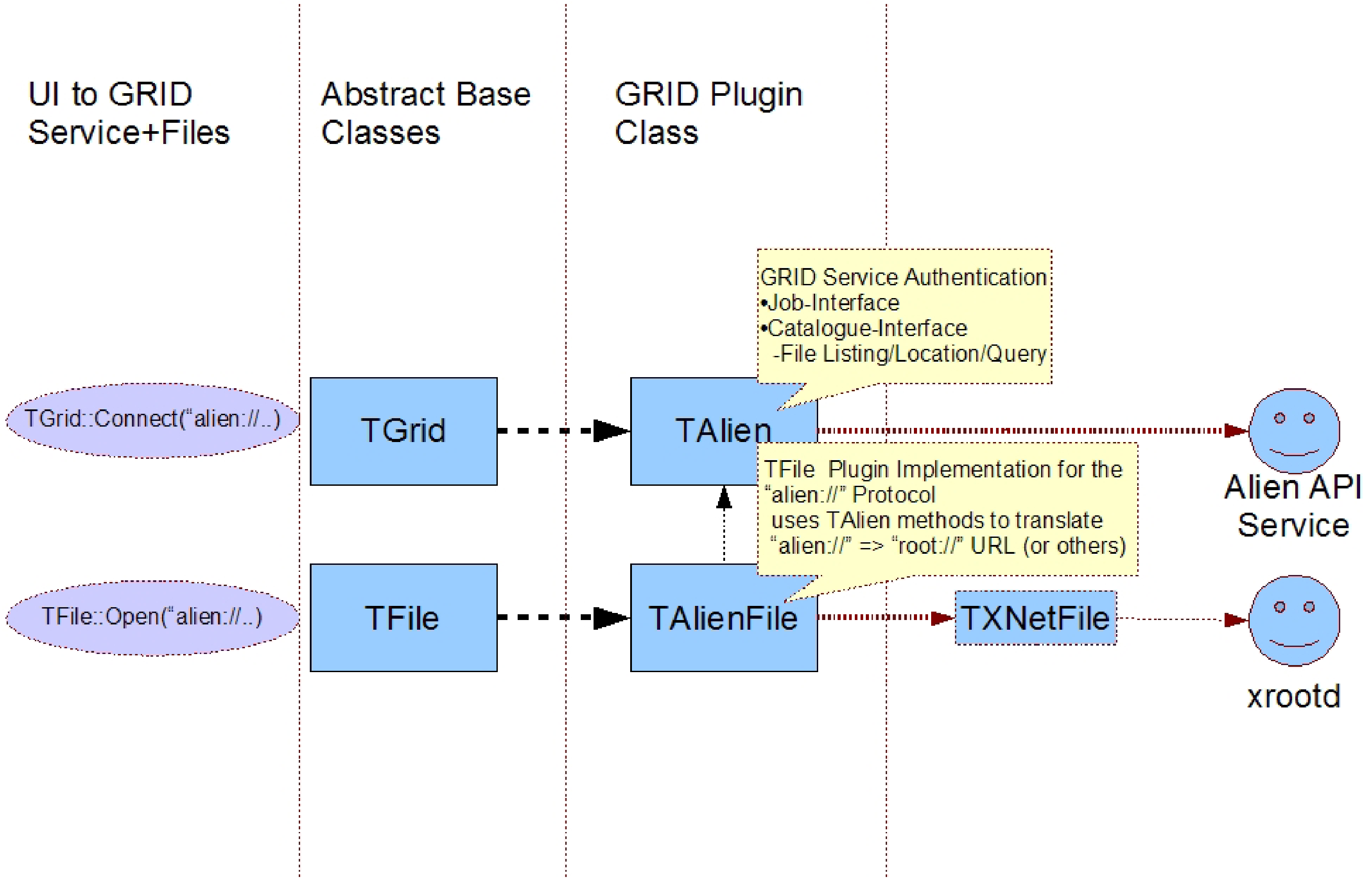
Remote processing

PROOF on wide area clusters

- Use “file catalogs” for file location
- Use “resource brokers” to identify best worker nodes
- Examples: interface with Condor COD, PEAC
  - see M. Ballintijn at ROOT04
- In GRID context
  - `TGrid`: abstract interface for all services

```
class TGrid : public TObject {
public:
   …
   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
};
```

TGrid: AliEn implementation


UI to GRID Service+Files
Abstract Base Classes
GRID Plugin Class
GRID Service Authentication
•Job-Interface
•Catalogue-Interface
-File Listing/Location/Query
TGrid::Connect("alien://..)
TGrid
TAlien
Alien API Service
TFile Plugin Implementation for the "alien://" Protocol uses TAlien methods to translate "alien://" => "root://" URL (or others)
TFile::Open("alien://..)
TFile
TAlienFile
TXNetFile
xrootd

TGrid: AliEn implementation (cnt'd)

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

// Query
TGridResult *res =
   alien->Query("/alice/cern.ch/user/p/peters/analysis/miniesd/",
                "*.root");
// List of files
TList *listf = res->GetFileInfoList();

// Create chain
TChain *chain = new TChain("Events", "session");
chain->AddFileInfoList(res);

// Start PROOF
TProof *proof = new TProof("remote");
chain->SetProof();

