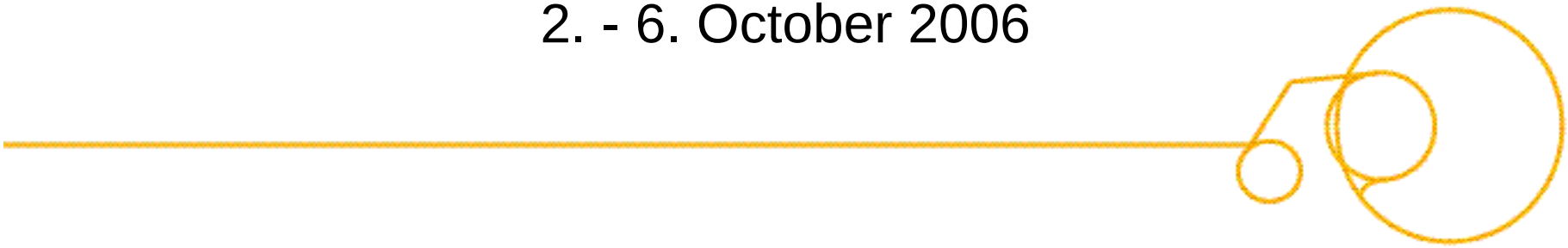
Analysis using PROOF on the CERN Analysis Facility

Jan Fiete Grosse-Oetringhaus

ALICE Offline Week
2. - 6. October 2006



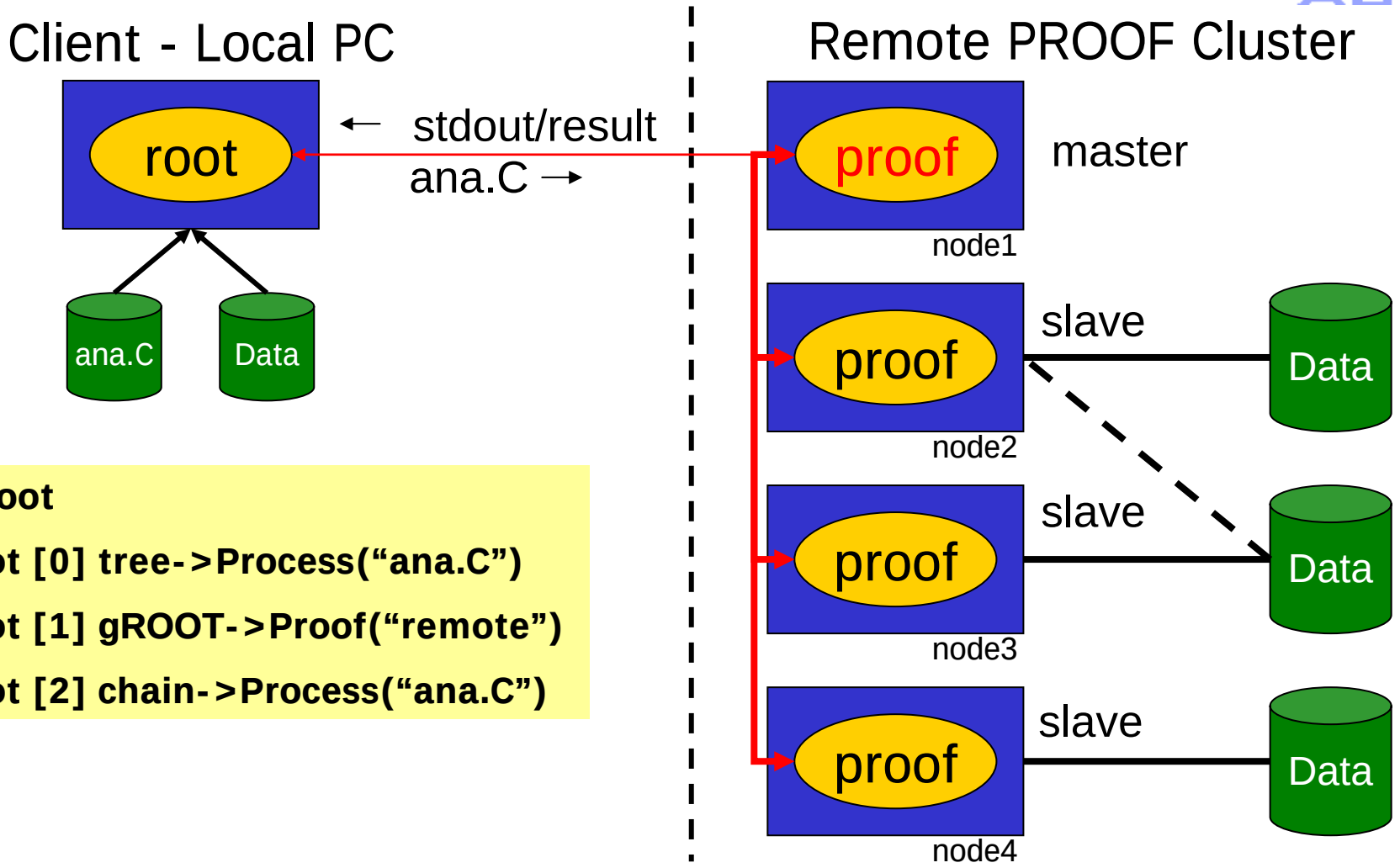
Content

- Theory part
  - Quick “repetition” about PROOF, CAF
  - Evaluation of the CAF test cluster
- Practical part (including demo)
  - Access ESD files
  - Access data via the RunLoader (needs full AliRoot)

