



PROOF

Status and Perspectives

G. GANIS
CERN / LCG

VII ROOT Users workshop, CERN, March 2007



Outline



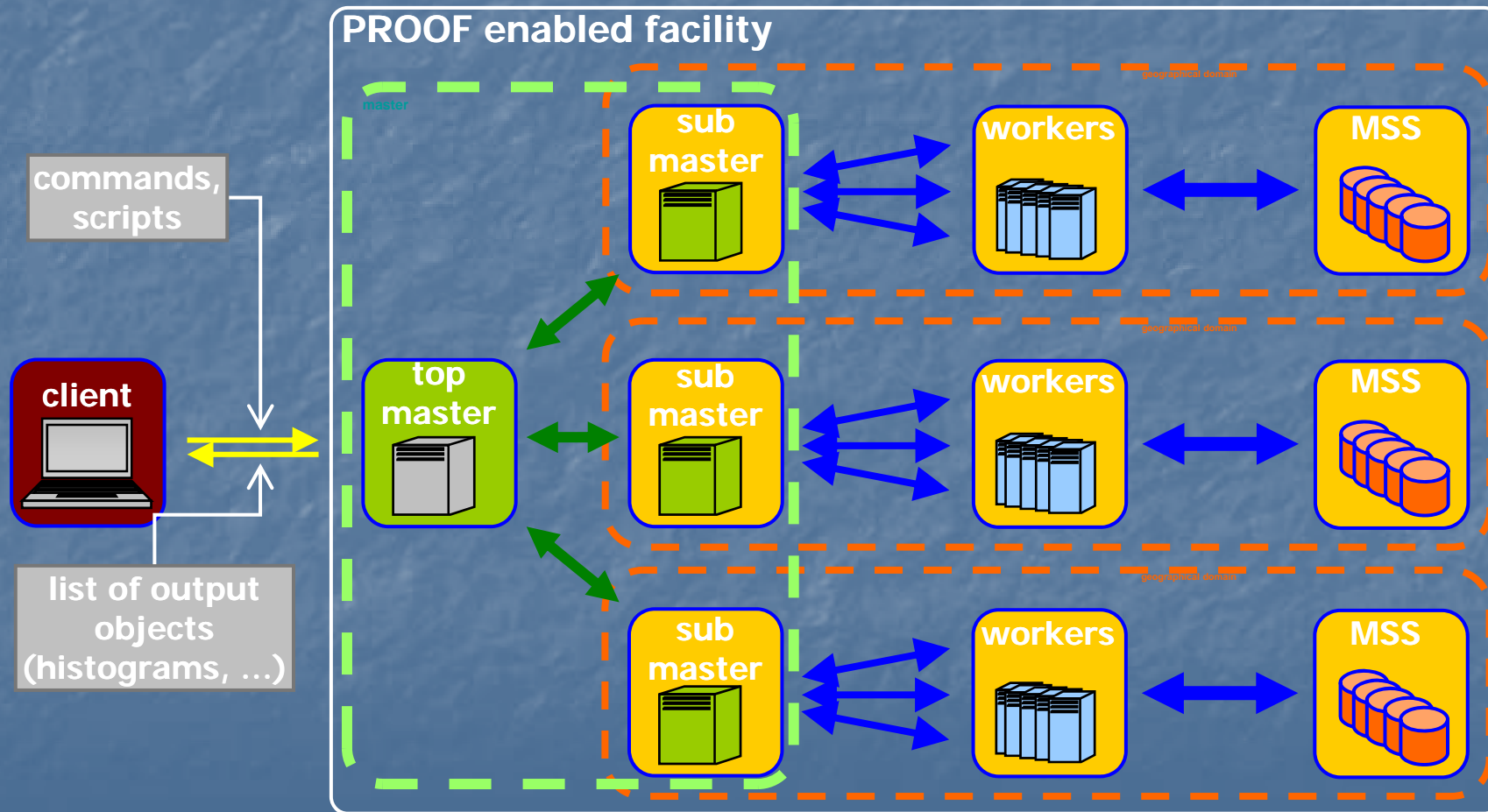
- (Very) quick introduction
- What's new since ROOT05
- Current developments and plans



PROOF in a slide



PROOF: Dynamic approach to end-user HEP analysis on distributed systems exploiting the intrinsic parallelism of HEP data (see Backup slides)





PROOF aspects / issues



- Connection layer
 - Xrootd, Authentication, Error handling
- Software distribution
 - Optimized package / class handling
- Data access
 - Optimized distribution of data on worker nodes
- Classification / handling of the results
 - Query result manager
- Resource sharing among users
 - Client gets one ROOT session on each machine
 - Scheduling



What's new since ROOT05



- Connection layer based on XROOTD
 - Coordinator functionality
 - Full implementation of “interactive batch” model
- Dataset management
- Packetizer improvements
- Progress in uploading / enabling additional software
- Restructuring of the PROOF modules
- Progress in the integration with experiment software
- [PROOF Wiki pages](#)
- ALICE experience at the CAF (see J.F. Grosse-Oetringhaus talk)



Coordinator functionality



- Independent channel to control the cluster
 - Global view
 - Independent access to information (e.g. log files)
 - Needed for full implementation of “interactive batch”
- Not directly achievable with proofd
 - Daemon instance “disappearing” into proofserv
 - Session lifetime same as client connection lifetime
 - Parent proofd not aware of childrens
- Natural candidate: XROOTD
 - Light weight, industrial strength, networking and protocol handler



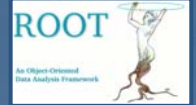
New connection layer based on XROOTD



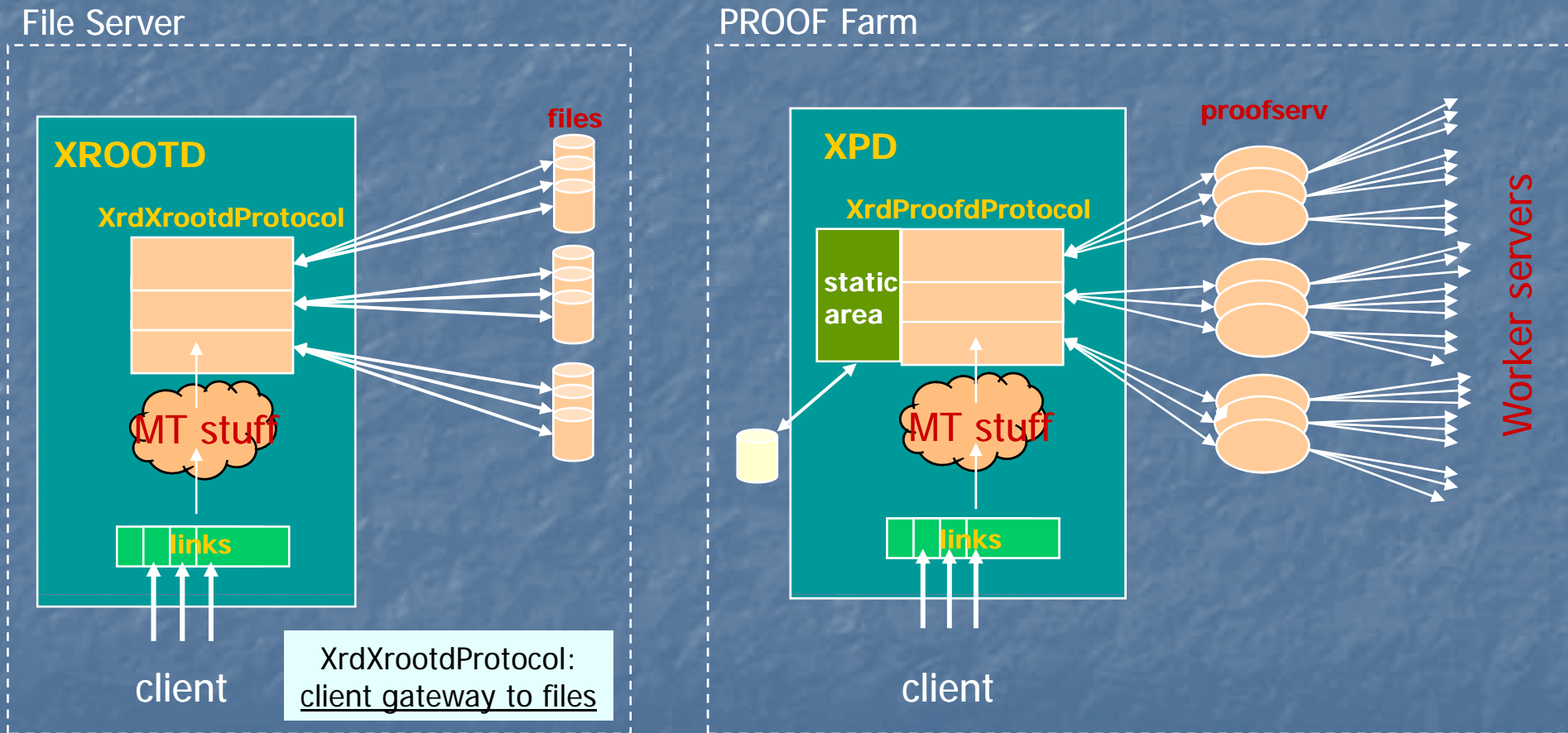
- New PROOF-related protocol:
 - XrdProofdProtocol (XPD)
 - XPD launches and controls PROOF sessions ([proofserv](#))
- Client connection (XrdProofConn) based on XrdClient
 - Concept of physical (per client) / logical (per session) connection
 - Asynchronous reading via dedicated thread
 - Messages read as soon as available and added to a queue
 - setup a **control interrupt network independent of OOB**
 - **Cleaner security system**
 - Physical connection authenticated
 - Associated logical connections inherit the "token"
- **Client disconnection / reconnection handled naturally**