// Process your query
chain->Process("selector.C");
```

DEMO

New developments

- Towards "interactive batch"
  - Support for asynchronous running mode
  - Support for multi sessions
  - Query result management
- GUI controller

Interactive batch

Why? Analysis session: example

AQ1: 1s query produces a local histogram

AQ2: a 10mn query submitted to PROOF1

AQ3->AQ7: short queries

AQ8: a 10h query submitted to PROOF2

Monday at 10h15 ROOT session on my laptop

BQ1: browse results of AQ2

BQ2: browse temporary results of AQ8

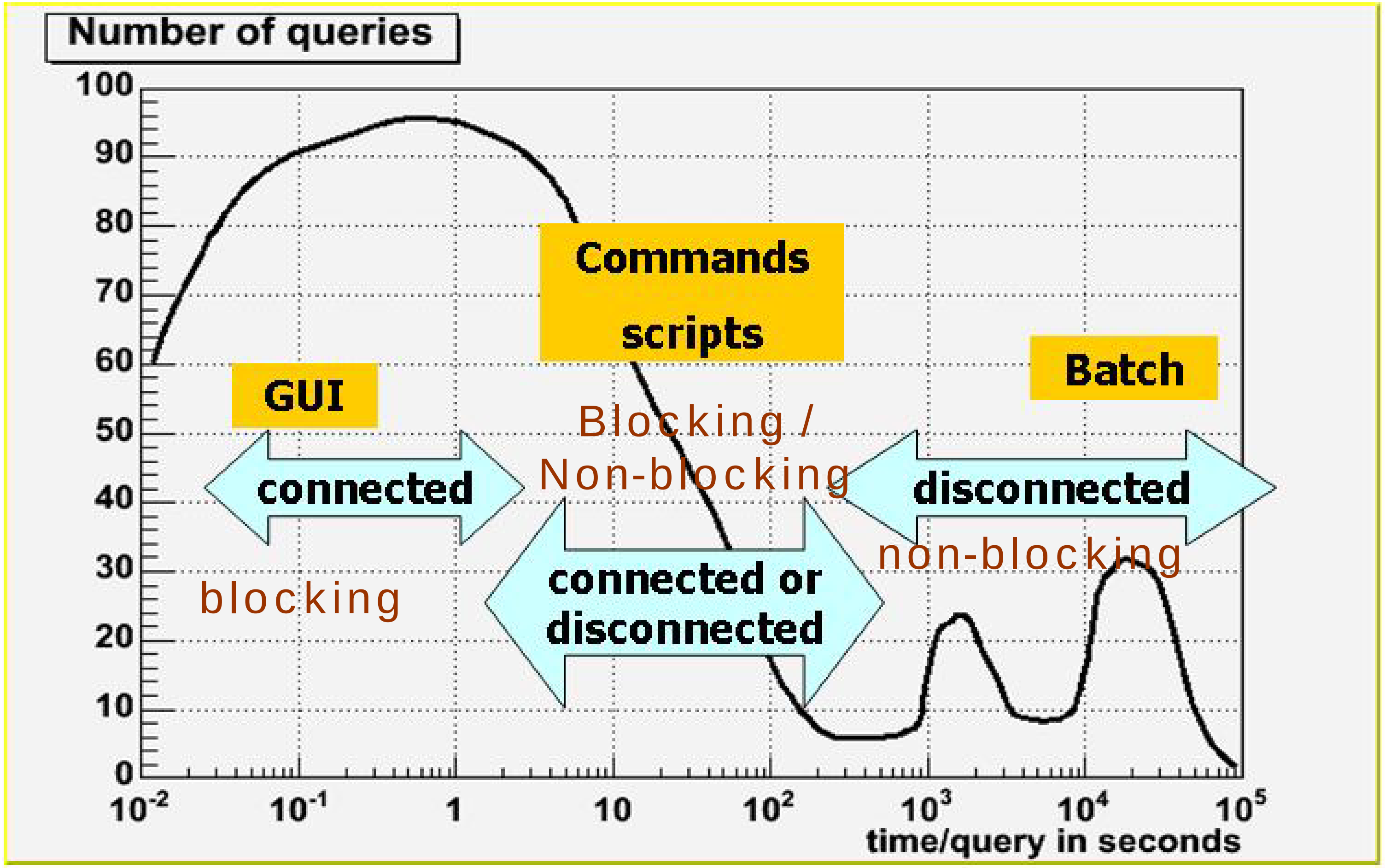
BQ3->BQ6: submit 4 10mn queries to PROOF1

Monday at 16h25 ROOT session on my laptop

CQ1: Browse results of AQ8, BQ3->BQ6

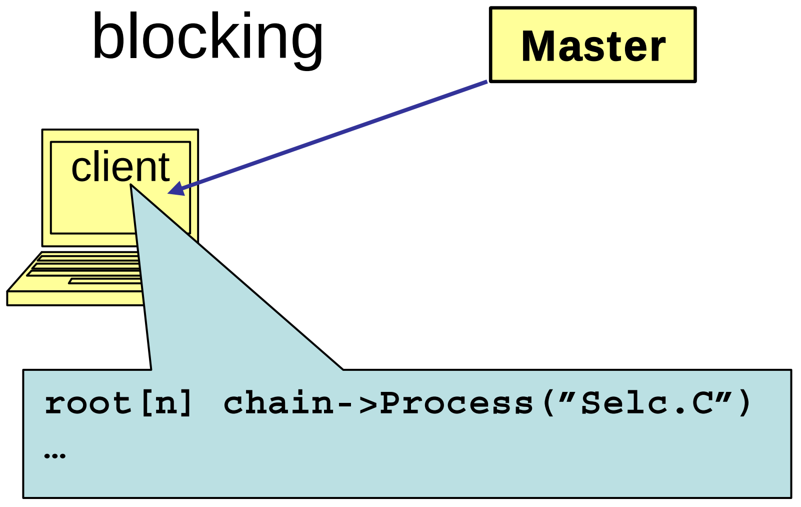
Wednesday at 8h40 Carrot session on any web browser

Typical query-time distribution

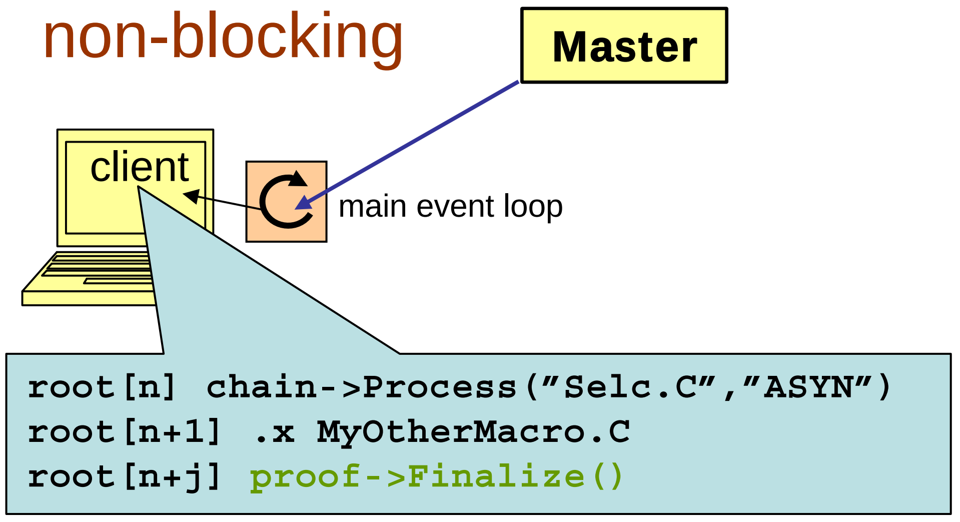


Asynchronous mode

- PROOF initially designed for short queries:
  - `TProof::Process()` blocks, as in a local session
- Inconvenient for longer queries:
  - *idle* client session waiting for master reply
- Non-blocking processing achieved handling input from master via the main event loop



```
root[n] chain->Process("Selc.C")
…
```



```
root[n] chain->Process("Selc.C","ASYN")
root[n+1] .x MyOtherMacro.C
root[n+j] proof->Finalize()
```

Asynchronous mode (cnt'd)

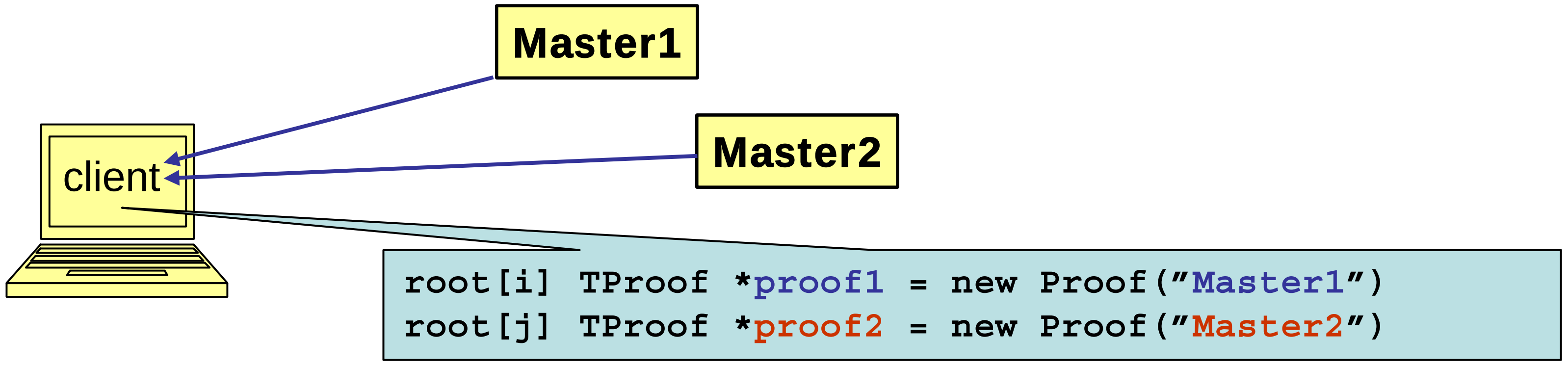
- End-of-Processing signaled by reception of a `TQueryResult` object (see below)

- `TProof::Finalize()`
  - Re-initialize `TSelector`, if needed
  - Merge outputs
  - Run `TSelector::Terminate()`

- *Submitted* queries are added to the waiting list on the master and processed sequentially

Multi sessions

- Start PROOF sessions to more than one cluster



