Table of Contents

Di-Boson studies

Introduction

This page describes the ongoing activities in the CAT team on di-boson analysis.

Analysis framework (work-in-progress)

The analysis is done with a C++ framework based on SFrame (https://svnweb.cern.ch/trac/atlasgrp/browser/Institutes/CERN/ElectroweakBosons/trunk, temporally available at https://svnweb.cern.ch/trac/atlasusr/browser/mbellomo/ElectroweakBosons/trunk). Instructions to checkout and run the code are provided below.

Overview

The framework is organized in a small number of simple SFrame packages, namely two packages for the "core" software:

- **AnalysisBase** : the main analysis cycle is called "AnalysisManager", this cycle executes a configurable number of analysis tools. The base analysis class is "AnalysisToolBase", base for analysis tools to be run inside the main cycle. The class to access D3PD variables is "Event"; and few other classes to handle particle corrections;

some external packages:

- **SFrame** : the underlaying analysis framework;
- **SMultiFrame** : an extension of SFrame to allow for multi-tool configuration;
- **RootCore** : small package to handle ROOT based detector performance codes (pile-up rw, grl package, muon corrections, etc...);

an analysis example package:

- **SMultiFrameUser** : contains a simple example to show how multi-tool cycle works;
- **AnalysisExample** : contains an analysis example: a couple of tools and a configuration xml file;

and finally packages for specific analysis (some still to be created...)

- **D3PDSkimmer** : code to skim D3PDs;
- **AnalysisZmumu** : Z -> mumu inclusive cross-section analysis code
- **AnalysisWmunu** : W -> munu inclusive cross-section analysis code
- **AnalysisWW** : WW analysis code
- **AnalysisWZ** : WZ analysis code
- **AnalysisZZ** : ZZ analysis code
- **AnalysisHWW** : H->WW analysis code

Checkout and compile

Set the lxplus shell to BASH
In order to checkout the trunk version of the code do:

```
svn co svn+ssh://$USER@svn.cern.ch/reps/atlasgrp/Institutes/CERN/ElectroweakBosons/trunk Electrow
```

If instead you want to check out the latest stable tag (reccomended) do:

```
svn co svn+ssh://$USER@svn.cern.ch/reps/atlasgrp/Institutes/CERN/ElectroweakBosons/tags/Electrowe
```

Instructions to get additional code (SFrame, RootCore packages), setup and compile are given in the README file.

Run the example

To run the example coded in AnalysisExample package, do:

```
sframe_main AnalysisExample/config/AnalysisExample_config.xml
```

Two analysis tools are execute event-by-event, namely AnalysisToolOne and AnalysisToolTwo. Look at them for details on how to connect and access input variables, book histograms, book output tree branches and loop over events.

Howto create your own analysis package

In order to create a new analysis package based on SMultiFrame a built-in command is available:

```
smultiframe_new_package.sh AnalysisXXX
```

This will create directories for header files (AnalysisXXX/include) for source files (AnalysisXXX/src), for configuration files (AnalysisXXX/config), PROOF support (AnalysisXXX/proof) plus a Makefile to compile it.

In order to create a new analysis tool within the analysis package AnalysisXXX do:

```
cd AnalysisXXX
smultiframe_create_tool.py -n MyAnalysisTool
```

This will create the header and source files plus a configuration file.

Run with PROOF

Example configuration file can be turned to run on PROOF-lite easily just changing mode from "LOCAL" to "PROOF". How fast does it run?

Running a selection of Zmumu events and filling, in each event, a set of about 50 histograms 10 times (after each cut of the selection, for a total of about 500 histograms) on 16-core lxplus over 5M signal events stored on castor:

```
....
Validating files: OK (500 files)
 ( INFO  )  TPacketizerAdap... : fraction of remote files 1.000000
[TProof::Progress] Total 4999129 events |================>...| 81.80 % [13243.8 evts/s, 59.8 MB/s
```

Run in GRID with prune

tba

Setup for PROOF on Demand () on lxplus

Remember to set the lxplus shell to BASH

0) Have a look at video tutorial from PoD site: http://pod.gsi.de/video_tutorials.html
They are quite quick and very much related to what you ar going to do...

1) Get PoD code from http://pod.gsi.de/download.html

```
cd $HOME
wget http://pod.gsi.de/releases/pod/3.6/PoD-3.6-Source.tar.gz
tar -xzvf PoD-3.6-Source.tar.gz
```

2) setup for PoD: source the setup file for PoD on lxplus that is shipped with the code together with the code setup itself (see below). For installation unrelated to ElectroweakBosons code see http://pod.gsi.de/documentation.html

```
cd ElectroweakBosons
source scripts/setup_lxplus.sh
source scripts/setup_pod_lxplus.sh
```

Note that you can safely ignore at this stage the error for the missing file PoD_env.sh. This file will be added later after installation is complete and will be sourced correctly each time the PoD environment is setup. The setup_pod_sh script is source here to setup the other evn variables needed for PoD compilation (see the script for details).

3) compile PoD

```
cd PoD-3.6-Source
mkdir build
cd build
cmake ../BuildSetup.cmake ..
make -j8 install
```

The "build" directory is used in order to build PoD. If you need to reinstall PoD from scratch just delete this "build" directory and restart.