XPD role

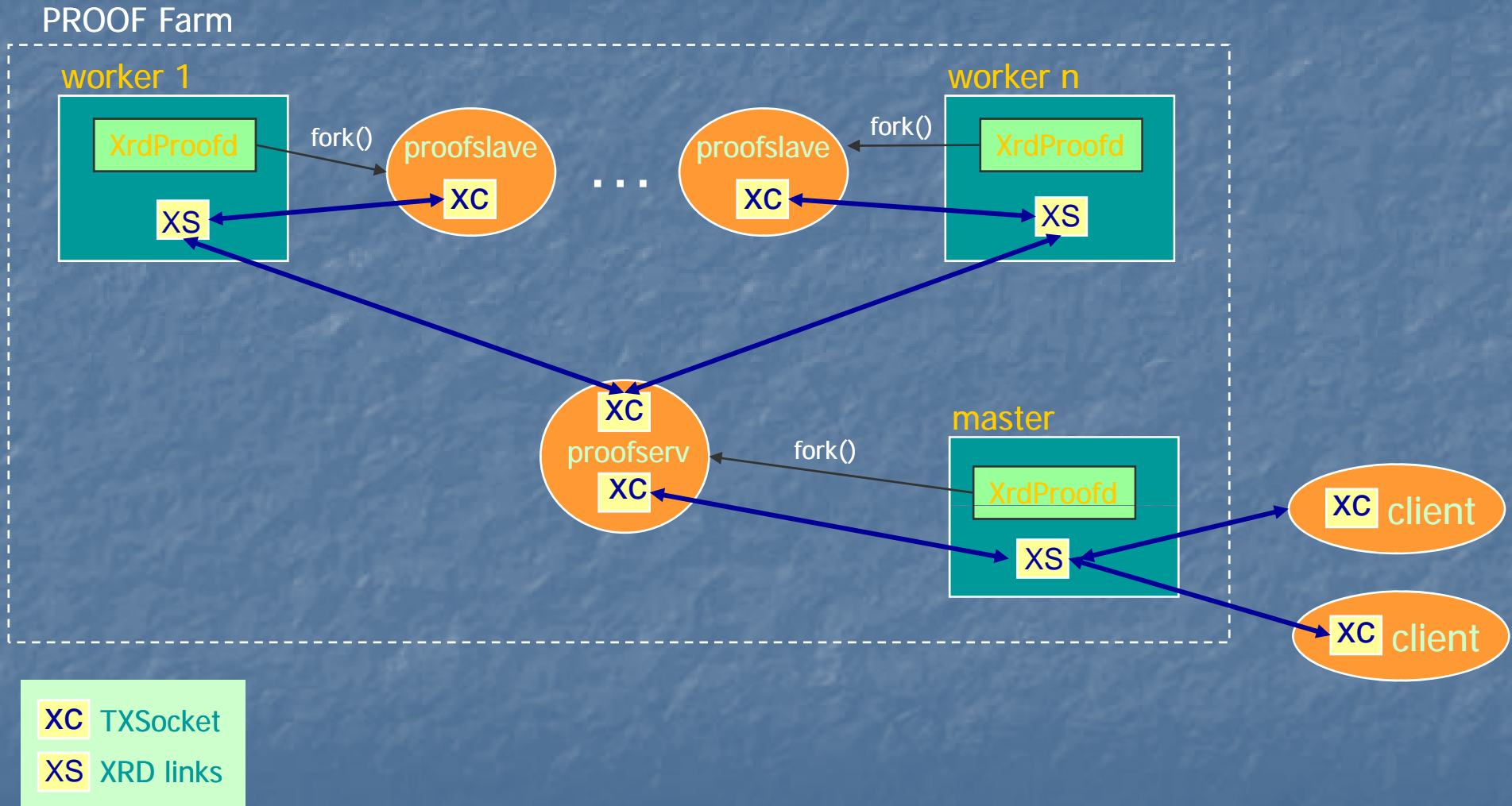


■ XrdProofdProtocol: client gateway to proofserv





XPD communication layer





Stateless connection and “Interactive batch”



- “Interactive batch”: flexible submission system keeping advantages of interactivity and batch
 - If a query is taking too long have the option to abort it, to stop and retrieve the results, or to leave it running on the system coming back later on to browse / retrieve / archive the results
- Ingredients
 - Non-blocking running mode (✓ v5.04.00, ROOT05)
 - Query result management (✓ v5.04.00, ROOT05)
 - Stateless client connection (✓ v5.08.00)
 - Ctrl-Z functionality (soon)



Exploiting the coordinator: client side



- Not yet fully exploited:
 - new functionality added regularly
- Examples:
 - Log retrieval

```
root [] TProofLog *pl = TProof::Mgr("user@master")->GetSessionLogs()  
root [] pl->Grep("violation")
```

- **TProofLog** contains log files as TMacro and implements display, grep, save, ... functionality

- Session reset

```
TProof::Reset("user@master")
```

- Cleanup of user's entry in the coordinator
 - Only way-out when something bad happen



Exploiting the coordinator: server side



- Static control of resource usage
 - Max number of users
 - Max number of workers per user
- Access, usage control
 - Role of server
 - List of users allowed to connect
- Define ROOT versions available on the cluster
 - Extendable to packages
- ...



