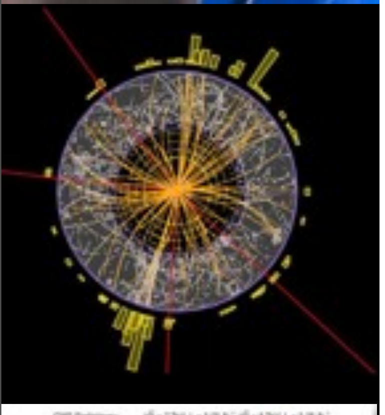
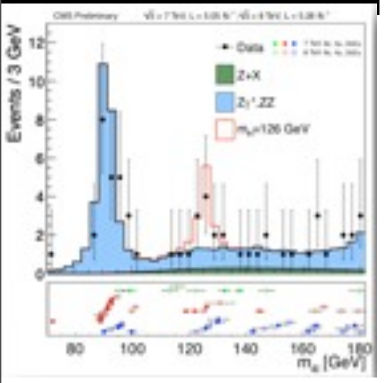
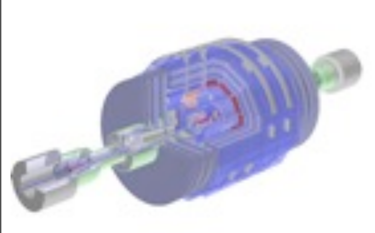






PROOF
Lessons Learned And Future Directions

G. Ganis
CERN PH-SFT

ROOT Users Workshop
Saas Fee, 11-14 March 2013

Gerardo.Ganis@cern.ch

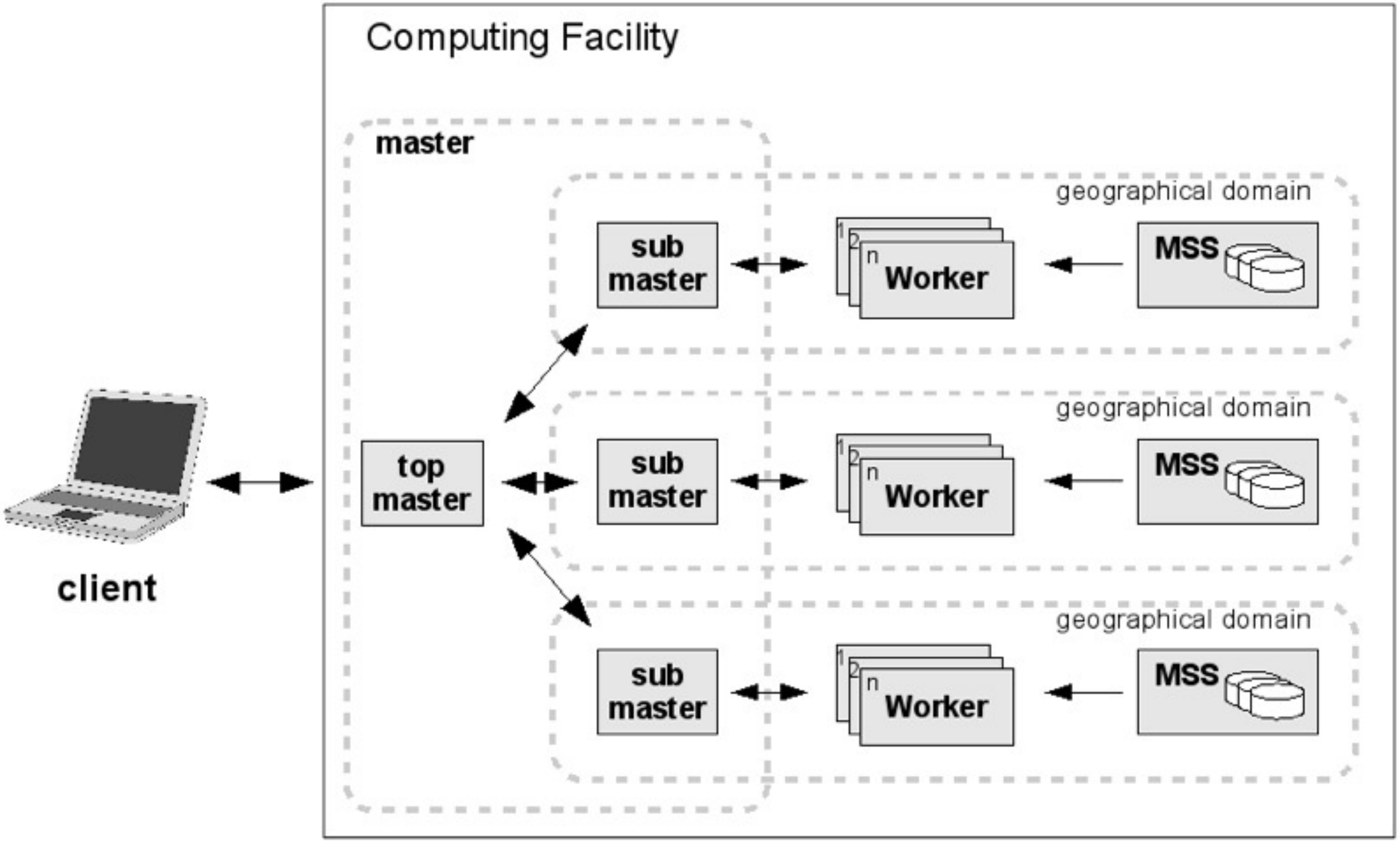
Outline

- Reminder
- The PROOF@LHC experience
  - PROOF-Lite
  - ‘Static’ AFs
  - PoD usage
- Current developments
- Summary

From where all this started

- HEP data samples are large
- Data mining is I/O bound but embarrassing parallel
- To increase effective I/O bandwidth:
  - Job splitting
    - Statically, a priori, push model (batch)
    - Dynamically, pull architecture: optimizes resource utilization but requires controller
  - Data locality
    - Real way to get large aggregate I/O rates
- PROOF: dynamic splitting for the ROOT way
  - TTrees for data, TSelector for code
  - Exploit data locality

Standard PROOF architecture



Not Important

Good connection

Very Important

Expectations

- C cores, U users, N cycles
  - cycle = files, events, ... units of work
- Average execution time $\overline{T}$

$$\overline{\mathrm{T}} = \overline{\mathrm{T}}_{\mathrm{init}} + \frac{\mathrm{U}}{\mathrm{C}} \cdot \mathrm{N} \cdot \overline{\mathrm{T}}_{\mathrm{cycle}} + \overset{\text{merging}}{\boxed{\frac{1}{\mathrm{F(U,C)}} \cdot \overline{\mathrm{T}}_{\mathrm{term}}}}$$

- Initialization ($\overline{T}_{init}$) affects each process/core
  - Can be big (load of calibrations, geometry, ...)
- Finalization ($\overline{T}_{term}$) includes merging
  - F(U,C)≥1 measures the degree of parallelization
  - Can be the bottleneck (more later)