PROOF

- **P**arallel **ROO**T **F**acility
- Interactive parallel analysis on a local cluster
- PROOF itself is not related to Grid
  - Accesses files that are produced on the Grid
- The usage of PROOF is transparent
  - The same code can be run locally and in a PROOF system (certain rules have to be followed)
- PROOF is part of ROOT

PROOF Schema




```
$ root
root [0] tree->Process("ana.C")
root [1] gROOT->Proof("remote")
root [2] chain->Process("ana.C")
```

CERN Analysis Facility

- The **C**ERN **A**nalysis **F**acility (CAF) will run PROOF for ALICE
  - Prompt analysis of pp data
  - Pilot analysis of PbPb data
  - Calibration & Alignment
- Available to the whole collaboration but the number of users will be limited for efficiency reasons
- Design goals
  - 500 CPUs
  - 100 TB of selected data locally available

Evaluation of PROOF

- Test setup since May 2006
  - 40 machines, 2 CPUs each, 200 GB disk
- Tests performed
  - Usability tests
  - Simple speedup plot
  - Evaluation of different query types
  - Evaluation of the system when running a combination of query types
- Goal: Realistic simulation of users using the system

Bugs Status

- 29 bugs closed since start of CAF tests
- 27 open bugs
  - 1 blocker
  - 10 important
  - 12 normal
  - 4 minor
- Main open issues
  - No log output when your selector crashed

Query Types

- A realistic stress test consists of different users that submit different types of queries

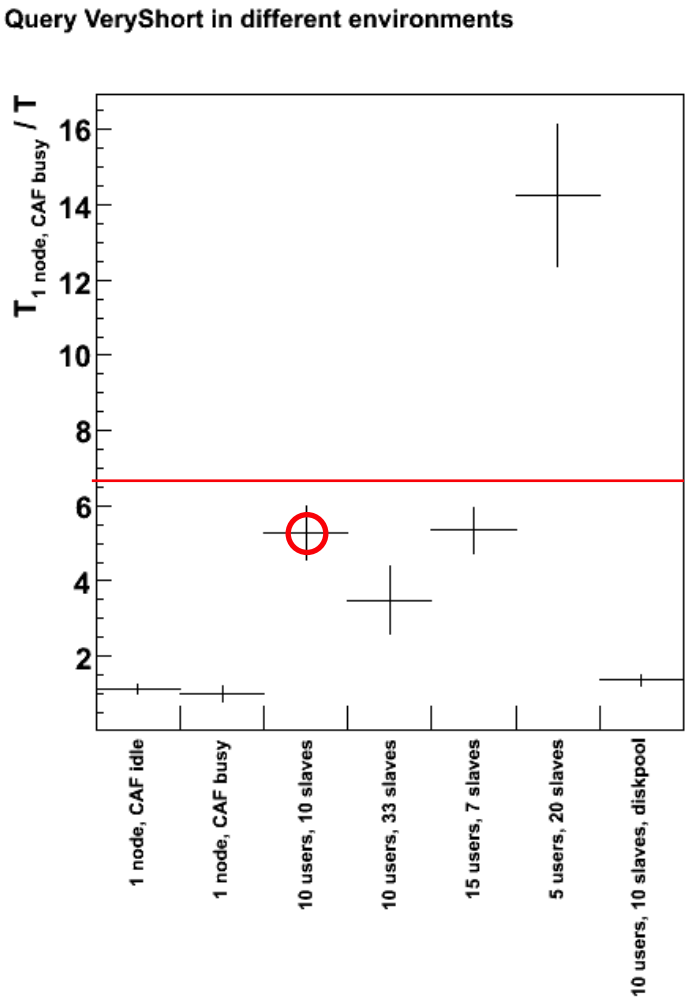
Name	# files	# evts	processed data	avg. time*	Submission Interval
VeryShort	20	2K	0.4 GB	9 ± 1 s	30 ± 15 s
Short	20	40K	8 GB	150 ± 10 s	120 ± 30 s
Medium	150	300K	60 GB	1,380 ± 60 s	300 ± 120 s
Long	500	1M	200 GB	4,500 ± 200 s	600 ± 120 s

