PROOF-based services

G. Ganis, F. Rademakers, CERN, PH-SFT

WLCG 2009 Data-Taking Readiness Planning Workshop
CERN, 14 Nov 2008

Outline

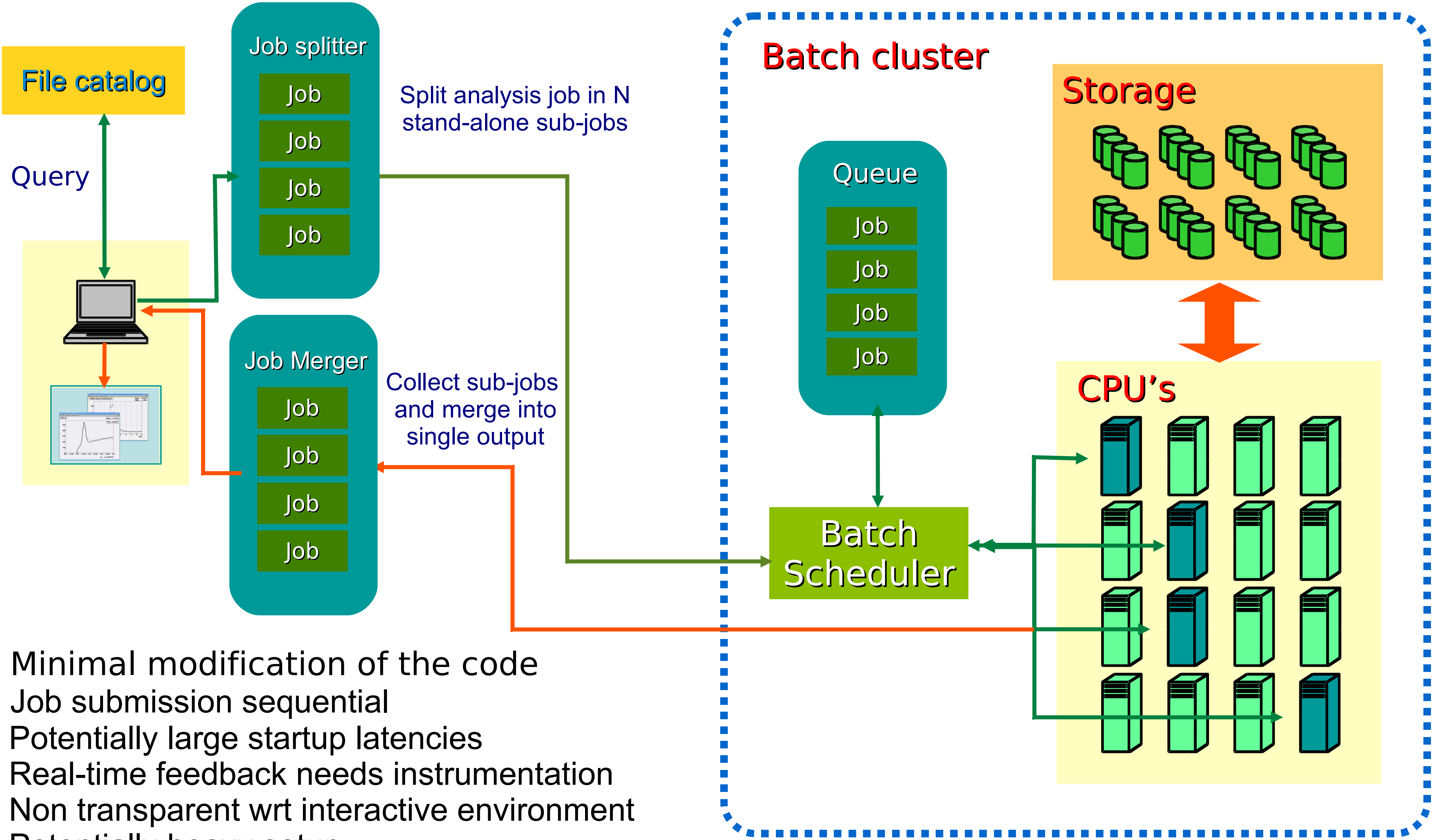
- PROOF reminder
- Performance considerations
- Installations
  - Setup and examples
  - Dataset handling
- Summary

PROOF – Parallel ROOT Facility

Designed for interactive processing of ideally parallel tasks at Tier 2 / Tier 3 facilities and many-core desktops

- Parallel coordination of distributed ROOT sessions
  - Transparent: extension of the local shell
  - Scalable: small serial overhead
- Multi-Process Parallelism
  - Easy adaptation to broad range of setups
  - Less requirements on user code
- Process the data where they are, if possible
  - Outputs much smaller than inputs
  - Minimize data transfers
- Dynamic load balancing
  - Minimize wasted cycles

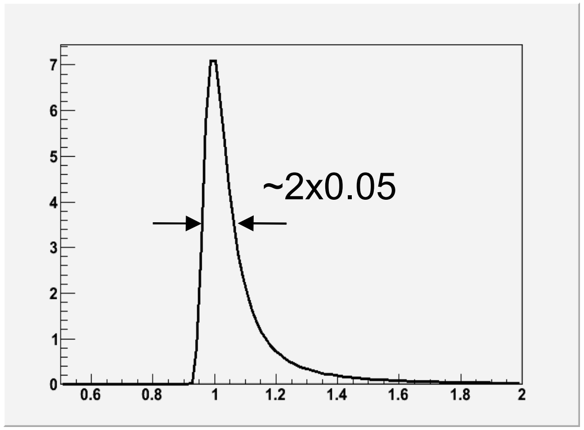
Traditional approach



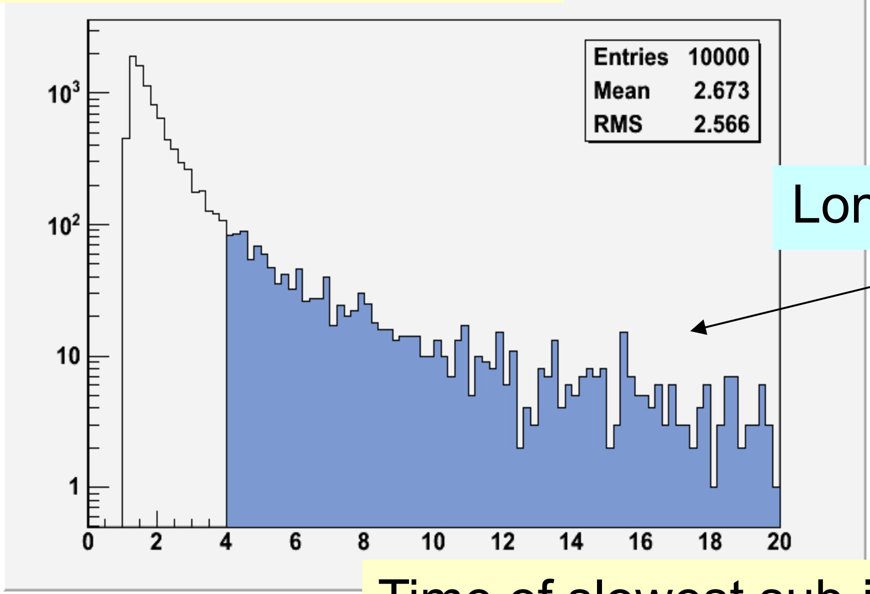
- Minimal modification of the code
- Job submission sequential
- Potentially large startup latencies
- Real-time feedback needs instrumentation
- Non transparent wrt interactive environment
- Potentially heavy setup