PROOF technology

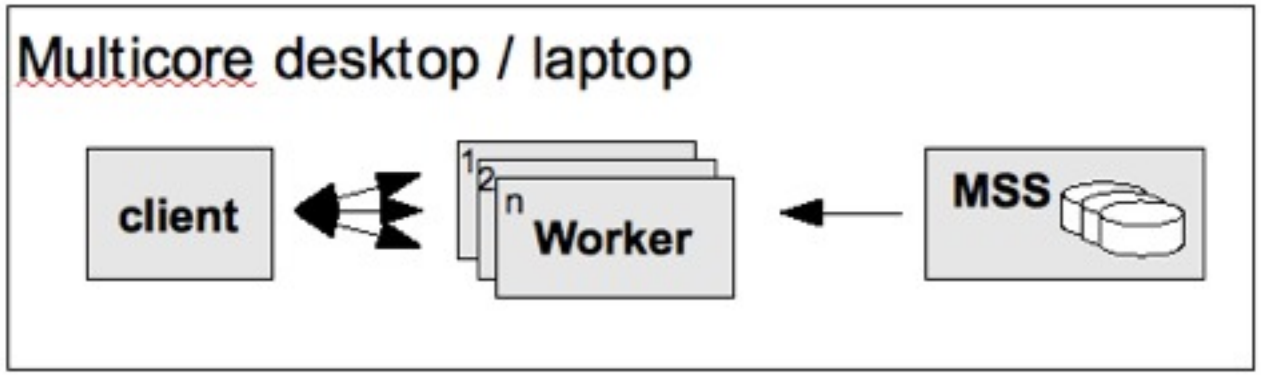
- Address embarrassing parallel tasks
  - Data mining, MC production, toy-MC, fits, ...
- Multi-process
  - Thread safe, range from desktop to super-cluster
- Interactive coordination of concurrent ROOT sessions
  - Dynamic load balancing
  - Dynamic adaptation to available resources
  - Merging optimizations
  - Realtime feedback
- Data locality
  - Scaling I/O rates
- ROOT User Interface

Standard PROOF components

- Connection layer
  - XRootD components used for low-level networking, {authentic,authoriz}ation, session management
  - Access to working areas
- Kernel
  - Protocol for Client-Master-Worker interaction
  - Setup, Work distribution
  - Results collection and merging
- Main ROOT components used
  - High-level networking (TMonitor, e.g. *select)*
  - I/O streaming for object exchange
    - TMessage : public TBuffer
  - Merging infrastructure

PROOF-Lite for Multi-Cores

- PROOF-Lite



  - 0-config setup (no config file, no daemon)
  - Communication via UNIX sockets
  - Same API as for standard PROOF

- Today’s desk-/laptops: multicore, performant storage
  - Fast HDD or SSD
  - Large RAM: can be used as full cache in many cases

The LHC experience

The LHC analysis context

- Many users
  - Scheduling, encapsulation, ...
- Non-dedicated facilities
  - Department clusters, grids, clouds
  - Sharing resources w/ other technologies
- Distributed data model
  - Dataset management
- Large outputs

PROOF @ LHC

- ALICE
  - Analysis Facility (AAF): full-featured model of Tier3
  - Online reconstruction
- ATLAS, D3PD analysis (standalone, SFrame, ...)
  - Dedicated clusters (SLAC, NYU, Barcelona, Milano, ...)
  - PROOF-Lite users
  - PoD under CernVM-FS
    - w/Condor@US, w/LSF@CERN, w/gLite (PanDa)@INFN
- CMS, private DPD
  - Some dedicated clusters (Oviedo, ...)
  - Bari, Firenze: mostly PROOF-Lite
- Indirect users, e.g. RooStat

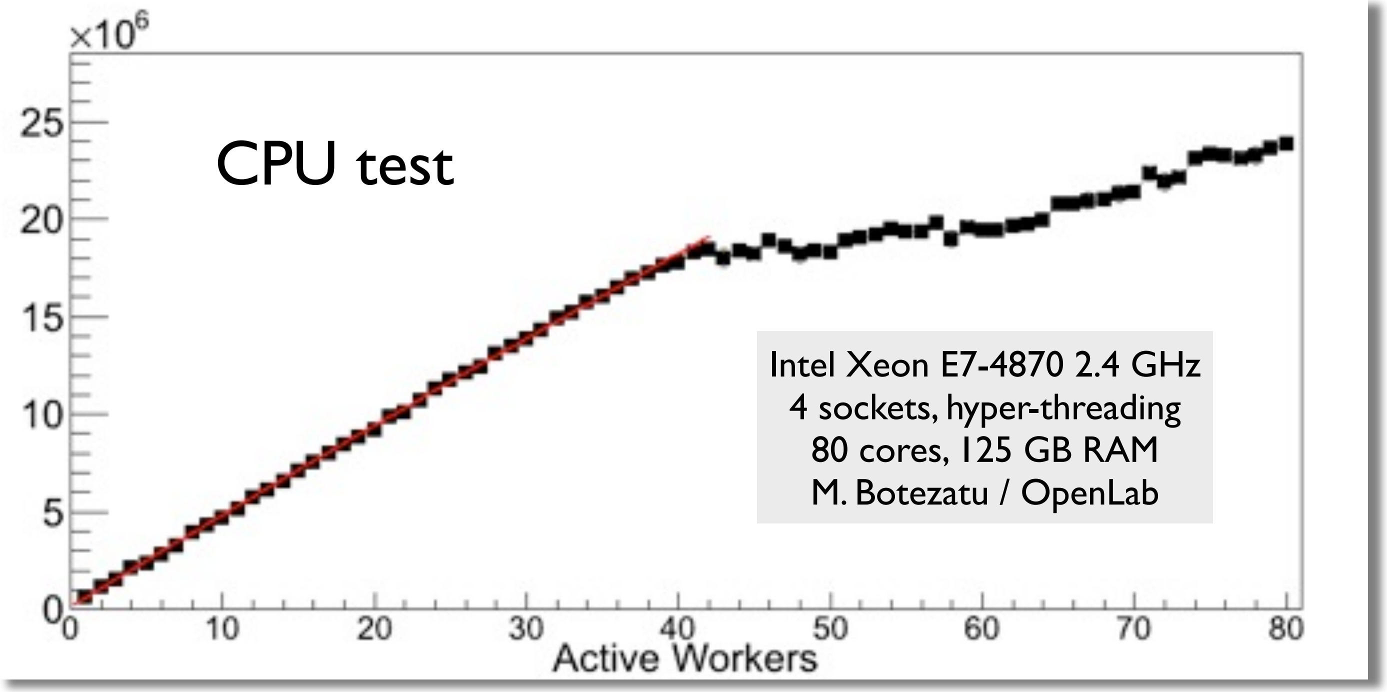
PROOF is TSelector-based

- Controls the event-loop
  - Possible conflict with experiment frameworks
- Requires to adapt to the TSelector paradigm
  {init, process, terminate}
- Typical solution: TSelector-based framework invoking user tasks
  - E.g. ALICE, CMS
- Or use experiment framework to produce TTrees
  - E.g. D3PD in ATLAS