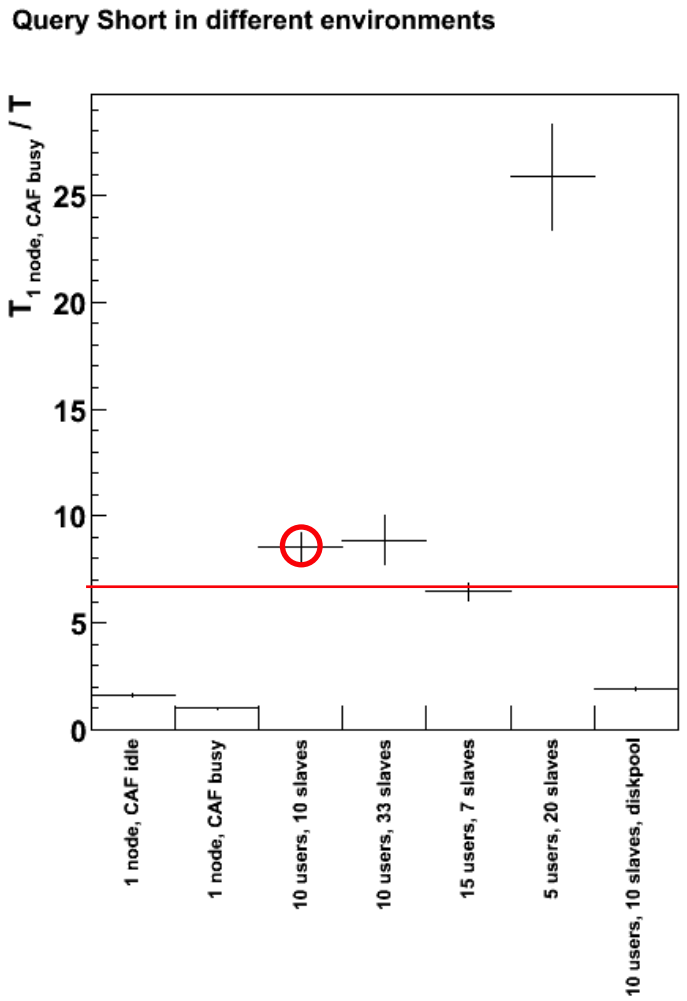
*run in PROOF, 10 users, 10 PROOFservs each

Query Type Cocktail

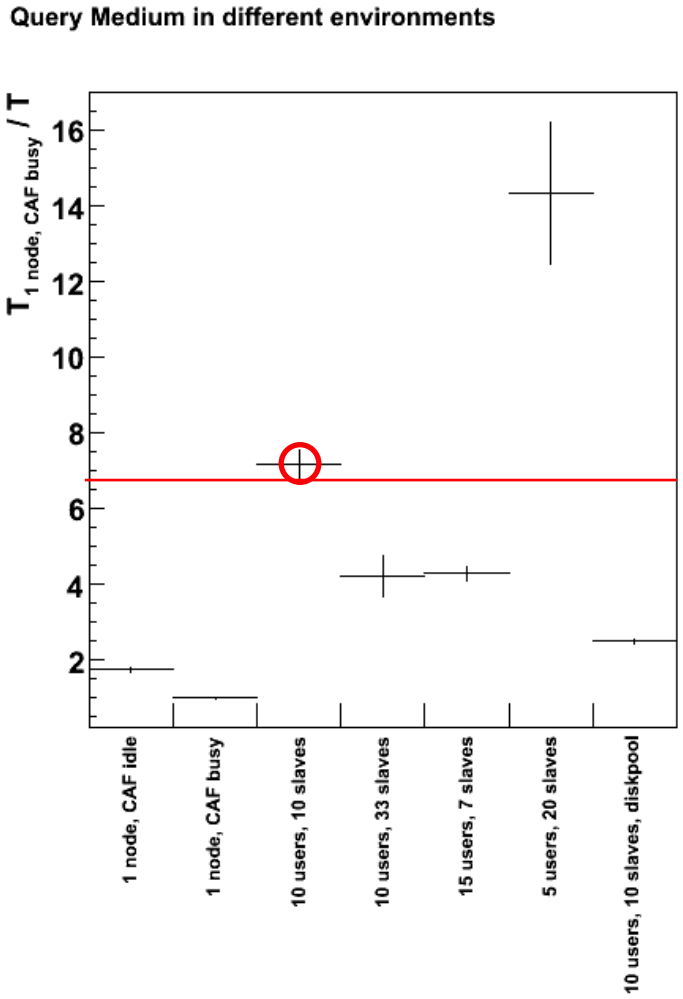
- 4 different query types
  - 20% very short queries
  - 40% short queries
  - 20% medium queries
  - 20% long queries
- User mix
  - 33 nodes available for the test
    - 10 users, 10 or 30 processes per user
    - 5 users, 20 processes per user
    - 15 users, 7 processes per user
  - Maximum average speedup for 10 users = 6.6 (33 nodes = 66 CPUs)