Traditional approach: sensitivity to tails

- Last sub-job determines the execution time
  - Basically a Landau distribution (see L. Betev talk)
- Example:
  - Total expected time 20h, target 1h
  - 20 sub-jobs, 1h +- 5%



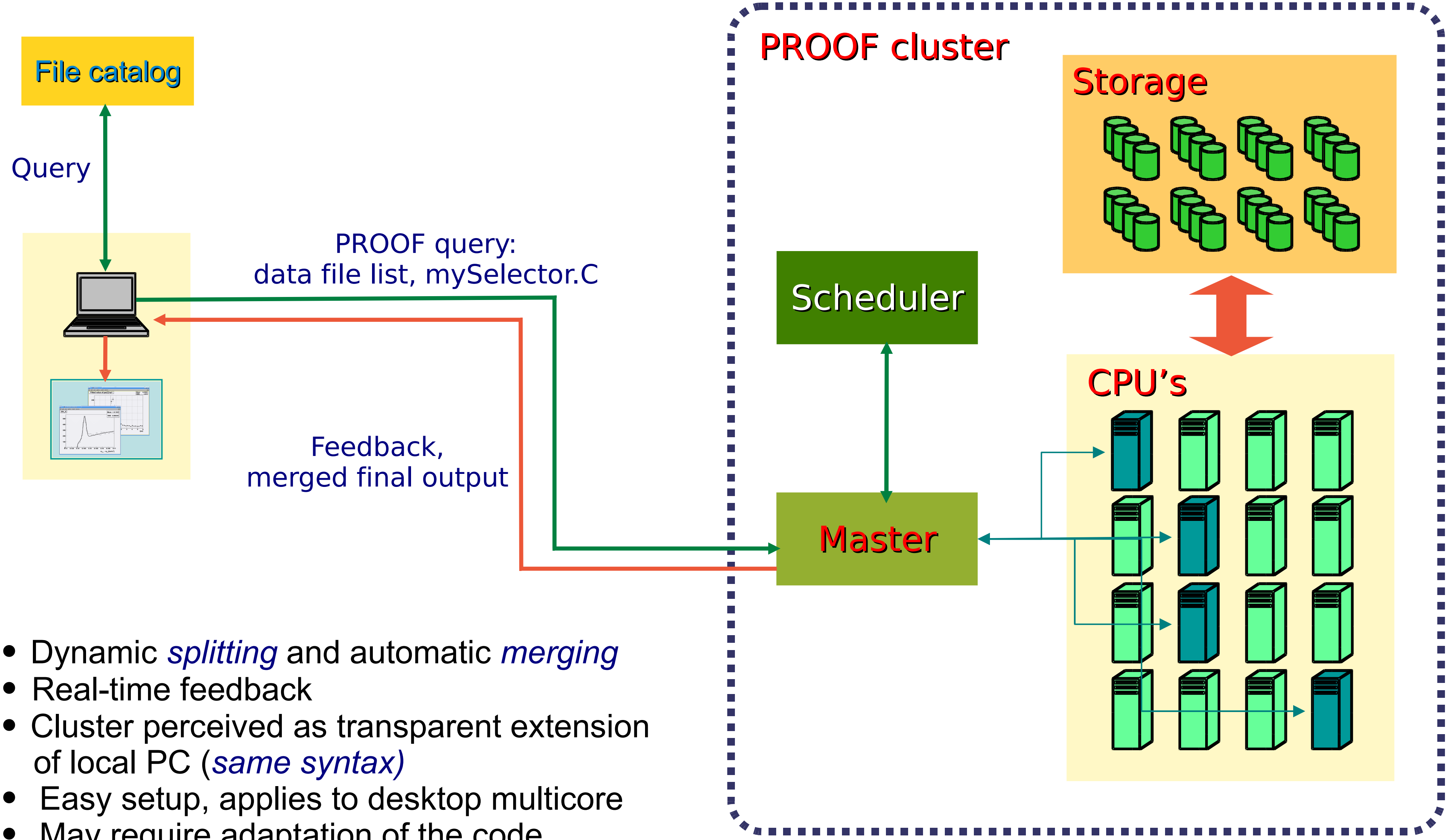
10000 toy experiments



Long tails: e.g. 15% > 4 h

Time of slowest sub-job

PROOF approach



File catalog

Query

PROOF query:
data file list, mySelector.C

Feedback,
merged final output

PROOF cluster

Storage

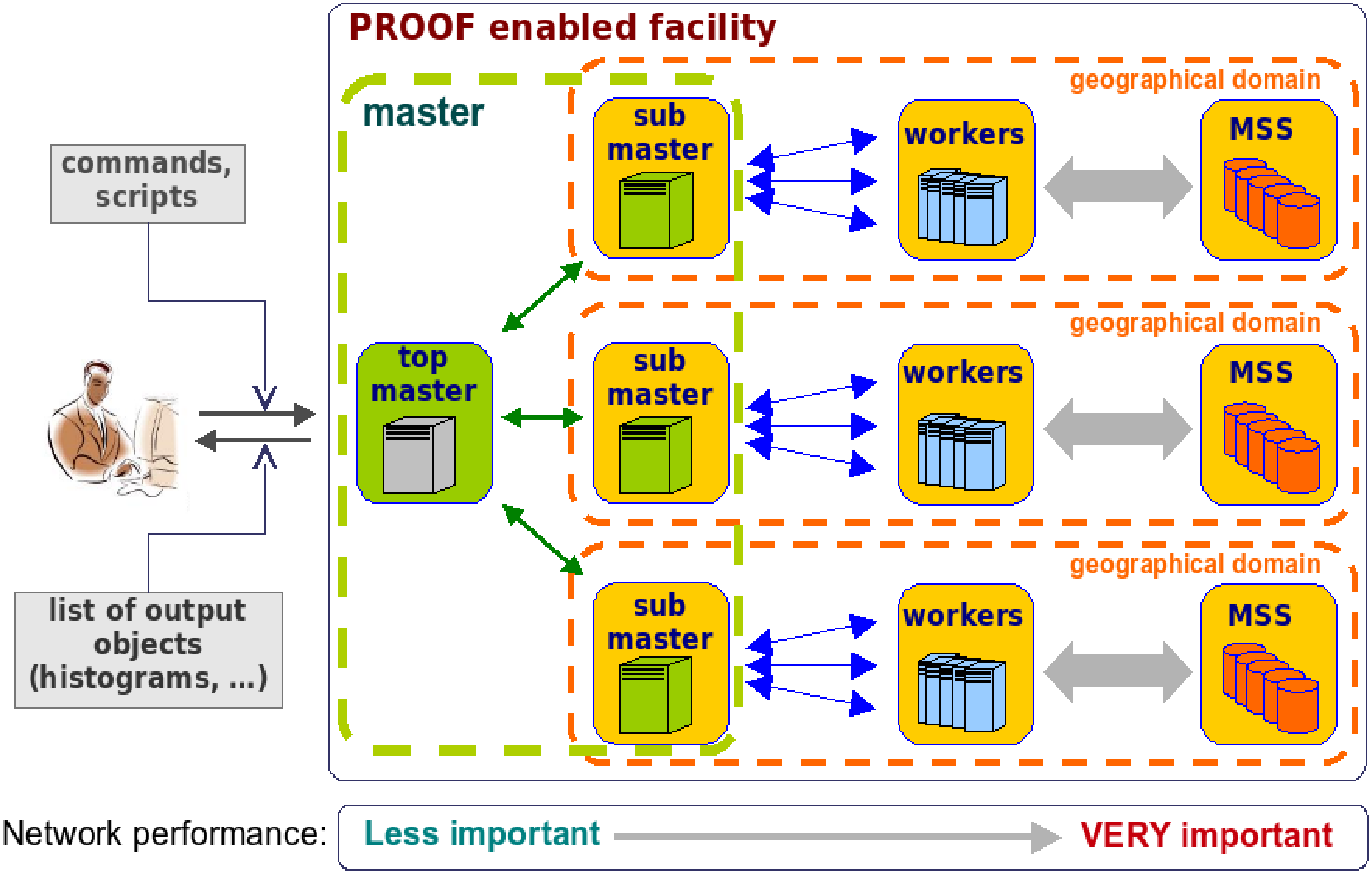
Scheduler

Master

CPU’s

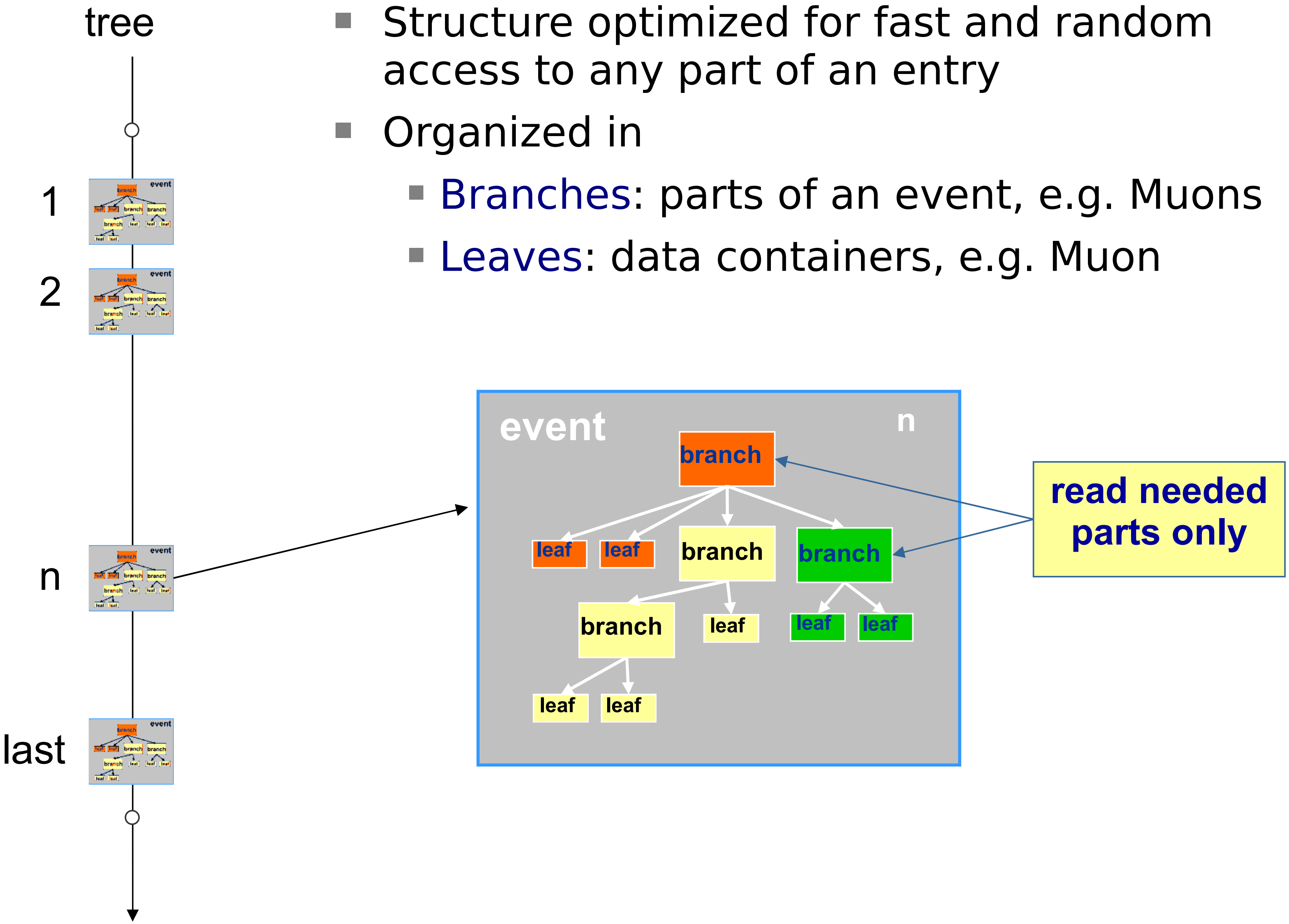
- Dynamic *splitting* and automatic *merging*
- Real-time feedback
- Cluster perceived as transparent extension of local PC (*same syntax)*
- Easy setup, applies to desktop multicore
- May require adaptation of the code

