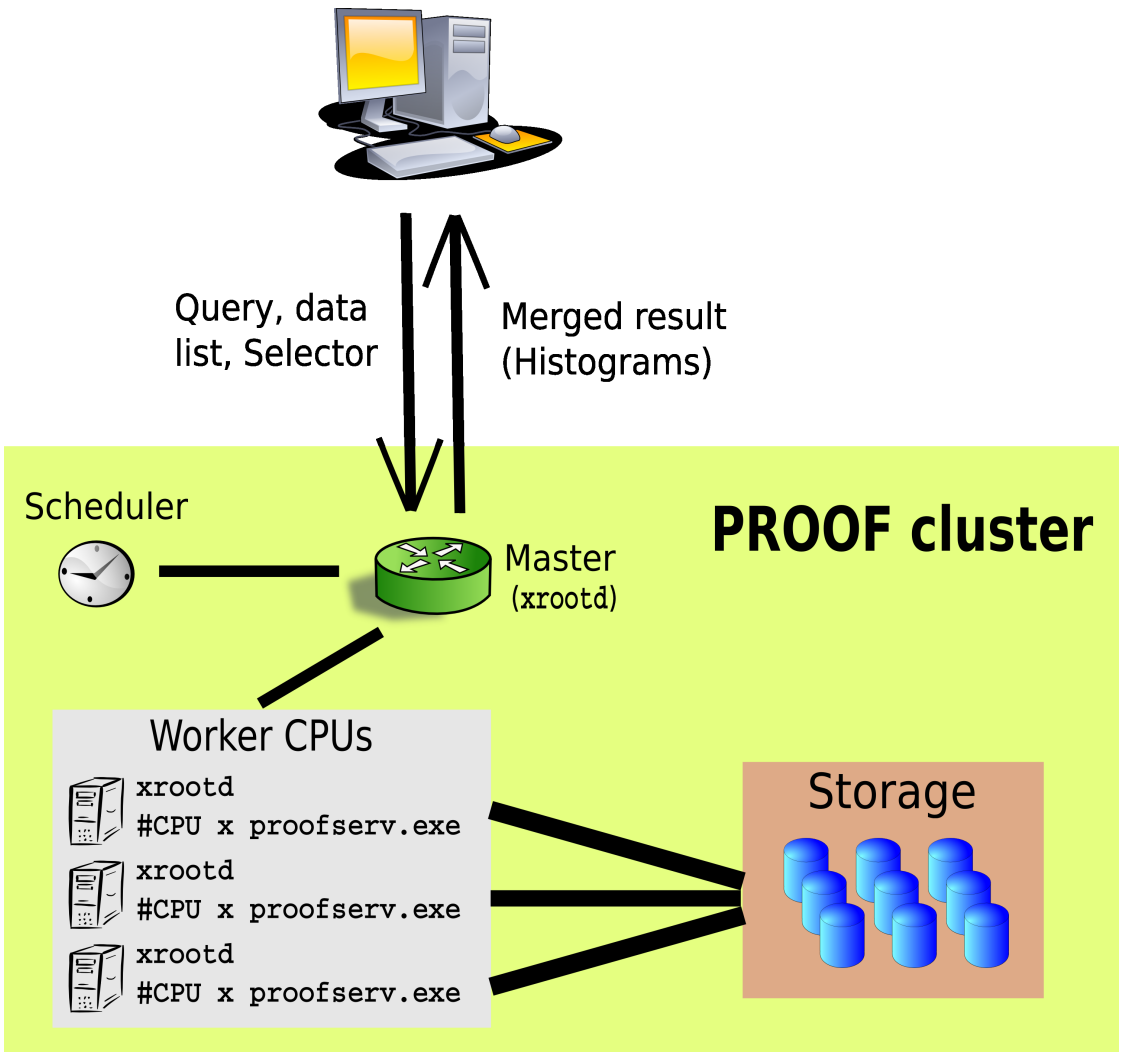
How does PROOF work?



- PROOF = "Parallel ROOT facility"
- Client connects to the PROOF master
- Selector and data path is send to master
- Master manages the data: data is split and sent to workers
- The results are joined and sent back to the client
- Transparent for the user!

