ROOT/PROOF based analysis at Tier2, Tier3

Torre Wenaus (BNL), Sergey Panitkin (BNL)

US ATLAS Tier 2 Workshop
Bloomington
June 21, 2007

Post AOD Analysis

- As discussed this morning, pathena is provided as a tool to perform parallel athena analysis with Panda
- What about post-Athena analysis? People want to run that in parallel as well
  - On official ATLAS computing resources, and probably also on their own university resources
  - No effort has been made to support this in Panda/pathena because it would be reinventing the wheel...
- PROOF provides this capability
  - Makes sense for us to try it out, study its capability, performance, usability, depoyment/operations complexity
  - In use by other experiments (RHIC, ALICE, CMS?, ...)

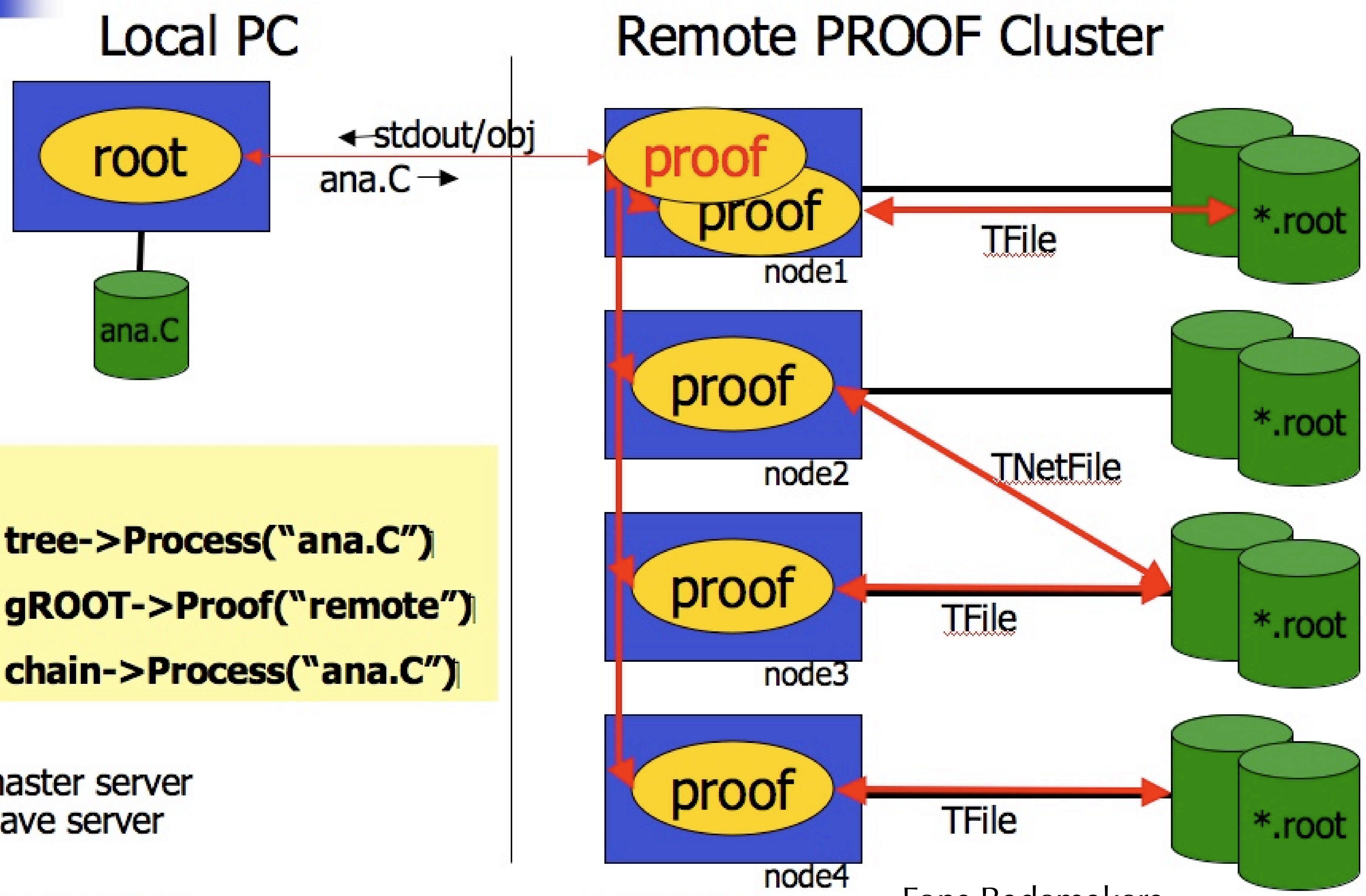
Post AOD Analysis

- There is no ‘final specification’ of post AOD data format (s) and their exact roles (AANT, DPD, ntuples,...)
- But we know they’ll all have something in common: ROOT
  - ROOT data format
  - Analyzed with ROOT
- Making PROOF applicable to parallelize their analysis
- What role should PROOF have, if any? Where can/ should it be deployed? Does it need firewall conduits to be useful? How much can it help analysis?
  - Discussion is starting

PROOF

- Original design goal: interactive parallel analysis on local cluster
  - Transparent: analysis code doesn’t change for parallel operation
  - Extended to operate also on wide area clusters
    - (Not something to try from day 1)
- Provides ‘interactive batch’
  - Submit long-running queries, disconnect/reconnect client
- GUI available if you like that sort of thing
- Scaling to ~1000 CPUs reported by developers
- Mates well with xrootd (but doesn’t require it)
  - xrootd storage pool can host proofd servers on storage nodes
  - PROOF authentication can use xrootd (supports grid certs)