PROOF architecture

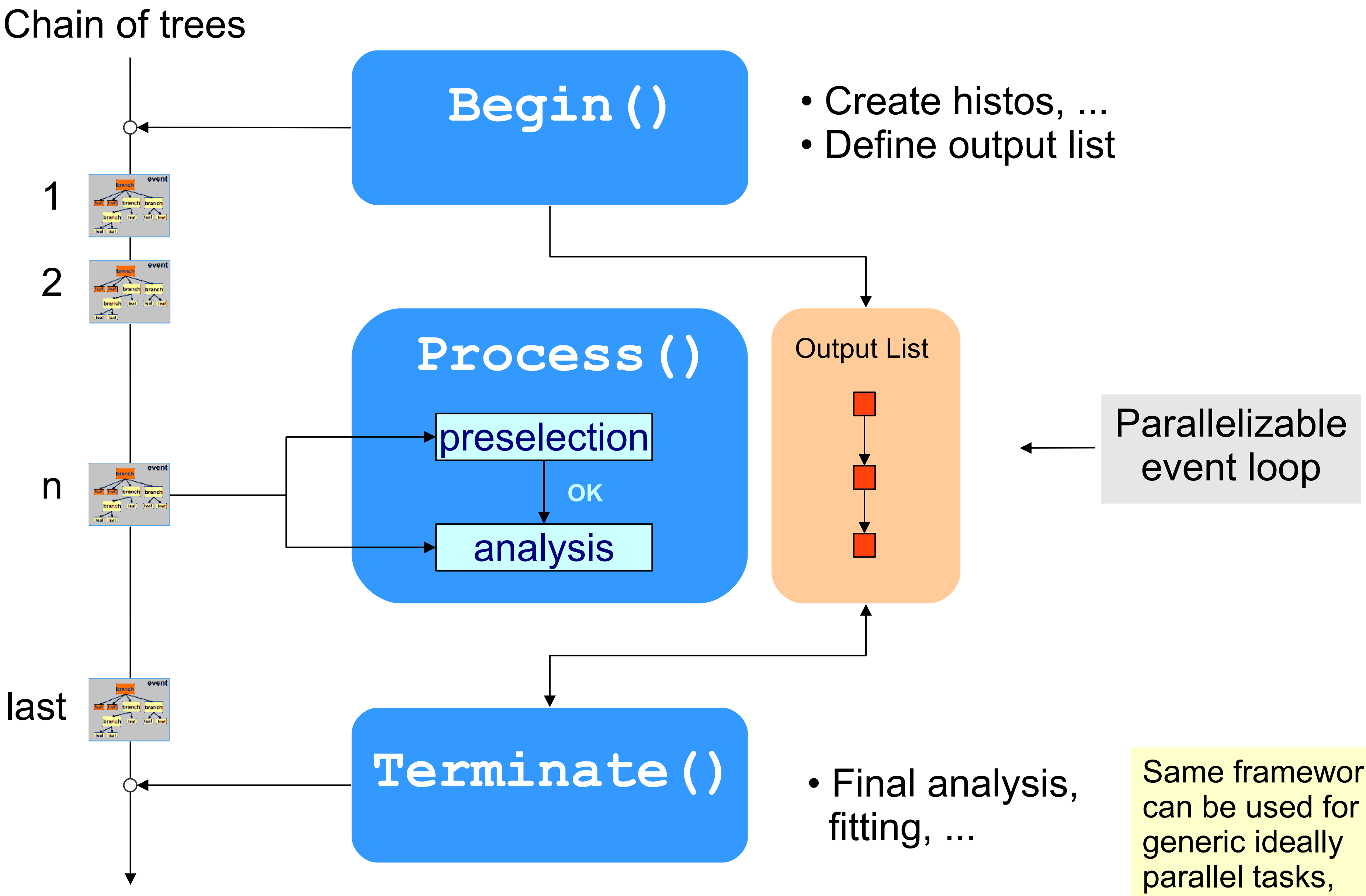


ROOT trees

- Structure optimized for fast and random access to any part of an entry
- Organized in
  - Branches: parts of an event, e.g. Muons
  - Leaves: data containers, e.g. Muon



Processing ROOT trees: TSelector framework



- Create histos, ...
- Define output list

Parallelizable event loop

- Final analysis, fitting, ...

Same framework can be used for generic ideally parallel tasks, e.g. MC simulation

Impact on existing frameworks

How difficult is to adapt a framework to PROOF?

- PROOF runs the event loop and opens the files
  - Possible interference with frameworks
- Modular approach to analysis algorithms and input / output handling
  - Allows to hide TSelector behind the scene
  - Examples
    - AliAnalysisTask (ALICE)
    - Tree-Analysis-Module (Phobos)
    - TFWLiteSelector template (CMS)
- TSelector framework is flexible
  - Can be used just to schedule tasks with file-level granularity
    - ATLAS interest
- Smooth transition typically possible

Hardware performance considerations

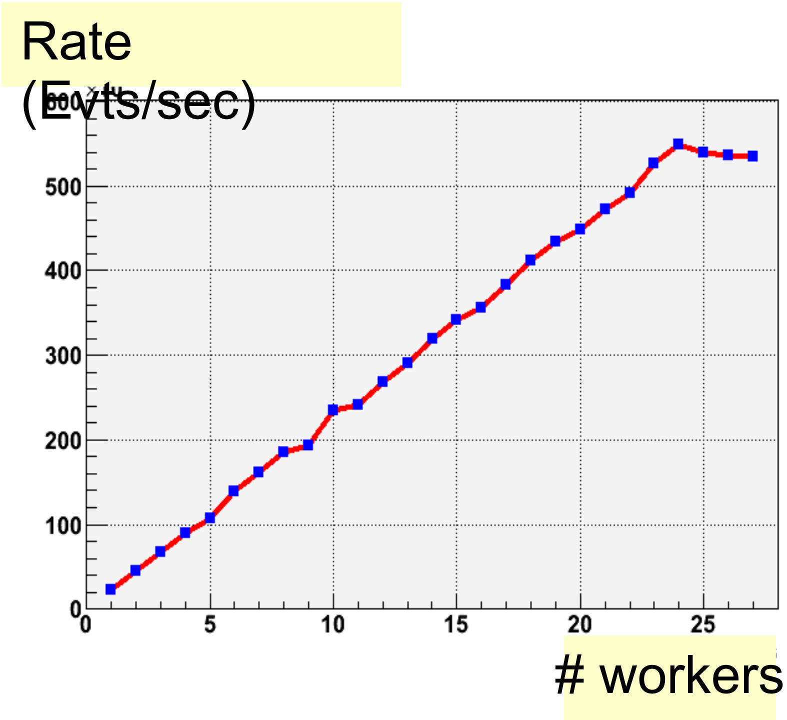
- Typical resource consuming end-user analysis
  - Data mining / processing ⇨ ~ I/O bound
  - Fits, {full,fast,toy}-simulations for systematic studies, ...
    ⇨ ~ CPU bound

