Suez and Event Display

Istvan Danko

(reusing Jim Pivarski's slides from last year)

June 7, 2007
Outline

- Overview of Suez
- Demonstration of Suez with the Event Display
- Raw data versus Pass2
- Walk-through of creating a processor for analysis
- Filling histograms
- Homework: repeat this presentation!
CLEO-c detector
Physics event

- Data consists of events

- Events contain objects like tracks and showers built from low level hits by our reconstruction software
CLEO-c data

Raw data (electronic signals from detector) -> Pass1/Caliper fast reconstruction for diagnoses/calibration

Pass2 reconstruction (tracks, showers, etc.) -> D-tag (D-skim) reconstructed $D_{(s)}$ meson candidates

Typical analysis uses pre-processed Pass2 and/or D-tag data

This makes analysis faster:

- Physics objects (tracks, showers) are already done
- Event sample is greatly reduced in case of D-tag (D-skim) data
Suez

- Suez is the User Interface to access and process CLEO data
- Functionality is provided by plug-ins (modules, producers, processors) loaded at run time
- Uses a command line interface with full scripting language (Tcl) – it is like a unix shell
Suez data access

- Data is accessed through the Frame

Frame: a “snapshot” of CLEO at a particular instant of time, formed by the most recent Record in each data Stream

![Diagram of data access through Frame]
CMS experiment at LHC
Follows the same model (thanks to Chris Jones)
Communication via the Frame

**Data Providers:** data returned when requested

**Sources:** data from storage
- Event Database
- Calibration Database

**Producers:** data from algorithm
- Pi0Finder
- RareBTracks

**Processors:** analyze and filter data
- SelectBtoKPi
- EventDisplay

**Sinks:** store data
- Event List

**Data Requestor:** sequentially run requestors for each new Record from a source
Processing data in SUEZ (1)

Typical data analysis

```plaintext
module GoGetDataModule

prod sel CalibrateDataProd
prod sel IdentifyTracksProd } order does not matter
proc sel FindGlueballsProc
proc sel MakePlotsOfGlueballsProc } order matters

```

GoGetDataModule → CalibrateDataProd → IdentifyTracksProd → FindGlueballsProc → MakePlotsOfGlueballsProc
Processing data in suez (2)

- module GoGetDataModule
- prod sel CalibrateDataProd
- prod sel IdentifyTracksProd
- proc sel FindGlueballsProc
- file out ToSkimFile
- go

Later, you can run MakePlotsOfGlueballsProc on your skim file
Demonstration: Event Display

The Event Display is a set of processors that draw CLEO data on the screen, event by event. It’s useful for

- Data sanity check (e.g. during data-taking)
- Identifying backgrounds
- Introduction to CLEO/data analysis
Setup (in bash)

Connect to a Linux computer (e.g. lnx102)

For bash/sh users (type “echo $SHELL” to identify),

- . /nfs/cleo3/offline/scripts/cleo3logins
- . /nfs/cleo3/offline/scripts/cleo3defs
- c3rel 20070330_FULL
- export USER_SRC=$HOME/my_src
- export USER_BUILD=/cdat/tem/mccann/build/$C3LIB
- export USER_SHLIB=$USER_BUILD/Linux/shlib
- c3rel $C3LIB (yes, again!)

Be sure to make the following directories (all shells):

- mkdir -p $HOME/my_src
- mkdir -p $HOME/my_tcl
Create a suetz (tcl) script

- cd $HOME/my_tcl
- favorite_text_editor hitsandeverything.tcl &

Fill it with the following:

```tcl
# Get raw events so we can look at hits
module sel EventStoreModule
eventstore in 20070607 daq all runs 226000 226042

# All the things you need to process raw data
run_file $env(C3_SCRIPTS)/getNewestConstants.tcl
run_file $env(C3_SCRIPTS)/trackingDataFull.tcl
run_file $env(C3_SCRIPTS)/CcP2.tcl

# Run the event display
run_file $env(C3_SCRIPTS)/view_command.tcl
view -display_only ShowerAttributes DRHits ZDHits SeedTrack
    KinePionFit DBEventHeader StandAloneGeom

go
```

raw data

No need to run view_command.tcl in recent releases
Run suez with your script

\[ \text{suez -f hitsandeverything.tcl} \]
Things to do with Event Display

1. Move windows within windows, zoom around, look at side view, select hits

2. Step through events ("Continue" button); zip through them ("Auto Advance" button)

3. SeedTracks (green) versus fitted tracks (KinePion, pink)

4. Calorimeter shower representation: label with energy

5. Make DR hits circles

6. Make ZD hits lines
Cheat shit for 4, 5, 6

View menu → Set 2D Representation...
Pass2 (reconstructed) data

```bash
$ cd $HOME/my_tcl
$ favorite_text_editor afterpass2.tcl &

# Get pass2’ed events (no hits, but much faster!)
module sel EventStoreModule
eventstore in 20070607 physics all

# Setup standard analysis and event display
setup_analysis

# Run the event display
run_file $env(CS_SCRIPTS)/view_command.tcl
view -display_only Pass2

go
$ suez -f afterpass2.tcl
```

1. Note the missing hits
2. See what happens when you press “Auto Advance”
First data analysis

Many events look like this:

- 2 tracks
- minimal calorimeter energy ($E/p \lesssim 10\%$)
- misses the beamspot
Selecting cosmic rays

We will make a processor that identifies cosmic rays

- cd $HOME/my_src
- mkproc -track -shower -histogram MySecondProcessor
- favorite_text_editor MySecondProcessor/Class/MySecondProcessor.cc &

The processor is already filled with a lot of example code. We’ll ignore that and add the following block at the beginning of the MySecondProcessor::event method (line 153). Before editing, the code looks like this:

```
ActionBase::ActionResult
MySecondProcessor::event( Frame& iFrame ) // anal3 equiv.
{
  report( DEBUG, kFacilityString ) << "here in event()" << endl;

  → insert new code here ←

  // Create a table of tracks and fill it.
  FATable< NavTrack > trackTable;
  extract( iFrame.record( Stream::kEvent ), trackTable );
```
```cpp
int number_of_tracks = 0;
double closest_to_beamline = 1000.; // meters

FATable<NavTrack> tracks;
extract(iFrame.record(Stream::kEvent), tracks);

for (FATable<NavTrack>::const_iterator track = tracks.begin();
    track != tracks.end();
    ++track) {

    double distance_from_beamline = fabs(track->pionHelix()->d0());
    if (distance_from_beamline < closest_to_beamline) {
        closest_to_beamline = distance_from_beamline;
    }

    double track_momentum = track->pionFit()->momentum().mag();
    if (track_momentum > 1.) { // greater than 1 GeV/c
        number_of_tracks++;
    }
}
```
The new code (part 2)

FATable<NavShower> showers;
extract(iFrame.record(Stream::kEvent), showers);

double biggest_shower_energy = 0.; // GeV

if (showers.size() > 0)
{
    // the showers table is sorted by energy
    biggest_shower_energy = showers.begin()->attributes().energy();
}

// Now filter the events
if (number_of_tracks == 2 &&
    closest_to_beamline > 0.05 &&
    biggest_shower_energy < 0.3) // two tracks above 1 GeV/c each
    // more than 5 cm from beamline
    // less than 300 MeV
{
    return ActionBase::kPassed;
}
else
{
    return ActionBase::kFailed;
}
Compile code and run

- `c3make`
- `cd $HOME/my_tcl`
- `favorite_text_editor afterpass2.tcl &`

In `afterpass2.tcl`, add the processor after setting up the view command but before calling it:

```tcl
run_file $env(C3_SCRIPTS)/view_command.tcl
proc sel MySecondProcessor
view -display_only Pass2
```

Run `suez`

- `suez -f afterpass2.tcl`

and press the “Auto Advance” button in the Event Display. See how you have biased the events!
1. In $HOME/my_src/MySecondProcessor/MySecondProcessor/MySecondProcessor.h, add HIHist1D* m_histphi; after HIHist1D* m_histo1;

2. In $HOME/my_src/MySecondProcessor/Class/MySecondProcessor.cc, add m_histphi = iHistoManager.histogram("phi", 100, 0., 2.*M_PI); at the end of MySecondProcessor::hist_book(){ } (line 144)

3. Also add the following just before your return ActionBase::kPassed; (line 181)
   
   for (FATable<NavTrack>::const_iterator track = 
       tracks.begin(); track != tracks.end(); ++track)
   {
       m_histphi->fill(track->pionHelix()->phi0());
   }

4. Recompile (described on previous page)
Histogram track $\varphi$ (cont.)

5. At the beginning of `$HOME/my_tcl/afterpass2.tcl`, add

   ```tcl
   # Load a histogram manager.
   module sel HbookHistogramModule
   hbook file myhistograms.rzn
   hbook init
   ```

6. Also add `proc sel HistogramViewerProc after` `proc sel MySecondProcessor`.

7. Run `suez -f afterpass2.tcl` in your `$HOME/my_tcl` directory.

8. Select Root $\rightarrow$ MySecondProcessor $\rightarrow$ phi from the heirarchy on the left of the HistogramViewer window.

Homework

1. Walk through the demonstrations on your own.

2. Study $e^+e^- \rightarrow \mu^+\mu^-$: plot the $\cos \theta$ distribution.

Hints:

- Collision-borne muons deposit the same energy in calorimeter showers as cosmic ray muons.
- But $e^+e^- \rightarrow \mu^+\mu^-$ events must come from the beam.
- Given a `FATable<NavTrack>::const_iterator` track, the momentum vector components are obtained by `track->pionFit()->momentum().x(), .y(), and .z()`.
- $\theta$ is the polar angle between $\sqrt{x^2 + y^2}$ and $z$.
- p. 137 Peskin & Schroeder: $P(\cos \theta) \propto 1 + \cos^2 \theta$
To learn more

- CLEO 3/c software page:
  https://wiki.lepp.cornell.edu/lepp/bin/view/CLEO/Private/SW/WebHome
- Source code cross reference:
  http://www.lepp.cornell.edu/restricted/webtools/cleo3/
- Class member functions:
- Navigation Objects overview (NavTrack, NvShower, etc.):
- Description of (some) individual libraries:
- Guide to EventStore: