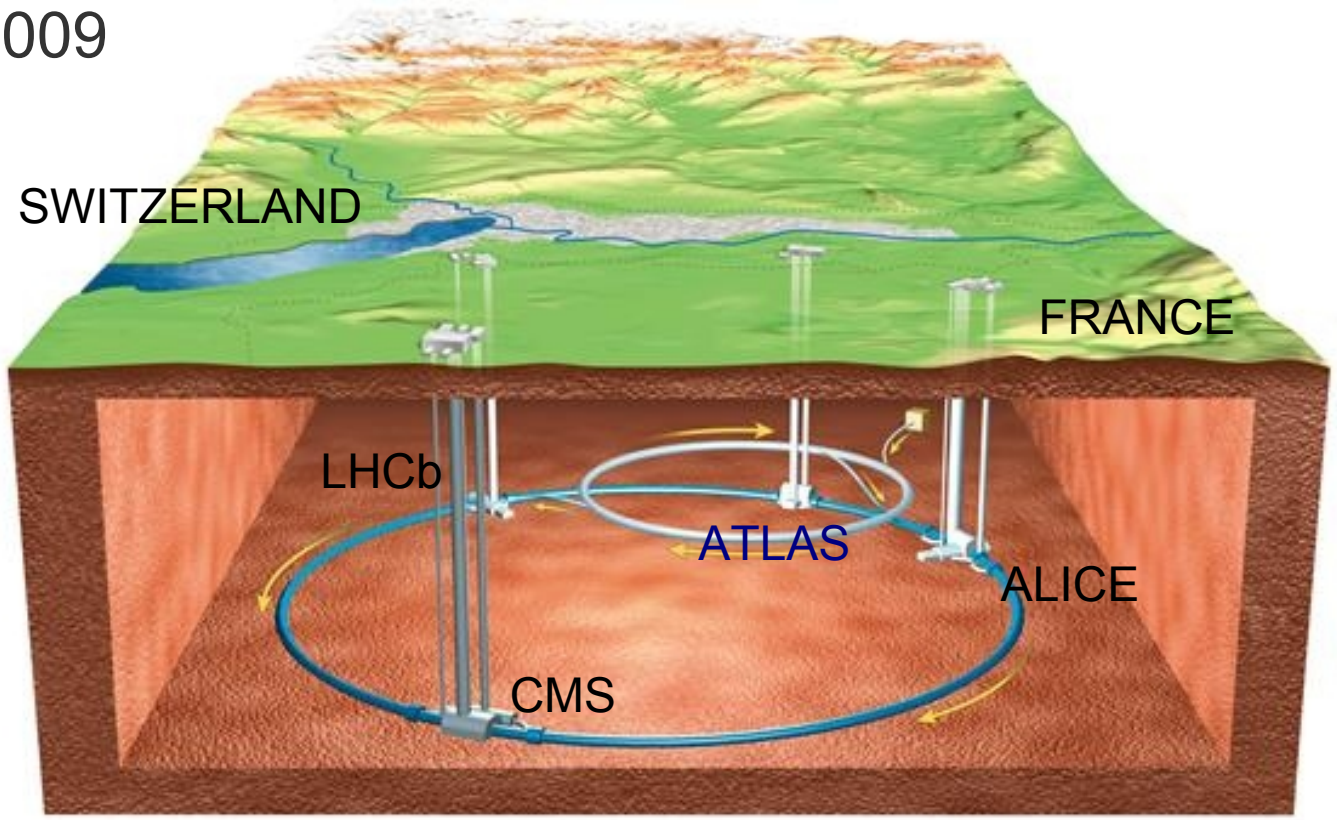
Data Analysis with PROOF

Gerardo Ganis
CERN PH-SFT

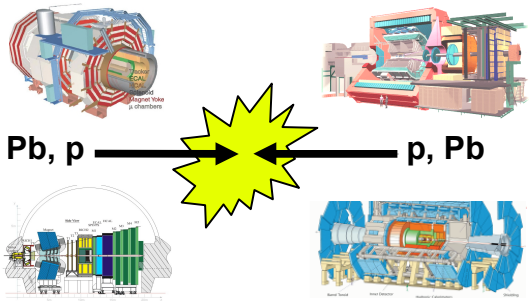
ACAT 2008, Erice, Italy, Nov 2008

The Large Hadron Collider (LHC)

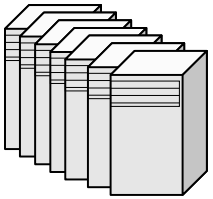
- p-p @ 14 TeV (Pb-Pb @ 2.76 TeV/n)
- $40 \cdot 10^6$ collisions / s, 100 Hz trigger rate
- 10 PB / y raw data (4 experiments)
- (Re)start spring 2009



