


Alignment Monitoring

Jim Pivarski, Dmitry Yakorev, Alexei Safonov

Texas A&M University

13 April, 2007

Alignment monitoring: roughly four categories

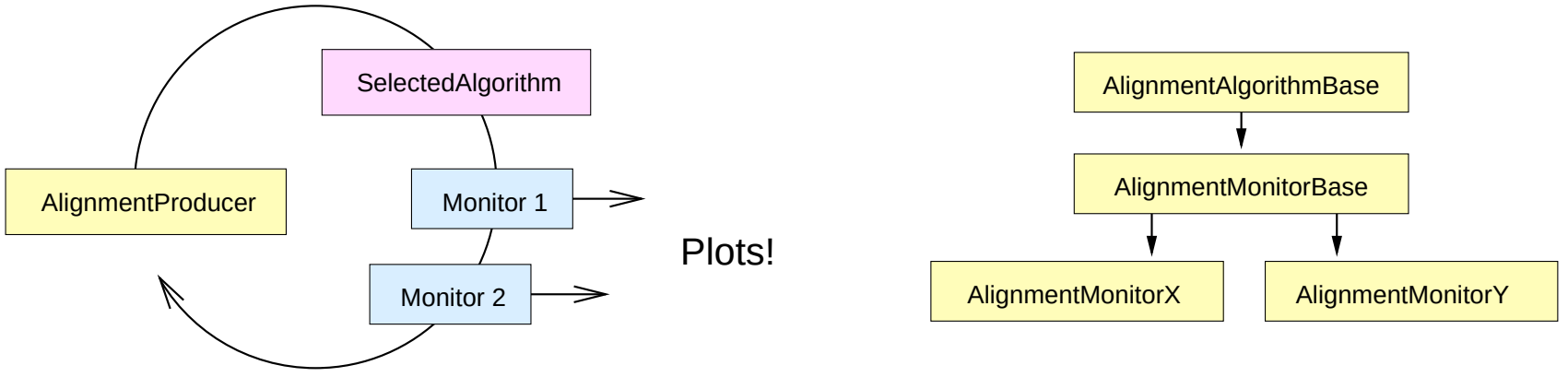
1. DQM-based monitoring upstream of alignment process (reports an error if alignment used online is wrong)
2. Sanity checks in AlignmentProducer (convergence, improvement in residuals, overlap plots, p_T)
3. Geometry Validation— compare output geometries from different alignments: have the chambers moved?
4. Validation with reconstructed tracks: is it better?

Status of 2: Foundation for plotting modules in AlignmentProducer

Status of 3: Reading from multiple SQLite files, plotting difference

Plotting Modules in AlignmentProducer (#2)

- Extends existing plugin mechanism to add monitoring modules



- Lowers "potential barrier" to adding plots
- Independent of algorithm (though we can make AlignmentMonitorHIP, AlignmentMonitorMillePede, etc.)
- Aware of iteration number, selected alignables and parameters
- Collects and merges histograms/profiles from a distributed job