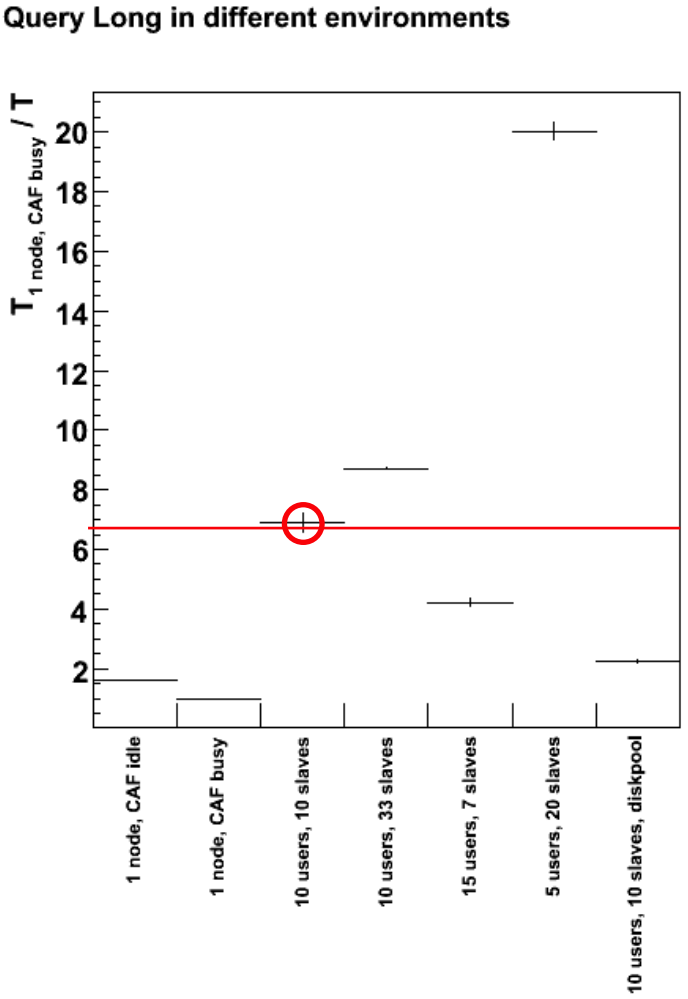
Relative Speedup





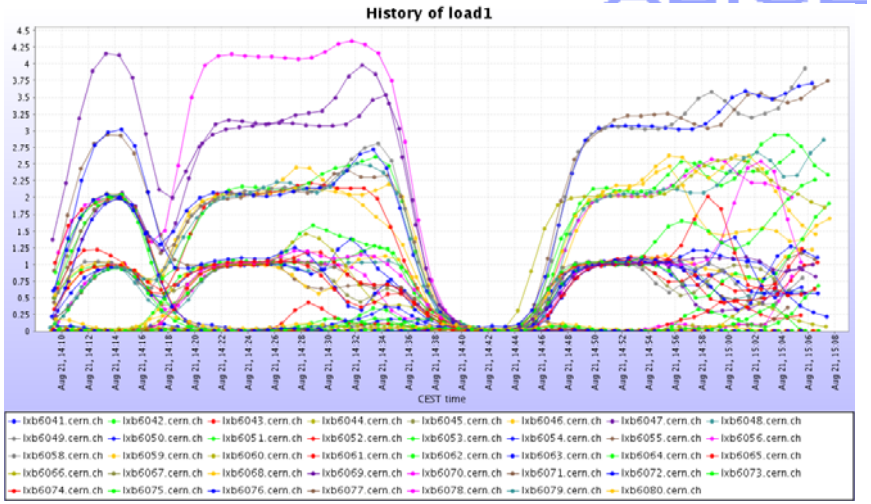
Relative Speedup (2)





Evaluation of the System Behavior

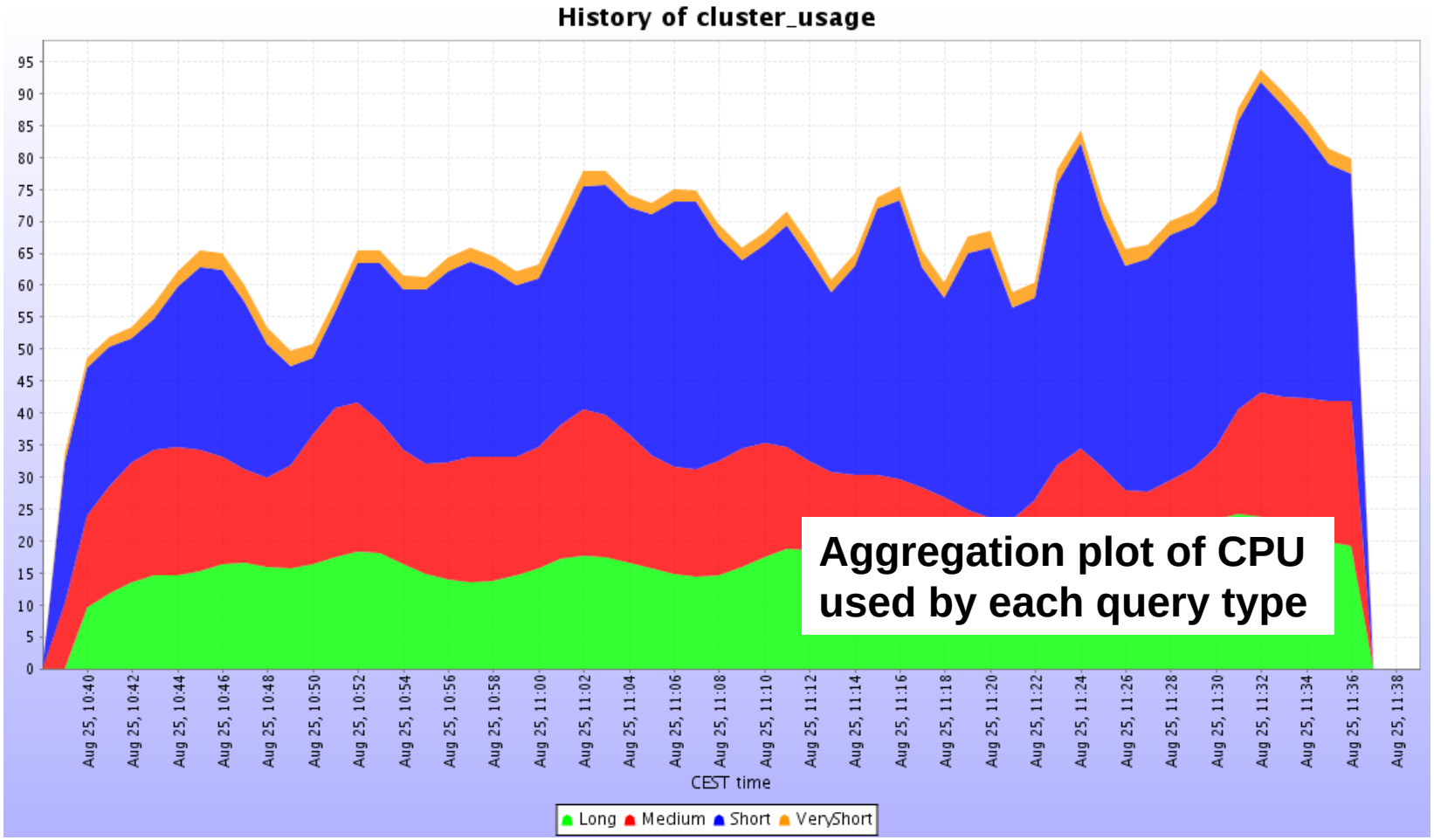
- MonALISA monitoring
  - Each host reports
  - Each slave reports
- Host
  - CPU, memory, swap, network
- Query (sum, per query type)
  - CPU, memory, event rate, file rate, IO vs. network rate
- pcalimonitor.cern.ch:8889
  - Click on CAF monitoring