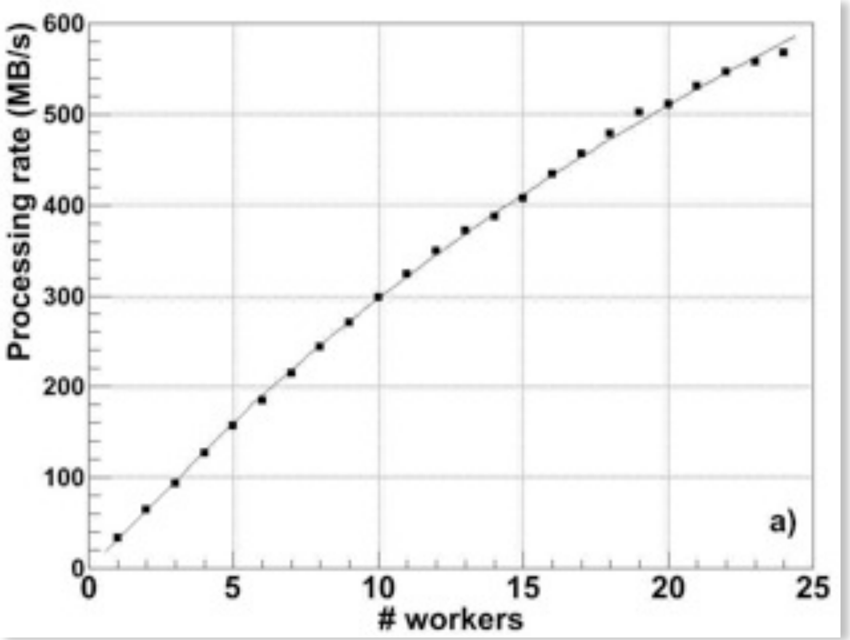
PROOF-Lite

- Used mostly with private sets of TTree's
  - But can be integrated into experiment framework
    - E.g. ALICE, CMS Bari, ...
- Straight-forward to start and easy to use, but some things felt as unnatural by users
  - Loading libraries on client does not do it on workers
  - Workers do not see same paths as the client
    - Need full paths in specifying files for workers
  - Draw functionality not exactly the same
  - Context not reusable for same selector and different parameters
- Large init data just replicated (memory issues)
  - Need fork after 1st event