The LHC Data Flow



~100 Hz
1÷12.5 MB
10 PB / y



MonteCarlo
Production
20÷100% / data

RAW

Reconstruction

ESD

Experiment Reduction

AOD

Individual / Physics Group
Selection / Reduction

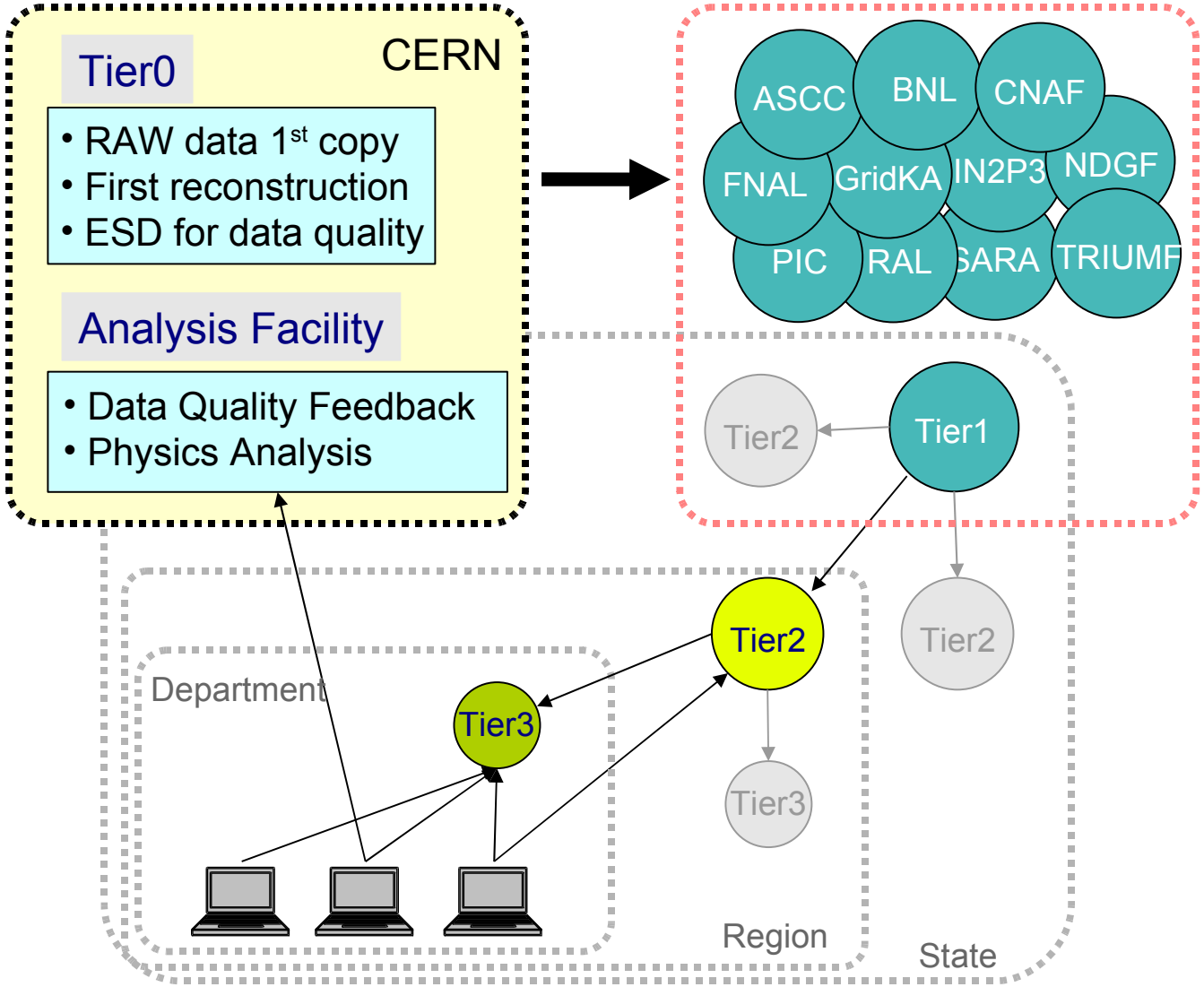
DPD

Event Summary Data
0.025÷ 2.5 MB/event
100÷1000 TB / y

Analysis Objects Data
4÷ 250 kB/event
30÷200 TB / y

Derived Physics Data
1÷10 TB / y

The LHC Data Hierarchical Distribution Model



Tier1, ~10 worldwide

- RAW data 2nd copy
- Re-reconstructions
- ESD, AOD

Tier2, ~5 / Tier1

- MC Simulation
- ESD, AOD repository
- Physics Analysis
- O(100) nodes

Tier3

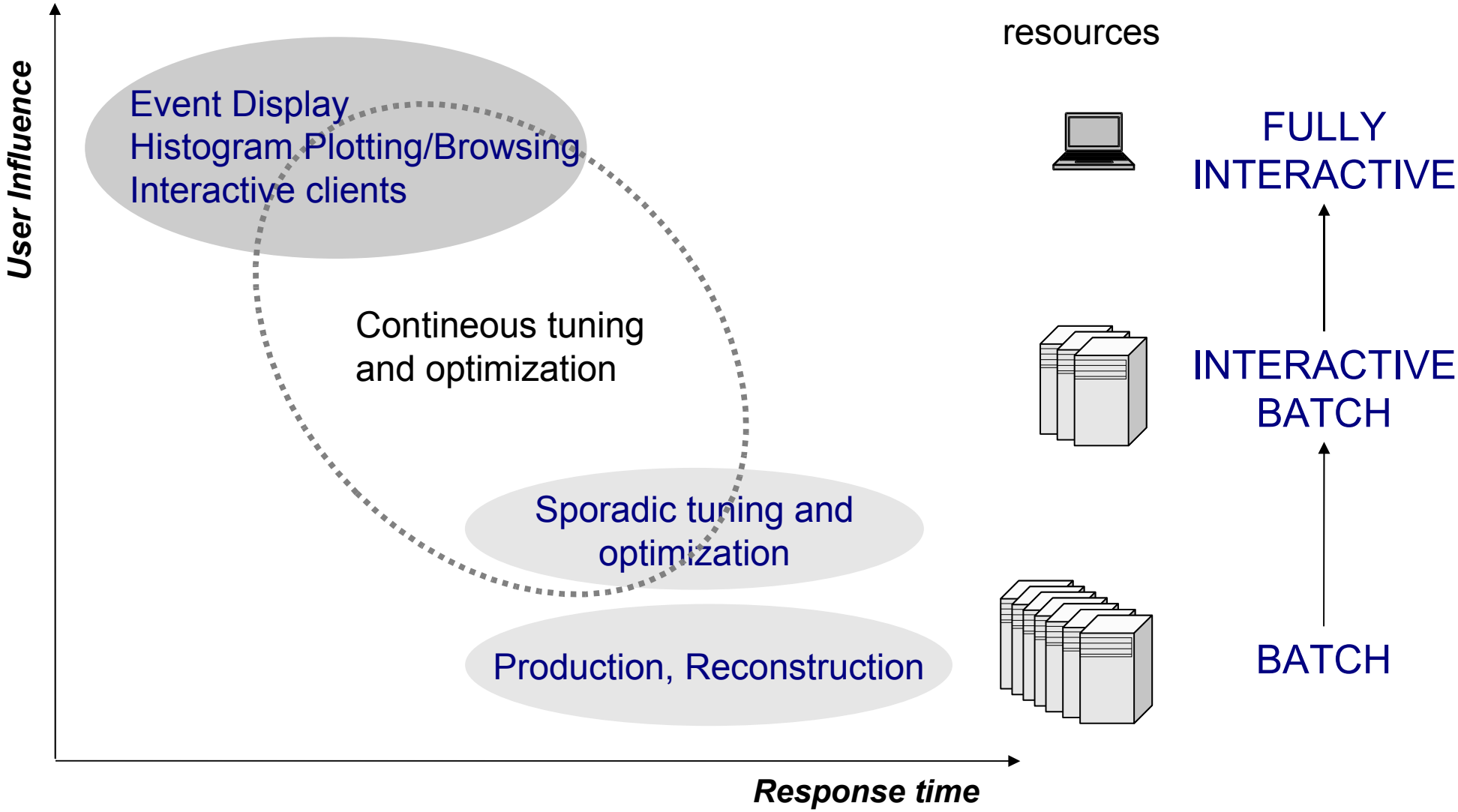
- AOD data set
- Physics Analysis
- O(20÷30) nodes

End-User Analysis activities

- Interactive tasks: desk/laptop
  - Browsing outputs, final fits, visualisation
- I/O bound tasks: data mining
  - O(1÷10 TB) data effectively read
  - ~10h @ 150÷ 250 MB/s (typical disk input rate)
  - ~1h @ ~2 GB/s (10 nodes or ... fancy hardware)
- CPU bound tasks:
  - {Full, Fast} “private” simulations
  - Toy Monte Carlo for systematic studies

Typically embarassingly parallel tasks: just split to get ideal parallel speedup

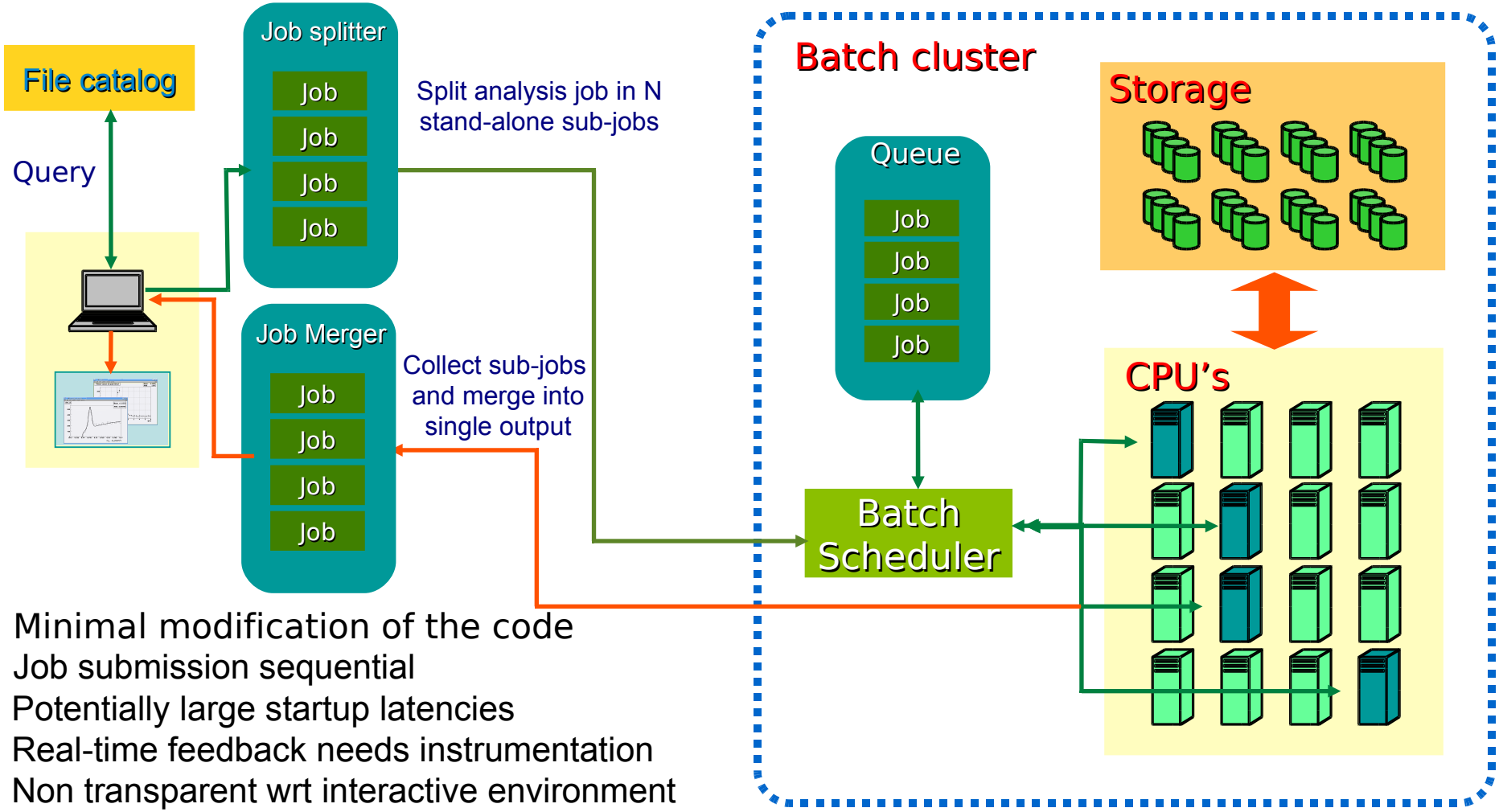
End-User Analysis Scenarios



Current Analysis Solutions

- Batch: all experiments have developed solid solutions
  - GANGA (LHCb, ATLAS)
  - PanDA (ATLAS)
  - CRAB (CMS)
  - AliEn (ALICE)
  - Condor, ...
- Fully interactive:
  - Event displays
  - Applications for analysis/visualization (e.g. ROOT, HippoDraw, JAS, ...)
- Interactive-batch: no common solution
  - Exploit the typical intrinsic parallelism of the task to reduce the response time