4) change in $HOME/.PoD/PoD.cfg

from

```
work_dir=$HOME/.PoD
```

to

```
work_dir=/tmp/$USER/PoD
```

5) add a file called "user_worker_env.sh" to $HOME/.PoD directory, with this content:

```
#! /usr/bin/env bash

echo "Setting user environment for workers ..."

export LD_LIBRARY_PATH=/afs/cern.ch/sw/lcg/external/qt/4.4.2/x86_64-slc5-gcc43-opt/lib:/afs/cern.

echo "LD_LIBRARY_PATH=$LD_LIBRARY_PATH"
```

Run analysis for PROOF on Demand () on lxplus

0) Start from a new shell and setup

```
source ElectroweakBosons/scripts/setup_lxplus.sh
source ElectroweakBosons/scripts/setup_pod_lxplus.sh
```

1) Start a PoD server

```
pod-server start
```

look for the "PROOF connection string" (you can get it also doing "pod-server status" once the server is running).
For instance:

```
mbellomo@lxplus406_trunk$ pod-server status
XPROOFD [10298] port: 21001
PoD agent [10317] port: 22001
PROOF connection string: mbellomo@lxplus406.cern.ch:21001
```

This string is the one that need to be set in the analysis SFrame xml file to run on the PoD cluster.
For instance:

```
 <Cycle Name="AnalysisManager" RunMode="PROOF" ProofServer="mbellomo@lxplus406.cern.ch:21001" Pro
    OutputDirectory="/tmp/$USER/" PostFix="" TargetLumi="1.0">
```

2) Start N workers:

```
pod-submit -r lsf  -q 1nh -n 20
```

Look with "pod-submit -h" for options. This will start "20" workers on the "lsf" batch system on the "1nh" queue.
Check with "bjobs" when jobs are running (these jobs will start the PROOF deamons on the worker nodes).
Check with:

```
pod-info -n
```

when all the requested workers are available. When done, you are ready to send you job on the PoD cluster!

Have fun!

Write out ntuples with SFrame/PoD

Writing of ntuples in SFrame when **running locally or with PROOF-Lite** is very easy. You just need:

- declare what variables need to be written in BeginInputData with calls like: DeclareVariable(m_myvar, "MyVar")
- fill the variable in the code
- specify in the xml that a ntuple has to be written adding in the InputData block a line with

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

Be aware that 1 global tree will be written by each instance of the same tool, that will be overwritten in case of multiple instances writing the same variable.

When **running with a PROOF cluster** everything is the same but you need make master and workers to talk each other through the network instead of using the filesystem (that is the default behavior). This you get easily adding under $HOME/.PoD directory a file called "user_xpd.cf" containing this line:

```
xpd.rootd allow
```

and restart the pod server if it was already running.

Documentation

- Tutorial meeting 28/09/2011: https://indico.cern.ch/conferenceDisplay.py?confId=156700

Tips and Tricks

How to get XML syntax highlighting in emacs

- Get nxml package from : http://www.thaiopensource.com/nxml-mode/
- Unzip under your $HOME
- Add these lines to your .emacs file

How to add a new skim and submit to the grid

everything should be quite simple if you check-out, compile and run from a clean bash shell on lxplus.

0) Get the trunk code and setup it as explained in README file

```
svn co svn+ssh://$USER@svn.cern.ch/reps/atlasgrp/Institutes/CERN/ElectroweakBosons/trunk Electrow
```

1) Add to AnalysisBase/include/EventBase.h the variables that are missing and that you want to add to your skim. All and only the variables listed in this class are skimmed. We need to check how many extra variables you may need to see if we want to have them in all skims or setup some flags to decide depending on the analysis.

2) Code a function in D3PDSkimmer/src/Skimmer.cxx to apply your skimming condition (D3PD variables can be accessed as ev.variable_name. This function need to be called in ExecuteEvent function, see for others. This is driven by a configuration flag (see for others.) The job is then started reading an xml file, where this flag need to be set to true (make it false by default).

3) Add your skim option (assign it a name "PhoSkim", whatever) to grid/prun.sh script. At the beginning is where the skim type is set. This flag is then propagated to a script (grid/grid_run.sh) that calls the corresponding grid configuration file depending on the skim. So edit grid/grid_run.sh, add appropriate condition for "PhoSkim", or whatever, to call the grid/grid_config_mc_photon.xml or grid/grid_config_data_photon.xml files that you also need to add (just copy existing ones and change the XML property value as appropriate to activate your photon skim).

4) Test locally using D3PDSkimmer/config/Skimmer_config.xml (setting an input file and activating your filter, see for others.)

```
sframe_main D3PDSkimmer/config/Skimmer_config.xml
```

5) prun_prod.sh, the script to submit jobs to grid, reads lists of data/mc datasets from data/MC11_p833 and data/Muons_p833_periods, etc...

6) To submit to the grid you need to tweak a bit prun_prod.sh script. - define the skim version (keep v0117) but maybe add a ".test" to run a single job for testing purposes. If you read the comments in the script you will see there is a way to run on a single dataset to test over grid. - define the skim flag (VarSkim, 2LepSkim, 3LepSkim, ecc) - add your username in the list at the beginning, as the only one uncommented - uncomment 50-53 the list of periods you want to process (it's read from an external file). - make the script executable

```
source grid/setup_prun.h
./grid/prun_prod.sh
```

```
(load "~/nxml-mode-20041004/rng-auto.el")

(setq auto-mode-alist
      (cons '("\\.\\(xml\\|xsl\\|rng\\|xhtml\\)\\'" . nxml-mode)
        auto-mode-alist))
```

* Set ALLOWTOPICVIEW = atlas-current-physicists * Set ALLOWTOPICCHANGE = atlas-current-physicists

Major updates:
-- MassimilianoBellomo - 12-Sep-2011

%RESPONSIBLE% MassimilianoBellomo
%REVIEW% **Never reviewed**

-- MassimilianoBellomo - 08-Dec-2011

This topic: Main > AtlasCATElectroweakBosons
Topic revision: r7 - 2012-07-20 - OldrichKepka