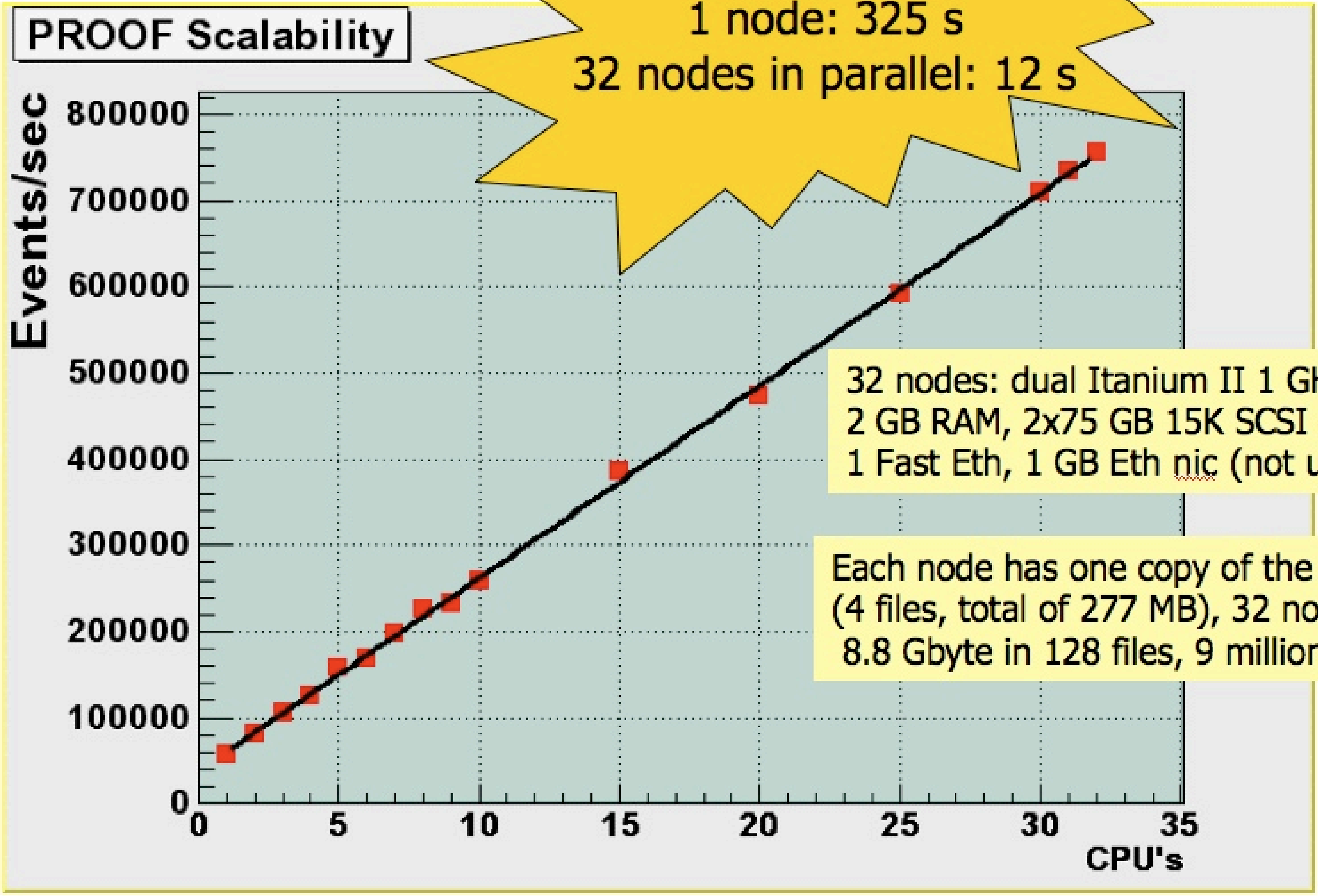
Parallel PROOF



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

proof = master server
proof = slave server

Scalability



8.8GB, 128 files
1 node: 325 s
32 nodes in parallel: 12 s

32 nodes: dual Itanium II 1 GHz CPU's,
2 GB RAM, 2x75 GB 15K SCSI disk,
1 Fast Eth, 1 GB Eth nic (not used)

Each node has one copy of the data set
(4 files, total of 277 MB), 32 nodes:
8.8 Gbyte in 128 files, 9 million events

