Recent PROOF developments

G. Ganis

CAF meeting, ALICE offline week , 11 October 2007

PROOF at the CAF

- CAF: first / main PROOF testbed in LHC environment
- Understand
  - Problems
    - Instabilities and error recovery
    - Performance (end-of-query tails)
  - Missing / improvable functionality
    - Handling of input data, additional software
    - Generic task processing
    - Quota (data/resources) control
    - Handling of big outputs
    - Diagnostics tools (memory usage)
  - Multi-user behaviour
    - Fair-sharing of resources

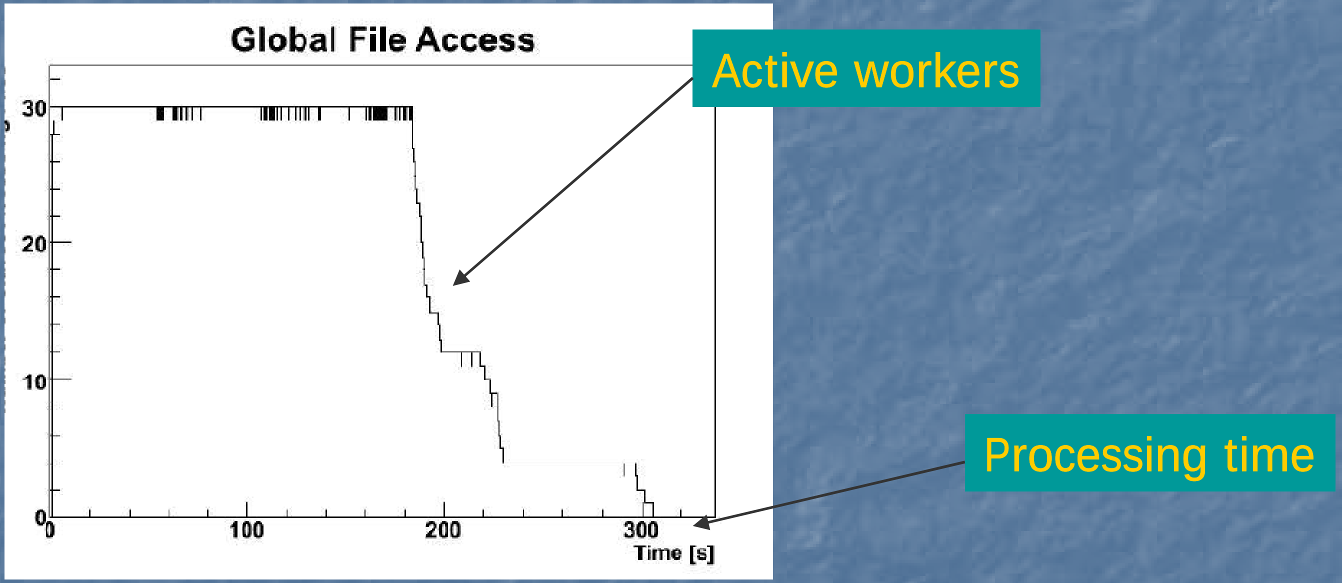
Outline

- User interface developments
  - Improvements
    - Packetizer
    - Software handling
    - Dataset handling
  - New features
    - Non-data driven processing
    - Output file merging
    - Memory monitoring
- Resource control developments
  - Fair share based on experiment policy
  - Central scheduler

Packetizer improvements

J. Iwaszkiewicz

- Packetizer's goal: optimize work distribution to process queries as fast as possible
- Standard TPacketizer's strategy
  - first process local files, than try to process remote data



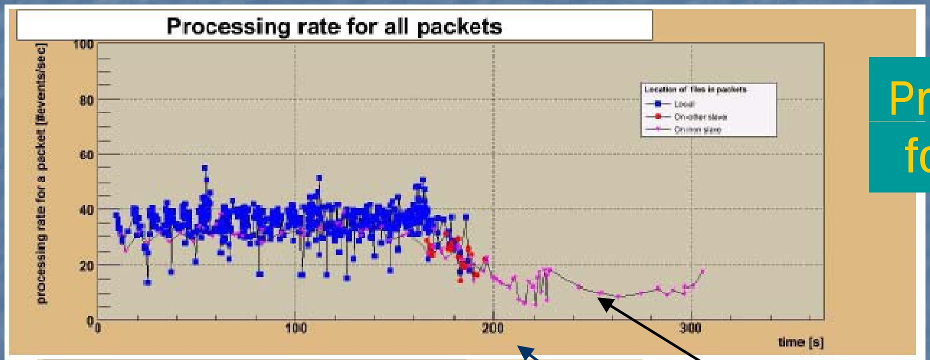
  - End-of-query bottleneck

New strategy: TPacketizerAdaptive

- Predict processing time of local files for each worker
- Keep assigning remote files from start of the query to workers expected to finish faster
- Processing time improved by up to 50%