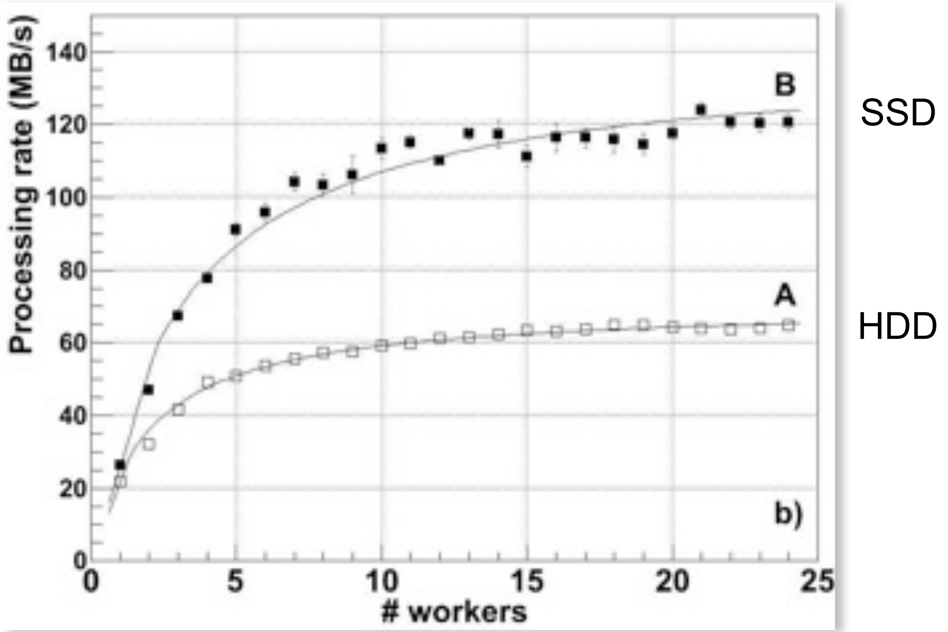
Benchmarking with PROOF-Lite



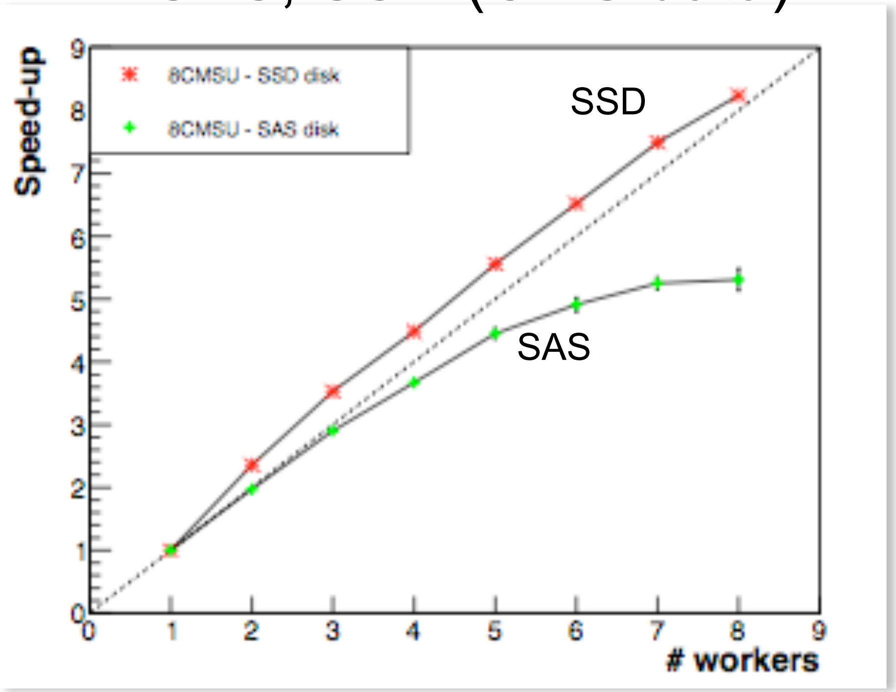
RAM



HDD, SSD



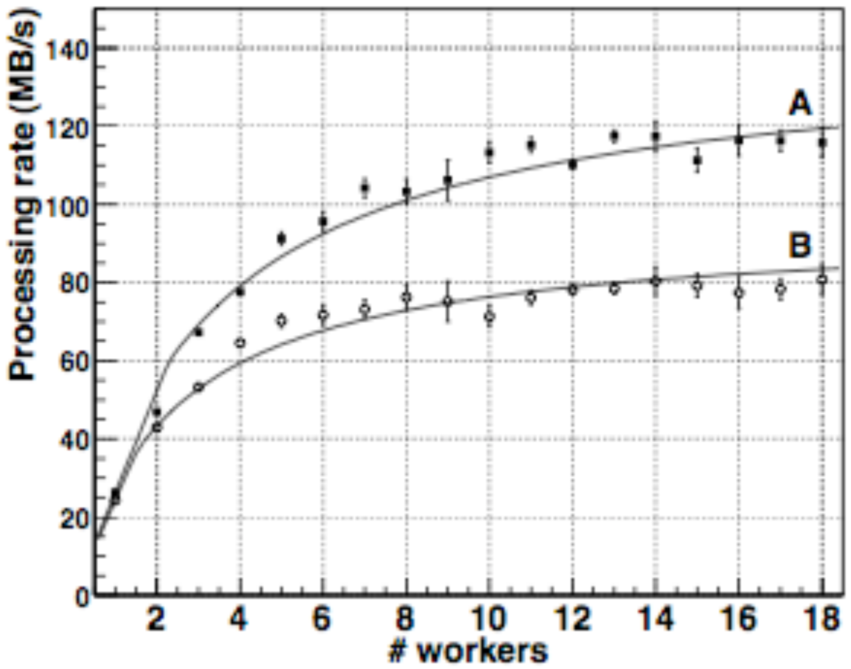
SAS, SSD (CMS data)



Barbone, Donvito, Pompilii CHEP2012

Benchmarking with PROOF-Lite (2)

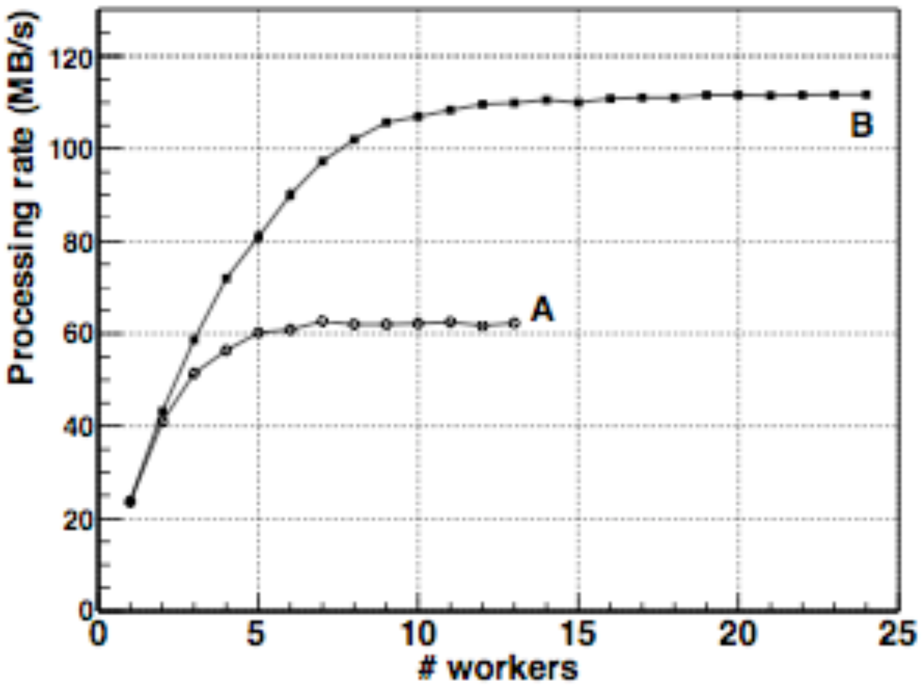
TTreeCache impact



ON

OFF

Reading over Network



5 servers

1 server

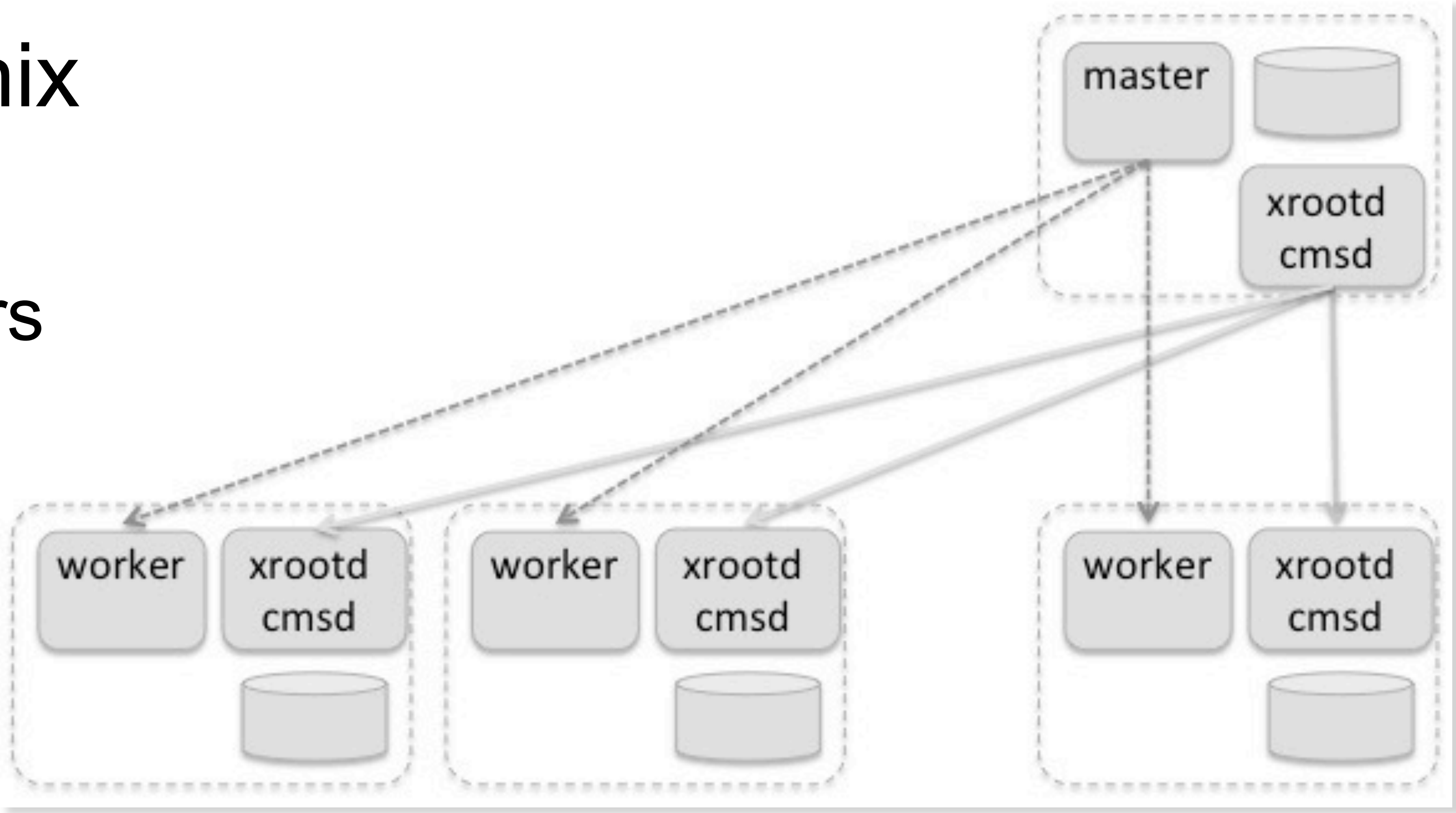
- TTreeCache enabled:
  less fragmented readout

