INTT Weekly Meeting

Joseph Bertaux

Purdue University

February 28, 2024

CDBTTree Intro

Overview

- `CDBTTree` is a wrapper class for normal ROOT `TTrees`
- Member functions save some of the trouble of creating and addressing `TBranches`
  - When writing or reading
- Output is stored in a normal .root file
  - Contains a `TTree` "Multiple"
    - Associated with calls to
    - `CDBTTree::SetDoubleValue`, etc
  - Contains a `TTree` "Single"
    - Associated with calls to
    - `CDBTTree::SetSingleDoubleValuestd::string const&, double)`, etc
- You can read/write files locally, and don't need to commit them to the calibrations database (CDB)
  - However, you retrieve files from the CDB by calling
  - `CDBInterface::instance()->getUrl`
  - to obtain a path

CDBTTree Intro

Relevant resources

- CDBTTree Source files on Github:
  - `#include <cdbobjects/CDBTTree.h>`
  - https://github.com/sPHENIX-Collaboration/coresoftware/blob/master/offline/database/cdbobjects/CDBTTree.h
  - https://github.com/sPHENIX-Collaboration/coresoftware/blob/master/offline/database/cdbobjects/CDBTTree.cc
- CDBInterface Source files on Github:
  - `#include <ffamodules/CDBInterface.h>`
  - https://github.com/sPHENIX-Collaboration/coresoftware/blob/master/offline/framework/ffamodules/CDBInterface.h
  - https://github.com/sPHENIX-Collaboration/coresoftware/blob/master/offline/framework/ffamodules/CDBInterface.cc
- I've found I need to link against PHOOL (`-lphool`) in addition to `-lcdbobjects` and `-lffamodules` to compile some local codes

CDBTTree Intro

Basic Example (Writing)

The basic structure of codes which generate `CDBTTrees` might look something like

```
CDBTTree cdb_ttree("cdb_ttree_file_name.root");
// Add some values, potentially in a loop
int size = 0, value = 0;
for(size = 0; size < 10; ++size) {
        // ...
        cdb_ttree.SetIntValue(size, "value_name", value);
}
// Probably store the size in its own single entry
cdb_ttree.SetSingleIntValue("size", size);
// Commit and write the CDBTTree
cdb_ttree.Commit();
cdb_ttree.CommitSingle();
cdb_ttree.WriteCDBTTree();
```

- You are free to add whatever single/multiple style of values you like
- However, the values should be POD types (specifically, `float`, `int`, `double`, `uint64_t`)
- Be sure to "commit" and write the `CDBTTree` when you finish

CDBTTree Intro

Basic Example (Reading)

The basic structure of codes which reads existing CDBTTrees might look something like

```
CDBTTree cdb_ttree("cdb_ttree_file_name.root");
cdb_ttree.LoadCalibrations();
// Loop over some values in the tree
for(int size = 0; size < cdb_ttree.GetSingleIntValue("size"); ++size) {
        // ...
        std::cout << cdb_ttree.GetIntValue(size, "value_name") << std::endl;
}
```

- You can see the convenience of storing the "size" of the CDBTTree
  - You can store multiple sizes if you have different "Multiple" branches
- Be sure to load the CDBTTree before attempting to retrieve values
- There are guards against loading values that don't exist, but there (currently) aren't guards against loading files that don't exist
  - You can implement such guards yourself, e.g.

```
#include <filesystem>
...
if(!std::filesystem::exists(filename)) // return, break, continue, throw
```

- https://github.com/sPHENIX-Collaboration/INTT/tree/main/general_codes/josephb/codes/cdb_example
- Implements most of what was discussed here, but a more complete/concrete example
- If you're curious, you can build it locally
  - run `git merge upstream/main` on a branch you are comfortable syncing with the main repository
  - run `make` to create main.o and main.exe
- run `./main.exe` to run the code main.cc and produce foo.root
- I'd encourage you to read main.cc to see what includes you need
- I'd encourage you to browse the contents of foo.root