```
root[i] TProof *proof1 = new Proof("Master1")
root[j] TProof *proof2 = new Proof("Master2")
```

- List of open sessions in **`TROOT`**

```
root[n] gROOT->GetListOfProofs()->ls()
```

- Results of queries saved in lists owned by **`TProof`**

```
root[m] proof1->GetQueryResults()->ls()
```

Query result management

- Asynchronous finalization needs selector, output list
- Keep track of what produced a given set of results

- `TQueryResult` : everything about a query
  - unique identifier (PROOF session + sequential #)
  - selector files (in compressed form, `TMacro`)
  - data set definition, list of loaded packages
  - processing info:
    - logs, start/end time, performance parameters
  - list of output objects
- Global list of `TQueryResult` in `TProofPlayer`

- Will also be used for local sessions

Query result management (cnt'd)

- `TQueryResult` object created by the master
  - Copy saved and continuously updated in sandbox

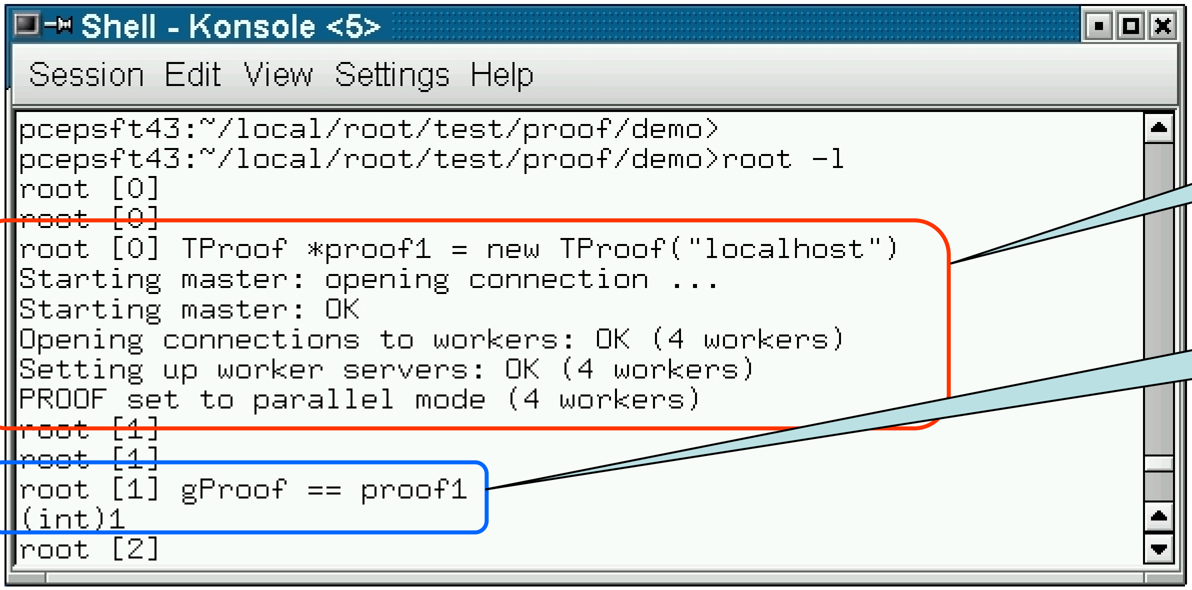
```
> ls <SandBox>/queries
./   ../   session-0-pcepsft43-1234678932-23456/
> ls <SandBox>/queries/session-0-pcepsft43-1234678932-23456
1/ 2/ 3/
> ls <SandBox>/queries/session-0-pcepsft43-1234678932-23456/2
query-result.root
```

- Master has access to all queries in the sandbox
  - can retrieve content at any time, from any session
  - can archive to any mass storage
  - can cleanup (if not in use)

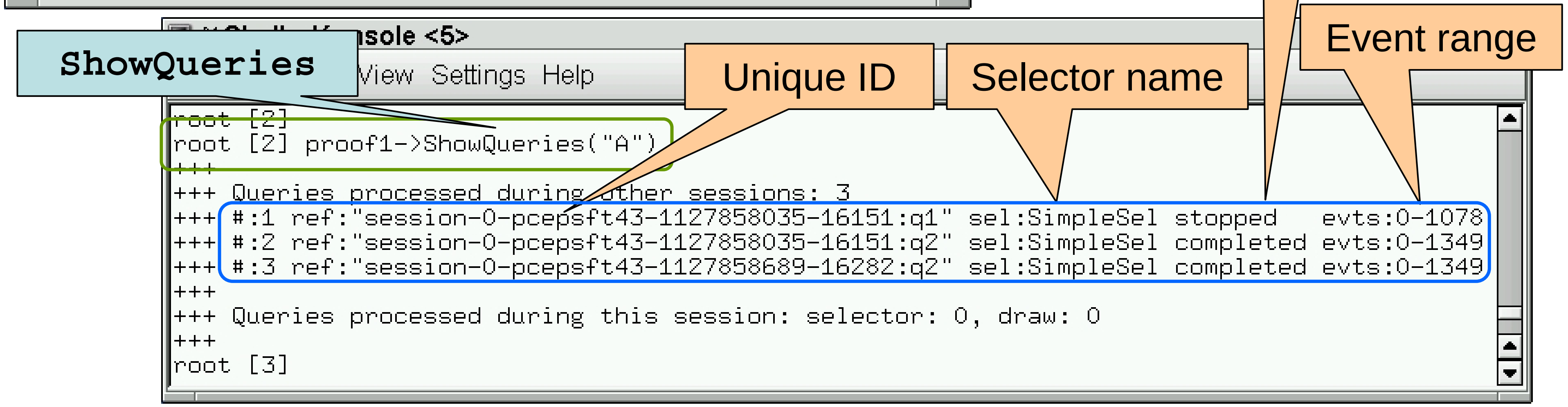
Query result management (cnt'd)

- `TProof` API to handle query results:
  - `ShowQueries("A")`
    - show list of queries, either local or known to master
  - `ShowLog(Int_t query)`
    - show logs from processing the query
  - `Finalize(Int_t query)`
    - run `Terminate()` of the output list (only once)
  - `Retrieve(Int_t query, const char *path)`
    - retrieve the result of processing from the master
  - `Remove(Int_t query)`
    - remove a query from the lists
  - `TList *GetQueryResults()`
    - get the list of `TQueryResult` already retrieved
  - …

New features at work - 1

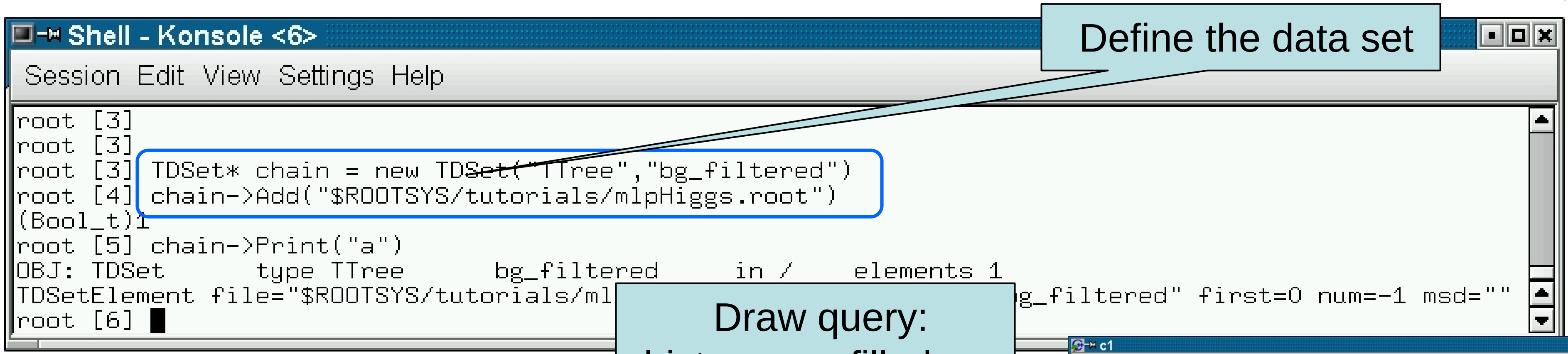

Shell - Konsole <5>
Session Edit View Settings Help
Start new session
gProof is the current session