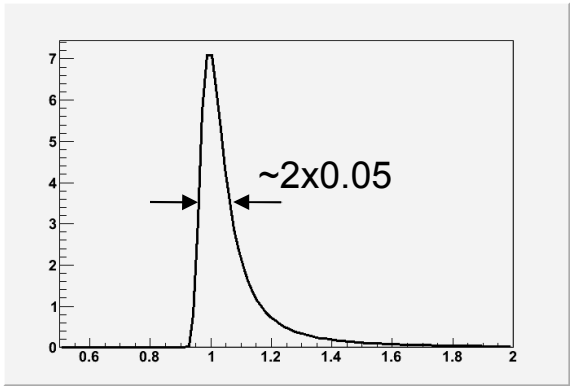
Traditional approach



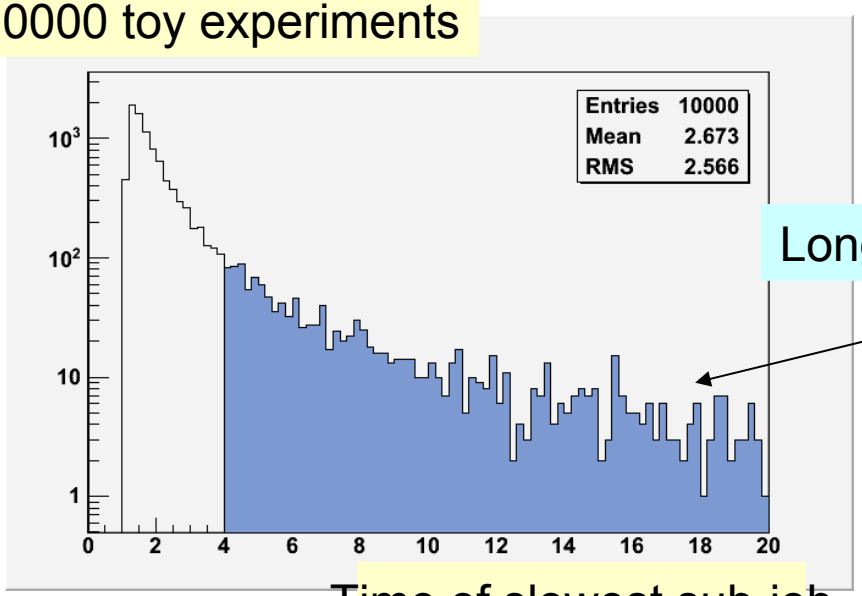
- Minimal modification of the code
- Job submission sequential
- Potentially large startup latencies
- Real-time feedback needs instrumentation
- Non transparent wrt interactive environment
- Potentially heavy setup

Traditional approach: sensitivity to tails

- Last sub-job determines the execution time
  - Basically a Landau distribution
- Example:
  - Total expected time 20h, target 1h
  - 20 sub-jobs, 1h +- 5%



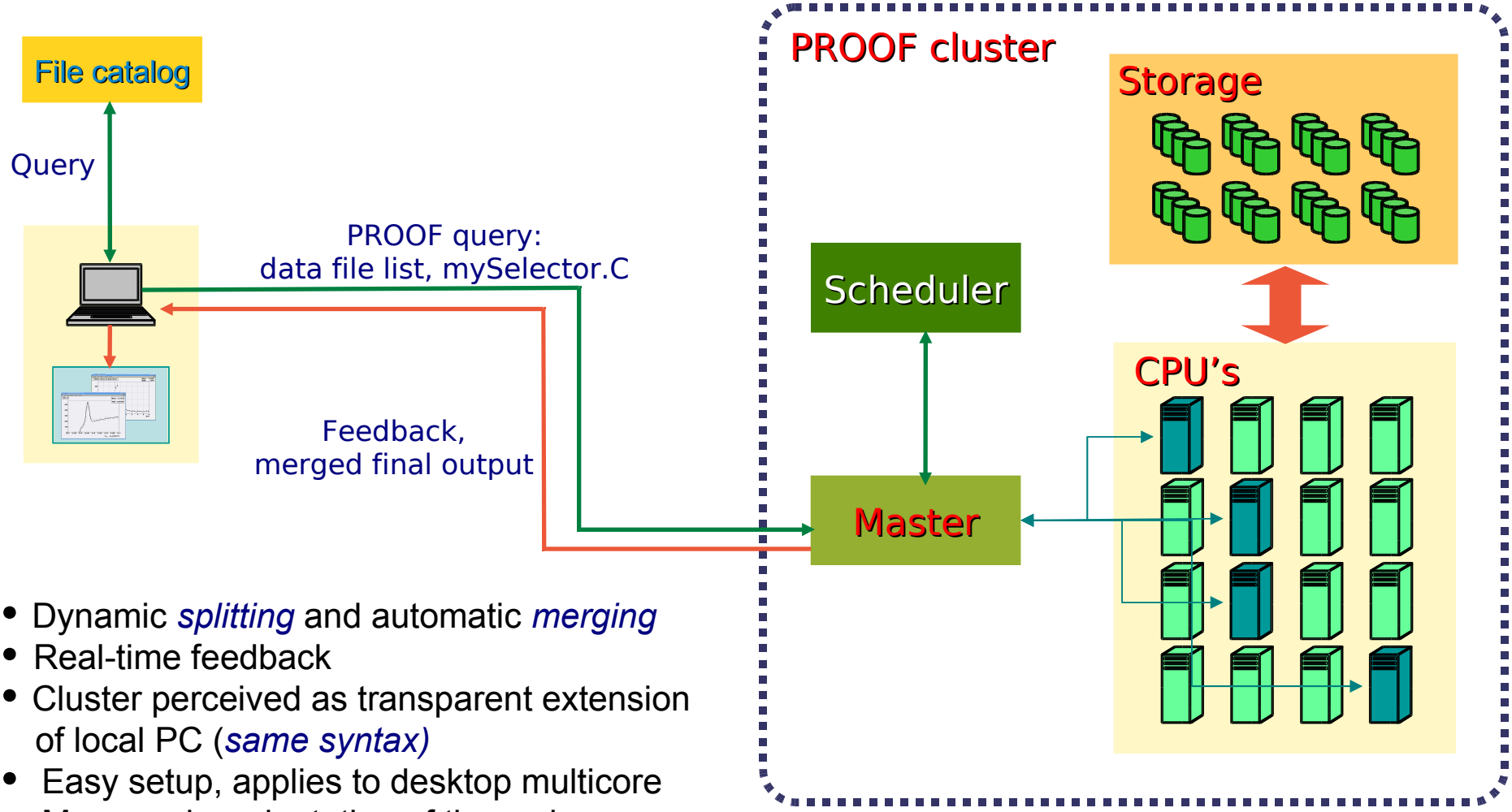
10000 toy experiments



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

Time of slowest sub-job

PROOF approach



- 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 approach: tail control

- Dynamic load balancing
  - Target: all workers finish at the same time
  - Use all free CPU cycles
- Slow workers get less to do
  - Can be discarded and the work re-assigned
- Can stop the job and save the results
  - If the problem cannot be solved

ROOT

- C++ Software Framework providing tools for
  - Storage optimized for HEP data
  - Visualization (2D, 3D, event display, ...)
  - Statistics, math functions, fitting, ...
  - Abstract interfaces (VMC, ...)
- 3 user interfaces: GUI, shell, applications