Traffic between the cluster machines (MB/sec) (last 0.5h average)

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	-	-	1.5
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	-	0.7
13. 6060	-	-	-	-	-	-	-	-	2.068	-	-	-	11.59	-	-	1.06	-	-	-	-	-	-	2.0
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	-	-	0.5
18. 6065	0	-	-	-	3.52	3.165	-	0	-	-	0	-	-	-	-	-	3.034	0	1.579	-	0	-	
19. 6066	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	

Overall Usage of the Cluster

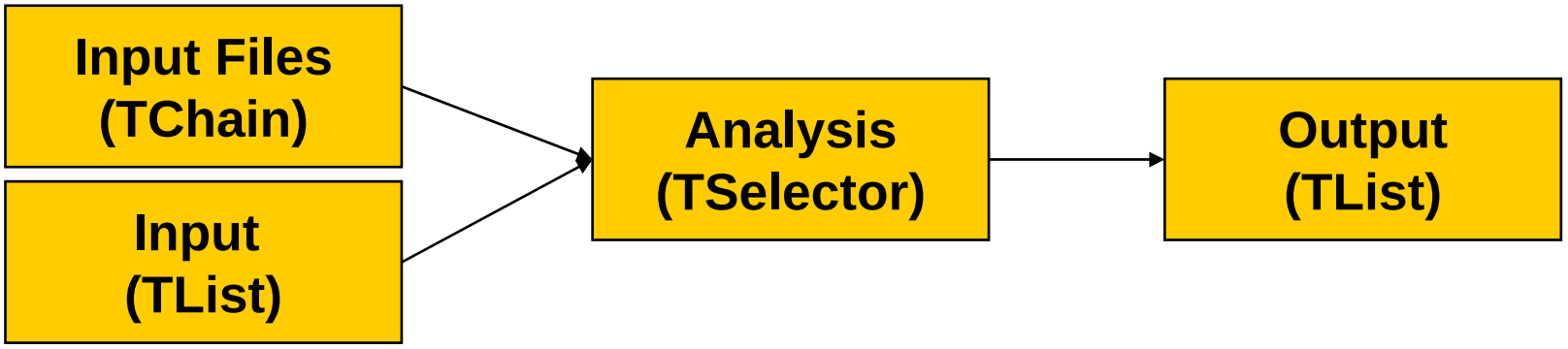


Content

- Theory part
  - Quick “repetition” about PROOF, CAF
  - Evaluation of the CAF test cluster
- Practical part (including demo)
  - Access ESD files
  - Access data via the RunLoader (needs full AliRoot)

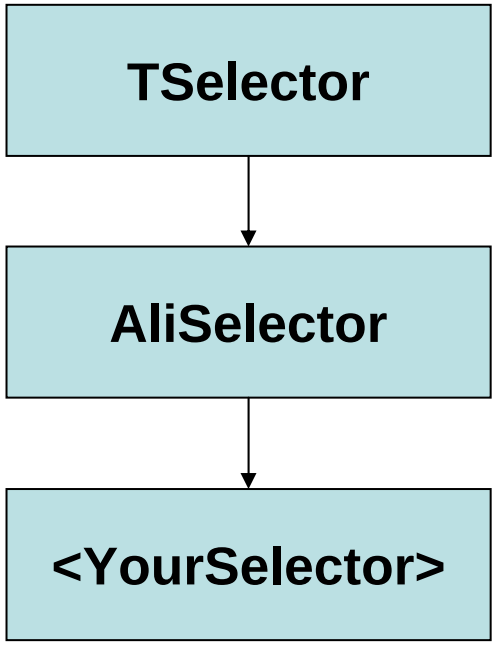
How to use PROOF

- Files to be analyzed are put into a chain (→ TChain)
- Analysis written as a selector (→ TSelector, AliSelector, AliSelectorRL)
- Input/Output is sent using dedicated lists
- If non-standard ROOT libraries are needed, these have to be distributed as a package