Dataset uploader



- Optimized distribution of data files on the farm using XROOTD functionality
 - By direct upload
 - By staging out from mass storage
- Direct upload
 - Sources: local directory, list of URLs
 - XROOTD/OLBD pool insures optimal distribution
 - No special configuration (except for clean-up)
- Using a stager
 - Requires XROOTD configuration
 - e.g. CASTOR for ALICE @ CAF



Dataset manager



- Data-sets are **identified by name**

```
root [0] TProof *proof = TProof::Open("master");
root [1] proof->UploadDataSet("MCppH", "/data1/mc/ppH_*");
Uploading file:///data1/mc/ppH_01.root to \
  root://poolurl//poolpath/ppH_01.root
[TFile::Cp] Total 20.34 MB |=====| 100.00 % [6.9 MB/s]

root [2] proof->ShowDataSets();
Existing Datasets:
MCppH
```

- Data-sets can be retrieved by name to automatically create TDSet's

```
root [] TDSet *dset = new TDSet(proof->GetDataSet("MCppH"));
```




Dataset manager



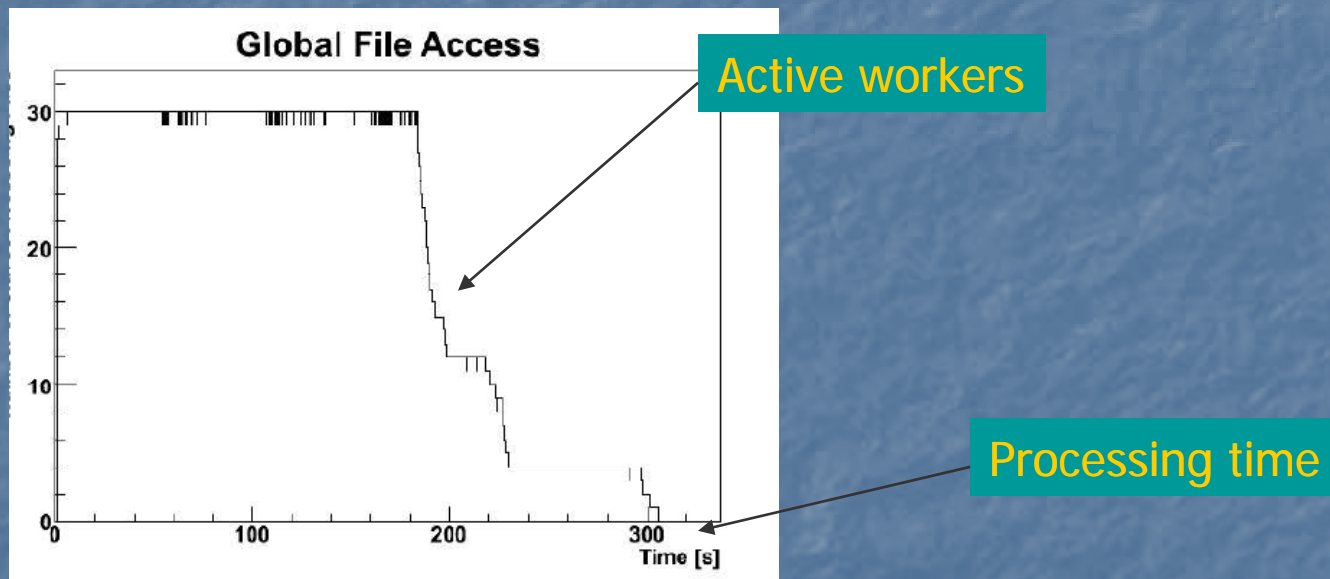
- Metadata stored in sandbox on the master
 - New sub-directory <SandBox>/dataset
- Concept of **private** / **public** data-sets
 - **User's private definitions**
 - readable / writable by owner only
 - **User's public definitions**
 - readable by anybody
 - **Global public definitions**
 - Workgroup- / experiment-wide (e.g. 2008 runs)
 - readable by anybody (group restrictions?)
 - writable by privileged account



Packetizer improvements

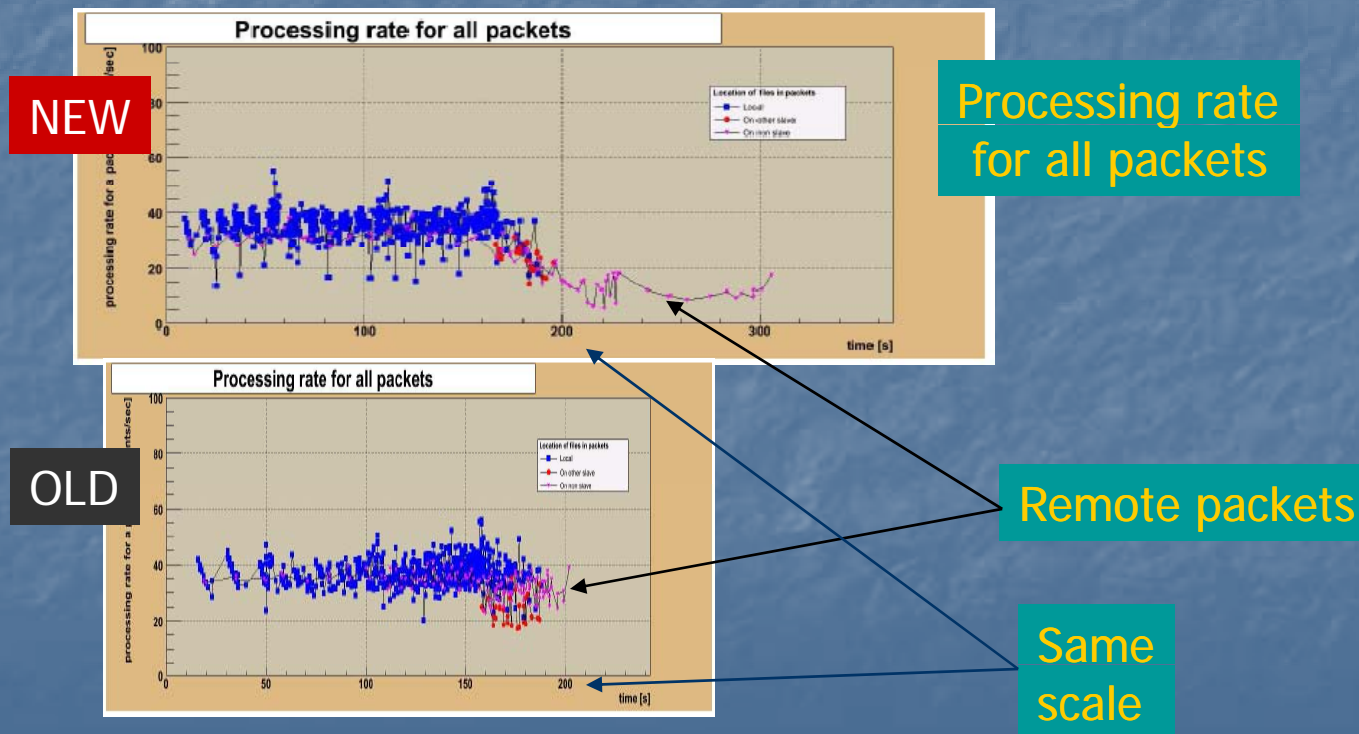


- Packetizer's goal: optimize work distribution to process queries as fast as possible
- Standard TPacketizer's strategy
 - first process local files, than try to process remote data



- End-of-query bottleneck

- Predict processing time of local files for each worker
- **Keep assigning remote files from start of the query to workers expected to finish faster**
- Processing time **improved by up to 50%**





Progress in using additional software



- Package enabling
 - Separated behaviour client / cluster
 - Real-time feedback during build
- Load mechanism extended to single class / macro

```
root [] TProof *proof = TProof::Open("master")  
root [] proof->Load("MyClass.C")
```

- Selectors / macros / classes binaries are now cached
 - Decreases initialization time
- API to modify include / library paths on the workers
 - Use packages globally available on the cluster