ROOT shell

```
$ root
root[0]  .x runSel.C(1)
```

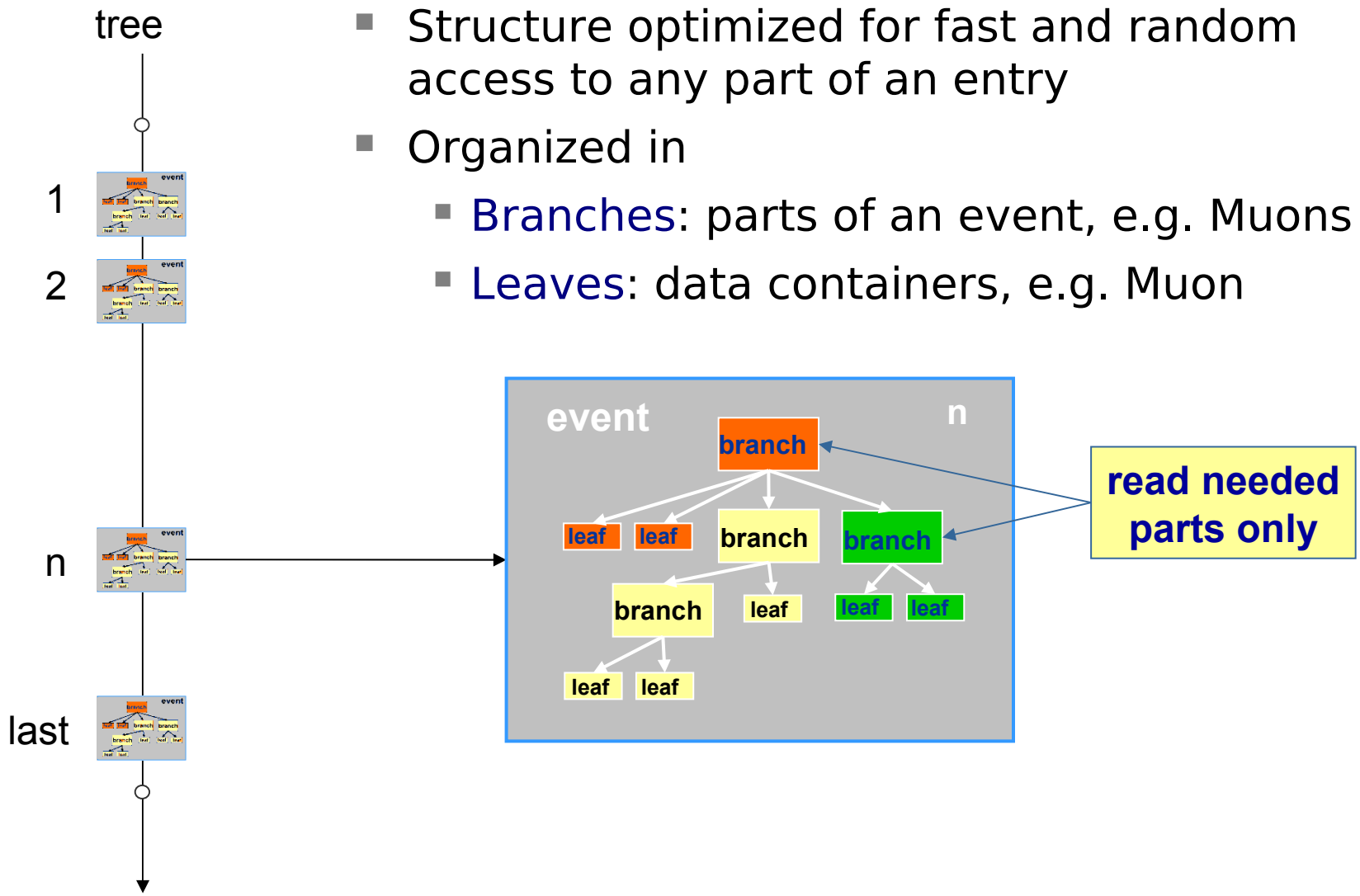
Same runSel.C

Applications

```
#include "runSel.C"
int main(int argc, char** argv)  {
   runSel(1);
  ...
}
```

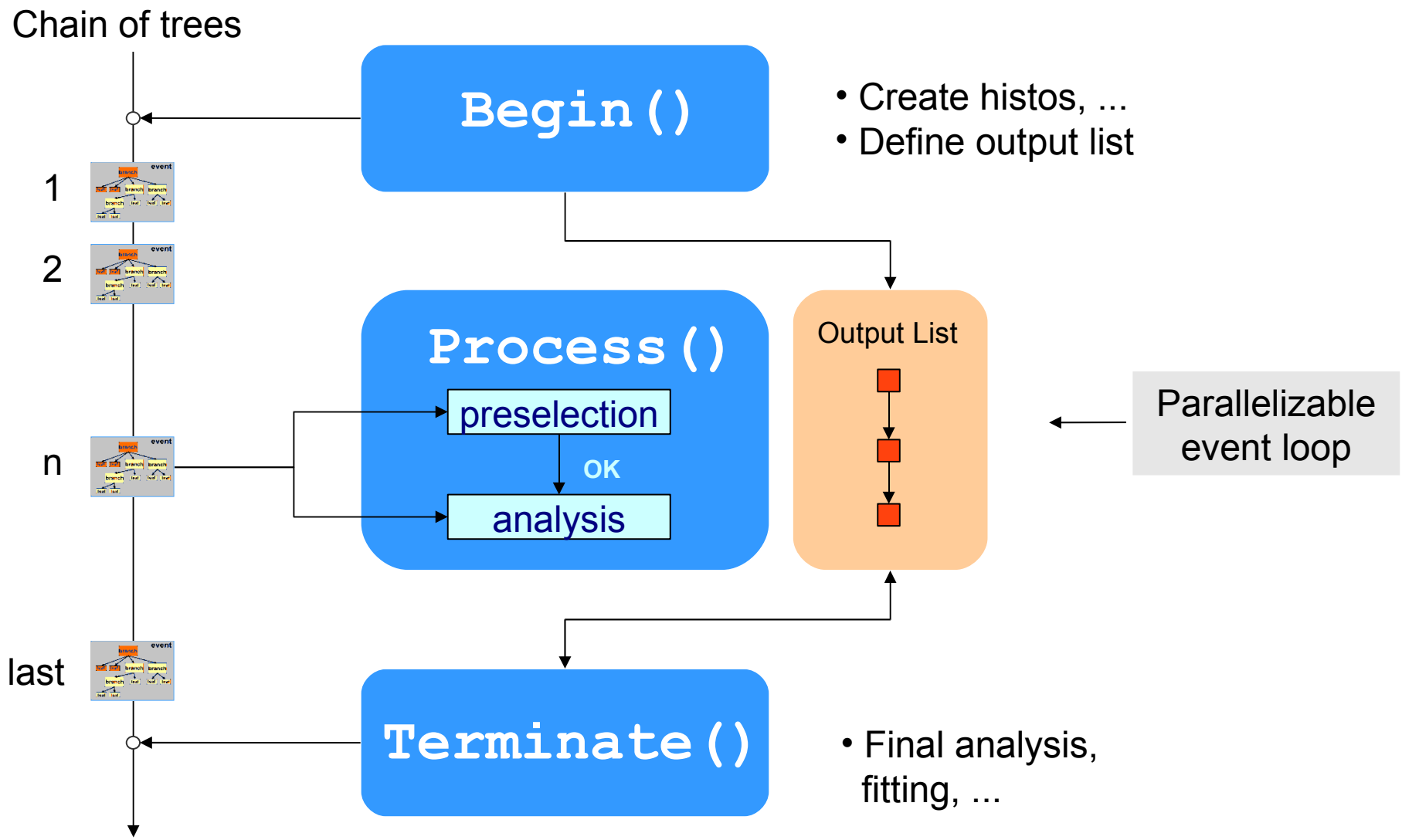
Event stores of all the LHC experiments are based on ROOT

The ROOT trees



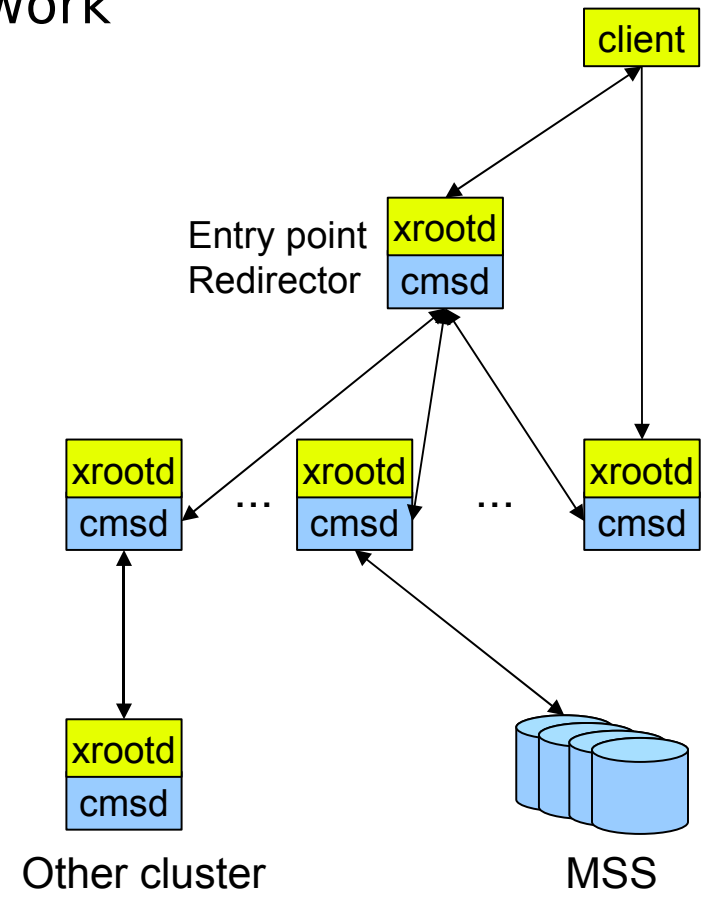
- Structure optimized for fast and random access to any part of an entry
- Organized in
  - Branches: parts of an event, e.g. Muons
  - Leaves: data containers, e.g. Muon