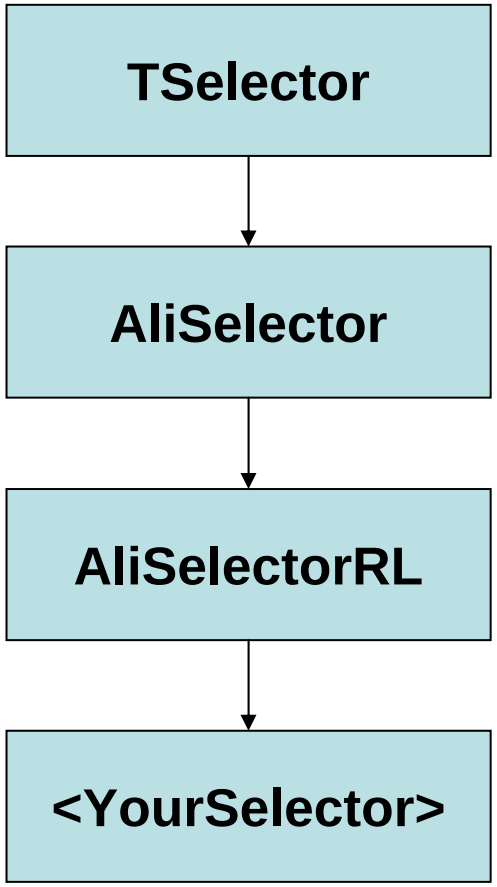
Accessing ESD

- To access AliESDs.root, the ESD.par package has to be uploaded into the PROOF environment
- Selector derives from AliSelector (in STEER)
- Access to data by member: fESD



Accessing the Runloader

- Access to Kinematics, Clusters, etc. requires access to the RunLoader
- Therefore (nearly) full AliRoot needs to be loaded
- A AliRoot version is already deployed on the CAF test system and can be enabled by a 6 line macro (~jgrosseo/public/proof/ProofEnableAliRoot.C)
  - ESD package is not allowed to be loaded
- Selector derives from AliSelectorRL (in STEER)
  - GetStack(), GetRunLoader(), GetHeader()



More Information

- 1M PDC06 events are available on the CAF
- Too few users at the moment, please use it!
- The two use cases (ESD, full AliRoot) will be exercised in the PROOF tutorials (next Mo, 9th Oct)
- CAF web site
  - http://aliceinfo.cern.ch/Offline/Analysis/CAF

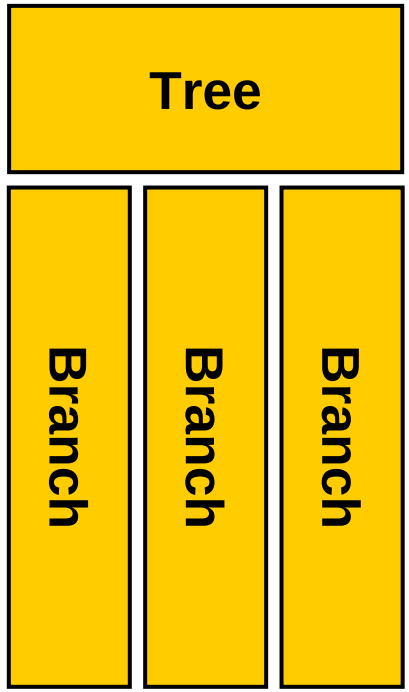
Conclusions

- PROOF has advantages over the case where each user gets a fraction of the nodes
  - Each user gets more total CPU time
  - The response time is much faster
- System is not underused
- For ALICE, the availability of PROOF/CAF in 2007 will be of critical importance for the fulfillment of the 'day one' physics program as well as calibration and alignment
  → Many bugs got resolved already. We encourage the PROOF team to continue to work hard!

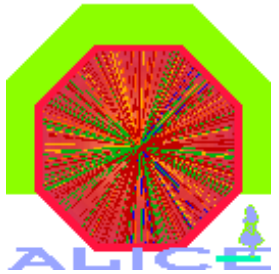
Backup

TTree

- A tree is an object for data storage
- It consists of several branches
- These can be in one or several files
- When reading a tree, certain branches can be switched off
  → significant speed up of analysis when not all data is needed (split mode)

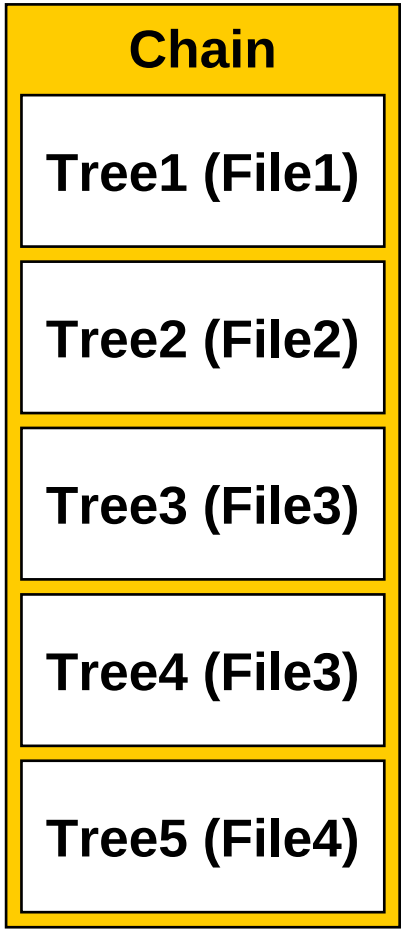


TChain



- A chain is a list of trees (in several files)
- Normal TTree functions can be used
  - **Draw(...), Scan(...)**

  → these iterate over all elements of the chain
- Selectors can be used with chains
  - **Process(const char* selectorFileName)**
- After using SetProof(...) these calls are run in PROOF