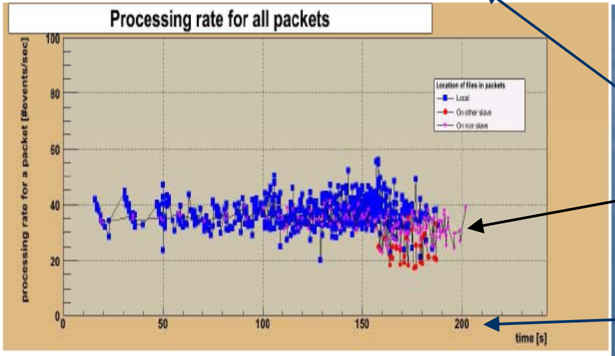
Default since Jun 5th

OLD



Processing rate for all packets

NEW



Remote packets

Same scale

Software handling

- Package enabling
  - Separated behaviour client / cluster
  - Real-time feedback during build
  - Soon: package versioning (e.g. ESD-v1.12.103-new)
- Load mechanism extended to single class / macro

```
root[] TProof *proof = TProof::Open(“master”)
root[] proof->Load(“MyClass.C”)
```

- Selectors / macros / classes binaries are now cached
  - Decreases initialization time
- API to modify include / library paths on the workers
  - Use packages globally available on the cluster
- Improved version check for binaries
  - Based also on SVN revision

Dataset manager

J. Iwaszkiewicz + G. Bruckner (more on Gerhard’s talk)

- Metadata about a set of files stored in sandbox on the master on dedicated subdirectory
    - <DatsetDir>/group/user/dataset or <SandBox>/dataset
- Data-sets are identified by name

```
root[0] TProof *proof = TProof::Open(“master”);
root[1] TFileCollection *fc = new TFileCollection(“dummy”);
root[2] fc->AddFromFile(“ESD5000_5029.txt”)
root[2] proof->CreateDataSet(“ESD5000_5029”, fc->GetList());
root[3] proof->ShowDataSets();
Existing Datasets:
ESD5000_5029
```

- Data-sets can be processed by name

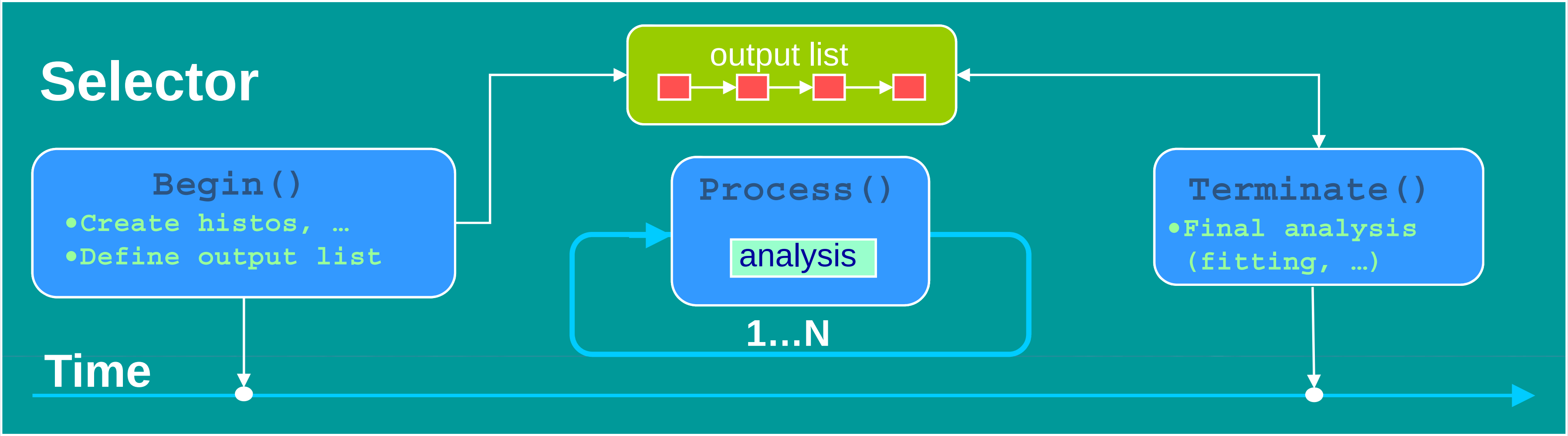
```
root[] proof->Process(“ESD5000_5029”, “MySelector.C+”);
```