Processing ROOT trees: TSelector framework



Remote Data Access

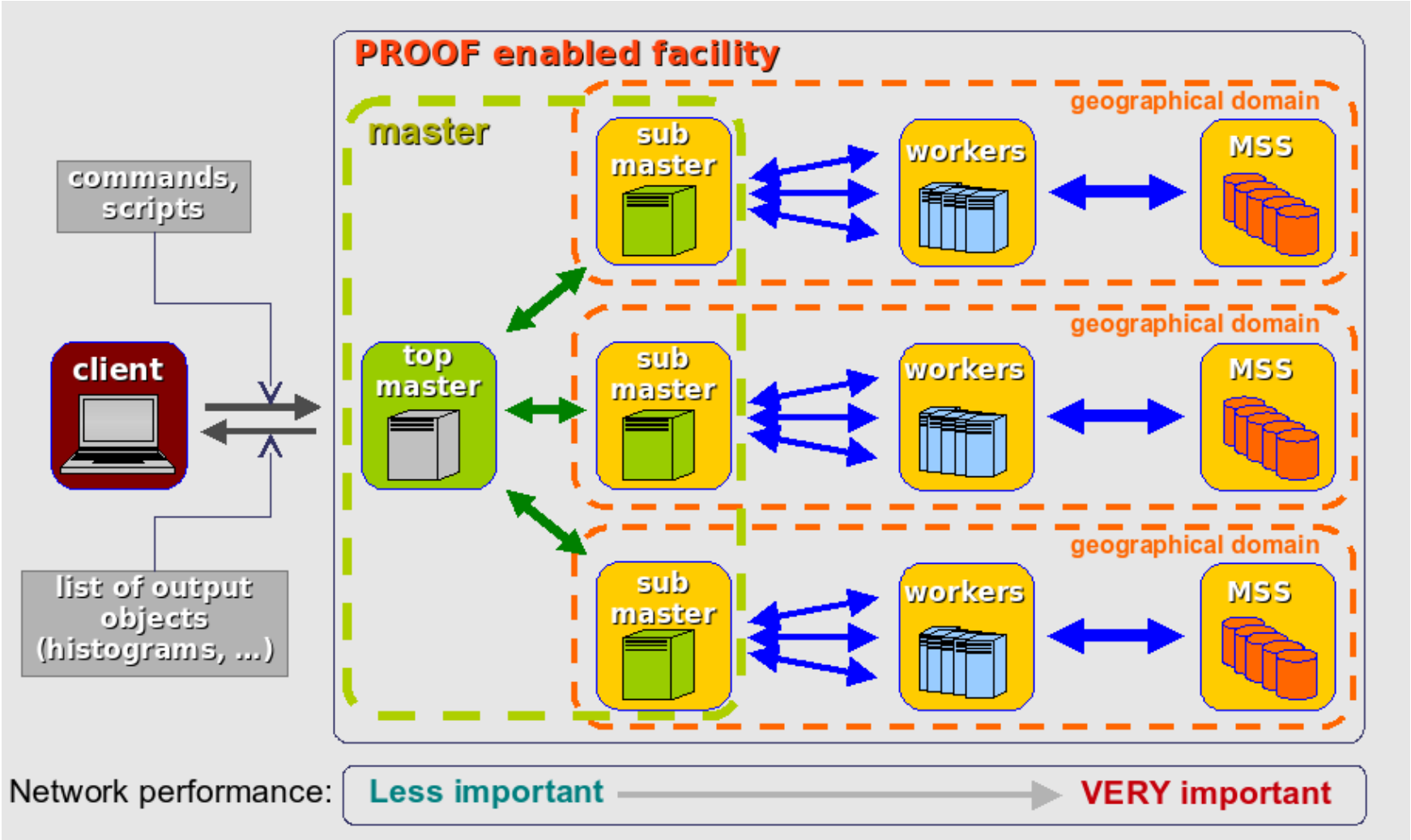
- File based, small sparse block, random access
  - O(10^6) files distributed over the network
  - Only fractions of files typically read
- Requires
  - Low latency (file open, stat, read)
  - Clustering
  - Mass Storage interface
    - Cannot keep everything on disk
- XROOTD/SCALLA
  - Efficient Byte-server system
  - Virtual Mass Storage
    - A. Hanushevsky, F. Furano talks
  - Data Access protocol for LHC



PROOF – Parallel ROOT Facility

- 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

PROOF architecture



PROOF features - 1

- Interactive parallel execution of independent tasks
  - Browsing / processing of datasets
  - Full / Fast / Toy Monte Carlo simulations
- Interactive-Batch
  - Can leave a processing session in the background, disconnect and reconnect later on to check the result
- Real-time Feedback
  - Define a subset of the output objects to be sent back at a tunable frequency

PROOF features - 2

- Flexible authentication infrastructure
  - GSI, Kerberos, Password-based
- Package manager
  - Add software to the system
- Flexible environment setting
- Flexible architecture
  - Adapt to a wide range of cluster sizes from the GRID (gLite interface, gLitePROOF) to multicore desktops
- Optimized 2-tier version for multicores
  - 0-config setup
  - Full access to the additional CPU power

PROOF and Google's MapReduce

- Similar approach
  - N mappers (workers) → M reducers (master, sub-masters)
- Main MapReduce usage @ Google:
  - Parallel grep (map) on huge amount of files
  - Parallel histogramming (reduction) of the results
  - Performance[1]: ~70 TB in ~7 mins w/ ~400 machines
- Many similarities in performance observations
  - Data locality, task granularity, networking performance, error handling, etc etc

(1) J. Dean and S. Ghemawat, *MapReduce: Simplified Data Processing on Large clusters*, Communications of the ACM, Jan 2008, Vol. 51, No. 1

PROOF: impact on existing frameworks

How difficult is to adapt my framework to PROOF?

- PROOF runs the event loop and opens the files
  - Possible Interference with frameworks
- Possible solution (CMS)
  - Modular approach to analysis algorithms and input/output handling
  - Same user code can be run in the experiment and PROOF/TSelector framework
- TSelector framework is flexible
  - Can be used just to schedule tasks
  - Smooth transition typically possible

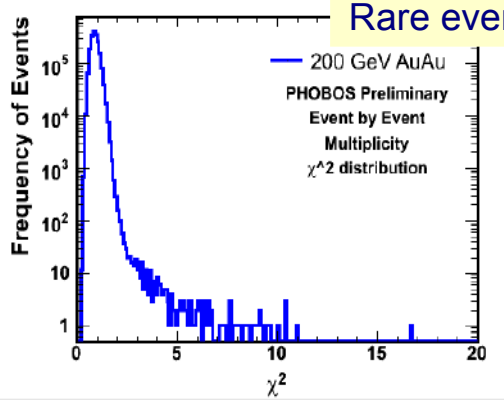
PROOF: who is using / testing?

- Developed/used by PHOBOS for end-user analysis since 2003 (M. Ballantjin et al.)
- At LHC
  - ALICE : integral part of the computing model
    - CAF, GSI (see talk by A. Kreshuk)
  - ATLAS : alternative analysis model based on PROOF
    - Facilities at BNL, Wisconsin, LMU, Munich, UTA
  - CMS : analysis framework adapted to PROOF
- CERN-IT
  - Under discussion PROOF service provision at CERN

Performance at PHOBOS

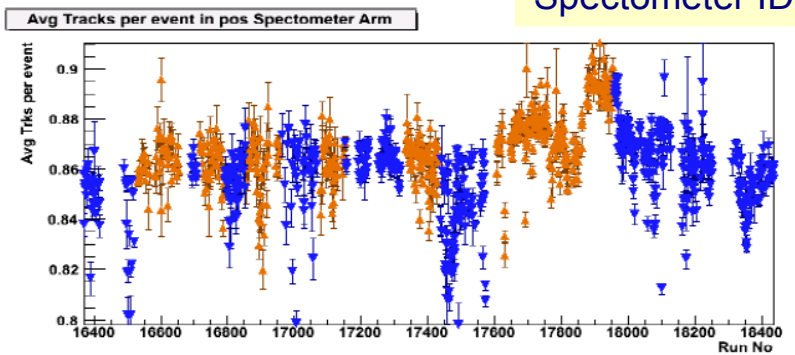
Collected by M. Ballintijn, MIT

Rare event search



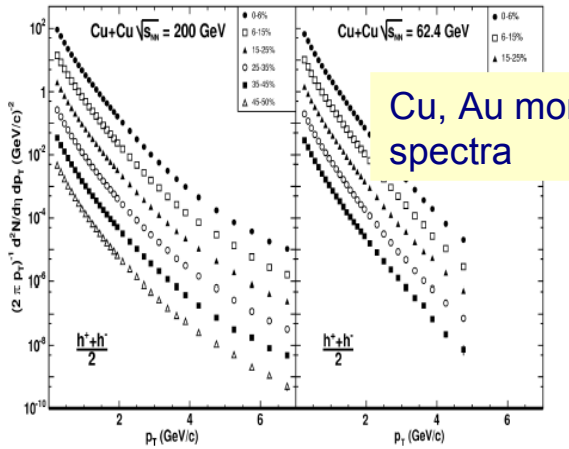
4.5 TB → 60 min (150 workers)

Spectometer ID



13.5 TB → 70 min (100 workers)

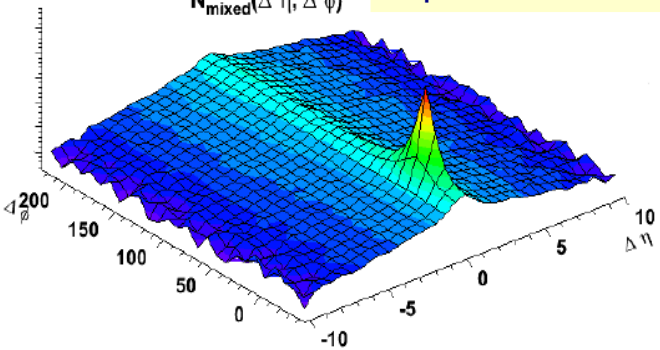
Cu, Au momentum spectra



13.5 TB → 45 min (100 workers)

2-particle correlations

$$C(\Delta\eta, \Delta\phi) = \frac{N_{real}(\Delta\eta, \Delta\phi)}{N_{mixed}(\Delta\eta, \Delta\phi)}$$



1.5 TB → 75 min (100 workers)

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
  - ~50 users registered
- See talk by A. Gheata