- Today typical hardware
  - Many-cores and reasonably large RAM
    - 4 or 8 (⇨ 64 next year?), 2 GB / core
  - Standard HDD

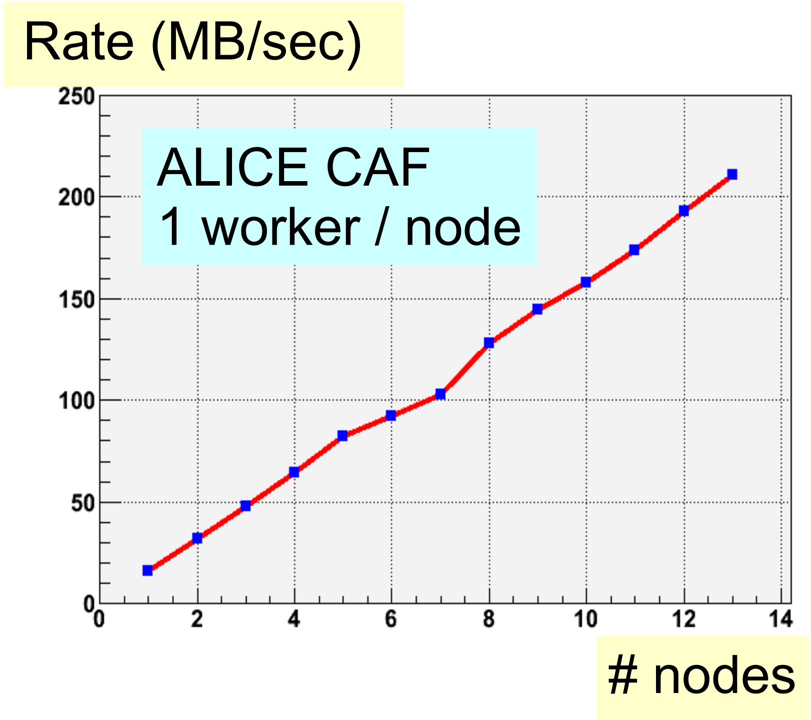
- Most likely the bottleneck is I/O
  - HDDs serve well ~2÷3 cores
  - Need performant I/O systems for data processing
    - Dedicated multi-HDD (HW or SW RAID)
    - Solid State Disks

Scalability tests

24 core machine
Toy MC simulation



ALICE CAF
ESD-based analysis

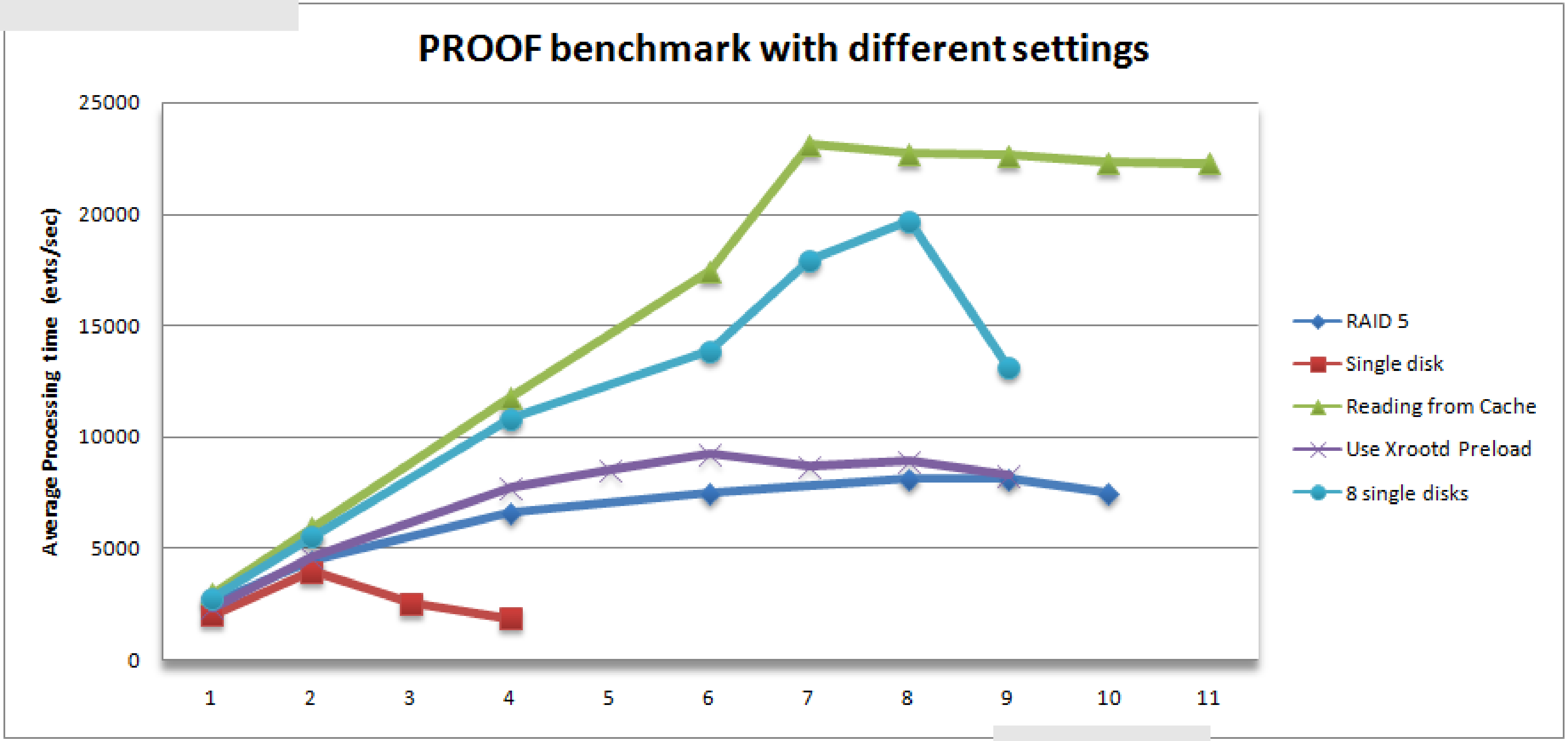


- Almost perfect scalability for CPU-bound tasks or I/O bound tasks with independent disk controllers