```
pcepsft43:~/local/root/test/proof/demo>
pcepsft43:~/local/root/test/proof/demo>root -l
root [0]
root [0]
root [0] TProof *proof1 = new TProof("localhost")
Starting master: opening connection ...
Starting master: OK
Opening connections to workers: OK (4 workers)
Setting up worker servers: OK (4 workers)
PROOF set to parallel mode (4 workers)
root [1]
root [1]
root [1] gProof == proof1
(int)1
root [2]
```

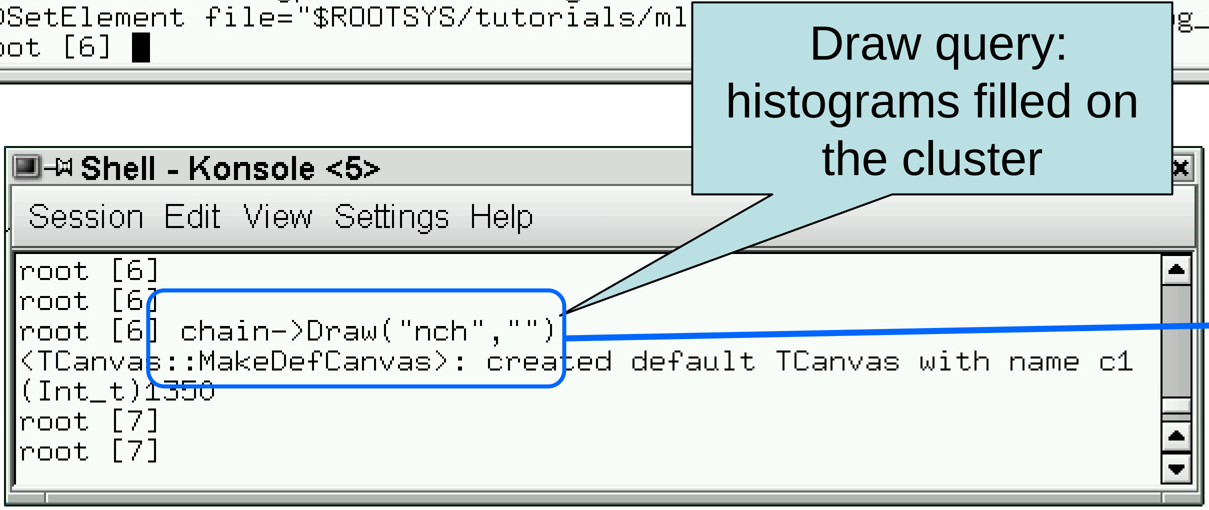


```
root [2]
root [2] proof1->ShowQueries("A")
+++
+++ Queries processed during other sessions: 3
+++ #:1 ref:"session-0-pcepsft43-1127858035-16151:q1" sel:SimpleSel stopped   evts:0-1078
+++ #:2 ref:"session-0-pcepsft43-1127858035-16151:q2" sel:SimpleSel completed evts:0-1349
+++ #:3 ref:"session-0-pcepsft43-1127858689-16282:q2" sel:SimpleSel completed evts:0-1349
+++
+++ Queries processed during this session: selector: 0, draw: 0
+++
root [3]
```

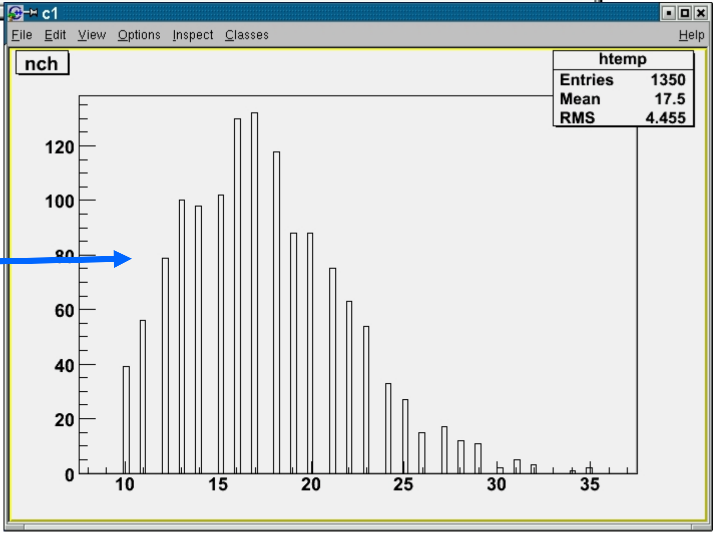
New features at work - 2



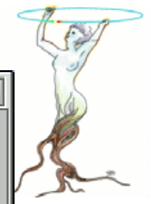
```
root [3]
root [3]
root [3] TDSet* chain = new TDSet("TTree","bg_filtered")
root [4] chain->Add("$ROOTSYS/tutorials/mlpHiggs.root")
(Bool_t)1
root [5] chain->Print("a")
OBJ: TDSet      type TTree      bg_filtered     in /    elements 1
TDSetElement file="$ROOTSYS/tutorials/ml                   g_filtered" first=0 num=-1 msd=""
root [6]
```

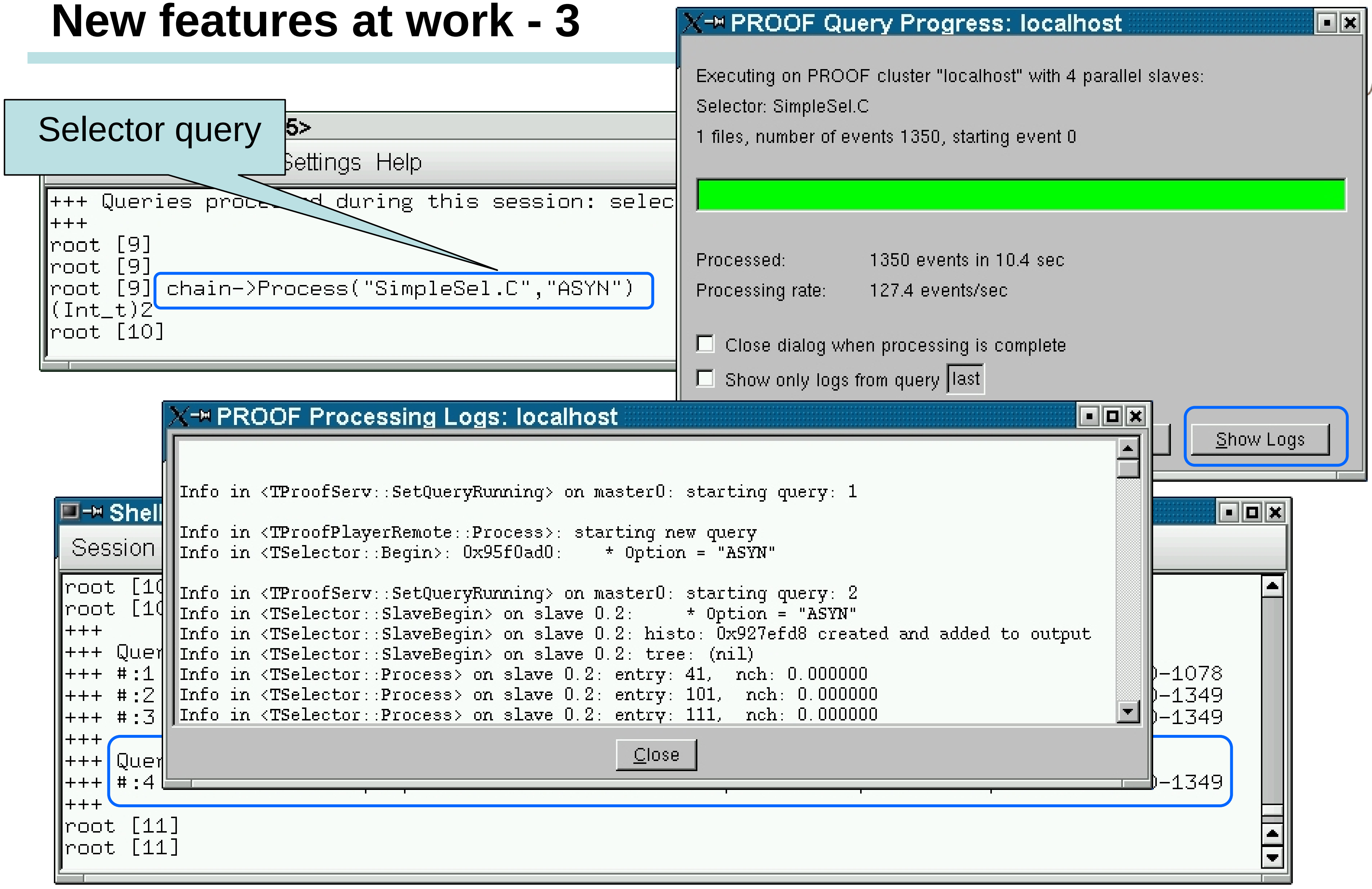

Shell - Konsole <5>
Session Edit View Settings Help
Draw query: histograms filled on the cluster

