# PROOF tutorial
# Analyzing trees

Gerardo Ganis, CERN, PH-SFT
gerardo.ganis@cern.ch

## TTree::MakeClass

- Method to create a class to loop over the TTree, read the entries and apply an algorithm

```
root [0] f = TFile::Open("data/event/event_tree_0_1_unsplit.root")
(class TFile*)0x102980580
root [1]
root [1] TTree *t = (TTree *) f->Get("EventTree")
root [2] t->MakeClass("ClassEvtUnsplit")
Info in <TTreePlayer::MakeClass>: Files: ClassEvtUnsplit.h and
    ClassEvtUnsplit.C generated from TTree: EventTree
(Int_t)0
```

- Do the same for the split case and compare the resulting classes

# TTree::MakeClass (2)

- Modify the macros to plot fNtrack and the Pt of tracks
  - See examples in macros/ClassEvt
- Run the two classes on the two files
- Compare

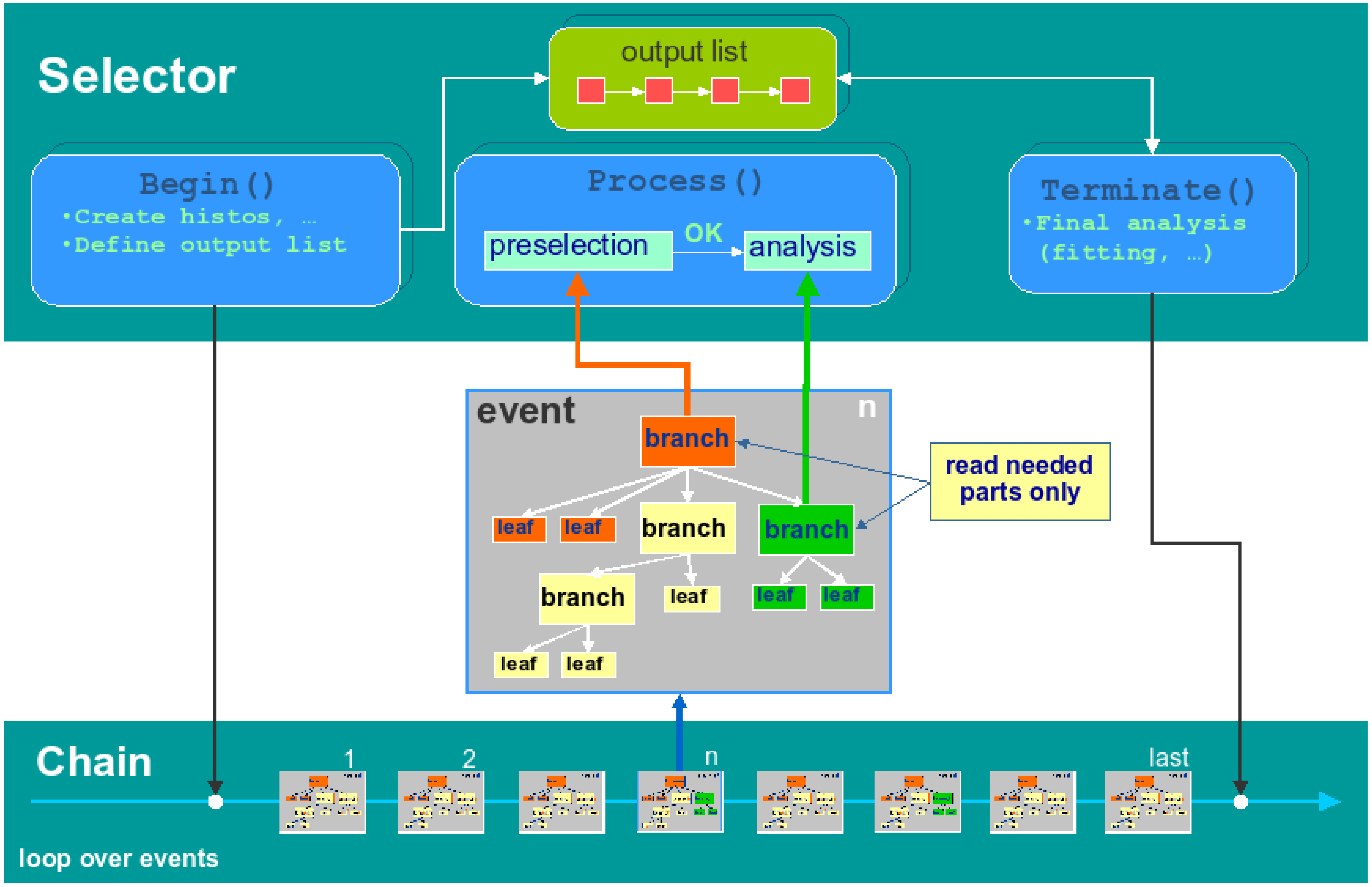
  TFile::GetFileBytesRead()

  the number of bytes read via file since the start of the session

# TTree::MakeSelector

- The classes produced by MakeClass work fine but they control the event loop (in Loop()) so that they cannot be used in PROOF
- TSelector in the class adapt to event-level parallelism
- TSelector does not control the event loop
- TSelector has been thought for PROOF

# ROOT model: Trees & Selector



# TSelector methods

- Begin()
  - Called on local session only
- SlaveBegin()
  - Called in the local session and on PROOF workers
  - This is te place where create output objects
- Process()
  - Called for each event in the workers
- SlaveTerminate()
  - Called on workers only
- Terminate()
  - Called on the client machine

## TTree::MakeSelector (2)

- Method to create a class to loop over the TTree, read the entries and apply an algorithm

```
root [0] f = TFile::Open("data/event/event_tree_0_1.root")
(class TFile*)0x1029829d0
root [1] TTree *t = (TTree *) f->Get("EventTree")
root [2] t->MakeSelector("SelEvt")
Info in <TTreePlayer::MakeClass>: Files: SelEvt.h and
 SelEvt.C generated from TTree: EventTree
(Int_t)0
```

- Do the same for the split case and compare the resulting classes
- Modify the macros to plot fNtrack and the Pt of tracks

## Read only used branches

- Both for ClassEvt and SelEvt you can choose to read only the required branches

```
    //   GetEntry(entry);
    b_event_fNtrack->GetEntry(entry);
    b_fTracks_fPx->GetEntry(entry);
    b_fTracks_fPy->GetEntry(entry);
```

- You can measure the effect of the selective read with TFile::GetFileBytesRead()

# Using the TTreeCache

- Selective pre-fetching and caching is an efficient way to increase performance when reading over the network
- The macro macros/createH1Chain.C defines a TChain with 4 files from ROOT HTTP
- The selector macros/h1analysis.C defines a simple analysis using those files
- The TTreeCache is enabled with
  TTree::SetCacheSize(Long64_t bytes)
- Compare running w/ and w/o enabling cache

# Using the TTreeCache (2)

```
root [0] root [0] .L macros/createH1Chain.C
root [1] TChain *chain = createH1Chain()
root [1] gROOT->Time()
root [2] chain->Process("macros/h1analysis.C+")
…
// Set cache at 30 MB
root [3] chain->SetCacheSize(30*1024*1024)
root [4] chain->Process("macros/h1analysis.C+")
…
```

- Cache helps also locally when having many processes using the disk, like in PROOF