GSI Analysis Facility, Darmstadt

- 160 cores, 150 TB Lustre
- Data analysis, TPC calibration
- 5-10 users
- See talk by A. Kreshuk

ATLAS

Wisconsin

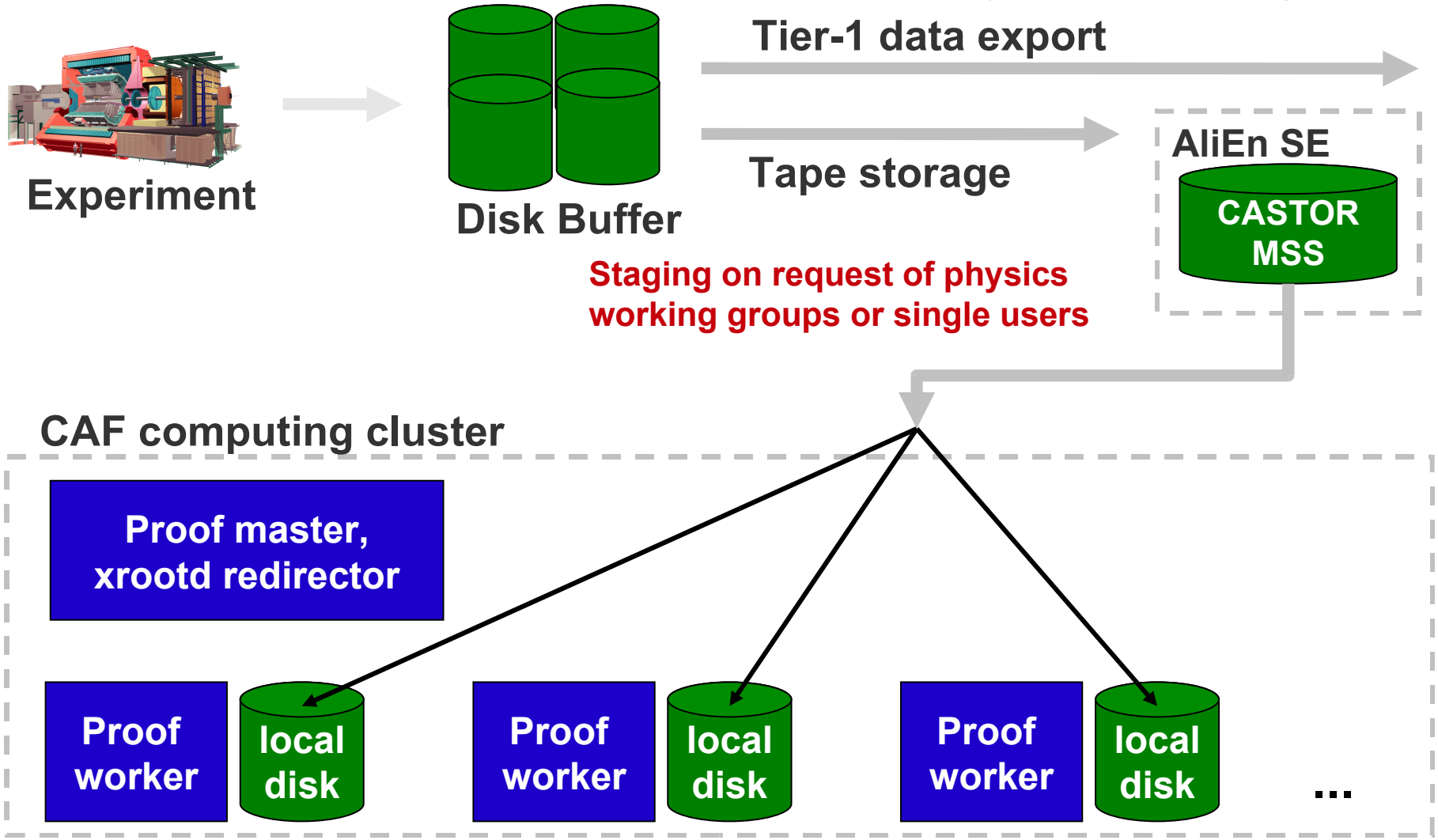
- 200 cores, 100 TB, RAID5
- Data analysis (Higgs searches)
- I/O perfomance tests w/ multi-RAID
- PROOF-Condor integration

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

ALICE CAF schematically

Courtesy of J.F Grosse-Oetringhaus, CERN



Issues addressed by ALICE/ATLAS tests

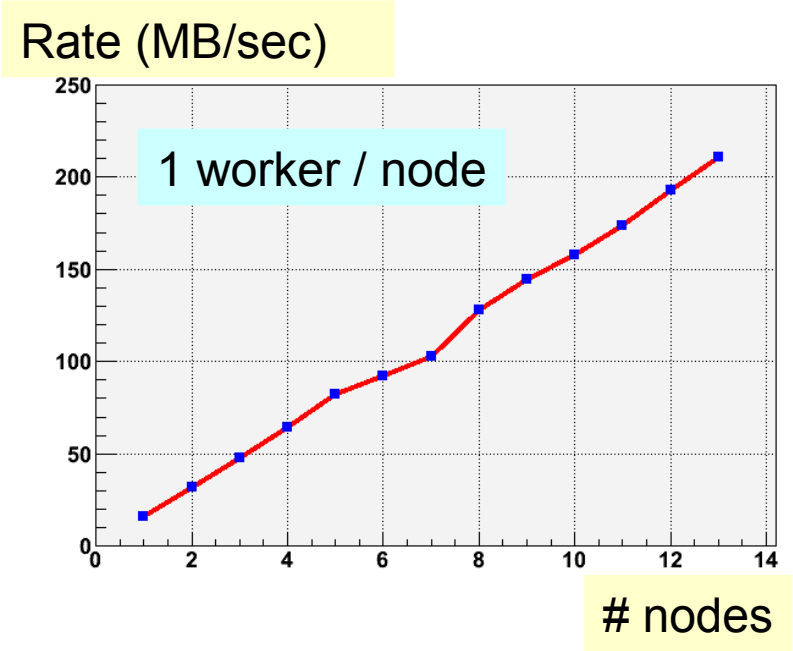
- Performance
- Dataset handling
- Scheduling