Restructuring of PROOF modules



- Reduce dependencies
- Better control size of executables (proofserv)
 - Faster worker startup
- First step:
 - Get rid of TVirtualProof and PROOF dependencies in 'tree'
 - All PROOF in 'proof', 'proofx', 'proofd'
 - Still 'proofserv' needs a lot of libs
- 2nd step (current situation):
 - Separate out TProofPlayer, TPacketizer, ... in 'proofplayer' (new libProofPlayer, v5.15.04)
 - proofserv size **on workers** reduced by a factor of ~2 at startup



Further optimization of PROOF libs



- Differentiate setups on client and cluster
 - Client:
 - Needs graphics
 - May not need all experiment software
 - TSelector: compile only Begin() and Terminate()
 - Servers:
 - Need all experiment software
 - Do not need graphics
 - TSelector: do not compile Begin() and Terminate()
 - Client and Server versions of basic libs



Additional improvements (incomplete)

- GUI controller
 - Integration of the data set manager
 - Integration of the new features of package manager
 - Improved session / query history bookkeeping
- Improved user-friendliness of parameter setting

```
root [] TProof *proof = TProof::Open("master")  
root [] proof->SetParameter("factor", 1.1)
```

- Automatic support dynamic environment setting
 - proofserv is a script launching proofserv.exe
 - Envs to define the context in which to run
 - Useful for experiment specific settings (see later) and/or for debugging purposes (e.g. run valgrind on worker ...)



Integration with experiment software



- Finding, using the experiment software
 - Environment settings, libraries loading
- Implementing the analysis algorithms
 - TSelector framework
 - Structured analysis and automated interaction with trees (chains) (+)
 - Tightly coupled with the tree (-)
 - New analysis implies new selector
 - Change in the tree definition implies a new selector
 - May conflict with existing experiment technologies
 - Add new layer to hide details irrelevant for the end-user



Setting the environment

- Experiment software available on nodes
- Additional dedicated software handled by the PROOF package manager
 - Allows user to run her/his own modifications
- The experiment environment can be set
 - **Statically** (e.g. ALICE)
 - before starting xrootd (inherited by proofserv)
 - **Dynamically** (e.g. CMS)
 - evaluating a user defined script in front of proofserv
 - Allows to select different versions at run time



Dynamic environment setting: CMS



- CMS needs to run SCRAM before proofserv
- PROOF_INITCMD contains the path of a script (**NEW**)

```
TProof::AddEnvVar("PROOF_INITCMD",  
    "~maartenb/proj/cms/CMSSW_1_1_1/setup_proof.sh")
```

- The script initializes the CMS environment using SCRAM

```
#!/bin/sh  
  
# Export the architecture  
export SCRAM_ARCH=slc3_ia32_gcc323  
  
# Init CMS defaults  
cd ~maartenb/proj/cms/CMSSW_1_1_1  
./app/cms/cmsset_default.sh  
  
# Init runtime environment  
scramv1 runtime -sh > /tmp/dummy  
cat /tmp/dummy
```



Examples of implementing analysis algorithms



- ALICE:
 - Generic AliSelector hiding details
 - User's selector derives from AliSelector
 - Access to ESD event by member fESD
 - Alternative technology using tasks
 - See J.F. Grosse-Oetringhaus talk
- TAM technology @ PHOBOS
 - Based on modularized tasks
 - Separate analysis tasks from interaction with tree
 - See C. Reed at ROOT05



Analysis algorithms in CMS



- CMSSW: provides EDAnalyzer for analysis
- Algorithms with **a well defined interface** can be used with both technologies (EDAnalyzer and TSelector)

```
class MyAnalysisAlgorithm {  
    void process( const edm::Event & );  
    void postProcess( TList & );  
    void terminate( TList & );  
};
```

- Used in a **TSelector templated framework TFWLiteSelector**

```
TSelector *mySel = new TFWLiteSelector<MyAnalysisAlgorithm>
```

- Selector libraries distributed as PAR file

```
// Load framework library  
gSystem->Load("libFWCoreFWLite");  
// Load TSelector library  
gSystem->Load("libPhysicsToolsParallelAnalysis");
```




Current developments and plans



- **Scheduling**
- Consolidation, error handling
 - Improved but still cases when we lose control of the session
- Processing error report
 - Associate to a query an object detailing what went wrong (e.g. data set elements not analyzed) and why
- Non-input-file-driven based analysis
 - Current processing is based on tree or object files
- Local multi-core desktop optimization
 - No daemons, UNIX sockets (no master?)
- GUI: integration in a more general GUI ROOT controller

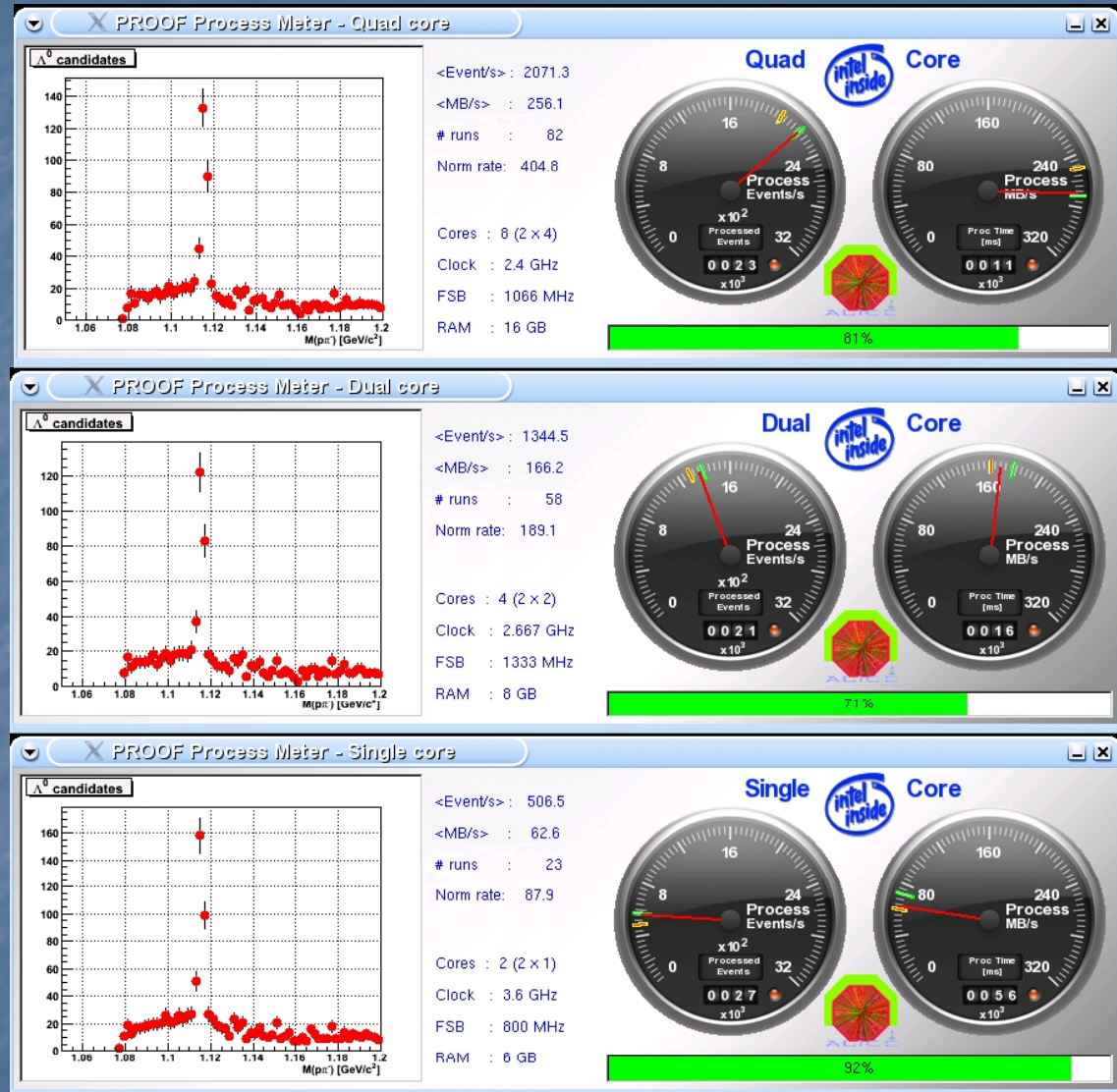


PROOF exploiting multi-cores



- Alice search for Λ^0 's
- 4 GB simulated data
- Instantaneous rates (evt/s, MB/s)
- Clear advantage of quad core

Additional computing Power fully exploited





PROOF: scheduling multi-users



- Fair resource sharing
 - System scheduler not enough if $N_{\text{users}} \geq \sim N_{\text{workers}} / 2$
- Enforce priority policies
- Two approaches
 - Quota-based worker level load balancing
 - Simple and solid implementation, no central unit
 - Group quotas defined in the configuration file
 - Central scheduler
 - Per-query decisions based on cluster load, resources need by the query, user history and priorities
 - Generic interface to external schedulers planned
 - MAUI, LSF, ...