- 1 xrootd server:
  bottleneck is the I/O on the server disk
- Cluster of 5 servers:
  bottleneck is network bandwidth

Toolkit for benchmarks:
http://root.cern.ch/drupal/content/proof-benchmark-framework-tproofbench

The ‘static’ Analysis Facilities

- Cluster of dedicated or shared physical nodes
  - Department cluster, experiment dedicate farm
- Some local storage
- Sandboxing à la unix
- Basic scheduling
  - FIFO, max # workers
- Basic monitorig



Example: the ALICE AF

- Full-featured, PROOF-enabled facility
  - Automatic installation suite
  - Local XRootD storage, dynamically populated from AliEn
  - Dataset manager daemon
  - Monitoring (MonAlisa)
- 7 instances: CAF, JRAF, KIAF, LAF, SAF, SKAF, TAF
  - Local storage: ~500 TB
  - TAF based on virtual machines
- User support: Savannah + mailing list

Issues addressed at static AFs

- Datasets
- Output merging
- Daemon stability
- Usability
  - Debugging user errors
  - User code enabling (package manager) (*)
- Also (*)
  - Monitoring
    - Query summary + dataset info
    - SQL or MonAlisa backends
  - Installation

(*) Not covering further

Datasets

- Dataset: named collection of files (run, MC set, ...)
  - In ROOT: TFileCollection, TFileInfo (URLs, metainfo)
- Convenient:
  - Avoid manual building of TChain
  - Reduce errors in specifying input data
  - In PROOF, speed-up initialization by caching meta-info
- The dataset manager controls a set of datasets
  - Interface class: TDataSetManager
- Local storage populated via XRootD
  - Stage-in files from a (virtual) Mass Storage System
- Staging steered by the *afdsmgrd* daemon
  - Synchronizes dataset requests with their availability

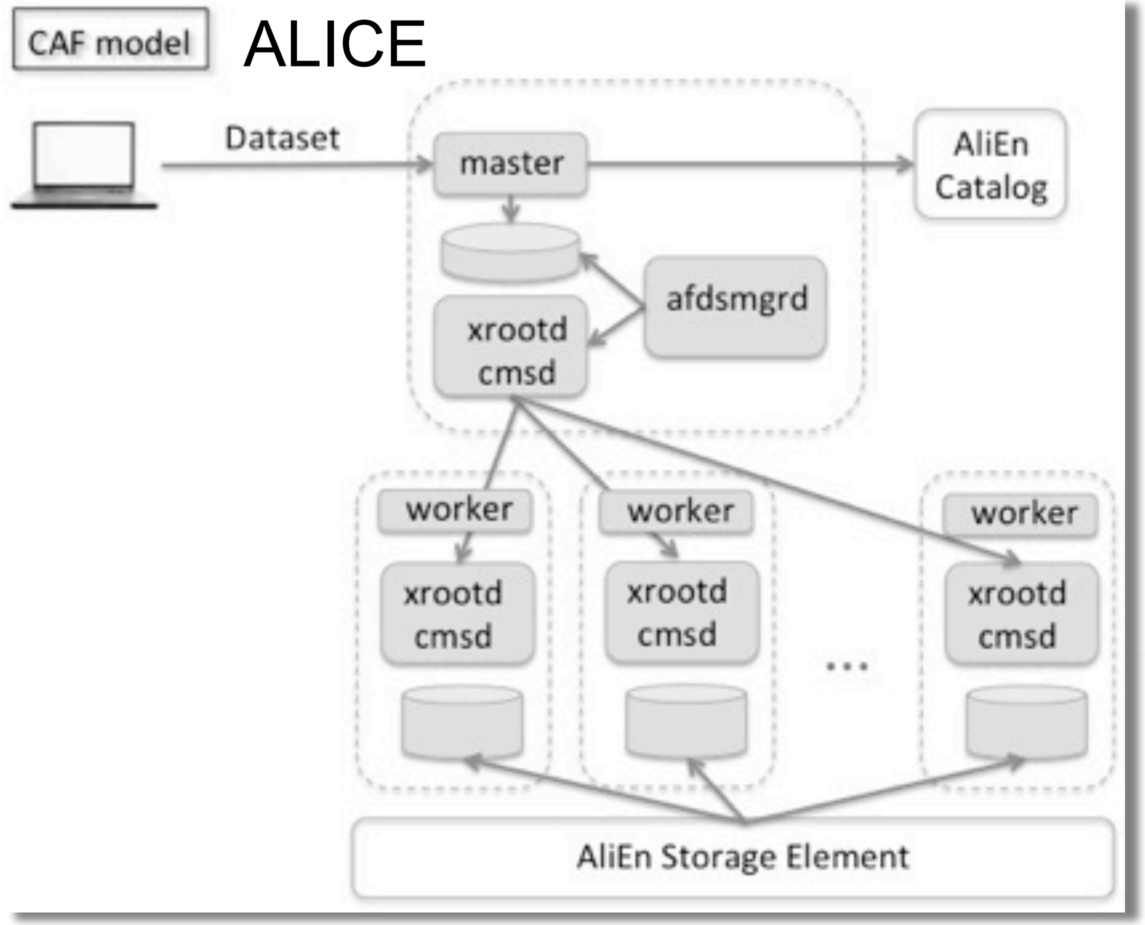
Remarks about datasets

- Working well with
  - Statically managed pools (delete by hand when required)
  - Limited number of datasets
- Scalability issues with pools in ‘caching’ mode
  - Rapid rotation of files: continuous synchronization
  - Slow response to new user requests
- Solution: dynamic model