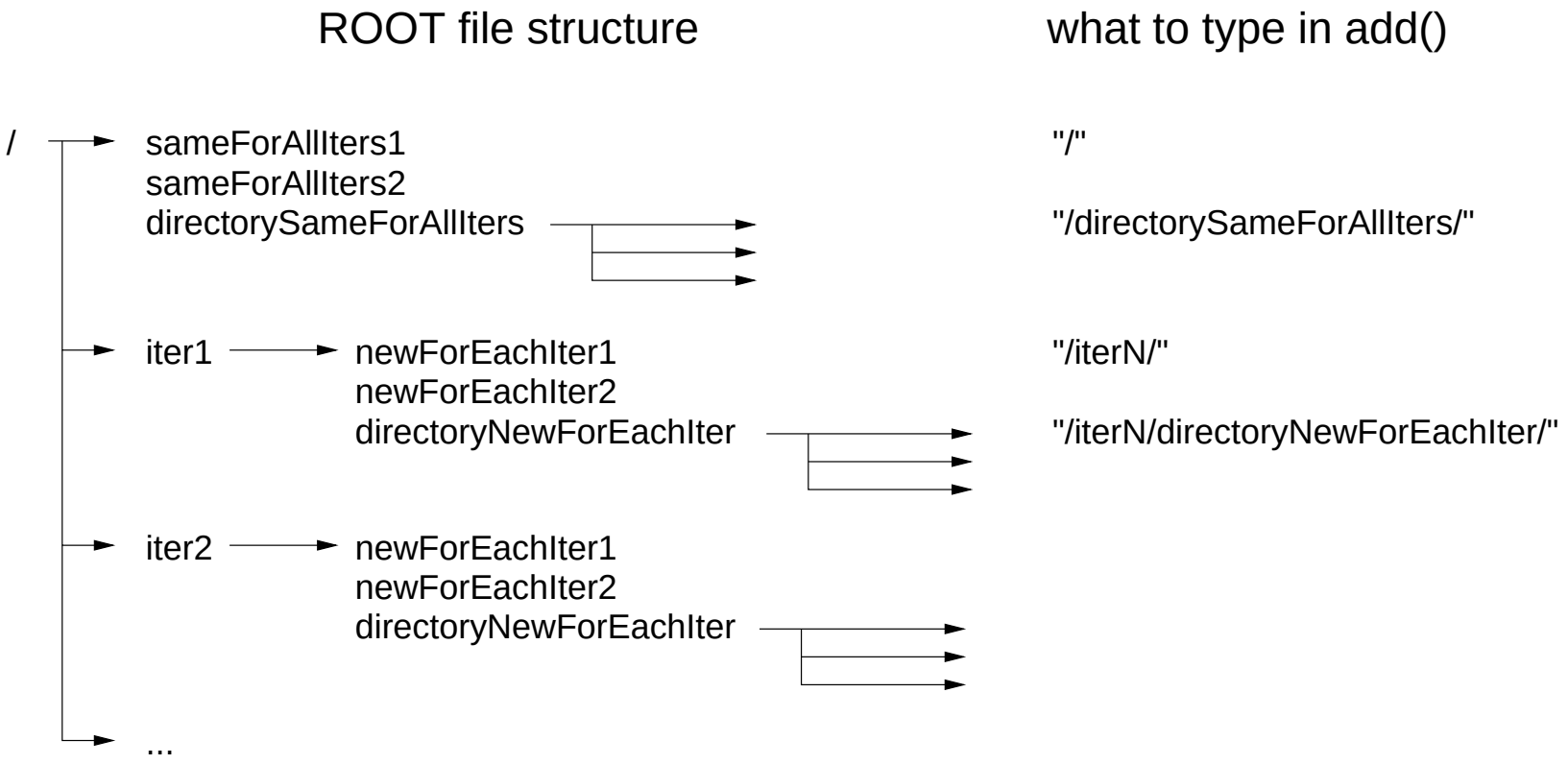
Plotting Modules in AlignmentProducer (#2)

```
replace AlignmentProducer.monitorConfig = {
   untracked vstring monitors = {"AlignmentMonitorHIP"}
   PSet AlignmentMonitorHIP = {
      string outpath = "./"
      string outfile = "histograms.root"
      bool collectorActive = false
      int32 collectorNJobs = 0
      string collectorPath = "./"
   }
}

void AlignmentMonitorHIP::book() {
  m_sameForAllIters = (TH1F*)(add("/", new TH1F(...)))
  m_newForEachIter = (TH1F*)(add("/iterN/", new TH1F(...)))
}
```

Plotting Modules in AlignmentProducer (#2)

```
m_sameForAllIters = (TH1F*)(add("/", new TH1F(...)))
m_newForEachIter = (TH1F*)(add("/iterN/", new TH1F(...)))
```



Nothing more is needed for collection jobs

Plotting Modules in AlignmentProducer (#2)

What works (tested for 25 events):

- Loading an arbirtary number of modules
- Arbitrarily-deep ROOT directory structure
- Iteration (via `AlignmentProducer.maxLoops` and/or files)
- Merging histograms/profiles from a distributed job
- Generating histograms from selected Alignables

What's next:

- Add lots of plots to AlignmentMonitorHIP
- Test with lots of events
- Put into CVS?