ATLAS I/O tests

8 core machine

Courtesy of Neng Xu, Wisconsin



# workers

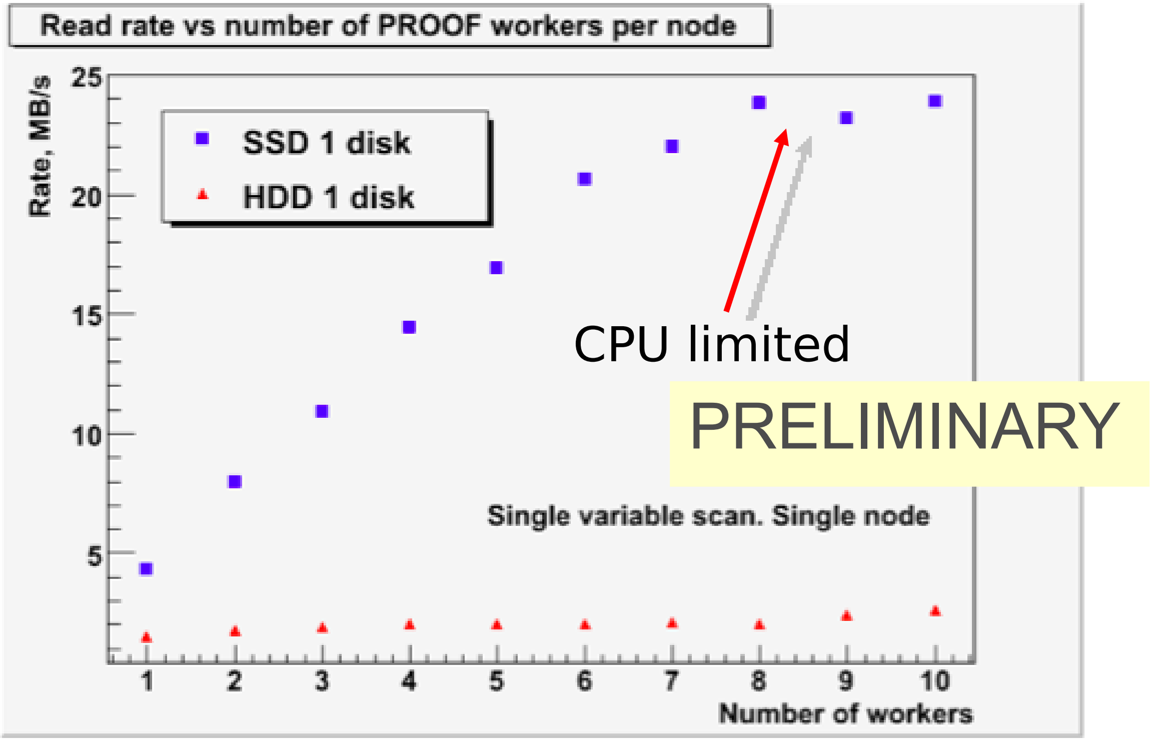
- 2 cores vs 1 disk seems to be a reasonable HW ratio
- Multi-disk systems allow to go beyond this limit
- Optimized use of memory caching techniques can also help

ATLAS tests using Solid State Disks (SSD)

Courtesy of S. Panitkin, BNL

BNL PROOF farm

- 10 nodes / 80 cores
- 2.0 GHz / 16 GB RAM
- 5 TB HDD / 640 GB SSD
- ProofBench analysis



CPU limited

PRELIMINARY

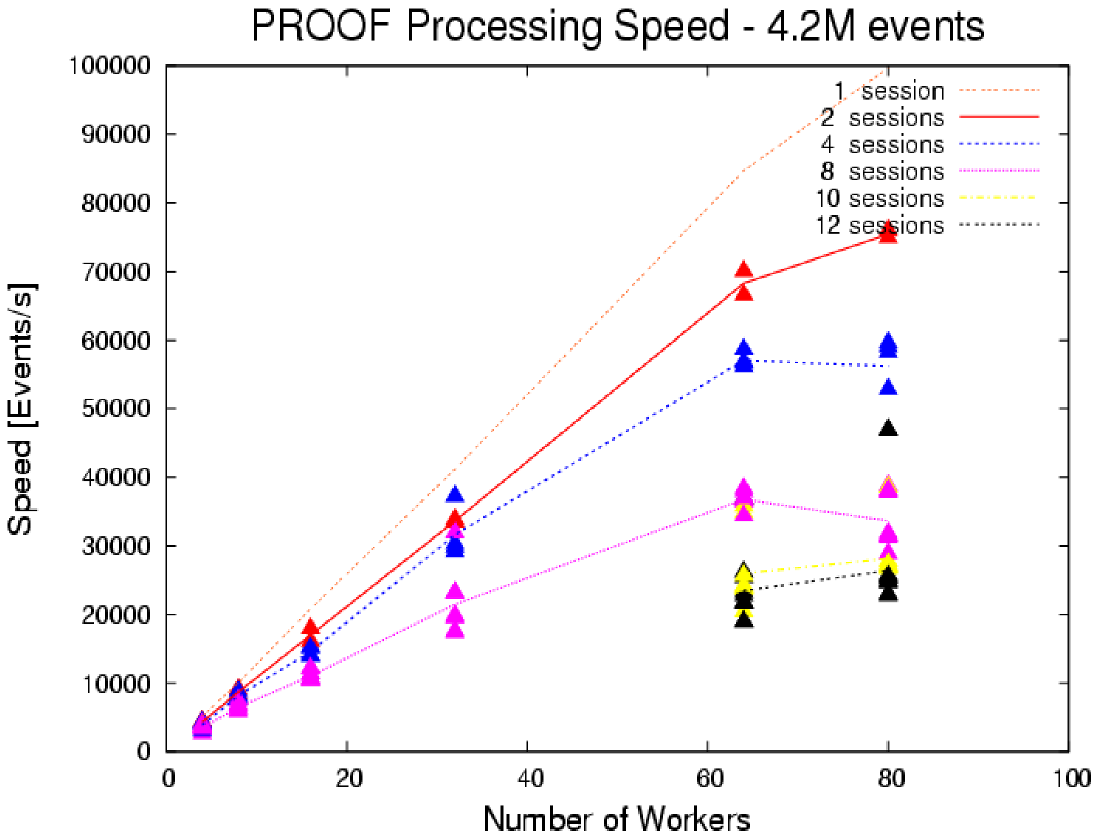
- SSD holds clear speed advantage
  - ~10 times faster in concurrent read scenario
- Price starts becoming affordable