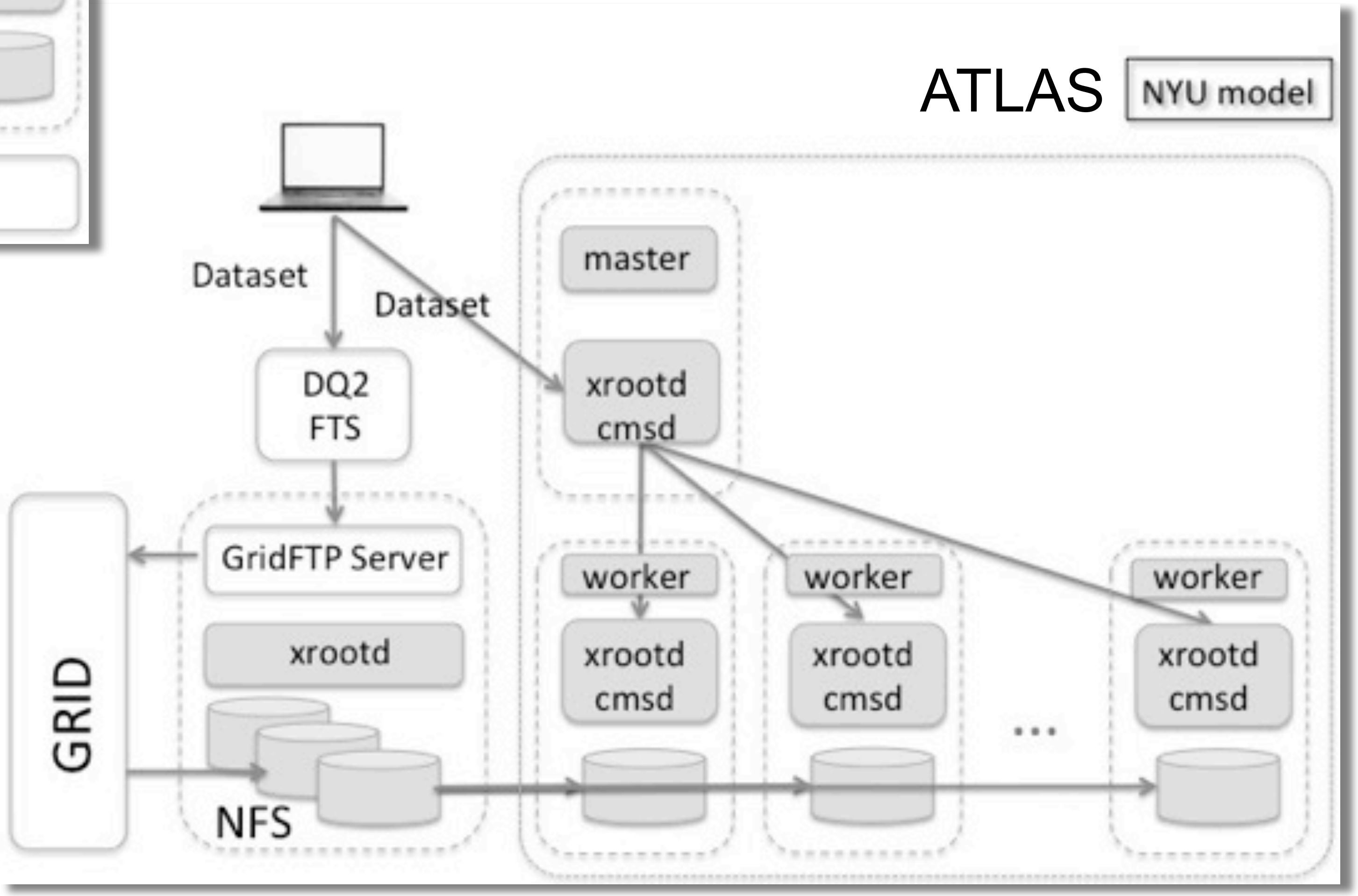
ROOT 5.34/05

  - Access directly experiment catalogs (AliEn, AMI, ...)
    - Exploit xrootd speed in stat-ing files
  - Synchronize only information about datasets being staged

Realizing data locality

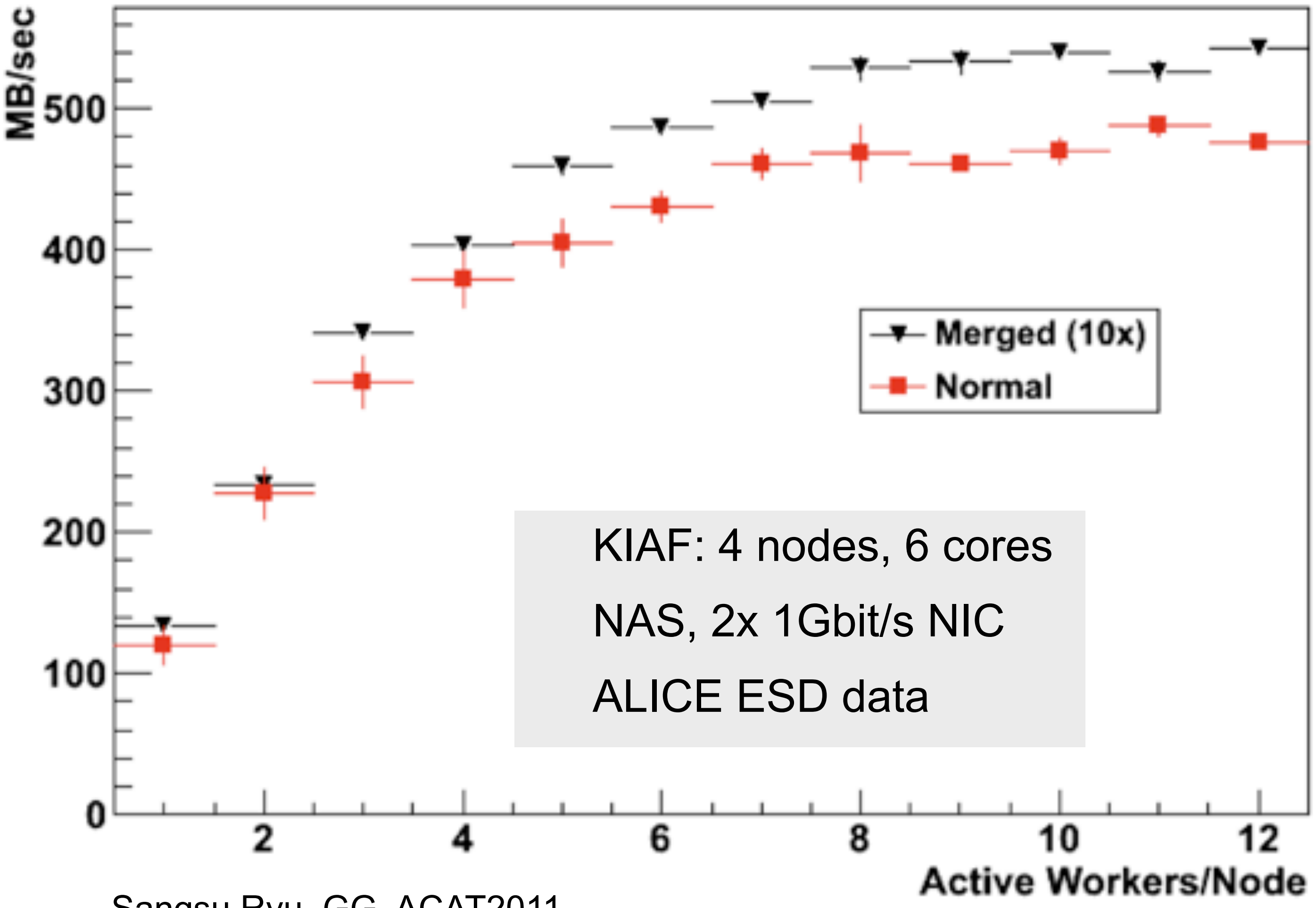


- Dataset files loaded on worker nodes from AliEn with FRM scripts
- Process controlled by afdsmgrd