Reading from multiple databases (#3, Dmitry)

Basic structure:

- Compiled C++ ROOT GUI
- Forks `cmsRun` processes which read SQLite files

What works:

- Loading two geometry files
- Calculating and displaying translation differences
- Tabs to switch between plots

What's next:

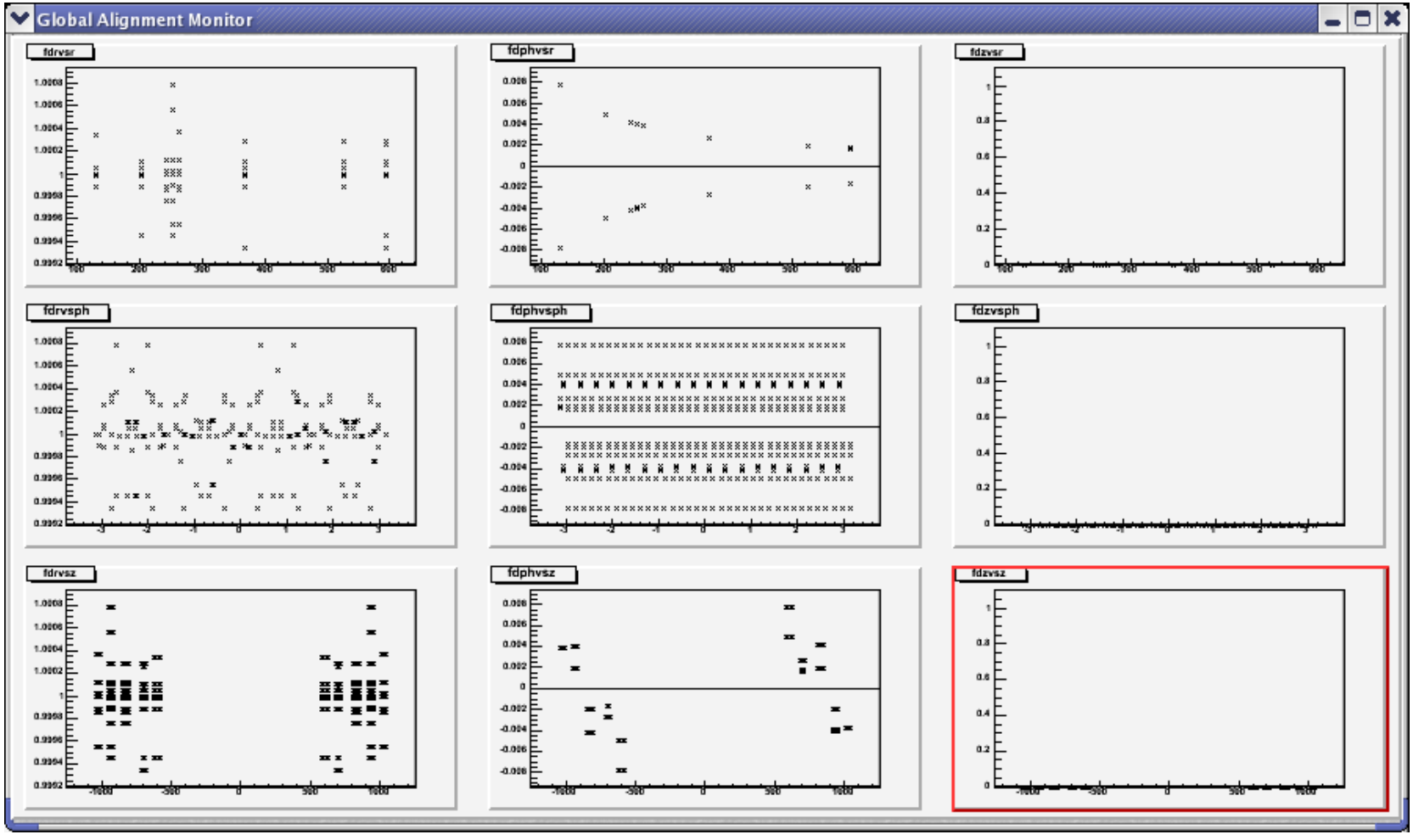
- Read from multiple databases— plot versus time
- Represent differences in rotation angles
- A different framework? DQM? Iguana? Compile database access into ROOT GUI?

Reading from multiple databases (#3, Dmitry)

vs. R

vs. ϕ

vs. Z



$(\vec{x_1} - \vec{x_2})$ $(\vec{x_1} - \vec{x_2}) \cdot \hat{\phi}$ $(\vec{x_1} - \vec{x_2}) \cdot \hat{Z}$

Summary

2. Sanity checks in AlignmentProducer (convergence, improvement in residuals, overlap plots, p_T)

Laid a foundation that handles multiple modules, ROOT directory structure with iterations, merging histograms after parallel processing

3. Geometry Validation— compare output geometries from different alignments: have the chambers moved?

Measuring geometry differences across multiple SQLite files, beginnings of a GUI tool