Shown by
F. Rademakers
March 2006

Using PROOF

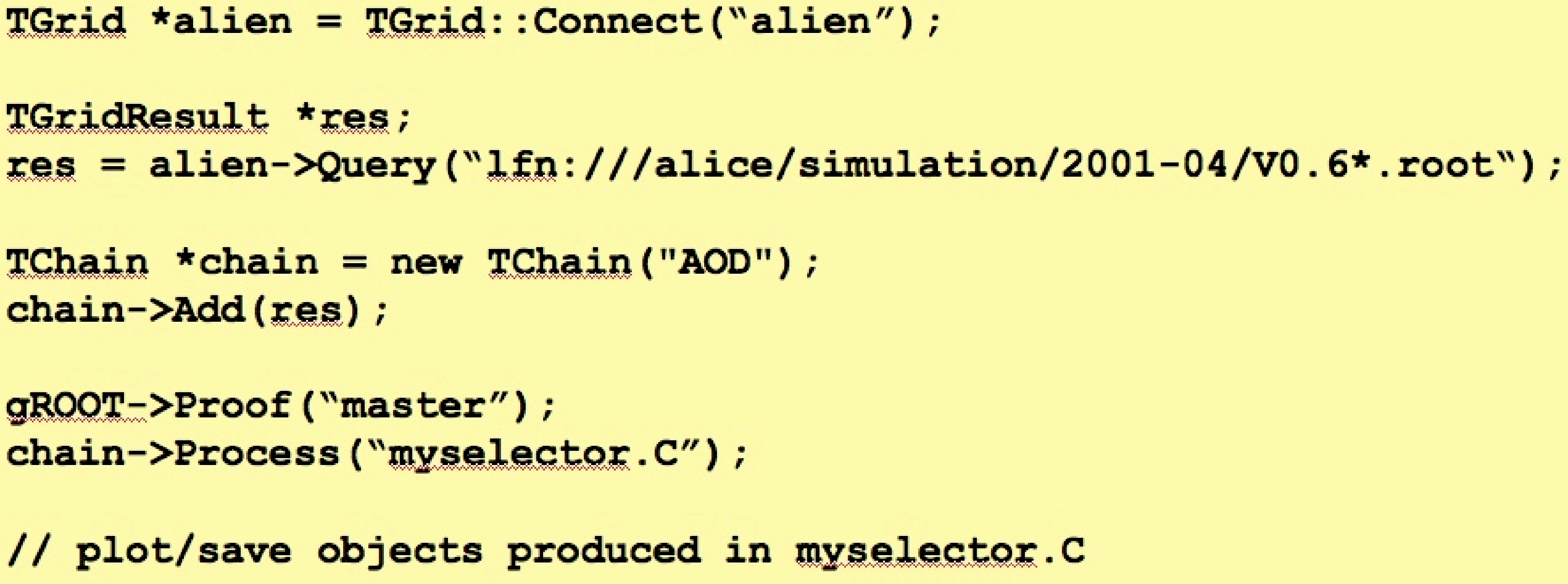
- User code: implemented using **TSelector** to process a set of inputs producing a set of outputs via a PROOF farm

```
// Abbreviated version
class TSelector : public TObject {
Protected:
    TList *fInput;
    TList *fOutput;
public
    void Init(TTree*);
    void Begin(TTree*);
    void SlaveBegin(TTree *);
    Bool_t Process(int entry);
    void SlaveTerminate();
    void Terminate();
};
```

- Data: a **TChain**, collection of TTree files

```
root[0] TChain *c = new TChain(“esd”);
root[1] c->Add(“root://rcrs4001/a.root”);
…
root[10] c->Print(“a”);
root[11] c->Process(“mySelector.C”, nentries, first);
```

- Running PROOF



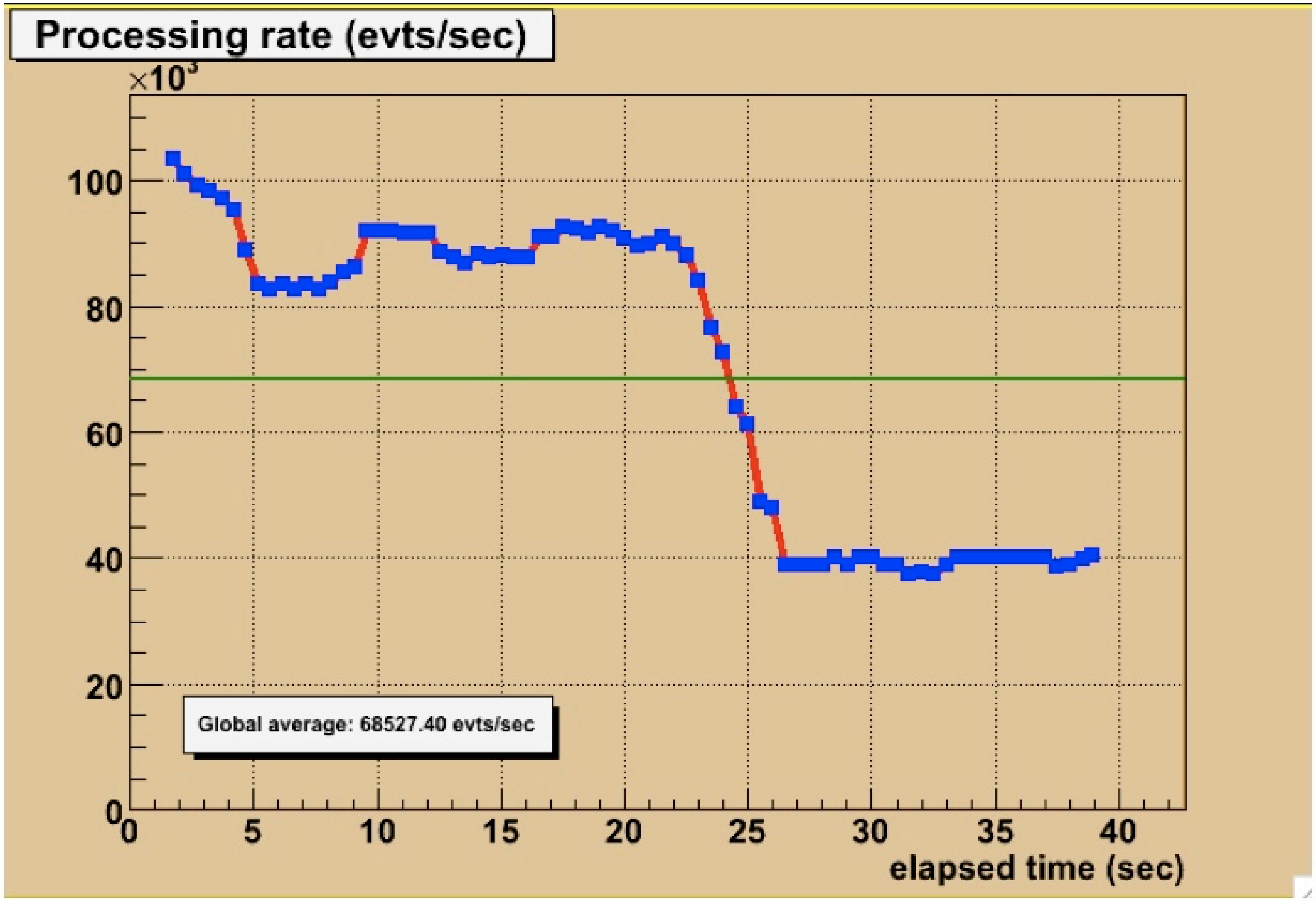
```
TGrid *alien = TGrid::Connect(“alien”);

TGridResult *res;
res = alien->Query(“lfn:///alice/simulation/2001-04/V0.6*.root“);

TChain *chain = new TChain("AOD");
chain->Add(res);

gROOT->Proof(“master”);
chain->Process(“myselector.C”);

// plot/save objects produced in myselector.C
```