Quota-based worker level load balancing



- Lower priority processes slowdown
 - sleep before next packet request
- Sleeping time proportional to the used CPU time
 - factor depends on # users and the quotas
- Example: userA, quota 2/3; userB, quota 1/3
 - After T seconds:
 - CPU(A) = T/2, CPU(B) = T/2
 - Sleep B form T/2 seconds
 - After T + T/2 seconds
 - CPU(A) = T/2 + T/2 = 2 * CPU(B) = T/2
- General case of N users brings a tri-diagonal linear system



Quota-based worker level load balancing



- Group quotas defined in the xrootd configuration file

```
xpd.group tpc usra,usrb  
xpd.grpparam tpc quota:70%
```

- Factors recalculated by the master XPD each time that a user start or ends processing
 - Only active users considered
 - A low priority user will get 100% of resources when alone
- Under linux processes SCHER_RR system scheduling enforced
 - The default, dynamic, SCHED_OTHER scheme screws up the all idea, as sleeping processes get higher priority at restart



Demo



- Same sample analysis (h1 slightly slowed-down) repeated for 20 times
- 2 users
 - gganis: reserved quota 70%
 - ganis: taking what left
- Histogram show processing rate in MB/s



Demo



The screenshot displays the ROOT analysis environment. It features two histogram windows and two progress dialog boxes.

Top Histogram: Titled "h1analysis analysis". The x-axis ranges from 0 to 20, and the y-axis from 0 to 7. A sharp peak is visible at approximately x=5. A performance table in the top right corner shows:

perf	
Entries	17
Mean	4.775
RMS	0.96

Bottom Histogram: Titled "h1analysis analysis <2>". The x-axis ranges from 0 to 20, and the y-axis from 0 to 4. It shows a distribution with several peaks between x=2 and x=6. A performance table in the top right corner shows:

perf	
Entries	9
Mean	2.594
RMS	0.8349

Progress Dialogs: Two "PROOF Query Progress" windows are shown. The top one indicates 4 files, 283813 events, and a processing rate of 81312.2 evts/sec. The bottom one indicates 4 files, 283813 events, and a processing rate of 34903.9 evts/sec. Both windows include a green progress bar and buttons for "Stop", "Cancel", "Close", "Show Logs", and "Rate plot".

Terminal Window: A terminal window at the bottom shows system messages including "killroot: line 34: kill: (9819) - no such process" and "a \$./startRun.sh gganis".



Central scheduling



- Entity running on master XPD, loaded as plug-in
 - Abstract interface XrdProofSched defined

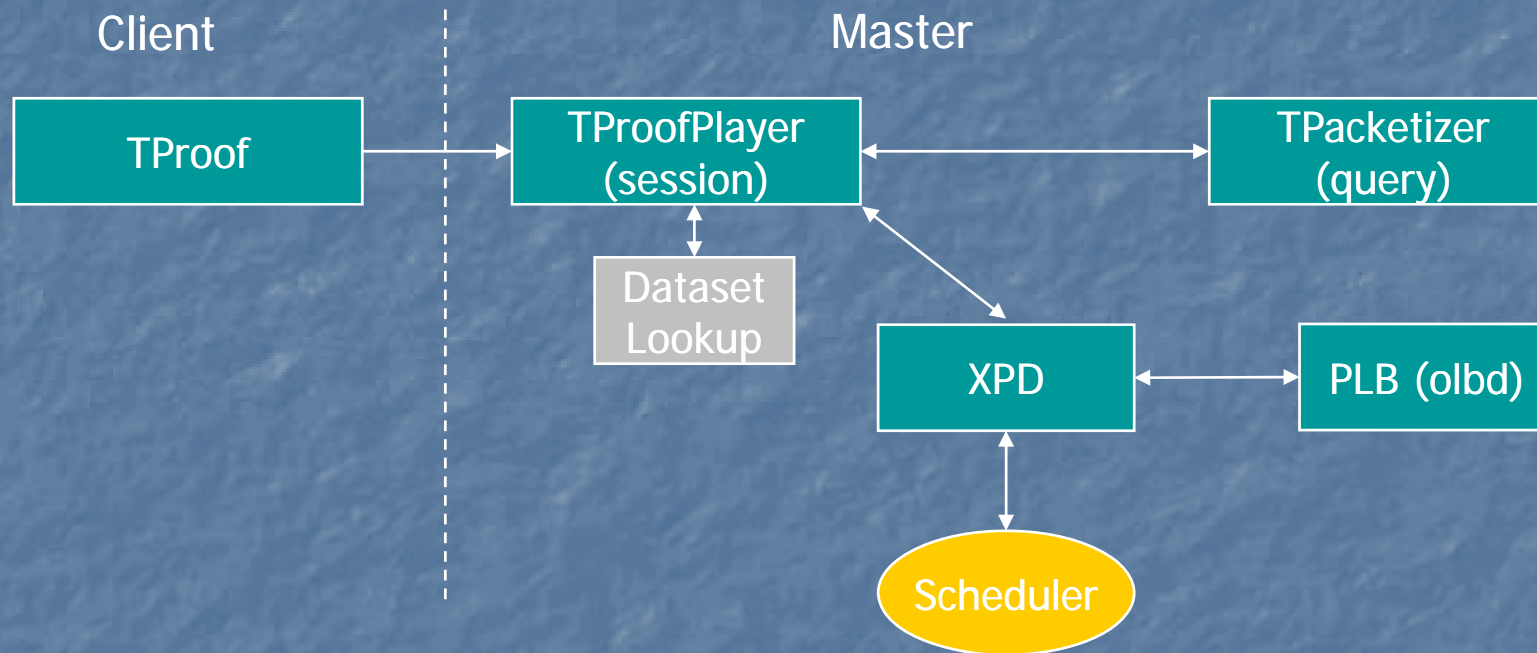
```
class XrdProofSched {  
    ...  
public:  
    virtual int GetWorkers(XrdproofServProxy *xps,  
                          std::list<XrdProofWorker *> &wrks) = 0;  
    ...  
};
```

- Input:
 - Query info (via XrdProofServProxy ->proofserv)
 - Cluster status via OLBD control network
 - Policy
- Output:
 - List of workers to continue with



Central scheduling

■ Schematic view



■ Needed ingredients:

- Full exploitation of the OLBD network
- Come&Go functionality for workers
- ...



