PROOF tutorial
Analyzing D3PD

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

Introduction

- The D3PD are the files currently used for ATLAS analysis
- Nothing special in principle, but there are some things to be kept in mind when processing them on PROOF especially if the files are read from DPM or dCache
- Dissecting a working example allows also to show other useful PROOF functionality

D3PD file content

- The D3PD files contain two TTrees
  - 'CollectionTree', 5, branches
  - 'physics', 7493 branches
- The 'physics' tree is a split tree with a large number of branches
- Automatically generated TSelectors or classes have large number of members, branches and branch address setters.
  - Difficult to read/manage the header file
  - Takes very long to compile

Running SelPhysics

- Under 'd3pd': SelPhysics.h, .C
- Creates:
  - 2 histograms, 1 TTree
  - Merges via file
- Example of
  - How to use parameters to control running
  - Merge objects via file
  - Access files via PROOF itself

TProofOutputFile

- Allows to steer automatic merging via file
- Objects are written to the associated file
- The TProofOutputFile object is added to fOutput
- TProofOutputFile::Merge takes care to produce a merged file …
  - … or a dataset for the files
- In the future this will be all automatized ...

PROOF w/ DPM/dCache

- DPM (Disk Pool Manager) and dCache are two different ways to manage files
- Their purpose is very different but from the user point of view the allow access at files
- Both they provide an 'XROOTD door', which means that they can be accessed with the XROOTD client
  - This is typically they way it is done in ATLAS
- This works well for pure file access, but certain the related TSystem implementation (file discovering, etc …) is not complete

PROOF w/ DPM/dCache (2)

- That's where there some problems with PROOF
- The main reason is that all what is file discovering in PROOF is very XROOTD oriented (conceptually)
- So 'Validation' and dataset handling does not work out of the box ...
- … but there are some tricks

- The coming versions of DPM should solve the issues (not sure for dCache)

PROOF file lookup

- During the 'Validation' step PROOF tries to locate the files, i.e. to know on which processing node, if any, the files are
- If the files are on external storage elements this operation should just transparently say so; but for the DPM and dCache XRD-doors this blocks, so it needs to be skipped

```
// Skip the lookup step
proof->SetParameter("PROOF_LookupOpt", "none");
```

- Unfortunately there is no way to find this out automatically

Dataset manipulation

- Similarly, the usual tools to make datasets useful do not work
- The workaround consist in
  - A PAR file pars/DSVerify.par
    - Contains a TSelector dedicated to file verification, in parallel
  - A few macros tools/registerDataSet.C
    - Use DSVerify to register and verify datasets

DSVerify

- Shows a different usage of TSelector
  - Not only data processing
- It has to be used with a different 'Packetizer' which distributes 'files' instead of events
  - TPacketizerFile
  - Works on list of files
    - The 'event' is the file on which something has to be done
  - It could be used to parallelize at file-level the execution of programs/macro taking a file as input

tools/registerDataSet.C

- Contains three functions
  - Int_t registerDataSet(TFileCollection *fc, ...)
    - Steers the setup of TPacketizerFile to verify and register the file collection 'fc'
  - Int_t verifyDataSet(const char *dsn, ...)
    - Verify an already registered dataset, called 'dsn'
  - Int_t registerDataSet(const char *filewithlist, const char *dsname, …)
    - Wrapper to the above starting from a file list

makeZee.C

- Shows how to use the tools in registerDataSet.C

```
root[] TProof *p = TProof::Open("<master>");
root[] gROOT->ProcessLine(".L tools/registerDataSet.C+");
root[] registerDataSet("data/AlpgenJimmyZeeNp2_pt20.txt","Zee");
```

- Creates a dataset named 'Zee'

runProofFC.C

- Shows how to process a TFileCollection

```
 void runProofFC (const char *where = "pod://",
  const char *filewithlist = "data/AlpgenJimmyZeeNp2_pt20.txt")
{
   TProof *p = TProof::Open(where);
   if (!p) {
      Printf("Could not start PROOF at %s", where);
      return;
   }
   p->Load("src/SelPhysics.C+");
   gROOT->ProcessLine(".L tools/getTFileCollection.C+");
   TFileCollection *fc = getTFileCollection();

   p->SetParameter("PROOF_LookupOpt", "none");

   gROOT->ProcessLine(".L tools/getProofInfo.C+");
   TString datadir = getProofInfo("datadir");
   p->SetParameter("PROOF_OUTPUTFILE",
    TString::Format("%s/SelPhysicsOut.root", datadir.Data()));

   p->Process(fc, "SelPhysics");
}
```

runProof.C

- Shows how to process a named dataset

```
void runProof(const char *where = "pod://",
              const char *dsname = "Zee") {
  TProof *p = TProof::Open(where);
  if (!p) {
     Printf("Could not start PROOF at %s", where);
     return;
  }

  p->Load("src/SelPhysics.C+");

  gROOT->ProcessLine(".L tools/getProofInfo.C+");
  TString datadir = getProofInfo("datadir");
  p->SetParameter("PROOF_OUTPUTFILE",
          TString::Format("%s/HwwOut.root", datadir.Data()));

  p->Process(dsname, "SelPhysics");
}
```

tools/getProofInfo.C

- Example of how to extract information from PROOF using
  - Output redirection
  - Parsing with TMacro, TString