Torre Wenaus, BNL

PROOF and xrootd

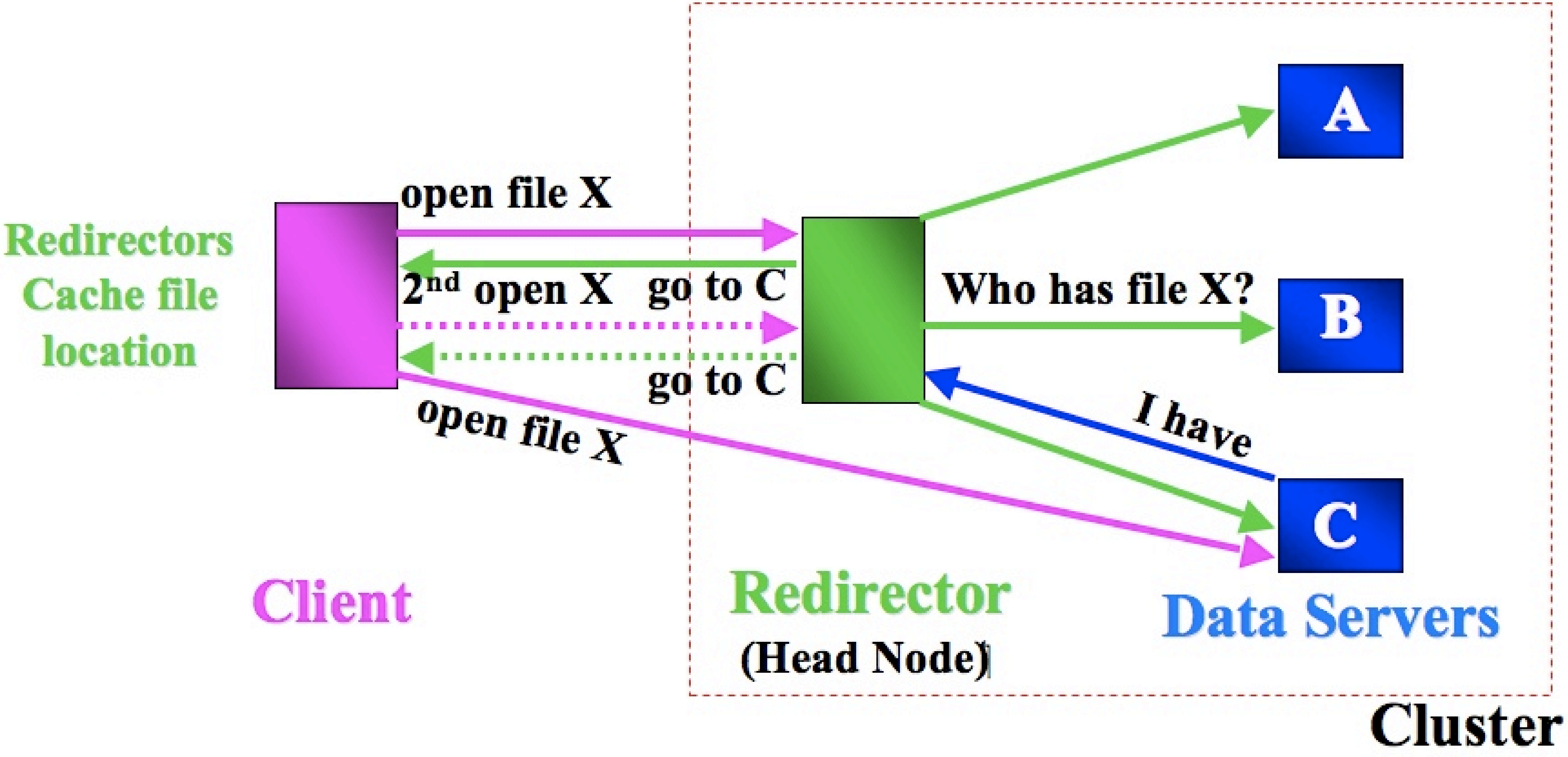
- Both distributed as part of ROOT
- PROOF daemon offered as a plugin for xrootd
- xrootd daemons on storage nodes use little CPU leaving lots of room for PROOF processing of local data
- PROOF will preferentially process data that is local
- PROOF can also access any ROOT-accessible data, not just xrootd