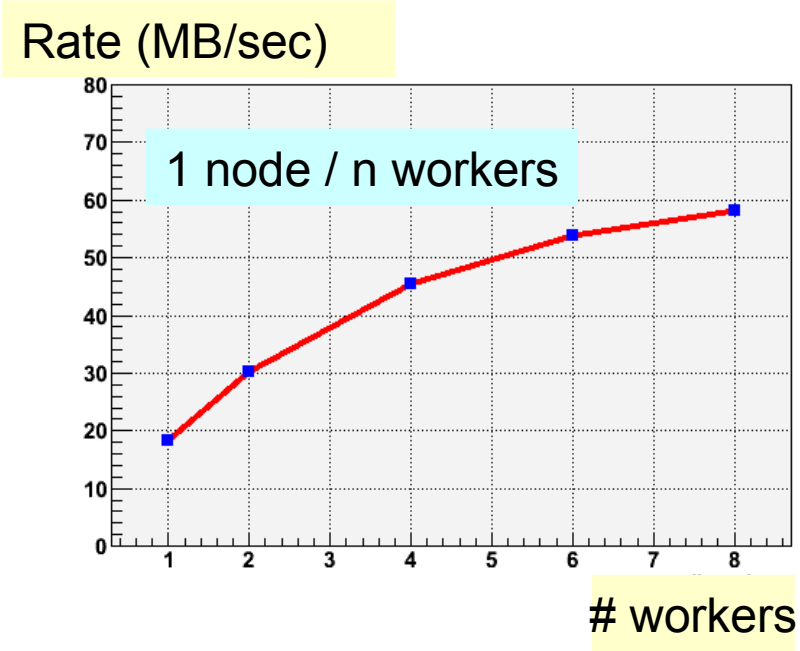
Issues addressed by ALICE/ATLAS tests

- Performance
- Dataset handling
- Scheduling

Performance: scalability at ALICE CAF

- ESD analysis, ~1 TB (~25% read)
  - ~10 min w/ 26 workers





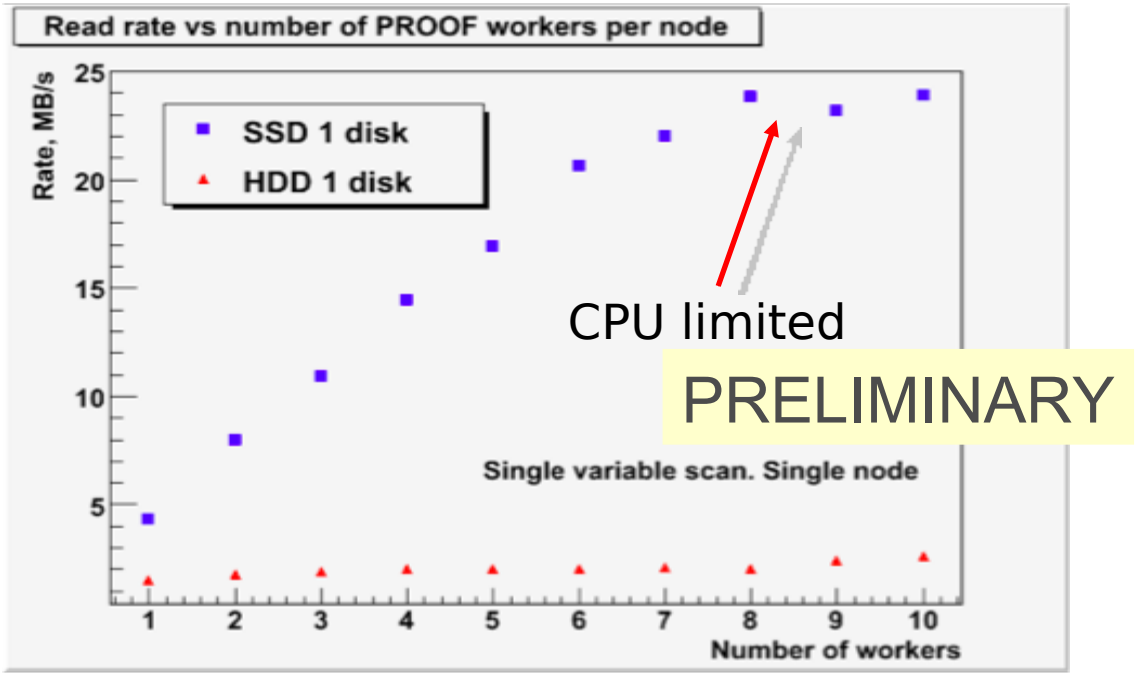
- I/O limitations limits scalability inside a machine
- But system behaviour predictable

Performance using Solid State Disks (SSD)

Courtesy of S. Panitkin, BNL

BNL PROOF Farm

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

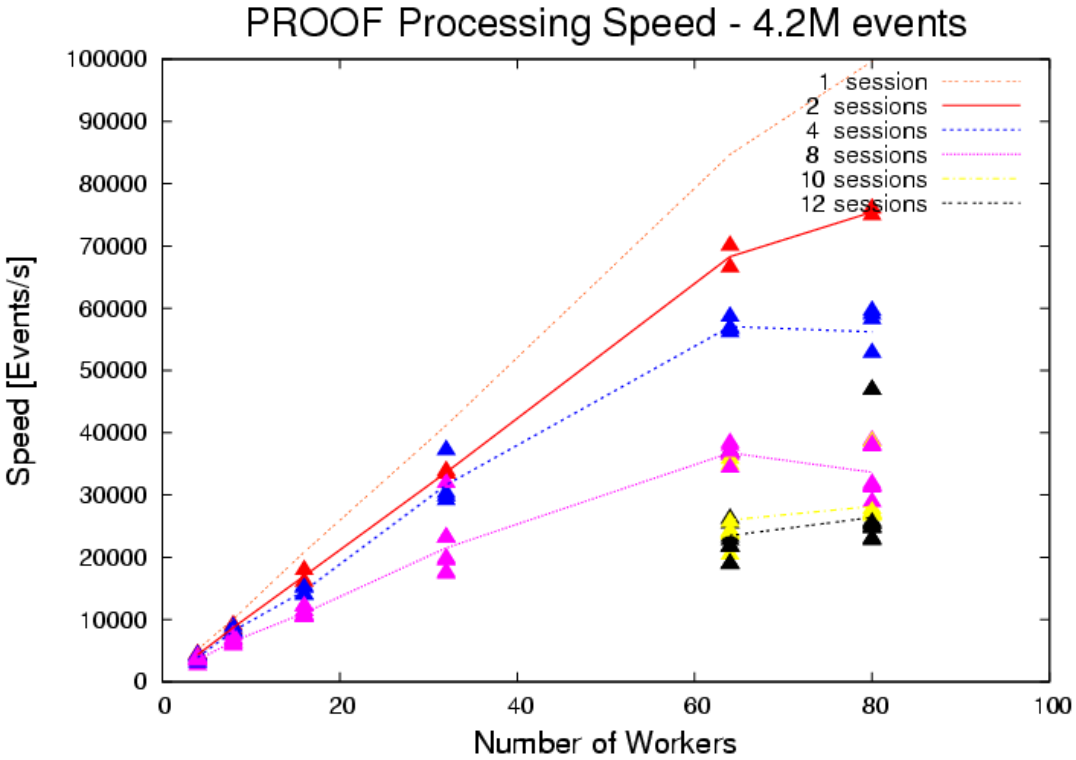


- SSD holds clear speed advantage
  - ~ 10 times faster in concurrent read scenario
- Price start becoming affordable

Performance: ATLAS analysis

Courtesy of G.C. Montoya, Wisconsin

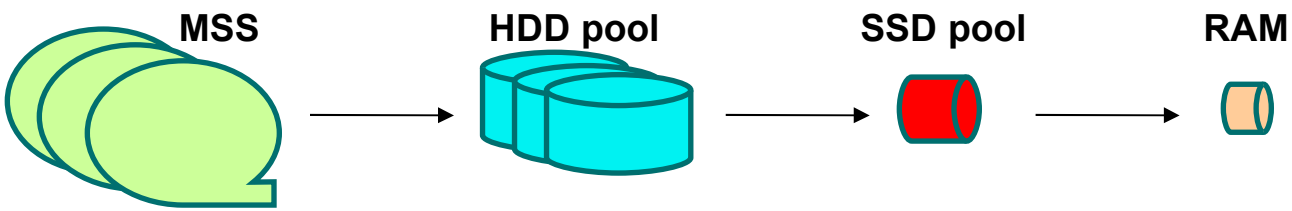
- Higgs 4-lepton analysis
- 50 nodes, AMD 64bit 4x, 4 GB RAM
- 4.5 M events, 68 GB
- 845 files
- Analysis include TMinuit fit



- 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

Considerations about data locality

- Data locality clear advantage
  - Remote access may work well but still behind
- Current favoured model
  - Local, unbackedup, pool populated from a MSS
    - More efficient to copy files once and read many
  - Requires pool management
    - Experiment policy
- Affordability of SSD technology makes possible an additional layer



Issues addressed by ALICE/ATLAS tests

- Performance
- Dataset handling
- Scheduling

Datasets

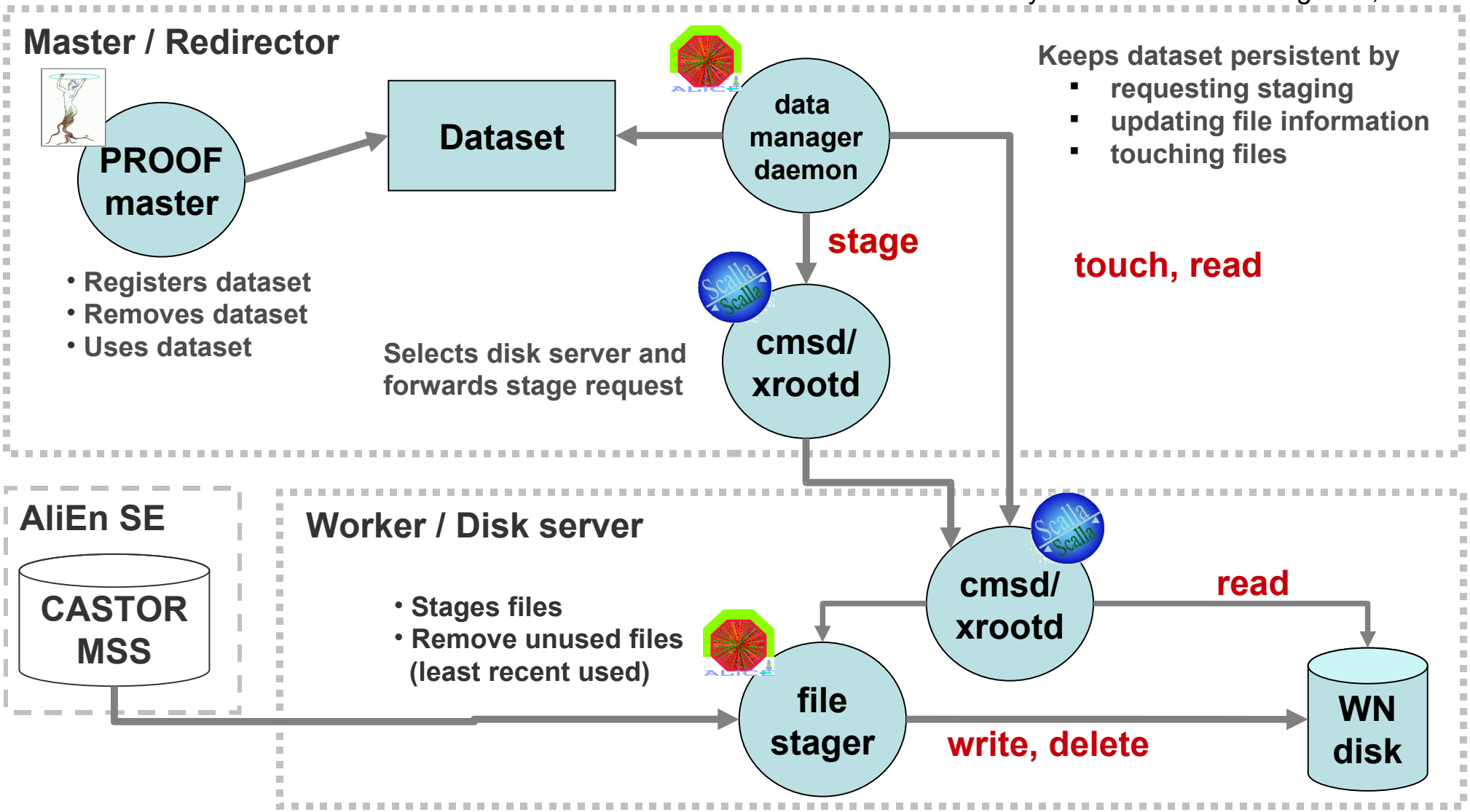
- Dataset: collection of data with homogenous conditions
  - Typically identified by name
- Different levels
  - Data runs of similar running conditions
  - Data / Simulation tags for Physics Working Group
  - Users making their own reductions
    - E.g. topological broad selection of Higgs candidates
- Information relevant to PROOF
  - List of files and their location
  - Meta information: tree names, entries, sizes, ...