Summary



- Several improvements in PROOF since ROOT05
 - Coordinator functionality
 - Data set manager
 - Resource control
- ALICE is stress testing the system in LHC environment using a test-CAF at CERN
 - a lot of useful feedback
- Efforts now concentrated on
 - Further consolidation and optimization
 - Scheduling
- PROOF is steadily improving: getting ready for LHC data



Credits



- PROOF team
 - M. Ballintijn, B. Bellenot, L. Franco, G.G., J. Iwaszkiewicz, F. Rademakers
- J.F. Grosse-Oetringhaus, A. Peters (ALICE)
- A. Hanushevsky (SLAC)



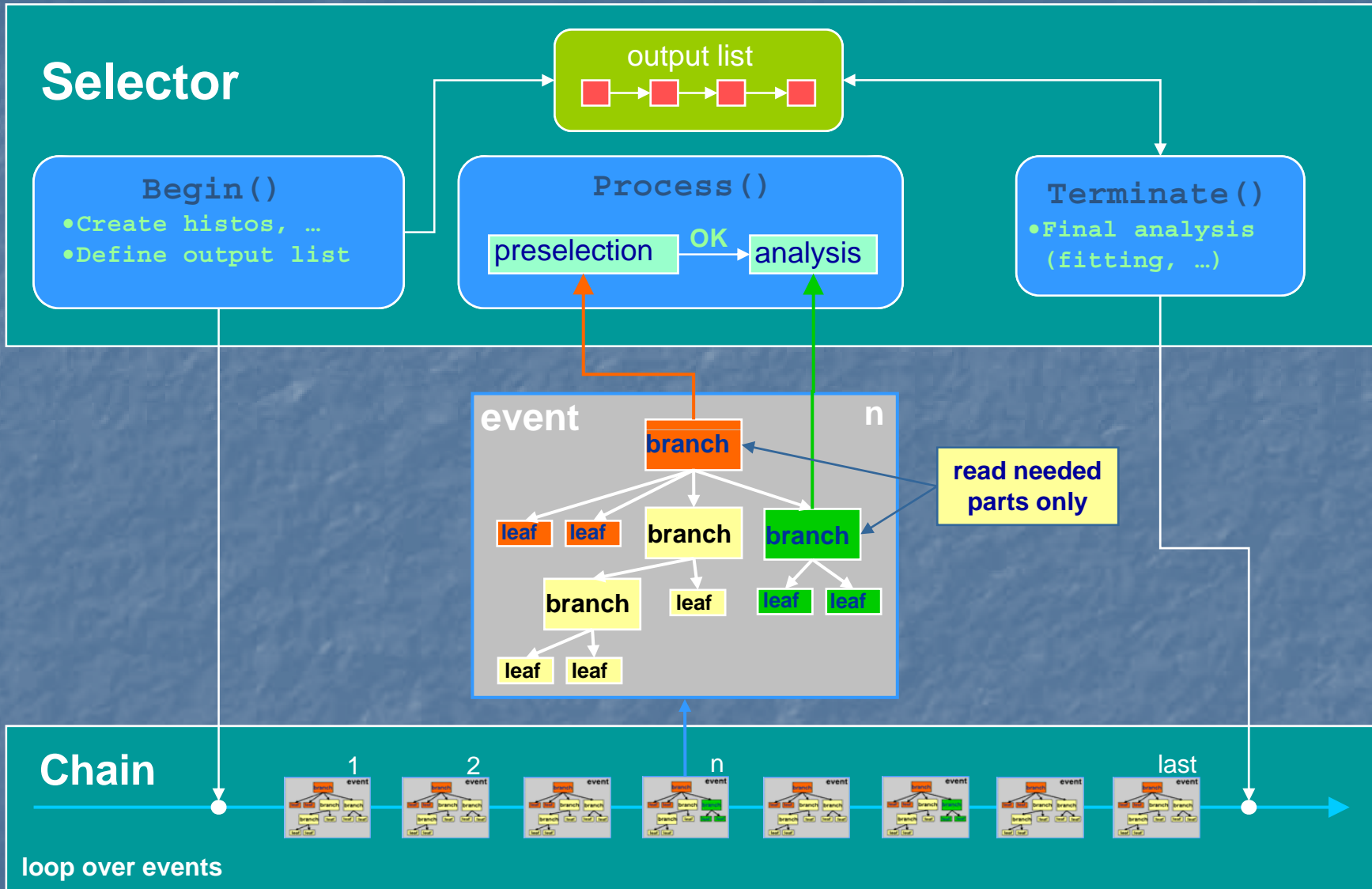
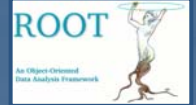
Backup



- See also presentations at previous ROOT workshops and at CHEPxx

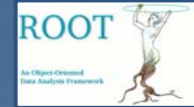


The ROOT data model: Trees & Selectors

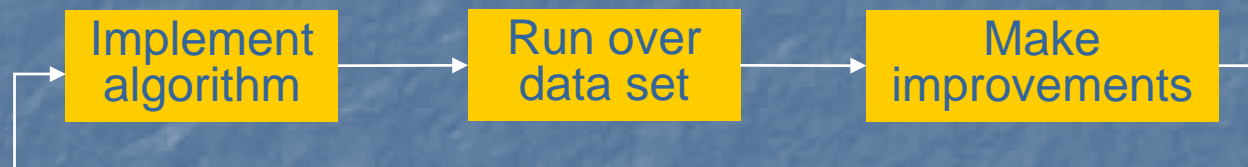




Motivation for PROOF



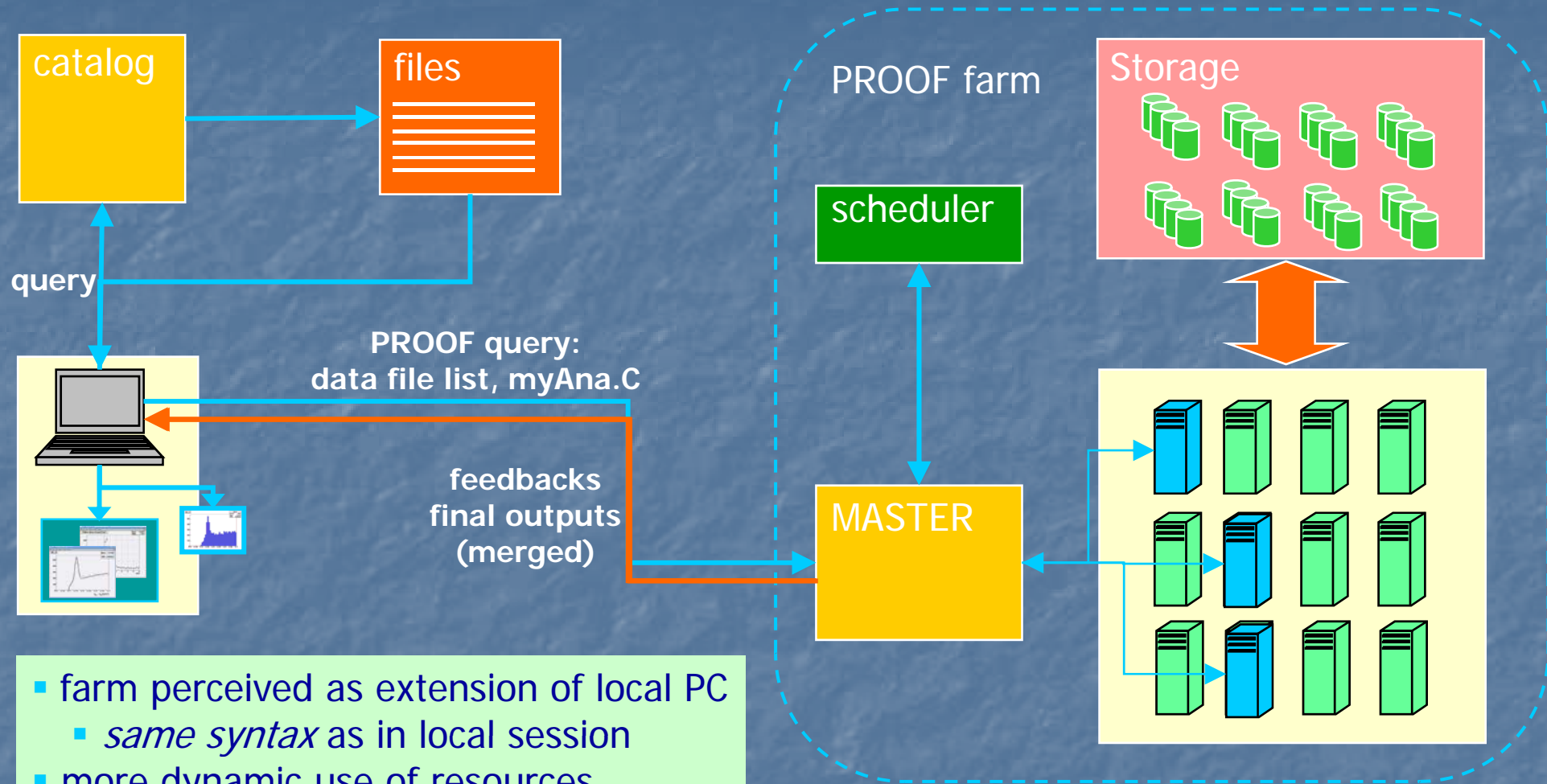
- Provide an alternative, dynamic, approach to end-user HEP analysis on distributed systems
- Typical HEP analysis is a continuous refinement cycle