PROOF – is it really faster?

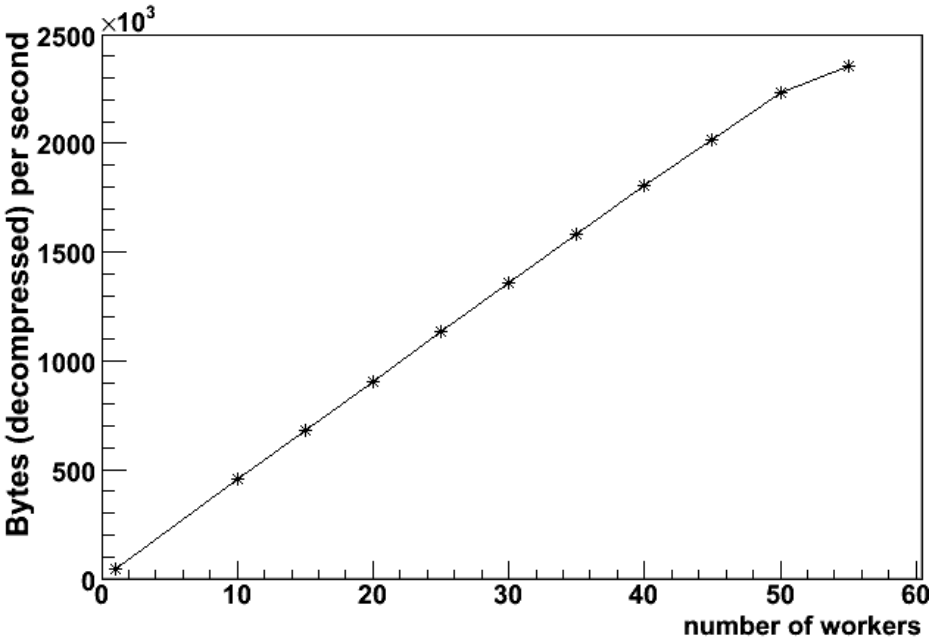
CPU bound jobs

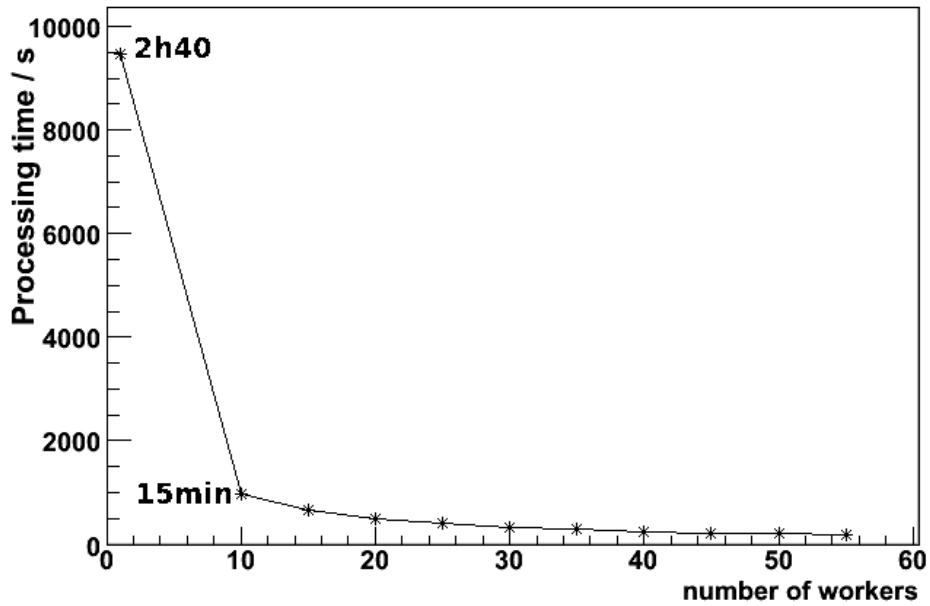
- Real world example: jet area algorithm
- What do you expect?

PROOF – is it really faster?

CPU bound jobs

- Real world example: jet area algorithm
- Rate scales linearly, time drops from several hours to a few minutes.





PROOF – is it really faster?