```
root [6]
root [6]
root [6] chain->Draw("nch","")
<TCanvas::MakeDefCanvas>: created default TCanvas with name c1
(Int_t)1350
root [7]
root [7]
```



New features at work - 3





```
+++ Queries proc    d during this session: selec
+++
root [9]
root [9]
root [9] chain->Process("SimpleSel.C","ASYN")
(Int_t)2
root [10]
```

PROOF Query Progress: localhost

Executing on PROOF cluster "localhost" with 4 parallel slaves:

Selector: SimpleSel.C

1 files, number of events 1350, starting event 0

Processed: 1350 events in 10.4 sec

Processing rate: 127.4 events/sec

☐ Close dialog when processing is complete

☐ Show only logs from query last

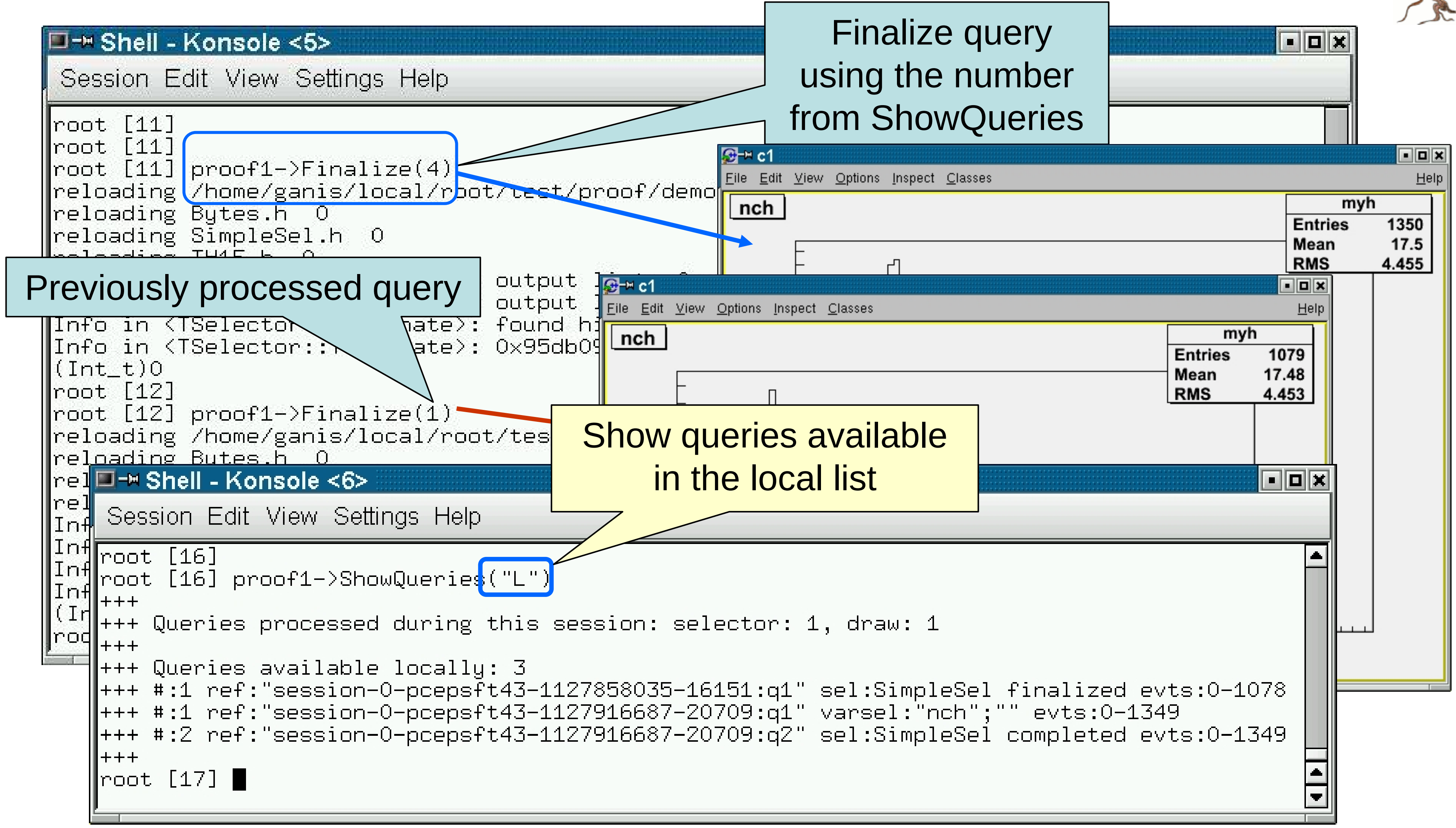
PROOF Processing Logs: localhost

```
Info in <TProofServ::SetQueryRunning> on master0: starting query: 1

Info in <TProofPlayerRemote::Process>: starting new query
Info in <TSelector::Begin>: 0x95f0ad0:    * Option = "ASYN"

Info in <TProofServ::SetQueryRunning> on master0: starting query: 2
Info in <TSelector::SlaveBegin> on slave 0.2:     * Option = "ASYN"
Info in <TSelector::SlaveBegin> on slave 0.2: histo: 0x927efd8 created and added to output
Info in <TSelector::SlaveBegin> on slave 0.2: tree: (nil)
Info in <TSelector::Process> on slave 0.2: entry: 41,  nch: 0.000000
Info in <TSelector::Process> on slave 0.2: entry: 101,  nch: 0.000000
Info in <TSelector::Process> on slave 0.2: entry: 111,  nch: 0.000000
```

Shell