Preferential local file processing

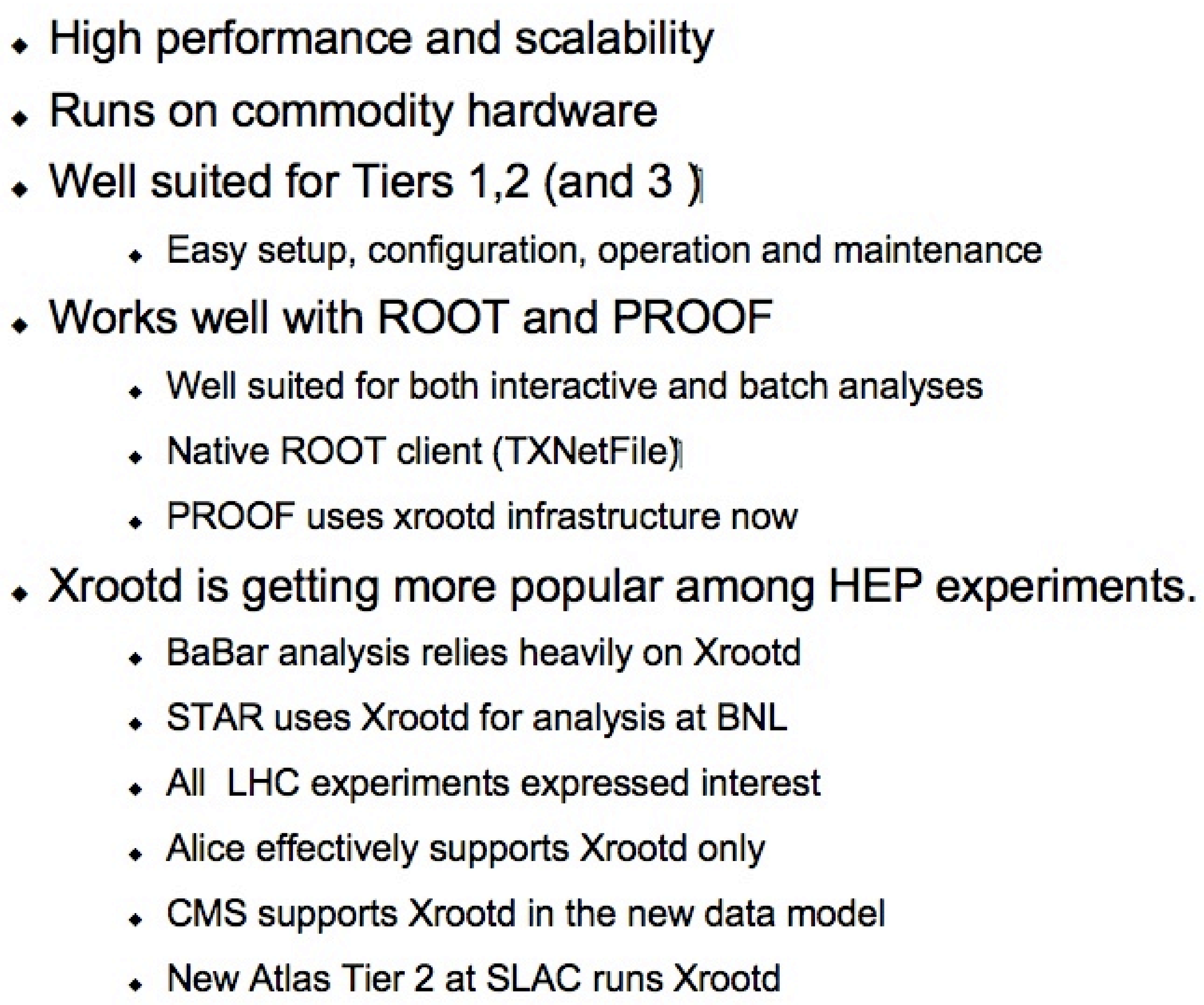


PROOF tries to match jobs running on a given node with files available locally on this node.
The "tail off" observed at ~25 sec happens when local file processing is exhausted and jobs are reading files from other xrootd nodes

How xrootd works



Why xrootd?



- High performance and scalability
- Runs on commodity hardware
- Well suited for Tiers 1,2 (and 3)
  - Easy setup, configuration, operation and maintenance
- Works well with ROOT and PROOF
  - Well suited for both interactive and batch analyses
  - Native ROOT client (TXNetFile)
  - PROOF uses xrootd infrastructure now
- Xrootd is getting more popular among HEP experiments.
  - BaBar analysis relies heavily on Xrootd
  - STAR uses Xrootd for analysis at BNL
  - All LHC experiments expressed interest
  - Alice effectively supports Xrootd only
  - CMS supports Xrootd in the new data model
  - New Atlas Tier 2 at SLAC runs Xrootd