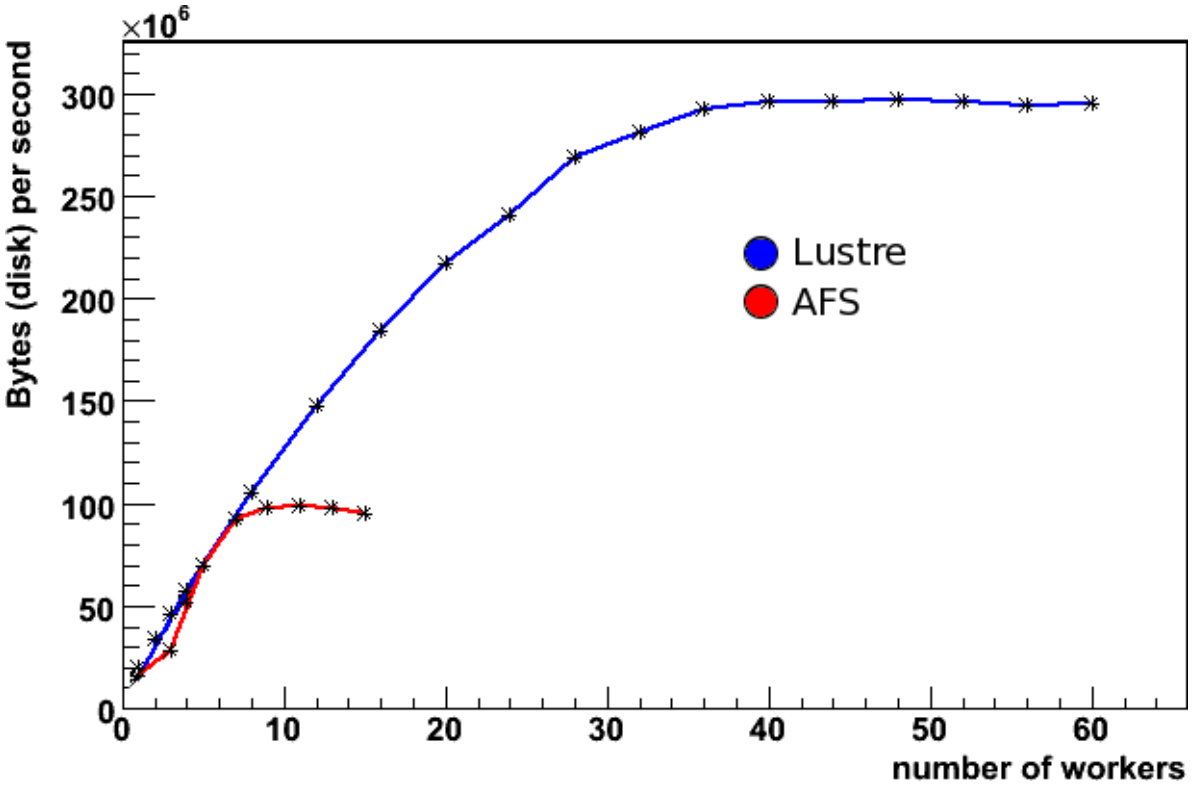
I/O bound jobs

- Selector on electron variables
- What do you expect?

PROOF – is it really faster?

I/O bound jobs

- Selector on electron variables
- Data throughput limited by I/O setup



Data

- 354 GB
- W+jets, CMSSW 1.6x

Speed depends on

- Storage system
- Network connection
- System load

Several possibilities to install PROOF

How to install PROOF?

- On a dedicated PROOF cluster
  - This is the standard way as described in the documentation
- On a batch farm
  - Every user can start his own cluster!
- On local desktop computer or laptop
  - Modern computers have at least 2 cores
  - You must have access to CMSSW and data files
  - Not covered in this talk

Comparing dedicated cluster and batch farm

Dedicated PROOF cluster	**PROOF on a batch farm**
One cluster for all users, one `xrootd` per machine	Users start "own" clusters, multiple daemons per machine
Master crashes → all users affected	Other users not affected
Version problems (new PROOF protocol in ROOT 5.17-06)	Independant clusters, user can select version
Instant startup (the PROOF cluster is already running)	Need to wait for free slots in the batch system
Fixed amount of resources used, wasted if cluster is not in use	Variable resources (SGE), shared with other batch jobs
Load balancing done by PROOF	Not needed (only one user)
Need to deal with security (Kerberos, running PROOF as `root`)	No issues, PROOF cluster is run under a user account