```
root [1(
root [1(
+++
+++ Quer
+++ #:1                                                   )-1078
+++ #:2                                                   )-1349
+++ #:3                                                   )-1349
+++
+++ Quer
+++ #:4                                                   )-1349
+++
root [11]
root [11]
```

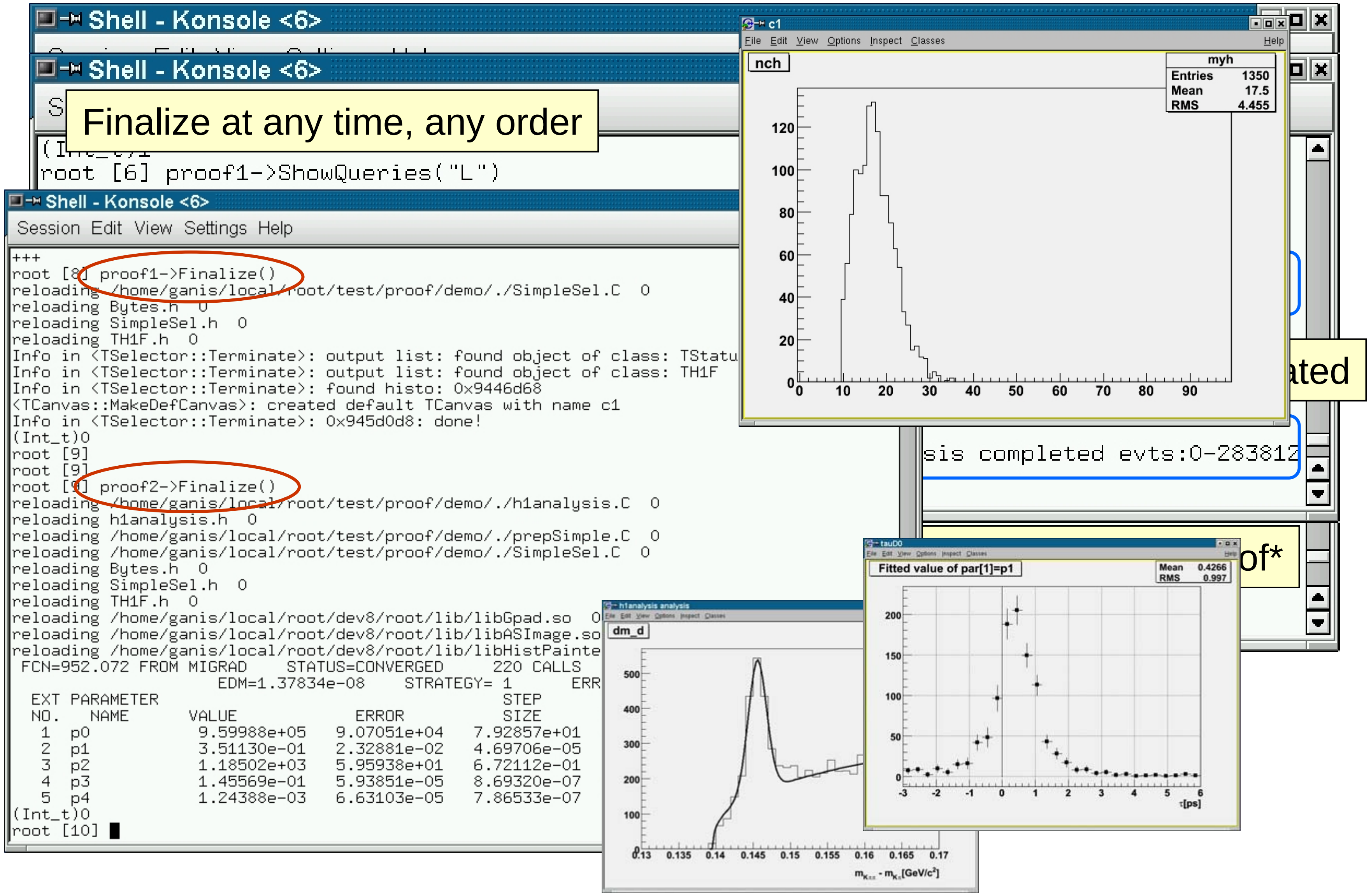
New features at work - 4


Finalize query using the number from ShowQueries
Previously processed query
Show queries available in the local list
Shell - Konsole <5>
Session Edit View Settings Help
c1
File Edit View Options Inspect Classes Help
nch
myh
Entries 1350
Mean 17.5
RMS 4.455
Entries 1079
Mean 17.48
RMS 4.453
Shell - Konsole <6>

```
root [11]
root [11]
root [11] proof1->Finalize(4)
reloading /home/ganis/local/root/test/proof/demo
reloading Bytes.h   0
reloading SimpleSel.h   0
...
Info in <TSelector ...ate>: found hi
Info in <TSelector::...ate>: 0x95db0
(Int_t)0
root [12]
root [12] proof1->Finalize(1)
reloading /home/ganis/local/root/tes
reloading Bytes.h   0
```

```
root [16]
root [16] proof1->ShowQueries("L")
+++
+++ Queries processed during this session: selector: 1, draw: 1
+++
+++ Queries available locally: 3
+++ #:1 ref:"session-0-pcepsft43-1127858035-16151:q1" sel:SimpleSel finalized evts:0-1078
+++ #:1 ref:"session-0-pcepsft43-1127916687-20709:q1" varsel:"nch";"" evts:0-1349
+++ #:2 ref:"session-0-pcepsft43-1127916687-20709:q2" sel:SimpleSel completed evts:0-1349
+++
root [17]
```

New features at work - 5



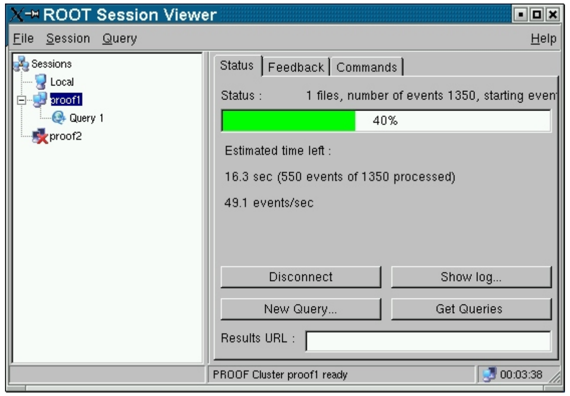
```
(Int_t)1
root [6] proof1->ShowQueries("L")
```