    - No need to locally create the chain (CreateESDchain)

Non-data-driven analysis

L. Tran-Thanh

Implement algorithm in a TSelector



Selector

output list

`Begin()`

- `Create histos, …`
- `Define output list`

`Process()`

analysis

1…N

`Terminate()`

- `Final analysis (fitting, …)`

Time

New TProof::Process(const char *selector, Long64_t times)

```
// Open the PROOF session
root[0] TProof *p = TProof::Open("master")

// Run 1000 times the analysis defined in the
// MonteCarlo.C TSelector
root[1] p->Process("MonteCarlo.C+", 1000)
```

Non-data-driven analysis

- New packetizer TPacketizerUnit
  - Time-based packet sizes
  - Processing speed of each worker measured dynamically
- Included in ROOT 5.17/04

Output file merging

L. Tran-Thanh

- Address the case of large output objects (e.g. trees) which create memory problems
- Idea: save them in files on the workers and merge them using TFileMerger
- New class TProofFile defines the file and provide tools to handle the merging
  - Unique file names are created internally to avoid crashes
- Merging will happen on the Master at the end of the query
- Final file is left in sandbox on the master or saved where the client wishes
- Included in ROOT 5.17/04

Output file merging: example

```
void PythiaMC::SlaveBegin(TTree *) {
    // Meta file object: to be added to the output list
    fProofFile = new TProofFile();
    fOutput->Add(fProofFile);
    // Output filename (any format understood by TFile::Open)
    TNamed *outf = (TNamed *) fInput->FindObject(“PROOF_OUTPUTFILE”);
    if (outf) fProofFile->SetOutputFileName(outf->GetTitle());
    // Open the file with a unique name
    fFile = fProofFile->OpenFile(“RECREATE”);
    // Create the tree and attach it to the file
    fTree = new TTree(…);
    fTree->SetDirectory(fFile);
    …
}
Bool_t PythiaMC::Process(Long64_t entry) {
    fTree->Fill();
}
void PythiaMC::SlaveTerminate() {
    if (fFile) {
       fFile->cd();
       // Write here big objects
       fTree->Write();
       fFile->Close();
    }
}
```

Memory consumption monitoring

A. Kreshuk

- Normal level
  - Workers monitor their memory usage and save info in the log file
  - Client get warned of high usage
    - The session may be eventually killed
  - New button in the progress dialog box to display the evolution of memory usage per node
- Advanced level
  - Possibility to save in a dedicated tree (TProofStats) very detailed information (e.g. interface to Marian Ivanov's memsta tool)
  - To be run as second pass when a problem shows up
- Coming soon

Scheduling multi-users

- Fair resource sharing
  - System scheduler not enough if N_{users} >= ~ $N_{workers}$ / 2
- Enforce priority policies
- Two levels
  - Quota-based worker level load balancing
    - Based on group quotas
  - Central level (scheduler)
    - Per-query decisions based on cluster load, resources need by the query, user history and priorities
    - Generic interface to external schedulers

Quota-based worker level load balancing

- Based on group priority information defined in dedicated files or communicated by masters
- Two technologies
  - Slowdown requests for new packets to match the quotas
    - Worker sleeps before asking for the next packet
    - PROS: quantitatively correct
    - CONS: large fluctuations if packet sizes are large and variable; requires round-robin system scheduling; acts only on CPU
  - “renice” low priority sessions
    - Priority = 20 – nice (-20 <= nice <= 19)
      - Limit max priority to avoid over killing the system
    - PROS: independent of packet size; controls all resources
    - CONS: quantitatively more difficult to control

Resource quotas based on experiment policy

- Feedback mechanism
  - At the end of each query the amount of resources used is reported to MonALisa per user/group
  - This information is used to calculate effective group priorities based target priorities (see Marco’s talk)
  - PROOF masters broadcast the effective group priorities to their workers

- The central scheduler will use the effective priorities to determine which workers to assign to a user