Sergey Panitkin

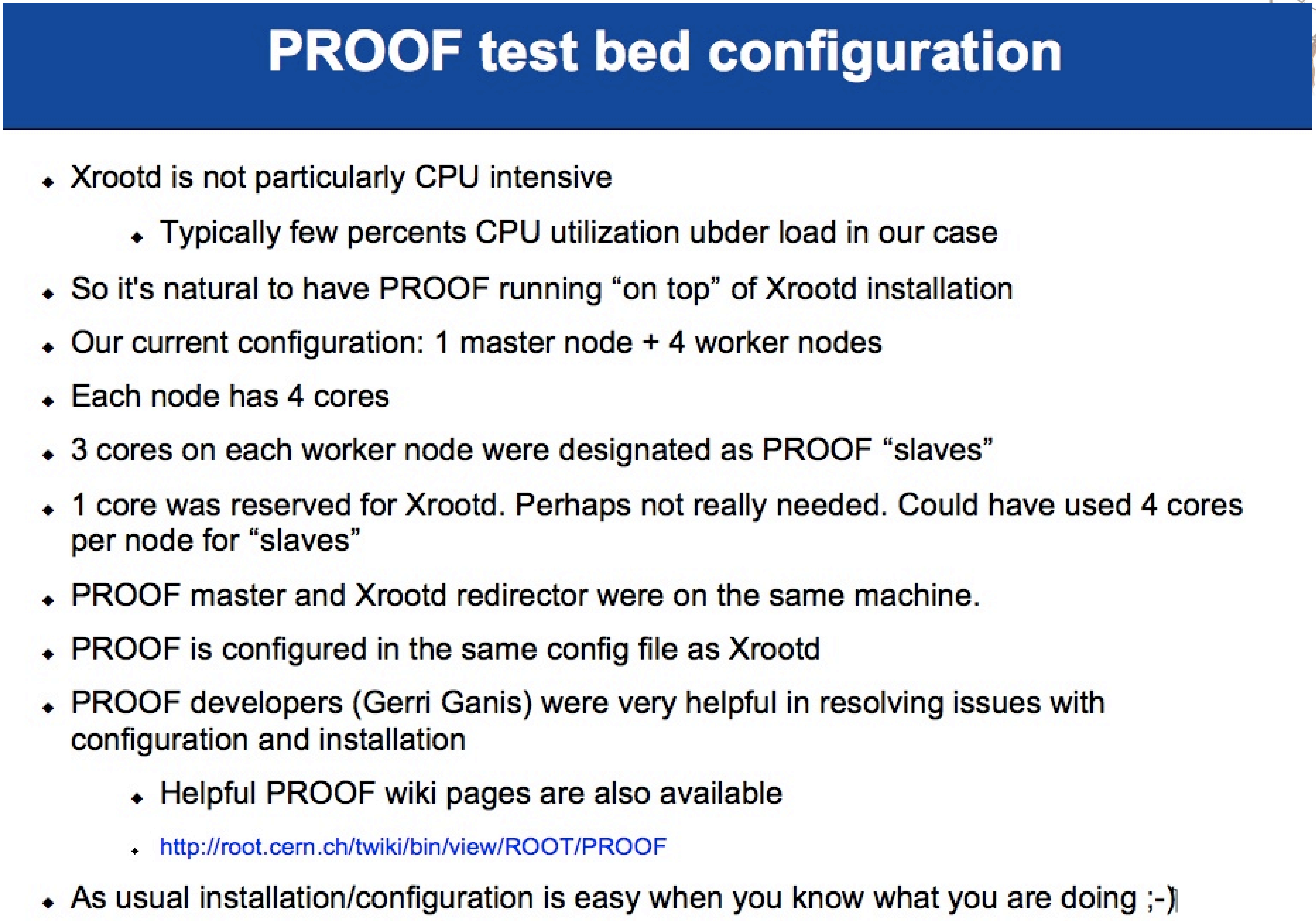
STAR Experience with xrootd

- STAR is a large heavy-ion experiment at RHIC BNL
  - ~500 physicists
  - ~100-1000 TB data per year
- STAR started with “rootd” (2003), moved to Xrootd (2005)
  - A highly scalable, self-configurable, fault-tolerant, plug-and-play component architecture tool suitable for technology evolution with ability to move hand-shake with Mass Storage Systems
- Cost effectiveness
  - Low human maintenance cost (< 1 FTE)
  - Hardware: order of magnitude (5-10) cheaper than leading centrally available solution
- Impact
  - 64 TB of centralized disk , 134 TB on distributed (xrootd) , data mostly on distributed disk IO aggregate scales linearly with data-servers
- Exceeds industry leading NAS&SAN i.e. analysis aggregate have faster turn around (TBC)
- Probably largest Xrootd installation in the world: ~320 analysis nodes

borrowed from Jerome Lauret (STAR)

xrootd/PROOF test farm at BNL

- 10 machines allocated so far for xrootd test farm
  - Two dual core Opteron CPUs at 1.8 Ghz per node
  - 8 GB RAM per node
  - 4x 500 GB SATA drives per node, configured as a 2 TB partition
  - Gigabit network
- 5 node configuration used for tests so far
  - 1 redirector + 4 data servers
  - 20 CPU cores
  - ~10 TB of available disk space
- PROOF deployed on the farm in conjunction with xrootd
  - PROOF installation was not easy! PROOF/xrootd integration still a work in progress. Versions must match precisely.
  - Configuration is straightforward once installed
- Behind ACF firewall, visible from ACF only
- 2 people did set up, installation, configuration, etc ~0.25 FTE (Sergey Panitkin, Ofer Rind)