```
+++
root [8] proof1->Finalize()
reloading /home/ganis/local/root/test/proof/demo/./SimpleSel.C  0
reloading Bytes.h  0
reloading SimpleSel.h  0
reloading TH1F.h  0
Info in <TSelector::Terminate>: output list: found object of class: TStatu
Info in <TSelector::Terminate>: output list: found object of class: TH1F
Info in <TSelector::Terminate>: found histo: 0x9446d68
<TCanvas::MakeDefCanvas>: created default TCanvas with name c1
Info in <TSelector::Terminate>: 0x945d0d8: done!
(Int_t)0
root [9]
root [9]
root [9] proof2->Finalize()
reloading /home/ganis/local/root/test/proof/demo/./h1analysis.C  0
reloading h1analysis.h  0
reloading /home/ganis/local/root/test/proof/demo/./prepSimple.C  0
reloading /home/ganis/local/root/test/proof/demo/./SimpleSel.C  0
reloading Bytes.h  0
reloading SimpleSel.h  0
reloading TH1F.h  0
reloading /home/ganis/local/root/dev8/root/lib/libGpad.so  0
reloading /home/ganis/local/root/dev8/root/lib/libASImage.so
reloading /home/ganis/local/root/dev8/root/lib/libHistPainte
 FCN=952.072 FROM MIGRAD    STATUS=CONVERGED     220 CALLS
                     EDM=1.37834e-08    STRATEGY= 1      ERR
  EXT PARAMETER                                STEP
  NO.   NAME      VALUE          ERROR          SIZE
   1  p0           9.59988e+05   9.07051e+04   7.92857e+01
   2  p1           3.51130e-01   2.32881e-02   4.69706e-05
   3  p2           1.18502e+03   5.95938e+01   6.72112e-01
   4  p3           1.45569e-01   5.93851e-05   8.69320e-07
   5  p4           1.24388e-03   6.63103e-05   7.86533e-07
(Int_t)0
root [10] █
```

GUI controller

M.Biskup’s talk + demo

Allows full *on-click* control on everything



Status : 1 files, number of events 1350, starting even

40%

Estimated time left :

16.3 sec (550 events of 1350 processed)

49.1 events/sec

Results URL :

PROOF Cluster proof1 ready

00:03:38

- define a new session
- submit a query, execute a command
- query editor
  - execute macro to define or pick up a **`TChain`**
  - browse directories with selectors
- online monitoring of feedback histograms
- browse folders with results of query
- retrieve, delete, archive functionality
- start viewer for fast **`TChain`** browsing

Improvements

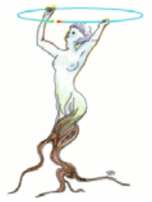
- Session startup
  - Optimized sequential startup
    - separate `proofd` phase from server setup
    - `execv("proofserv")` run in parallel
    - significant speed-up for medium/slow machines
  - Parallel startup
    - Start workers in dedicated threads
    - almost full parallelism
      - protect authentication, lists updates
    - `Proof.ParallelStartup: yes` (on the master)
  - Startup status and progress bar
    - Psychological speed-up …

Improvements (cnt’d)