Dataset management

- Dataset manager
  - Handle datasets
    - Register a new dataset or remove an existing one
    - Retrieve information
    - Verify the availability of the files
    - ...
- 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



Dataset handling at ATLAS (proposal)

Courtesy of Neng Xu, Wisconsin

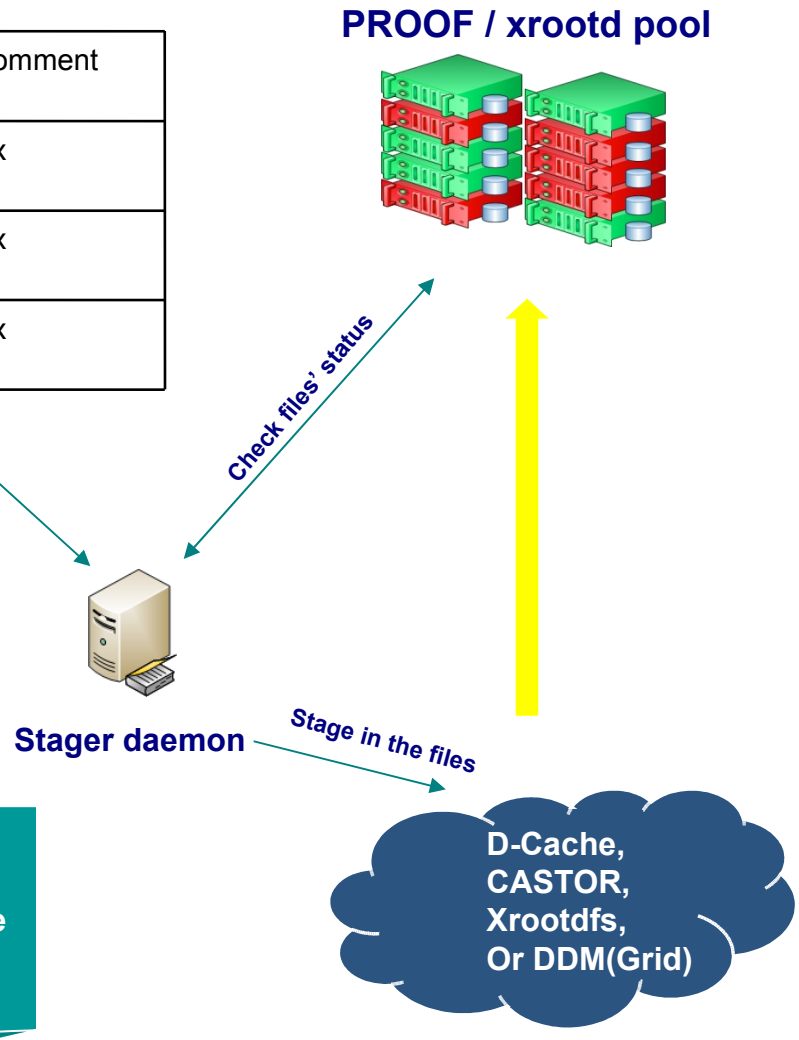
Database for Datasets

Dataset name	#of req	# of file	Last req date	Status	comment
mc08.017506.PythiaB_bbmu6mu4X.evgen.e306	2	50	2008/2/20	waiting	xx
mc08.017506.PythiaB_bbmu6mu4X.evgen.e306	1	50	2008/2/20	waiting	xx
mc08.017506.PythiaB_bbmu6mu4X.evgen.e306	1	500	2008/2/20	waiting	xx

PROOF datasets requests

```
mc08.017506.PythiaB_bbmu6mu4X.evgen.e306   500
mc08.017506.PythiaB_bbmu6mu6X.evgen.e306   400
mc08.0175068.PythiaB_bbmu6mu4X.evgen.e306  30
mc08.017506.PythiaB_bbmu6mu4X.evgen  50
mc08.017888.PythiaB_bbmu6mu4X.evgen.e306  100
mc08.017506.PythiaB_bbmu6mu4X.evgen.e306   120
```

- Dataset stage-in has priority which depends on the number of requests, number of files, waiting time, etc..
- May be synchronized w/ Condor jobs submission in "Held" state: the Stage Server release them once the dataset is staged into the PROOF / Xrootd pool.



Issues addressed by ALICE/ATLAS tests

- Performance
- Dataset handling
- Scheduling

Scheduling

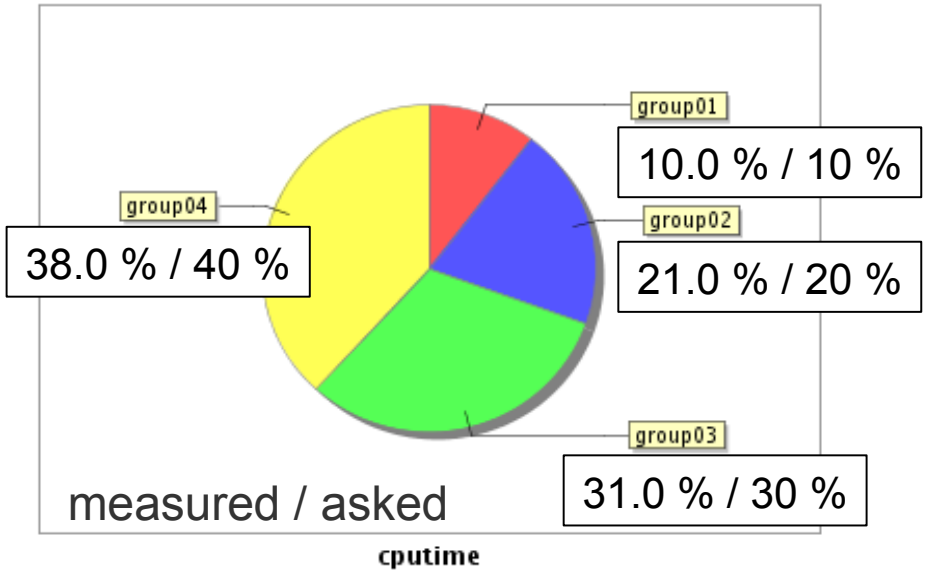
Levels of scheduling

- Control how resources are shared between running sessions
  - Fair-sharing
  - Enforce experiment policies
- Control how many resources to assign to a session
  - Increase throughput by avoiding congestions
  - Enforce experiment policies

Scheduling: resource sharing

- Based on group priority information
- Renicing technology
- Central control w/ monitoring system (e.g. MonAlisa)

- Example: stress-test run 4 groups over 1 day



Scheduling: resource assignment

- System scheduler does well when

  # worker processes ~ # cores

- Controlled scaling wrt # workers and wrt # sessions
  - Can predict what one gets given the number of effective worker processes
- Needs control on the number of effective processes
  - Internal queuing
  - Delayed query processing startup, check pointing, ...

Beyond data analysis?

Additional ALICE tests

- Calibration
  - Fast feedback on detector conditions
  - 1÷2 TB of data processed in ~20 mins (160 workers)
- Reconstruction
  - Fast feedback on data quality
  - New use case
    - Large input condition data
    - Large outputs
    - Variable event size

Sharing resources w/ batch systems?

- Integration w/ Grid middleware
  - gLitePROOF (A. Manafov, GSI)
    - Use gLite to start workers adding up dynamically
- Integration w/ Condor (US ATLAS interest)
  - Joint project w/ Condor team and ATLAS Wisconsin
- Virtual PROOF (see S. Bagnasco talk)
  - Use virtual machines to provide LCG and PROOF services

Conclusions

- Data Analysis at LHC is challenging
  - Large amount of data, large number of users, complex analysis environment
  - Need to optimally exploit the resources
- PROOF emerges as a complement to batch systems addressing use-cases where fast turn around is needed
- Lot of developments driven by experiments addressing issues of general interest
  - Data access, multicore exploitation
- The system is getting ready for the real data

Credits

- CERN / PH-SFT: R. Brun, J. Iwaszkiewicz, F. Rademakers
- ALICE: J.F. Grosse-Oetringhaus, M. Meoni
- ATLAS: Neng Xu, G.C. Montoya, S. Panitkin
- F.Furano, A. Hanushevsky
- ...

Link

- http://root.cern.ch/twiki/bin/view/ROOT/PROOF