- Data sets are collections of independent events
 - Large (e.g. ALICE ESD+AOD: ~350 TB / year)
 - Spread over many disks and mass storage systems
- Exploiting intrinsic parallelism is the only way to analyze the data in reasonable times



The PROOF approach



- farm perceived as extension of local PC
 - *same syntax* as in local session
- more dynamic use of resources
- real time feedback
- automated splitting and merging



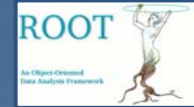
PROOF design goals



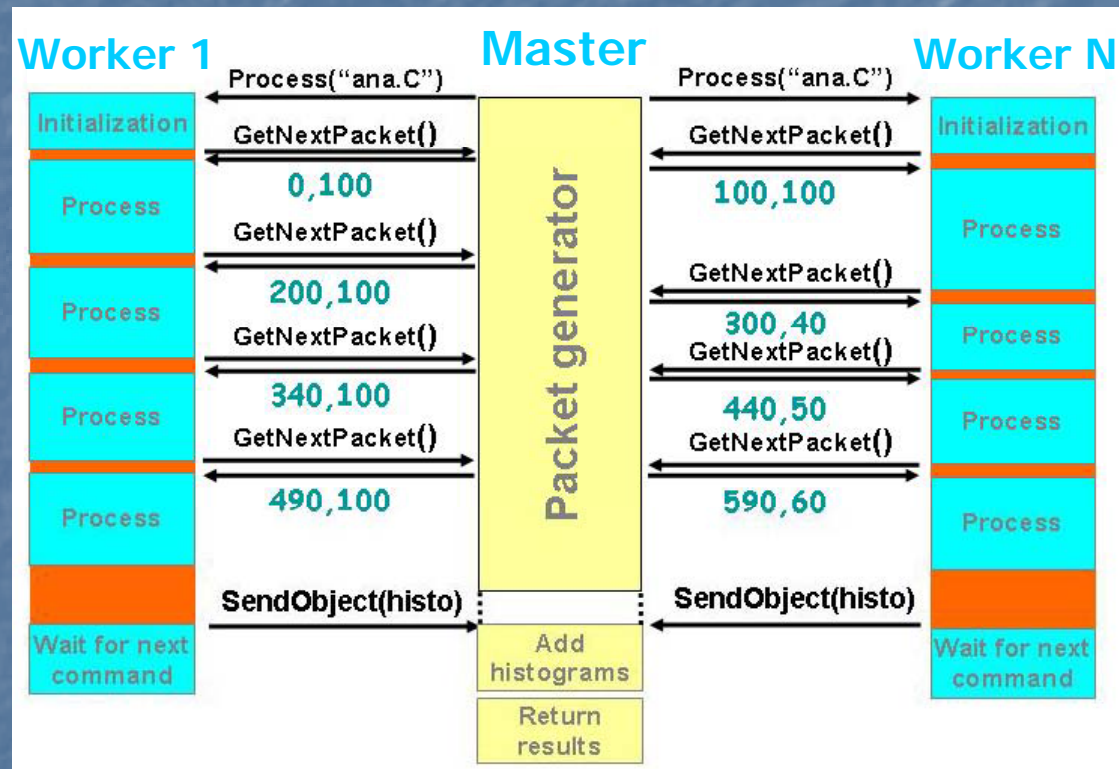
- Transparency
 - Minimal impact on the ROOT user habits
- Scalability
 - Full exploitation of the available resources
- Adaptability
 - Cope transparently with heterogeneous environments
- Preserve Real-time interaction and feedback
- Intended for
 - Central Analysis Facilities
 - Departmental workgroup computing facilities (Tier-2's)
 - Multi-core / multi-disk desktops



PROOF dynamic load balancing



- Pull architecture guarantees scalability



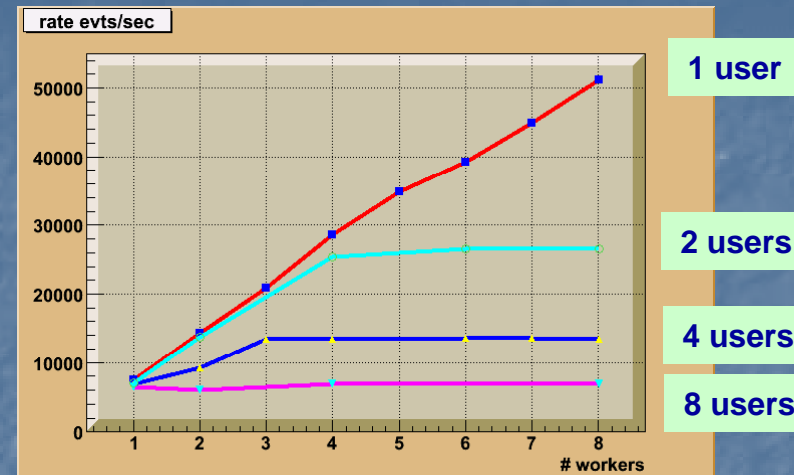
- Adapts to variations in performance



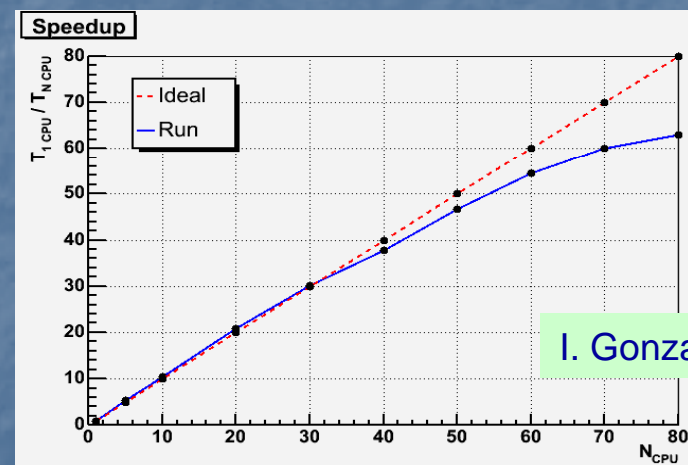
PROOF intrinsic scalability



- Strictly concurrent user jobs at CAF (100% CPU used)
- In-memory data
- Dual Xeon, 2.8 GHz



- CMS analysis
- 1 master, 80 workers
- Dual Xeon 3.2 GHz
- Local data: 1.4 GB / node
- Non-Blocking GB Ethernet



I. Gonzales, Cantabria



PROOF essentials: what can be done?