TSelector

- Classes derived from TSelector can run locally and in PROOF
  - **Begin()** — **once on your client**
  - **SlaveBegin()** — **once on each slave**
  - **Init(TTree* tree)** — **for each tree**
  - **Process(Long64_t entry)** — **for each event**
  - **SlaveTerminate()**
  - **Terminate()**

Input / Output

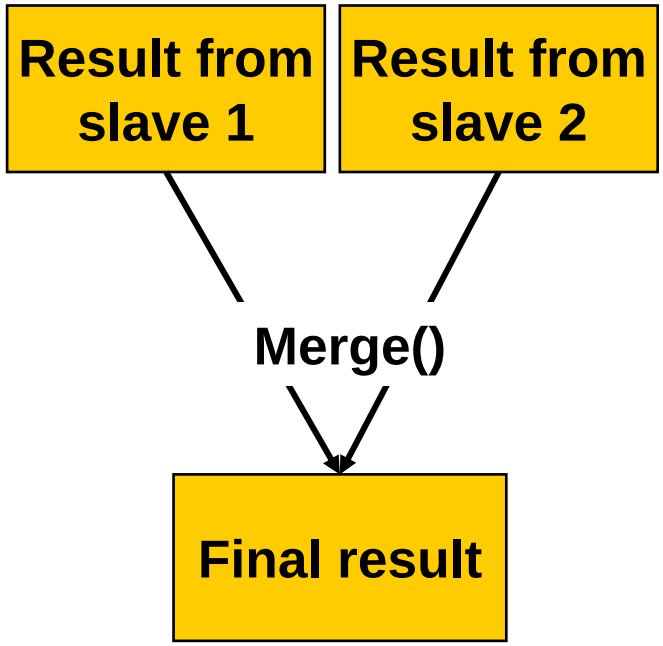
- The TSelector class has two members of type TList: fInput, fOutput
  - These are used to get input data or put output data
- Input list
  - Before running a query the input list is populated **proof->AddInput(myObj)**
  - In the selector (**Begin**, **SlaveBegin**) the object is retrieved: **fInput->FindObject("myObject")**

Input / Output (2)

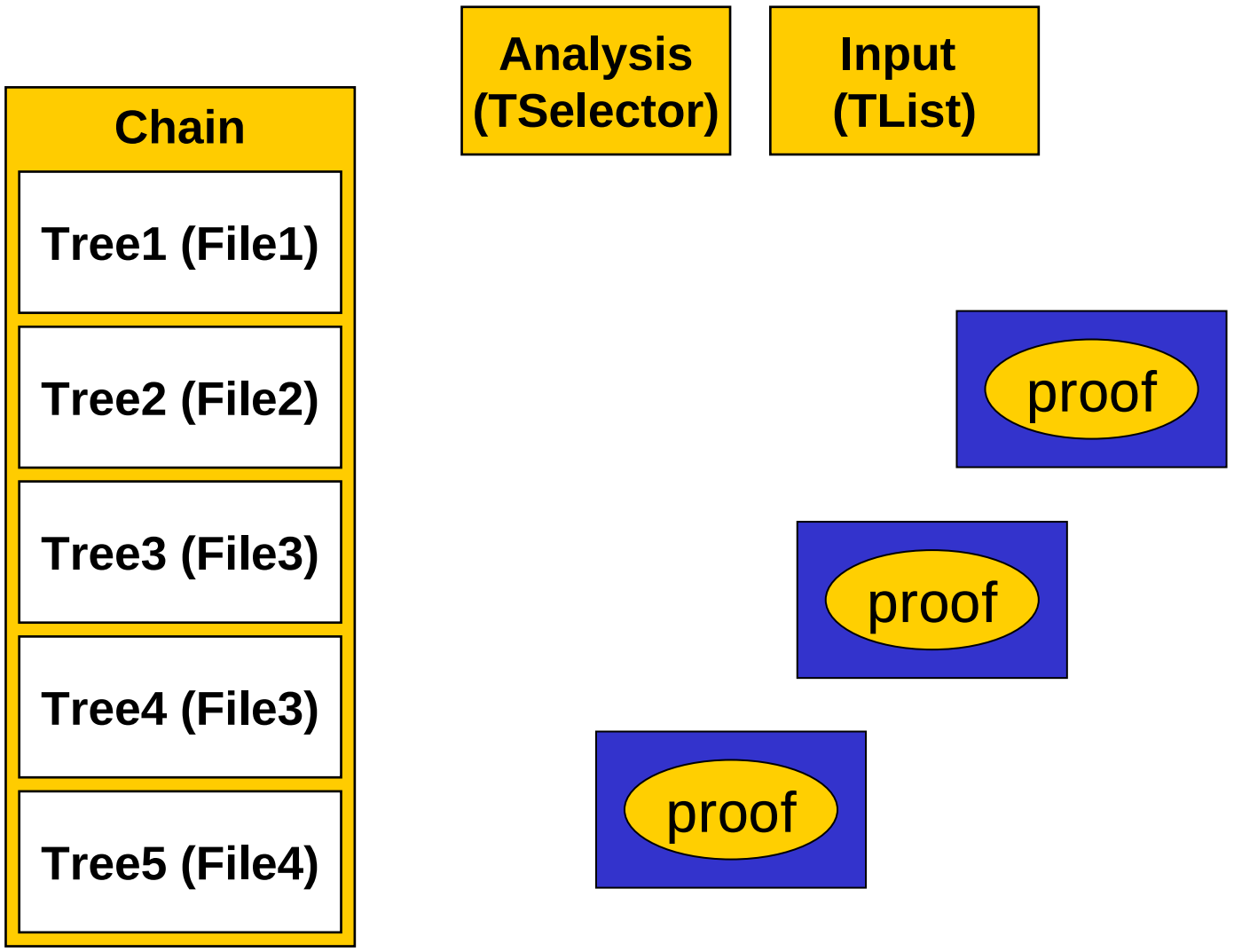
- Output list
  - After processing, the output has to be added to the output list on each slave (in **SlaveTerminate**) **fOutput->Add(fResult)**
  - PROOF merges the results from each query automatically (see next slide)
  - On your client (in **Terminate**) you retrieve the object and save it, display it, ... **fOutput->FindObject("myResult")**