ATLAS physics analysis

Courtesy of G.C. Montoya, Wisconsin

- Higgs 4-lepton analysis
- 50 nodes, AMD 64bit 4x, 4 GB RAM
- 4.5 M events, 68 GB
- 845 files
- Analysis include TMinuit fit

- Single session
  - 1.5 kEvt/s ⇒ 50 min
- PROOF 1 user (80 wrks)
  - 100 kEvt/s ⇒ ~1 min
- PROOF 8 users (64 wrks)
  - 40 kEvt/s ⇒ ~2.5 min



Major current PROOF installations

ALICE

CERN Analysis Facility

- 112 cores, 35 TB
  - Target: 500 cores, 110 TB
- Prompt analysis of selected data, calibration, alignment, fast simulation
- 5-10 concurrent users
  - ~80 users registered

GSI Analysis Facility, Darmstadt

- 160 cores, 150 TB Lustre
- Data analysis, TPC calibration
- 5-10 users
- Performance example:
  - ~1.4 TB processed in ~20 min

ATLAS

Wisconsin

- 200 cores, 100 TB, RAID5
- Data analysis (Higgs searches)
- I/O perfomance tests w/ multi-RAID
- PROOF-Condor integration
- ~20 registered users

BNL

- Users: 40 cores, 20 TB HDD
- Test: 72 cores, 25 TB HDD, 192 GB SSD
- I/O perfomance tests with SSD, RAID
- Tests of PROOF cluster federation
- ~25 registered users

Test farms at LMU, UA Madrid, UTA

PROOF-enabling a cluster

- PROOF is part of ROOT
  - No additional package

- PROOF service runs as an XROOTD plug-in
  - Same XROOTD can be used to serve files and PROOF sessions
    - Port 1094 for data serving, port 1093 for PROOF

- Configuration files
  - Dedicated part in the XROOTD config file
    - Can be the same physical file for all nodes
  - File defining the role of the nodes (proof.conf)
  - File defining the groups of users and their properties
    - Priorities, quotas, ...

Example: ALICE CAF - 1/2

- ROOT versions installed via RPM
- Relevant files on AFS
  - Configuration files
  - XROOTD MPS scripts to populate the local pool space
- ALICE-specific RPM to setup a machine
  - Setup init.d scripts
    - xrootd, cmsd, monalisa
  - Configure relevant directories
    - Local data pools
      - /pool/alien, /pool/castor
    - Local dataset management
      - /pool/dataset/<group>/<user>
    - User sandboxes
      - /pool/proofbox/<user>

Example: ALICE CAF - 2/2

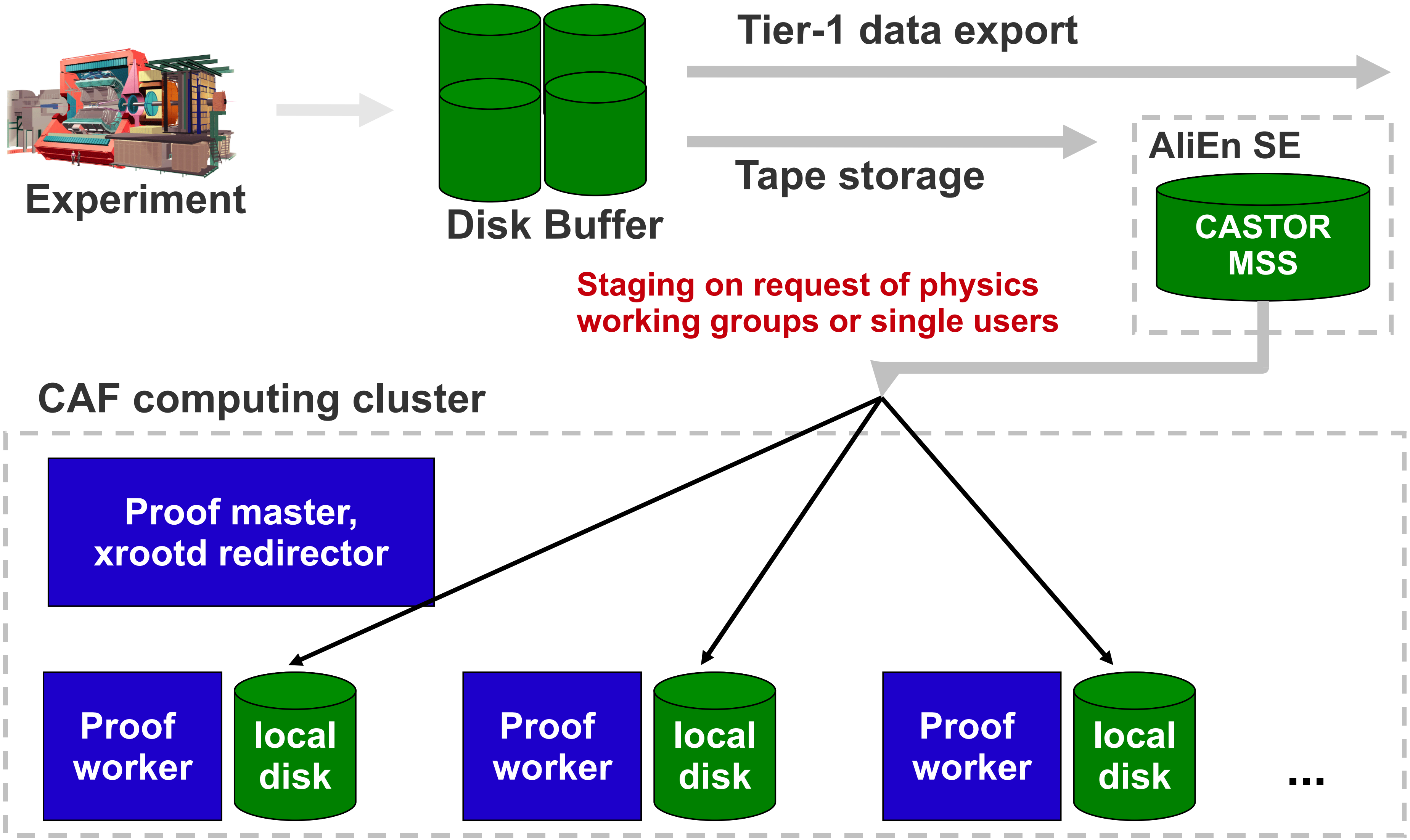
- Cluster managed using set of scripts based on 'wassh'

```
$ cafpro installrpm 5-21-05-alice
$ cafpro restart
```

- User Support & bugs
  - ROOT Savannah
  - Dedicated mailing list
    - alice-project-analysis-task-force@cern.ch