- Big NFS server populated from the GRID with DQ2/GridFTP
- Hot files loaded on worker nodes with FRM scripts

Aggregating I/O bandwidth



KIAF: 4 nodes, 6 cores

NAS, 2x 1Gbit/s NIC

ALICE ESD data

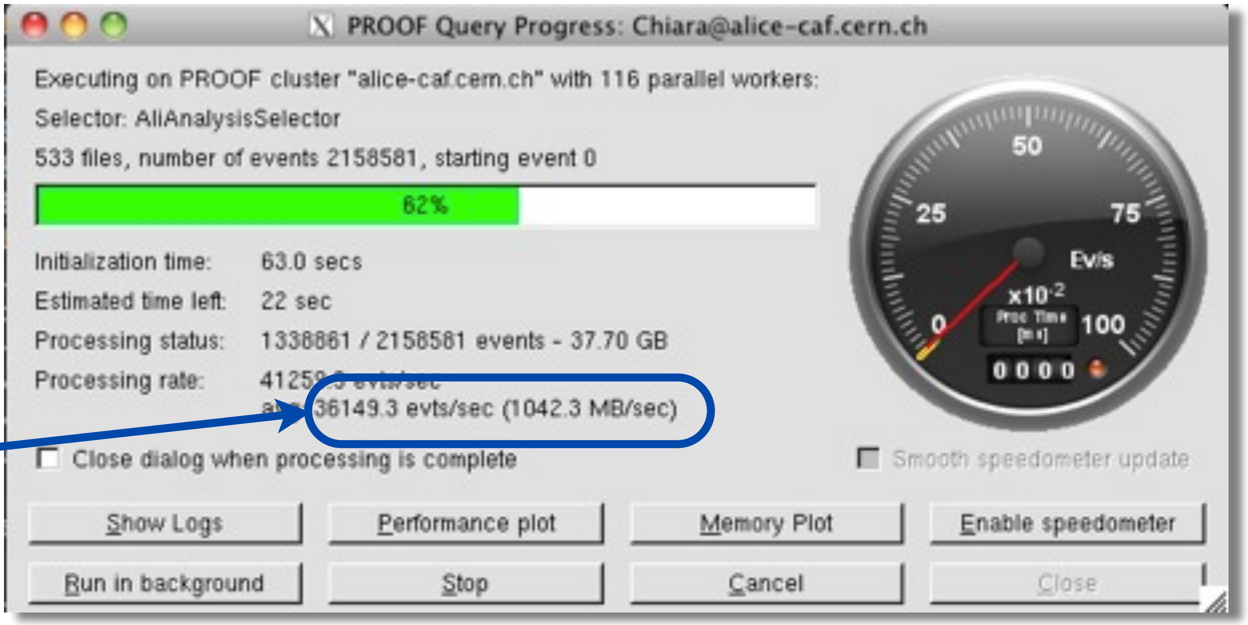
Sangsu Ryu, GG, ACAT2011

~ 4 x 120 MB/s

TProofBench run

Not only in test runs: screen shot from end user analysis run

~1 GB/s



PROOF Query Progress: Chiara@alice-caf.cern.ch

Executing on PROOF cluster "alice-caf.cern.ch" with 116 parallel workers:

Selector: AliAnalysisSelector

533 files, number of events 2158581, starting event 0

62%

Initialization time: 63.0 secs

Estimated time left: 22 sec

Processing status: 1338861 / 2158581 events - 37.70 GB

Processing rate: 41259.3 evts/sec
av. 36149.3 evts/sec (1042.3 MB/sec)

