Table of Contents

Analysis in SFrame

Introduction

The manTree package contains a set of classes for storing physics objects for final analysis. It can be used outside of athena, for example in SFrame. With the increasing size of data sets, for 8 TeV analysis onwards manTree ntuples shall not be created. Instead D3PD ntuples can be read into SFrame and then analysed either as D3PDs or converted to manTree objects at run time to allow existing analysis code to be used. The recommended way to compile the package is to follow the instructions below.

Analysis Framework

In order to do a complete analysis, there are a set of packages provided by ATLAS for applying common prescriptions. These packages are made available through the TopRootCore package and

we interface to these packages using the topUtils package.

First Time Setup

The following instructions should be followed to setup up & compile sframe, manTree and the common analysis packages:

Export CERN_USER variable:

```
export CERN_USER=yourcernlxplususername
```

Setup root v5.28:

```
source /opt/root/x86-64/root5.28.00/bin/thisroot.sh
```

Setup root core:

```
source /nfs/software/AtlasSoftware/RootCore/Root_v528_Summer2012Data/RootCore/scripts/setup.sh
```

Export SVNMAN variable:

```
export SVNMAN=svn+ssh://$CERN_USER@svn.cern.ch/reps/atlasinst/Institutes/Manchester
```

Go to a directory where you want to work from:

```
cd somedir
```

Check out sframe:

```
svn co https://sframe.svn.sourceforge.net/svnroot/sframe/SFrame/tags/SFrame-03-05-17 SFrameArea
cd SFrameArea
```

Build sframe:

```
source ./setup.sh
make
```

Check out manTree and D3PD packages:

```
svn co $SVNMAN/SFrame/manTree/trunk manTree
svn co $SVNMAN/SFrame/SFToolInterfaces/trunk SFToolInterfaces
svn co $SVNMAN/SFrame/convertD3PDObjects/trunk convertD3PDObjects
svn co $SVNMAN/SFrame/d3pdSFrameBase/trunk d3pdSFrameBase
svn co $SVNMAN/SFrame/topUtils/trunk topUtils
svn co $SVNMAN/SFrame/PlottingUtil/trunk PlottingUtil
svn co $SVNMAN/SFrame/exampleD3PDCycle/trunk exampleD3PDCycle
```

Build manTree packages:

```
cd manTree
make
```

```
cd ../SFToolInterfaces
make
cd ../convertD3PDObjects
make
cd ../d3pdSFrameBase
make
cd ../topUtils
make
cd ../PlottingUtil
make
cd ../exampleD3PDCycle
make
```

Make a soft link to the RootCore par file (needed for proof running):

```
cd $SFRAME_LIB_PATH
ln -s $ROOTCOREDIR/../RootCore.par .
```

Setup for Each Session

Each time you start a new terminal and want to work on the analysis you need to do the following steps:

Go to your sframe directory:

```
cd pathto/SFrameArea
```

Export CERN_USER variable:

```
export CERN_USER=yourcernlxplususername
```

Setup root v5.28:

```
source /opt/root/x86-64/root5.28.00/bin/thisroot.sh
```

Setup root core:

```
source /nfs/software/AtlasSoftware/RootCore/Root_v528_Summer2012Data/RootCore/scripts/setup.sh
```

setup sframe:

```
source ./setup.sh
```

It's probably most convenient to put these commands into a script to save typing them in each time.

Running the Example Cycle

There is an example sframe cycle in exampleD3PDCycle. It can be run with the following (don't forget the setup commands):

```
cd $SFRAME_DIR
cd exampleD3PDCycle
sframe_main config/TestCycle_config.xml
```

Converting from manTreeSFrame to d3pdSFrame Cycles

It was intended that the user code would be as similar as possible to when running over manTrees, however some changes were necessary. For an example of how these changes are implemented you may wish to compare src/TestCycle.cxx located in exampleD3PDCycle with its equivalent located in exampleManTreeCycle.

include/YourCycle.h

Within your cycle's header file there are only two modifications required to access the D3PD. The header file include to the parent class, and the subsequent inheritance.

D3PD code required	Old manTree equivalent
`#include "d3pdSFrameBase/include/d3pdSCycleBase.h"`	`#include "manTreeSFrameBase/include/manTreeSCyc`
`class YourCycle : public d3pdSCycleBase {`	`class YourCycle : public manTreeSCycleBase {`

src/YourCycle.cxx

There are several modifications that should be made to allow previous manTree cycles to run on D3PDs due to the nature in which the data is read.

Similar to the header file the constructor's call to its parent class constructor should be changed.

D3PD code required	Old manTree equivalent
`YourCycle::YourCycle() : d3pdSCycleBase() {`	`YourCycle::YourCycle : manTreeSCycleBase() {`

With manTree analysis in order to optimise the cycle it was necessary to remove branches from the list to be read in. This is no longer necessary as within the D3PD framework variables are only read in as they are needed for the first time. Inclusion of the command to disable branches within the D3PD framework will result in a warning at runtime.

D3PD code required	Old manTree equivalent
-	`disableBranch( std::string );`

Within the new framework it is now necessary to tell the D3PD reader which event it should be reading. This is done by adding the line

```
GetEntry(data);
```

as the first line of your ExecuteEvent member function.

As the physics objects are only read and filled as they are needed, rather than at the start of the analysis, it is no longer possible to access objects one of the previously available manners. Taking the muons as an example, previously they could be accessed either by

```
m_muon
```

or alternatively by

```
m_event->getMuons()
```

In the new regime, only

```
m_event->getMuons()
```

is valid.

config/YourCycle_config.xml

The config file for your cycle needs to contain links to the new D3PD package libraries in place of the manTreeSFrameBase package.

D3PD code required	Old manTree equivalent

`<Library Name="libManTopD3PDReaderGen"/>`
`<Library Name="libSFToolInterfaces"/>`
`<Library Name="libconvertD3PDObjects"/>`
`<Library Name="libd3pdSFrameBase"/>`
	`<Library Name="libmanTreeSFrameBase"/>`

For the input data sets, the data's version should contain the type of D3PD being run over. For instance to use NTUP_SM, you may wish to set the version to

```
<InputData Version="NTUP_SMWZ.Reco">
```

In addition the input tree name should be set to physics using

```
<InputTree Name="physics" />
```

ManTree Completeness

There are a number of variables that were previously held in manTrees which have no D3PDs equivalent. For this reason code written to work purely with manTrees may attempt to access data members that are no longer available.

Accessing the D3PD Directly

It is also possible to access the D3PD directly from the cycle without converting it to a manTree, however this code is dependent on the type of ntuple being read. Some variables held within the D3PD do not have a manTree counterpart, and so this is the only way of accessing them.

This can also be useful for optimising code, as when a D3PD object is read into a manTree object, every data member is filled. Hence if a cut is being performed on a single variable which is held in the D3PD, a significant amount of time can be saved by accessing the variable directly from the D3PD and cutting there, prior to filling the manTree.

-- SteveMarsden - 03-Jul-2012

This topic: AtlasSandbox > SFrameD3PD
Topic revision: r5 - 2012-07-10 - MarkOwen