- Ideally everything made of independent tasks
- Currently available:
 - Processing of trees
 - Processing of independent objects in a file
- Tree processing and drawing functionality complete

LOCAL

```
// Create a chain of trees
root[0] TChain *c = CreateMyChain.C;

// MySelec is a TSelector
root[1] c->Process("MySelec.C+");
```

PROOF

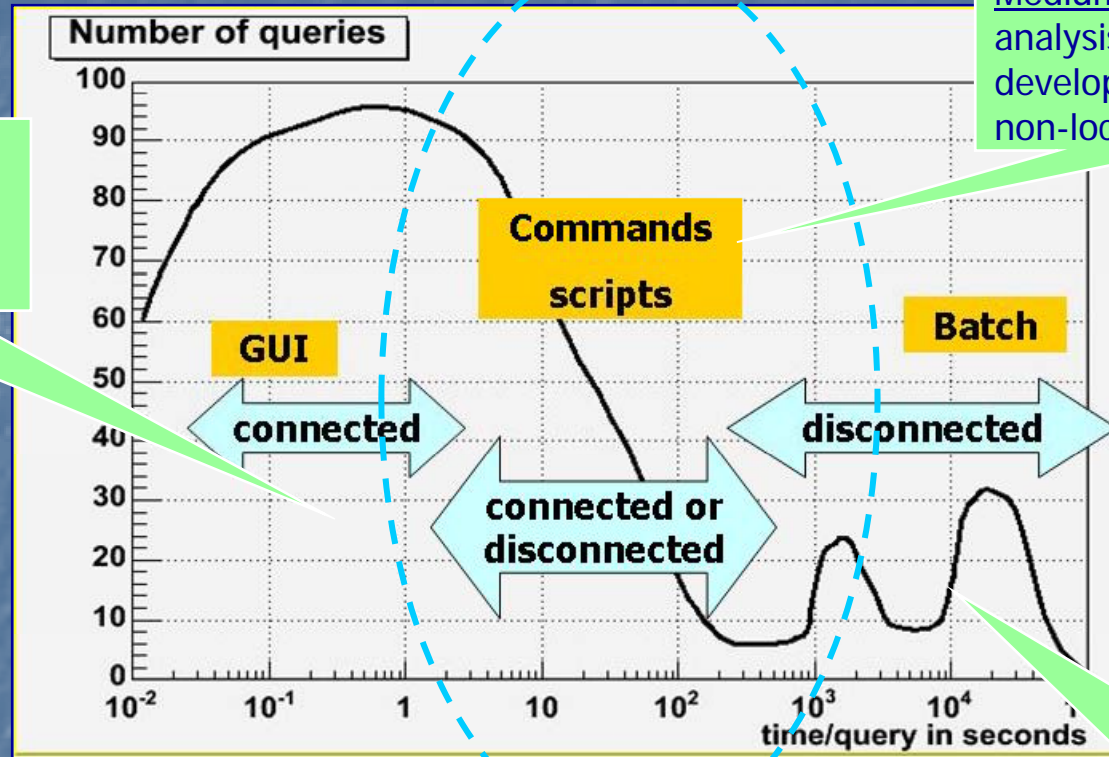
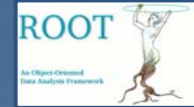
```
// Create a chain of trees
root[0] TChain *c = CreateMyChain.C;

// Start PROOF and tell the chain
// to use it
root[1] TProof::Open("masterURL");
root[2] c->SetProof();

// Process goes via PROOF
root[3] c->Process("MySelec.C+");
```



The PROOF target



Medium term jobs, e.g. analysis design and development using also non-local resources

Short analysis using local resources, e.g.
- end-analysis calculations
- visualization

Long analysis jobs with well defined algorithms (e.g. production of personal trees)

- Optimize response for short / medium jobs
- Perceive medium as short



PROOF: additional remarks



- Intrinsic serial overhead small
 - requires reasonable connection between a (sub-)master and its workers
- Hardware considerations
 - IO bound analysis (frequent in HEP) often limited by hard drive access: N small disks are much better than 1 big one
 - Good amount of RAM for efficient data caching
- Data access is The Issue:
 - Optimize for data locality, when possible
 - Low-latency access to mass storage



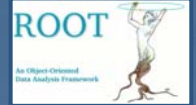
PROOF: data access issues



- Low latency in data access is essential for high performance
 - Not only a PROOF issue
- File opening overhead
 - Minimized using asynchronous open techniques
- Data retrieval
 - caching, pre-fetching of data segments to be analyzed
 - Recently introduced in ROOT for TTree
- Techniques improving network performance, e.g. InfiniBand, or file access (e.g. memory-based file serving, PetaCache) should be evaluated



PROOF: PAR archive files



- Allow client to add software to be used in the analysis
- Simple structure
 - *package/*
 - Source / binary files
 - *package/PROOF-INF/BUILD.sh*
 - How to build the package (makefile)
 - *package/PROOF-INF/SETUP.C*
 - How to enable the package (load, dependencies)
- A PAR is a gzip'ed tar-ball of the *package* tree
- Versioning support being added



PROOF essentials: monitoring



- Internal
 - File access rates, packet latencies, processing time, etc.
 - Basic set of histograms available at tunable frequency
 - Client temporary output objects can also be retrieved
 - Possibility of detailed tree for further analysis
- MonALISA-based
 - Each host reports
 - CPU, memory, swap, network
 - Each worker reports
 - CPU, memory, evt/s, IO vs. network rate
 - pcalimonitor.cern.ch:8889

Network traffic between nodes

Machine	6047	6048	6049	6050	6052	6053	6054	6055	6056	6057	6058	6059	6060	6061	6062	6063	6064	6065	6066	6067	6068	6069	607
1. 6047	0	-	-	-	-	-	2.927	2.018	-	-	1.094	-	-	-	1.908	4.112	-	-	0.974	0.614	0	0	
2. 6048	-	9.406	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
3. 6049	-	-	8.678	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
4. 6050	-	-	-	6.692	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
5. 6052	-	-	-	-	3.913	-	1.454	-	-	-	-	3.084	-	0.317	0	0	-	0	-	-	0.985	4.447	
6. 6053	-	-	-	-	-	6.603	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
7. 6054	0	-	-	1.363	-	-	6.195	-	-	-	0	-	-	-	0	-	-	-	-	-	0	-	1.50
8. 6055	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
9. 6056	-	-	-	-	-	-	-	4.962	-	2.442	0.525	-	-	-	-	-	-	-	-	-	-	-	-
10. 6057	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
11. 6058	1.164	-	-	-	0	-	-	2.531	-	0	0	-	-	-	-	1.103	-	0	-	-	-	-	-
12. 6059	3.755	-	0.622	-	-	-	-	-	-	-	11.76	1.955	0	0.677	1.848	0	-	-	-	-	2.812	-	0.70
13. 6060	-	-	-	-	-	-	-	2.068	-	-	-	11.69	-	-	1.06	-	-	-	-	-	-	-	2.00
14. 6061	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15. 6062	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
16. 6063	-	-	-	-	1.655	0.27	2.416	-	-	-	-	-	0	-	6.38	-	0	-	0	-	-	-	-
17. 6064	-	-	-	-	-	1.123	-	2.822	-	-	-	-	1.621	-	0	-	3.117	-	0	0	-	-	0.50
18. 6065	0	-	-	-	3.52	3.165	-	0	-	0	-	-	-	-	-	3.034	0	1.579	-	0	-	-	-
19. 6066	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-



PROOF GUI controller



- Allows full *on-click* control
- define a new session
- submit a query, execute a command
- query editor
 - create / pick up a chain
 - choose selectors
- online monitoring of feedback histograms
- browse folders with results of query
- retrieve, delete, archive functionality