☐ Close dialog when processing is complete

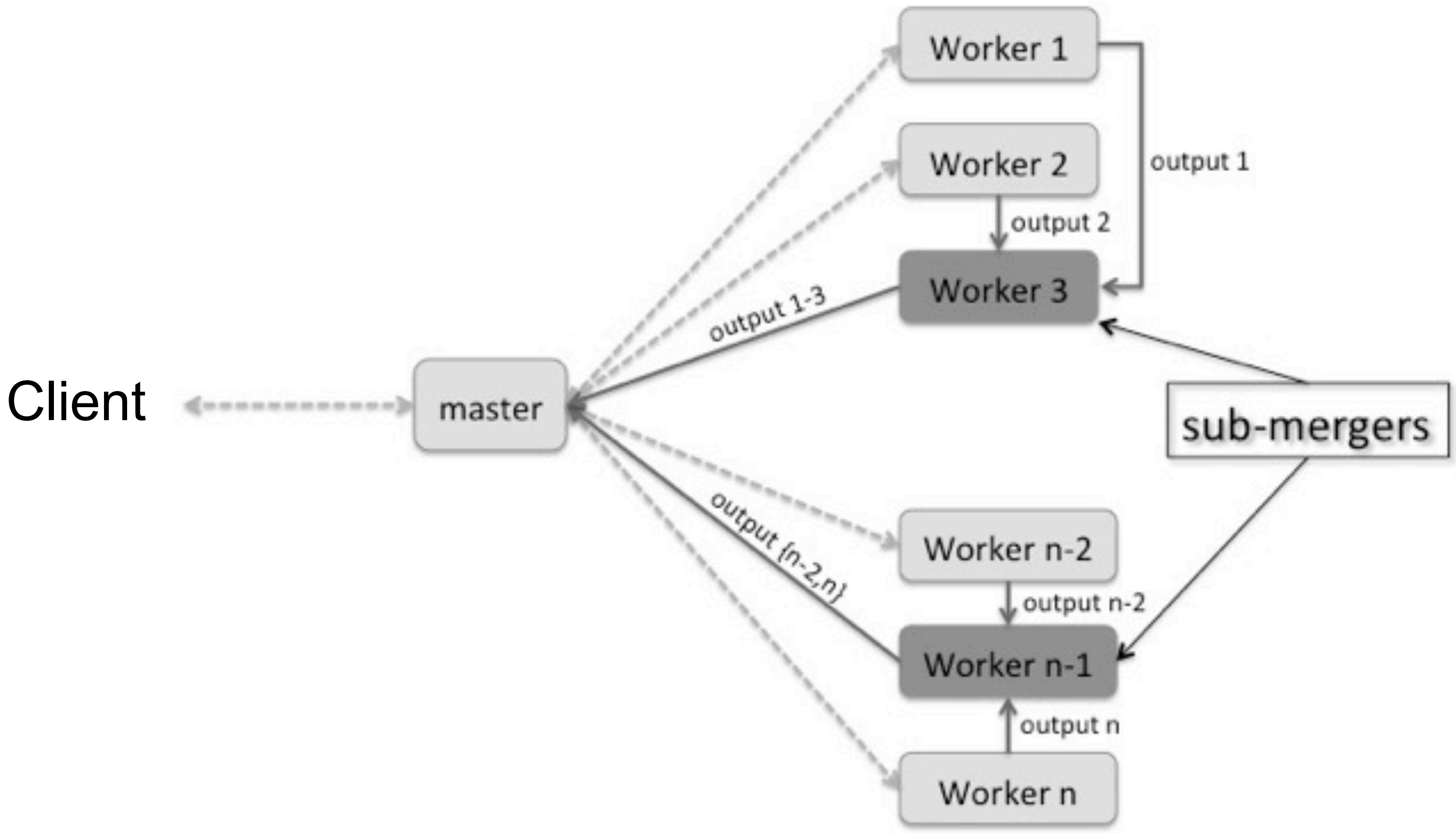
☐ Smooth speedometer update

Handling outputs

- PROOF designed assuming small outputs
  - Objects kept in memory, merged by the master
  - RAM problems for large outputs
- Solutions implemented
  - Merge 1-by-1 (not N in one go) if big objects
    - But asynchronous reader thread reads everything in memory as arrives
  - Save objects to file and merge the files
  - If output are TTrees, save tree to file and provide metadata for transparent access (e.g. dataset)
    - May also optimize subsequent access to the output (files already local to workers)
  - Use sub-mergers

Merging with submergers

- Sub-merger: faster workers promoted merger to help the master (basically map-reduce ...)



- Optimal number: ~ Sqrt(# of workers)
- Used by default on ALICE AF

Additional merging improvements

- Start merging during processing (i.e. not at the end)
- Works if objects sizes increase with # of events
  - Examples are TTrees:
    - Integration w/ multi-producer/collector technique (see P. Canal talk)
- Histograms have fixed sized and need to be fully merged at the end
  - No simple solution, a part from increasing parallelism
  - Collector technique could help for sparse histograms

Debugging user issues

- Difficult on distributed systems
  - Need good logging
    - E.g. Log processed file, last event #, on exception
  - Access to logged info
    - Log file retrieval
- Automatic valgrinding of PROOF sessions
  - Many issues memory-related
- Memory usage limitations
  - Large RAM usage can put node in weird state
    - Gracefully stop RAM-hungry sessions

Debugging user issues (2)

- What we have is not enough
  - valgrind requires debug symbols (and it is heavy)
  - Users do not automatically retrieve/look-at logs
- Ideas for further improvements
  - Memory analysis with TMemStat
  - Automatic creation of a tarball with all the information to be sent to developers
  - Watchdog technology to increase coverage of logging in case of problems
  - Global memory monitoring (with xproofd?)

Daemon stability

- Instabilities in multi-user mode due to its multi-threaded nature and complex handling of error conditions
- Deep debugging (thanks to B. Butler) during last year
- Version 5-34-05 includes the fixes
  - Very stable according to ATLAS (NYU, SLAC) and ALICE
- New binary ‘xpdtest’ available in 5-34-05 to test the responsiveness of the daemon
  - Could be used in monit or a cron job to regularly check its status

PoD based clusters

- PoD uses an Resource Management System to start daemons
  - Master runs on a dedicated machine, e.g. the desktop
  - Easy installation, complete CLI
  - RMS drivers as plug-ins: gLite, PanDa, HTCondor, PBS, OGE, LSF
    - SSH plug-in to control resources w/o an RMS
  - See next talk for more details
- Here we note that
  - Each user her/his own cluster:
    - Sandboxing, daemon in single-user mode (stability)
  - Scheduling, authent-/authorization done by RMS

Current plans of development

- New low-level connection client
  - XrdClient is being phased-out by Q1/2014
  - Opportunity to review requirements and implementation
- Protocol modifications to
  - Control output object sending
    - Control memory usage by multiplexed clients
  - Control packet distribution
- Dynamic addition of workers
  - Required to gradually ramp-up a session started on a PoD-managed cluster w/o a new TProof::Open
- Parallel merging techniques (see above)
- New packetizer

New packetizer

- Packetizer: component of the system generating packets of work aiming at having all workers finishing at the same time
- Current packetizers do not behave well when worker performance and/or file distributions are (strongly) non-homogeneous
  - Tails in the processing times
  - Sometimes manually switching between packetizers or changing some parameters helps
- Also, need to be able to start processing w/o exact number of entries
  - Required by new dataset management model

New packetizer (2)

- Idea: heuristic approach to follow the theoretical result minimizing the execution time

$$\frac{1}{t_0} = \frac{1}{N} \cdot \sum_{i}^{K} \bar{R}_i$$

based on the updated worker-server rates.

  - N = number of events
  - t_0 = execution time
  - $\bar{R}$i = average processing rate of worker i-th

– Technique used for CPU-based tasks

- Force workers to stop processing a packet when time goes beyond the expected one

PoD-based solution for static clusters

- Install free RMS (e.g. HTCondor) or use SSH
- If SSH
  - Install private key of authorized users
    - Can extract one associated to user’s X509 certificate with web-based technique (see D.Berzano’s talk)
  - Enable host-based pass-less SSH among nodes
- Users start their own cluster
  - Additional files needed can go in the PoD payload

PROOF and Clouds

- More and more computing resources available through clouds as virtual machines
- Already a few tests on cloud based test beds used as ‘static’ resources
  - Amazon EC2, Google CE (ATLAS/BNL)
  - Frankfurt Cloud (GSI)
- Dedicated CernVM Virtual Appliance with the relevant services to deploy PROOF on cloud resources
  - PROOF Analysis Facility as a Service
    - See D.Berzano talk

AoB

- PROOF as an external package
  - More efficient and less disruptive deployment of new functionality and/or fixes
  - Distributed as a tarball in ROOT
    - No visible changes for the user that wants it in ROOT
    - But disabling and usage-as-external possible
- New package manager
    - Support for versions, multi architecture
- PROOF-Lite improvements (see above)
- Improve documentation
- Multi-Master setups

PROOF Support

- Documentation
  http://root.cern.ch/drupal/content/proof
- PROOF section in ROOT forum
  http://root.cern.ch/phpBB3/
- ROOT Savannah
  http://savannah.cern.ch/projects/savroot

- Need for more (alive) tutorials!

Summary and outlook

- PROOF@LHC:
  - PROOF-Lite is a powerful tool for a MC desktop/laptop
  - ‘Static’ AFs reached a sort of equilibrium
  - Resource sharing works well thanks to PoD
- Current developments aim to
  - Consolidation
  - Full exploitation of PROOF capabilities
  - Improve user experience
- Exploit at best available resources
  - Desktops/Laptops
  - Static dedicated or shared clusters
  - Clouds