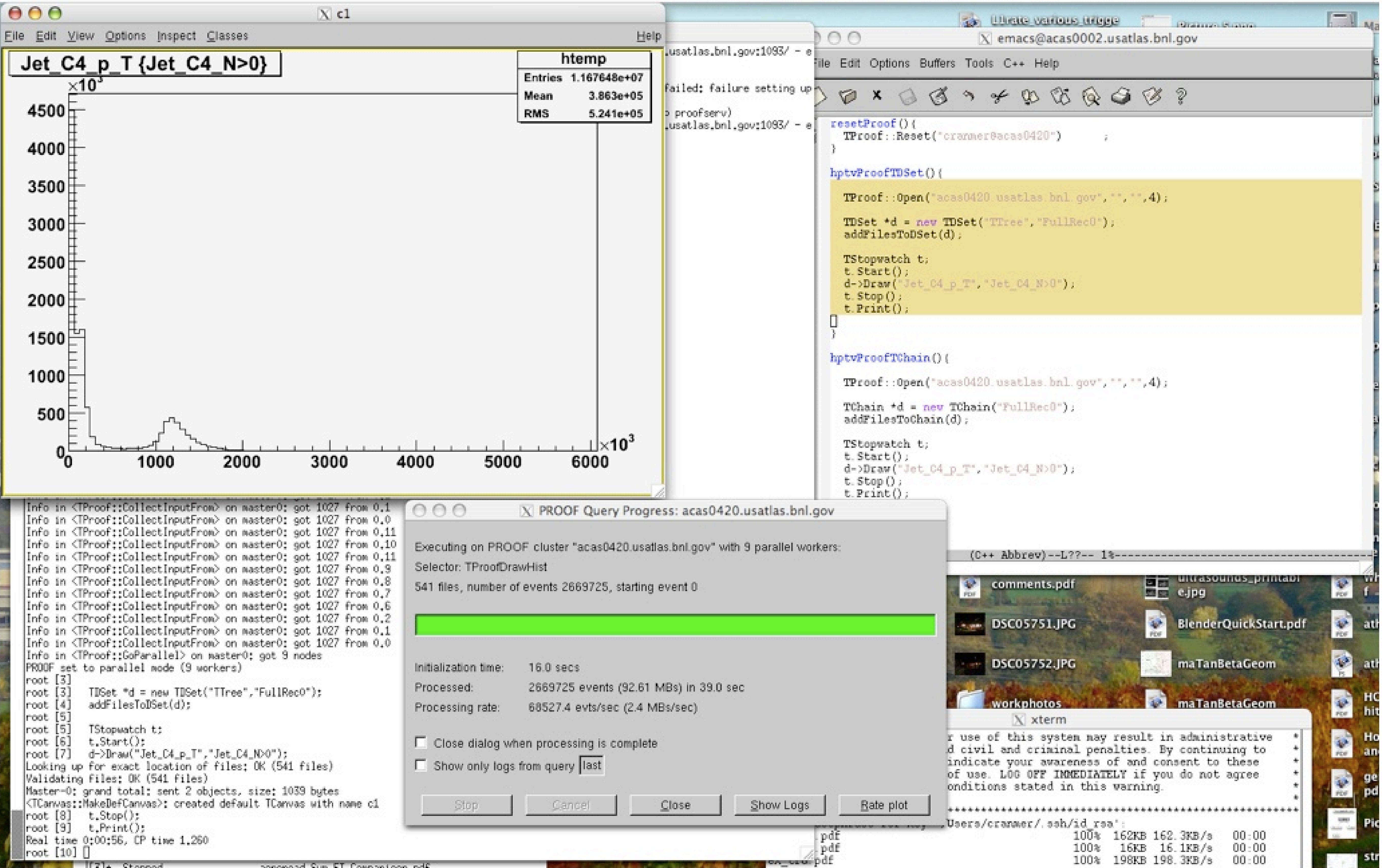


PROOF test bed configuration

- Xrootd is not particularly CPU intensive
  - Typically few percents CPU utilization ubder load in our case
- So it's natural to have PROOF running “on top” of Xrootd installation
- Our current configuration: 1 master node + 4 worker nodes
- Each node has 4 cores
- 3 cores on each worker node were designated as PROOF “slaves”
- 1 core was reserved for Xrootd. Perhaps not really needed. Could have used 4 cores per node for “slaves”
- PROOF master and Xrootd redirector were on the same machine.
- PROOF is configured in the same config file as Xrootd
- PROOF developers (Gerri Ganis) were very helpful in resolving issues with configuration and installation
  - Helpful PROOF wiki pages are also available
  - http://root.cern.ch/twiki/bin/view/ROOT/PROOF
- As usual installation/configuration is easy when you know what you are doing ;-)