Input / Output (3)

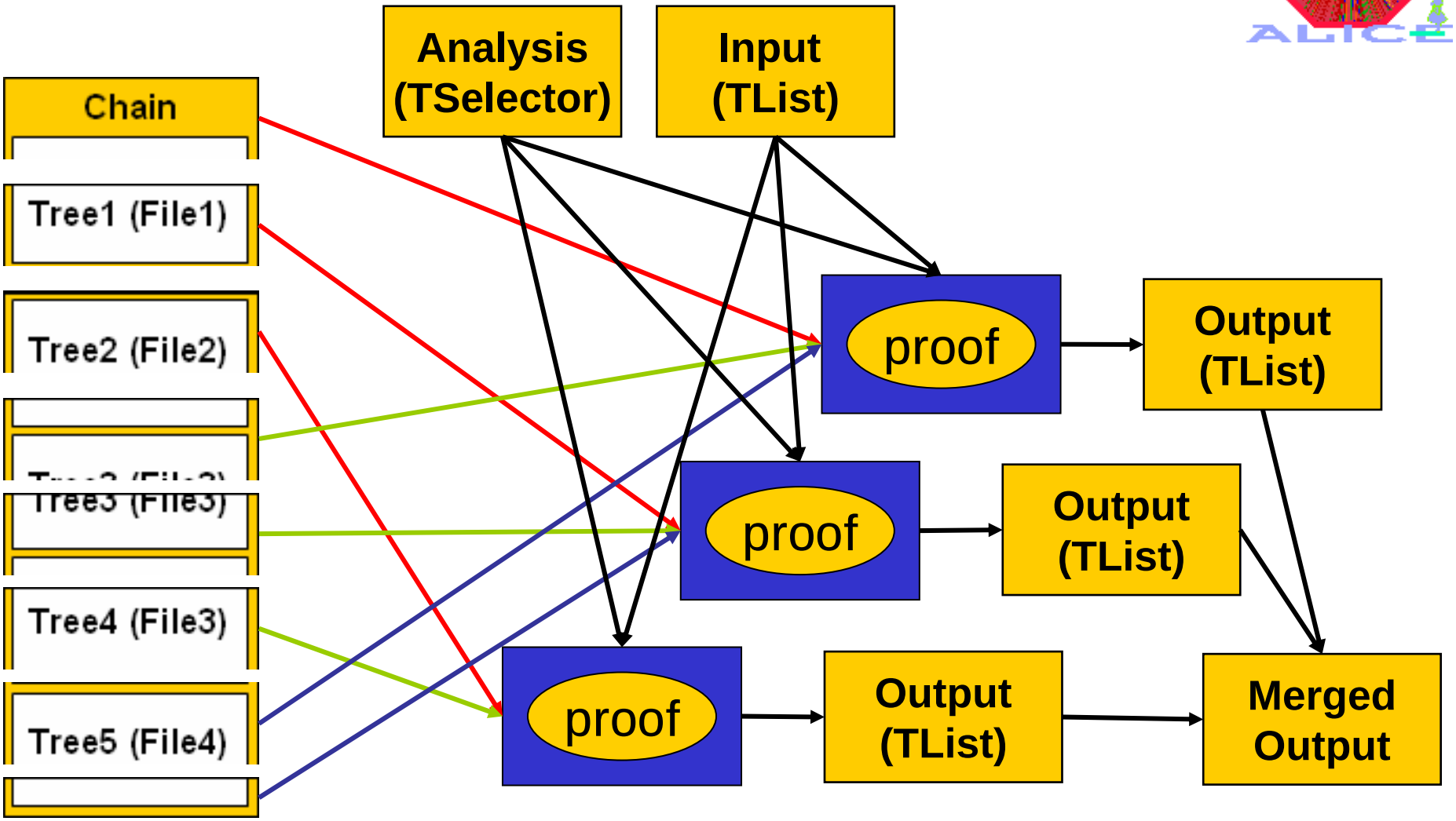
- Merging
  - Objects are identified by name
  - Standard merging implementation for histograms available
  - Other classes need to implement **Merge(TCollection*)**
  - When no merging function is available all the individual objects are returned



Workflow Summary



Workflow Summary



Packages

- PAR files: **P**ROOF **AR**chive. Like Java jar
  - Tar file
  - PROOF-INF directory
    - BUILD.sh, building the package, executed per slave
    - SETUP.C, set environment, load libraries, executed per slave
- API to manage and activate packages
  - **UploadPackage(“package.par”)**
  - **EnablePackage(“package”)**