ALICE CAF schematically

Courtesy of J.F Grosse-Oetringhaus, CERN



Experiment

Disk Buffer

Tier-1 data export

Tape storage

AliEn SE

CASTOR MSS

Staging on request of physics working groups or single users

CAF computing cluster

Proof master, xrootd redirector

Proof worker

local disk

Proof worker

local disk

Proof worker

local disk

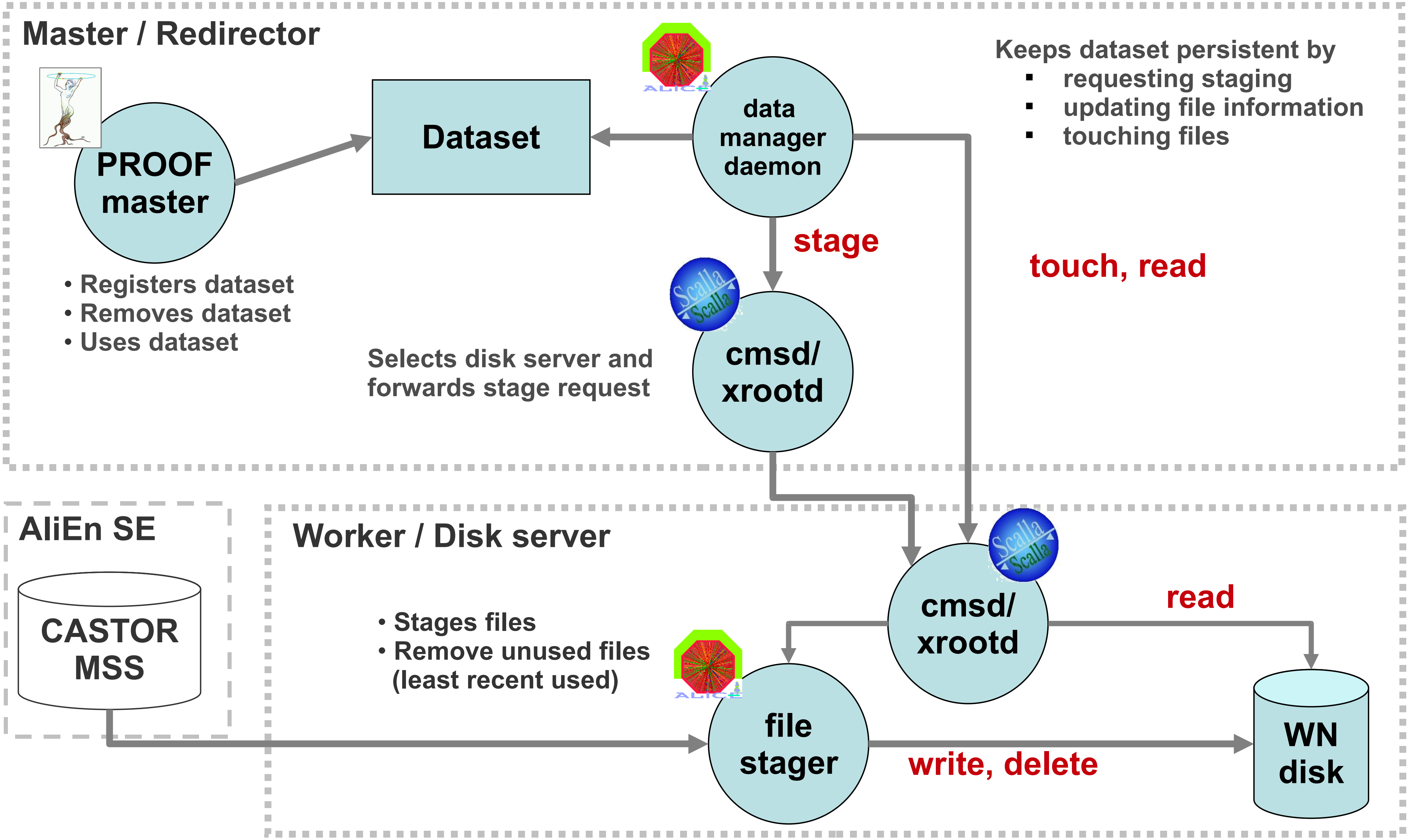
...

Dataset management

- Dataset: named collection of files
- Dataset manager
  - Handle datasets
    - Register a new dataset or remove an existing one
    - Retrieve information
    - Verify the availability of the files
  - Basic quota management
- Information sources: different backends
  - Dedicated ROOT files on the master
    - E.g. created from the AliEn catalog (ALICE)
  - Experiment dataset databases
    - E.g. SQL based (ATLAS)

Dataset handling at ALICE CAF

Courtesy of J.F Grosse-Oetringhaus, CERN



Master / Redirector

PROOF master

- Registers dataset
- Removes dataset
- Uses dataset

Dataset

data manager daemon

Keeps dataset persistent by

- requesting staging
- updating file information
- touching files

Selects disk server and forwards stage request

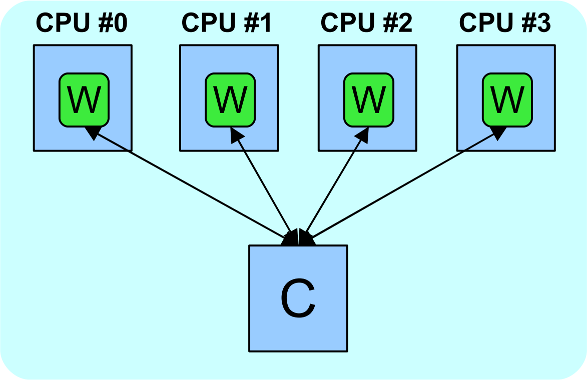
AliEn SE

Worker / Disk server

- Stages files
- Remove unused files (least recent used)

PROOF-Lite

- Realization of PROOF in two tiers optimized for multi-cores



- The client starts / controls directly the workers (# $\equiv N_{CPU}$)
  - No need of daemons, works out of the box
  - Communication goes via UNIX sockets for optimal resource usage
- Very efficient: very good scalability for CPU-bound analysis
- Allows to transparently exploit the additional CPU power for a ROOT-based analysis

Summary

- PROOF technology is a viable solution for interactive end-user analysis at Tier3 facilities
  - Code development with large statistics
  - CPU intensive systematic studies

- Provides straight-forward extension of ROOT-based analysis of distributed resources
  - Comes with ROOT
    - No additional dependencies

- Lot of constructive feedback from ALICE / ATLAS users
  - Realistic use-cases
  - New functionality (e.g. dataset management)