Typical PROOF session

courtesy of Kyle Cranmer



```
.usatlas.bnl.gov:1093/ - e
failed: failure setting up
 proofserv)
.usatlas.bnl.gov:1093/ - e
```

```
Info in <TProof::CollectInputFrom> on master0: got 1027 from 0.1
Info in <TProof::CollectInputFrom> on master0: got 1027 from 0.0
Info in <TProof::CollectInputFrom> on master0: got 1027 from 0.11
Info in <TProof::CollectInputFrom> on master0: got 1027 from 0.10
Info in <TProof::CollectInputFrom> on master0: got 1027 from 0.11
Info in <TProof::CollectInputFrom> on master0: got 1027 from 0.9
Info in <TProof::CollectInputFrom> on master0: got 1027 from 0.8
Info in <TProof::CollectInputFrom> on master0: got 1027 from 0.7
Info in <TProof::CollectInputFrom> on master0: got 1027 from 0.6
Info in <TProof::CollectInputFrom> on master0: got 1027 from 0.2
Info in <TProof::CollectInputFrom> on master0: got 1027 from 0.1
Info in <TProof::CollectInputFrom> on master0: got 1027 from 0.0
Info in <TProof::GoParallel> on master0: got 9 nodes
PROOF set to parallel mode (9 workers)
root [3]
root [3]   TDSet *d = new TDSet("TTree","FullRec0");
root [4]   addFilesToDSet(d);
root [5]
root [5]   TStopwatch t;
root [6]   t.Start();
root [7]   d->Draw("Jet_C4_p_T","Jet_C4_N>0");
Looking up for exact location of files: OK (541 files)
Validating files: OK (541 files)
Master-0: grand total: sent 2 objects, size: 1039 bytes
<TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root [8]   t.Stop();
root [9]   t.Print();
Real time 0:00:56, CP time 1.260
root [10]
```

```
resetProof(){
  TProof::Reset("cranmer@acas0420")        ;
}

hptvProofTDSet(){

  TProof::Open("acas0420.usatlas.bnl.gov","","",4);

  TDSet *d = new TDSet("TTree","FullRec0");
  addFilesToDSet(d);

  TStopwatch t;
  t.Start();
  d->Draw("Jet_C4_p_T","Jet_C4_N>0");
  t.Stop();
  t.Print();

}

hptvProofTChain(){

  TProof::Open("acas0420.usatlas.bnl.gov","","",4);

  TChain *d = new TChain("FullRec0");
  addFilesToChain(d);

  TStopwatch t;
  t.Start();
  d->Draw("Jet_C4_p_T","Jet_C4_N>0");
  t.Stop();
  t.Print();

(C++ Abbrev)--L??-- 1%------------------------------------------
```

Executing on PROOF cluster "acas0420.usatlas.bnl.gov" with 9 parallel workers:

Selector: TProofDrawHist

541 files, number of events 2669725, starting event 0

Initialization time: 16.0 secs

Processed: 2669725 events (92.61 MBs) in 39.0 sec

Processing rate: 68527.4 evts/sec (2.4 MBs/sec)

☐ Close dialog when processing is complete

☐ Show only logs from query last

```
r use of this system may result in administrative   *
d civil and criminal penalties. By continuing to    *
indicate your awareness of and consent to these     *
of use. LOG OFF IMMEDIATELY if you do not agree     *
onditions stated in this warning.                   *
                                                    *
*****************************************************
Users/cranmer/.ssh/id_rsa':
.pdf                          100%  162KB 162.3KB/s   00:00
.pdf                          100%   16KB  16.1KB/s   00:00
pdf                           100%  198KB 198.3KB/s   00:00
```

DQ2 Access from ROOT

- Tadashi has already provided an interface for DQ2 access from ROOT to enable using ATLAS datasets/files in PROOF sessions

```
[tmaeno@lxplus204 tmp]$ root.exe
root [0] TPython::LoadMacro("DQ2IF.py")
root [1] DQ2IF dq2;
root [2] string ds = dq2.getDatasets("ideal0_csc11.005012.J3_pythia_jetjet.digit.RDO.v12000604*")
root [3] cout << ds << endl;
ideal0_csc11.005012.J3_pythia_jetjet.digit.RDO.v12000604_tid010481_dis1923369,ideal0_csc11.0050
12.J3_pythia_jetjet.digit.RDO.v12000604_tid010481_dis1923173,ideal0_csc11.005012.J3_pythia_jetjet
.digit.RDO.v12000604_tid010481_dis1923177,ideal0_csc11.005012.J3_pythia_jetjet.digit.RDO.v1200
0604_tid010481_dis1954374,...
```

PROOF trials at BNL

Kyle Cranmer, Fabien Tarrade

- HighPtView ntuples for all CSC v12 AODs at BNL loaded in xrootd
- Comparing 1 machine with 9-node PROOF:
  - >500 AANT files, 2.6M events processed, selecting all jets (11M), applying cuts, making plots
    - 1 machine: 7 minutes. PROOF: 37 seconds.
    - With PROOF: ~70k events/sec (2.4 MB/s)
  - Heavier analysis code with similar size samples, reading all data, can take a day to run serially
    - Implementing a TSelector to do this analysis in PROOF
    - Will also be able to pad out the calculation time to adjust CPU to disk I/O ratio to study scaling
- Firewall conduit issue already apparent. No conduit for proofd traffic means the user must run root on an acas machine, with X graphics traversing the LAN or WAN. *Slow*. The popular

Near term objective

- For mid-July: support pathena DPD loading to xrootd, usable by PROOF, on the BNL testbed
  - Leverage existing work done for Panda/pilot support of SLAC xrootd SE
  - Support distinct SEs for input/output in Panda pilot
  - Finish ROOT/DQ2/PROOF interface for dataset-based PROOF processing

No conclusions, just questions!

- If PROOF turns out to be valuable in expediting the last stage of analysis,
  - Where do we make PROOF available, and who operates it? T1, T2, T3s? What resources to we dedicate to it?
  - How does it mate with pathena?
    - pathena/Panda can send job outputs (eg DPDs) directly to xrootd, making them immediately available to PROOF
  - Will every T2 *really* have the full AOD? Or will AOD processing remain focused at BNL, with later-stage PROOF analysis taking place 'closer to the physicist' (and away from BNL security issues) being an important T2 role?
- Implement everything (pathena, PROOF, xrootd) at T1 and all T2s, and interested T3s, and let users 'vote with their feet' on the model that works?