Central scheduling

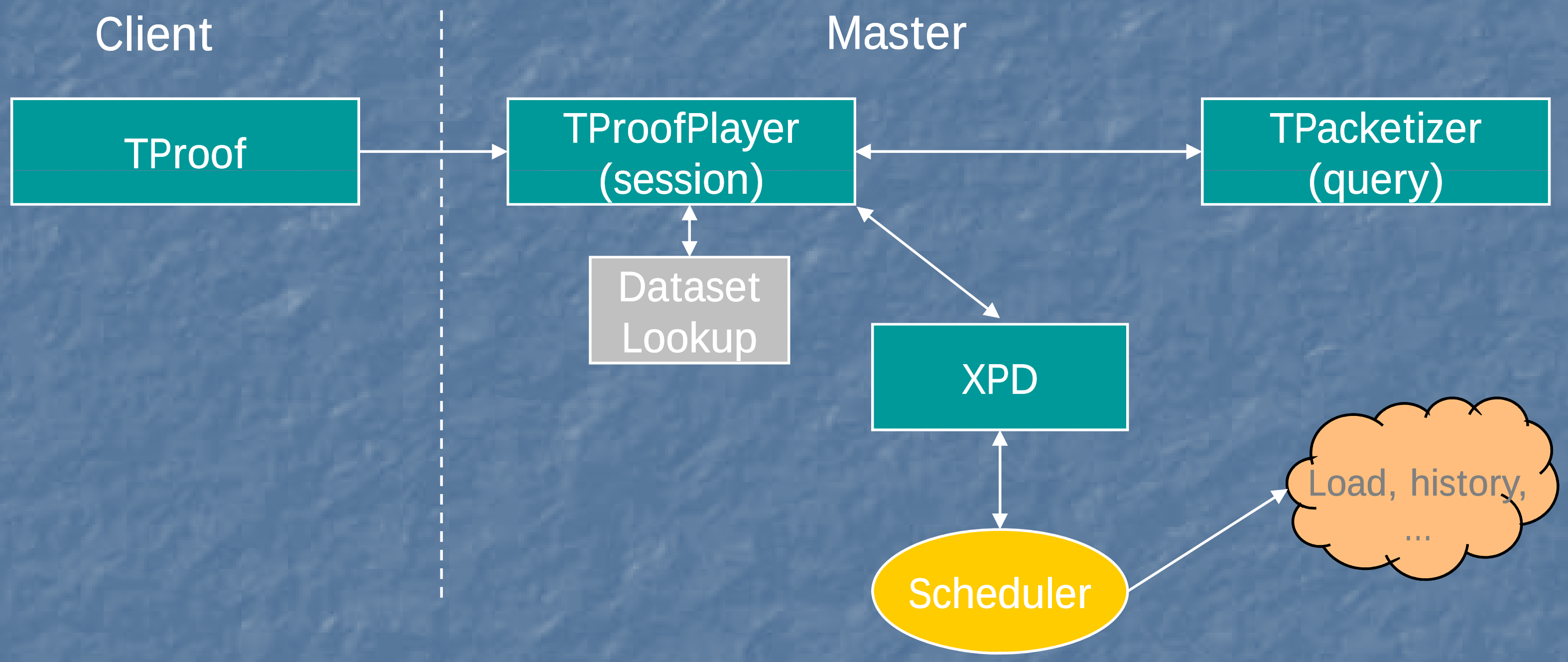
- Entity running on master XPD, loaded as plug-in
  - Abstract interface XrdProofSched defined

```
class XrdProofSched {
  …
public:
  virtual int GetWorkers(XrdproofServProxy *xps,
                         std::list<XrdProofWorker *> &wrks)=0;
  …
};
```

- Input:
  - Query info (via XrdProofServProxy ->proofserv)
  - Cluster status and past usage (e.g. from ML)
  - Policy
- Output:
  - List of workers to continue with

Central scheduling

- Schematic view



Central scheduling status

- Basic version in place (but not always enabled)
  - Selection a subset of workers based on
    - Round-robin, random, load (# of sessions)
- Version using the ML information to chose the best set of workers for a given user under test

Coming versions at CAF

- Later this week
  - Non-data driven processing
  - Output file merging
  - Fair share based on experiment policy
- Next (end of October)
  - Memory monitoring
  - Improved dataset handling

PROOF and SVN

- PROOF development branch
  - http://root.cern.ch/svn/root/branches/dev/proof
  - Synchronized daily with the main trunk
- Versions installed on CAF correspond to a revision on the dev branch
  - vPROOFDEV_r20285

Questions?

- Credits
  - B. Bellenot, G.G., J. Iwaszkiewizc, A. Kreshuk, F. Rademakers, L. Tran-Thanh (summer student ‘07)
  - G. Bruckner, M. Meoni, J.F. Grosse-Oetringhaus, A. Peters (ALICE)
  - A. Hanushevsky (SLAC)