Running PROOF on the NAF using SGE

The master is started on the current WGS

- Set up ROOT and start the PROOF master daemon (`xrootd`) via `ssh MASTER start.sh`

Running PROOF on the NAF using SGE

The master is started on the current WGS

- Set up ROOT and start the PROOF master daemon (`xrootd`) via `ssh MASTER start.sh`

The workers run on the SGE farm

- Submit a parallel batch job
  `qsub -pe proof 5-10 -notify -l h_vmem=600M -l s_cpu=00:30:00 -l site=hh -o PathToStdout -e PathToStderr proofjob`
- Batch job file `$PE_HOSTFILE` contains available host names and number of cores to use
- Update configuration with list of available host and corresponding number of cores
- start `xrootd` via `qrsh -inherit -nostdin HOST start.sh NUMBER`
- The `xrootds` create the worker process(es) when `TProof::Open()` is called in ROOT

How users set up a PROOF cluster

How you can set up your own PROOF cluster on the NAF

- A simple script does the setup!
- `https://twiki.cern.ch/twiki/bin/view/CMS/HamburgWikiComputingNAFPROOF`

Steps

0. Write code and compile it
1. Call `proofcluster.pl start` (asks for configuration if called for the first time)
2. Run ROOT
3. Connect to the cluster, load and run your selector
4. Call `proofcluster.pl stop` (this is important, don't forget!)

Setting up the cluster: details

What can the `proofcluster.pl` script do for you?

- starts/stops/configures the cluster on the NAF
- automatically chooses CMSSW/PROOF version
- creates a ROOT macro which you need to execute. The macro...
  - connects ROOT to the PROOF cluster
  - sets up the CMSSW environment for the workers
  - executes `CURRENT_DIR/rootlogon.C` on every worker

proofcluster.pl

proofcluster.pl command [clustername]

Command can be one of the following:

- **start**: Starts the cluster and sets up the environment
- **stop**: Stop the cluster and free the resources
- **restart**: Executes the stop command followed by the start command
- **status**: Shows if master and worker are running
- **config**: Configure the cluster
- **copylog**: Copy log files from the workers to `/scratch/current/cms/user/NAME/proofcluster/log`
- **wc**: Write config files without starting the cluster

A cluster can be named (2nd paramter).

- Multiple cluster configurations can be saved
- Start more than one cluster at the same time

Examples

Drawing histograms

- Connect to PROOF (`TProof::Open()`)
- Add data files to a `TChain`
- Call `chain.SetProof()`
- Call `chain.Draw("whatever")`

Full CMSSW Framework

- Connect to PROOF (`TProof::Open()`)
- Load compiled libraries (`gProof->Load(...)`)
- Add data files to a `TChain`
- Call `chain.SetProof()`
- Call `chain.Process("NameOfSelector")`

Backup: List of important commands

List of commands

- **Set up CMSSW**
  ```
  export SCRAM_ARCH=slc4_ia32_gcc345;
  export VO_CMS_SW_DIR=/afs/naf.desy.de/group/cms/sw;
  source $VO_CMS_SW_DIR/cmsset_default.sh
  ```
- **Create project area:**
  `cmsrel CMSSW_2_0_12; cd CMSSW_2_0_12/src; cmsenv`
- **/afs/naf.desy.de/group/cms/proof/proofcluster.pl**: Start/stop the PROOF cluster
- **/afs/naf.desy.de/group/cms/proof/makechain.pl**: creates macro containing Chain.Add for a list of files, also works for files on dCache
- **mktsel**: Create TFWLiteSelector skeleton

List of ROOT commands

- `gSystem->Load("lib.so") / gProof->Load("lib.so")` – load a .so file
- `gSystem->Execute("any root command")` – run on all PROOF workers
- `chain.SetProof()` – enable/disable PROOF for a chain
- `chain.Process("selector",option="",max)` – run the selector