- New packetizer (**`TPacketizerProgressive`**)
  - optimized data file opening
- Support for **`TTree`** friends in PROOF
- Full draw functionality via the master

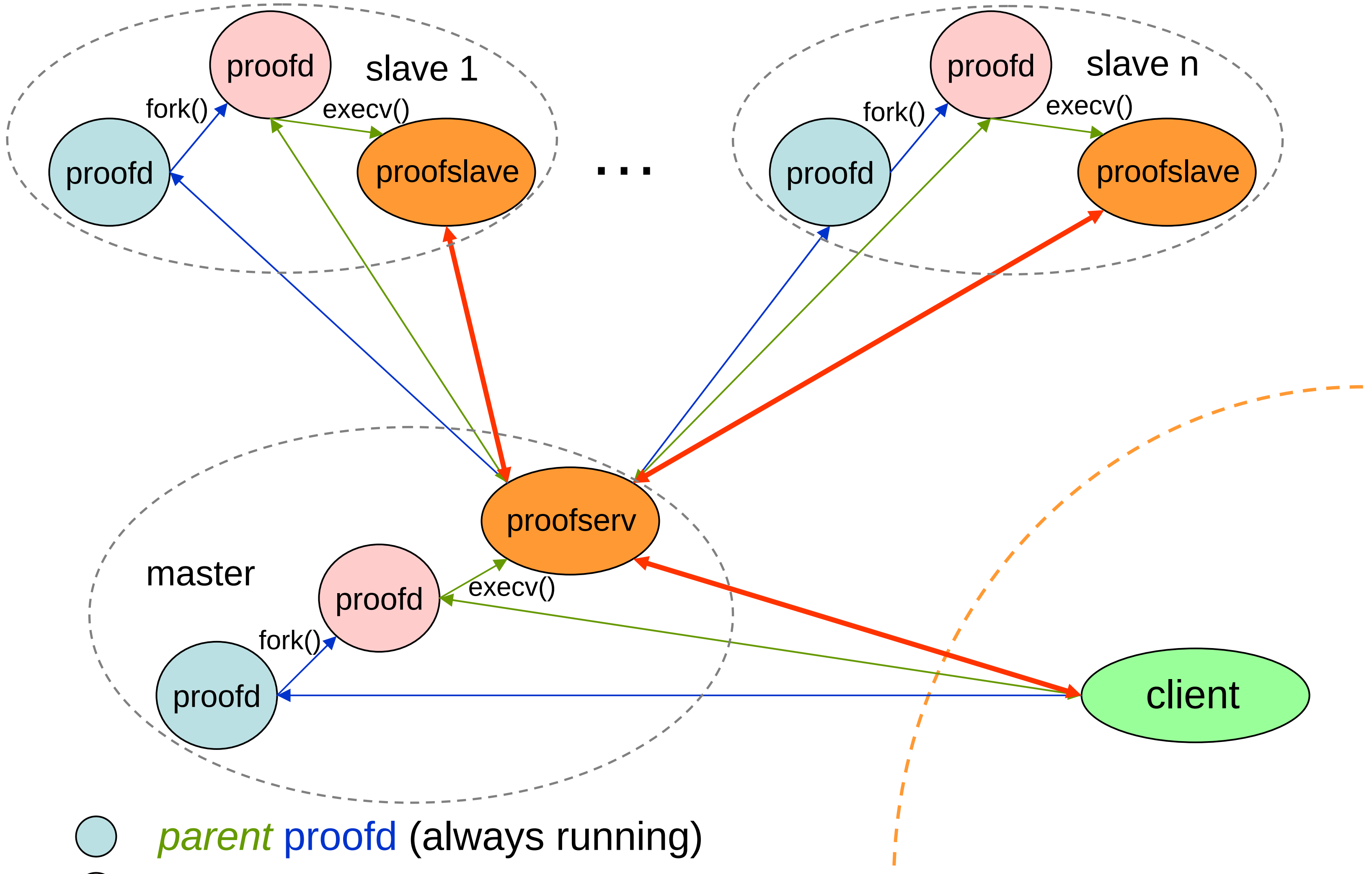
M.Biskup’s talk

Plans



- Consolidate what in place
  - Error handling, fault tolerance
- Complete wiring, tuning of the GUI
- Complete implementation of “interactive batch”
  - Stateless connection (see next)
- Cluster configuration
  - dynamic master-worker setup
  - allow workers come and go
- Documentation

PROOF – Connection layer



● *parent* proofd (always running)

● *child* proofd (executing proofserv / proofslave)

● proofserv / proofslave : `TProofServ` instances (`TApplication`)

Stateless connection

- Introduce a coordinator on the server side able to keep the `proofserv` processes alive when the client goes away
- `proofserv` must be separate processes
  - occasional crashes should not touch other clients

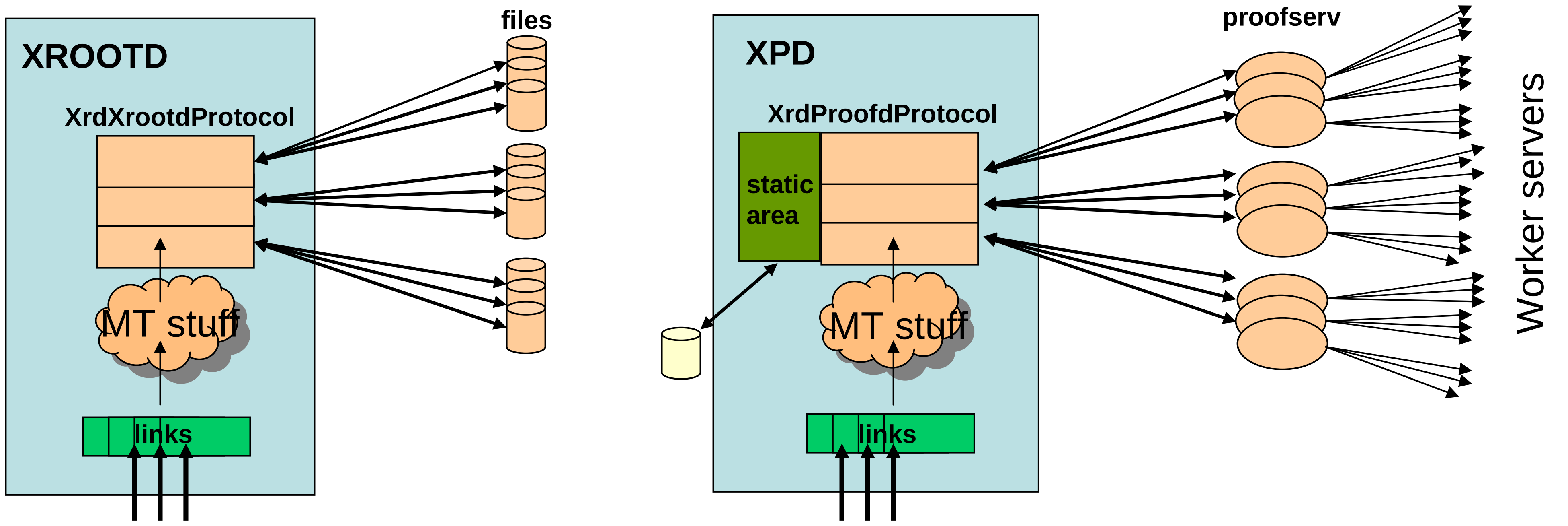
- Main component of `xrootd` (*xrd*) handles networking, security, generic work dispatching

A. Hanushevsky's talk

- Good candidate: already in ROOT, thoroughly tested

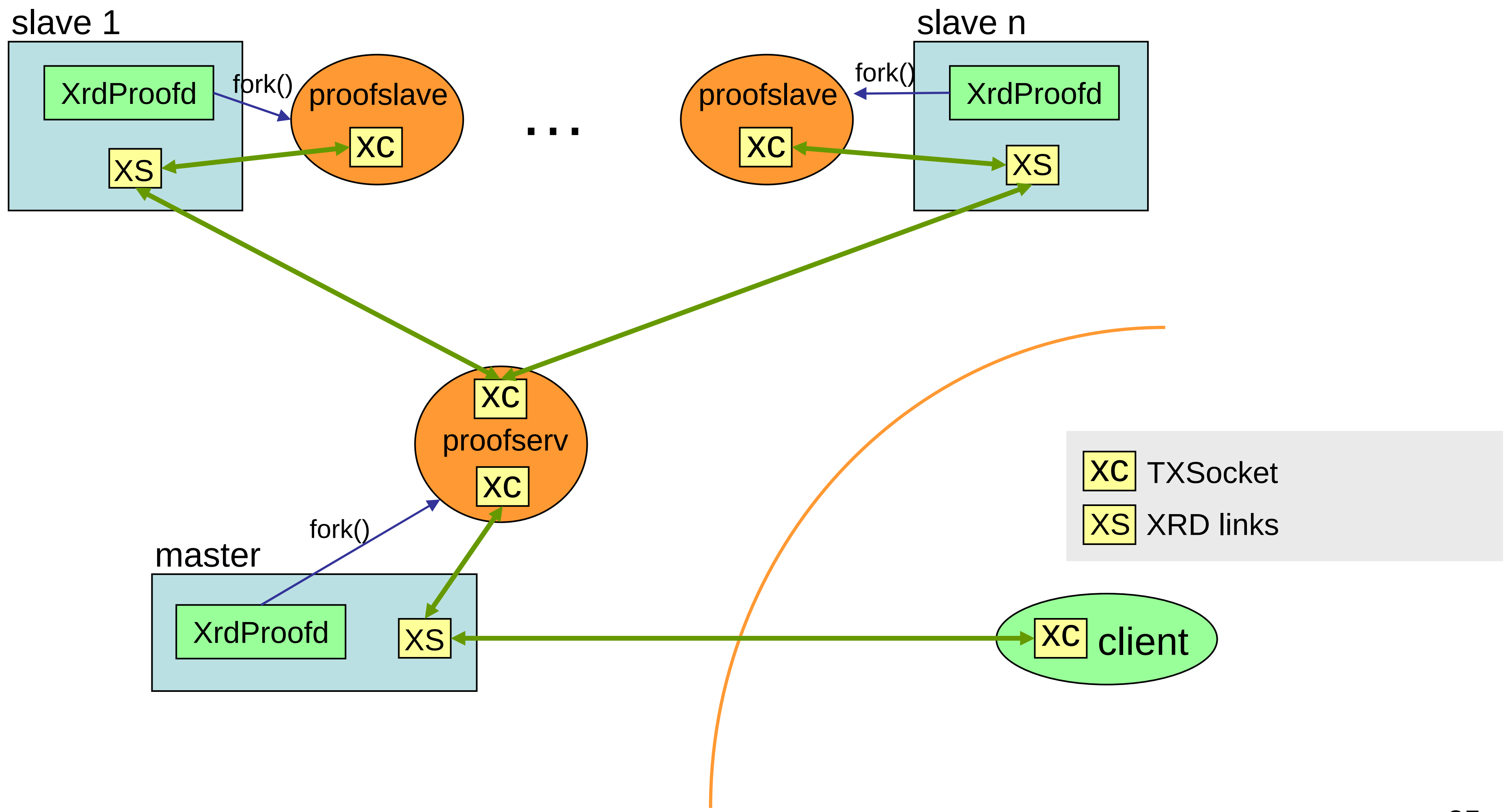
XPD

- Prototype based on XROOTD



- XrdProofdProtocol: client gateway to proofserv
- static area for all client information and its activities

XPD communication layer



XPD remarks

- First impressions for the prototype positive
  - Disconnect / reconnect can be implemented naturally
  - Asynchronous reading allows to setup a control interrupt network independent of OOB
  - Cleaner security system
- Main issue is to understand the impact on performances of the additional layer

Demo

- CERN testbed:
  - 32 dual Pentium III 800 MHz / 512 MB memory
  - 100 MBit Ethernet
  - 600 GB hard disk

- CMS simulation data
  - 20 files, 1.4 GB total

Demo: CMS selector

```
Bool_t TCjets::Process (Long64_t entry)
{
   b_CaloJetCollection_obj_->GetEntry(entry);
   if (CaloJetCollection_obj_ < 11) return kFALSE;
   fChain->GetEntry(entry);

   for (Int_t i = 0; i < CaloTowerCollection_obj_; i++) {

      Float_t e   = CaloTowerCollection_obj_e[i];
      Float_t et  = CaloTowerCollection_obj_et[i];
      Float_t eta = CaloTowerCollection_obj_eta[i];
      fHe  ->Fill(e);
      fHet ->Fill(et);
      fHeta->Fill(eta);
      fHete->Fill(e,et);
   }
   return kTRUE;
}
```

Summary

- Task force working on PROOF work-package since ROOT restructuring
- Several new features introduced
  - “interactive batch” mode
  - GUI controller
- Many other expected during coming months
- Next step is to get the experiments start using PROOF
- Test-bed being setup at CERN
  - experiments welcome to send data samples and complex selectors to better tune the system