

# Java RootIO



## ➤ Implementation:

➤ *Reading*

➤ *Writing*

➤ *Xrootd*

## ➤ Tools:

➤ *Root Object Browser*

➤ *AIDA Tools*

➤ *Web Interface*

➤ *ColMan*

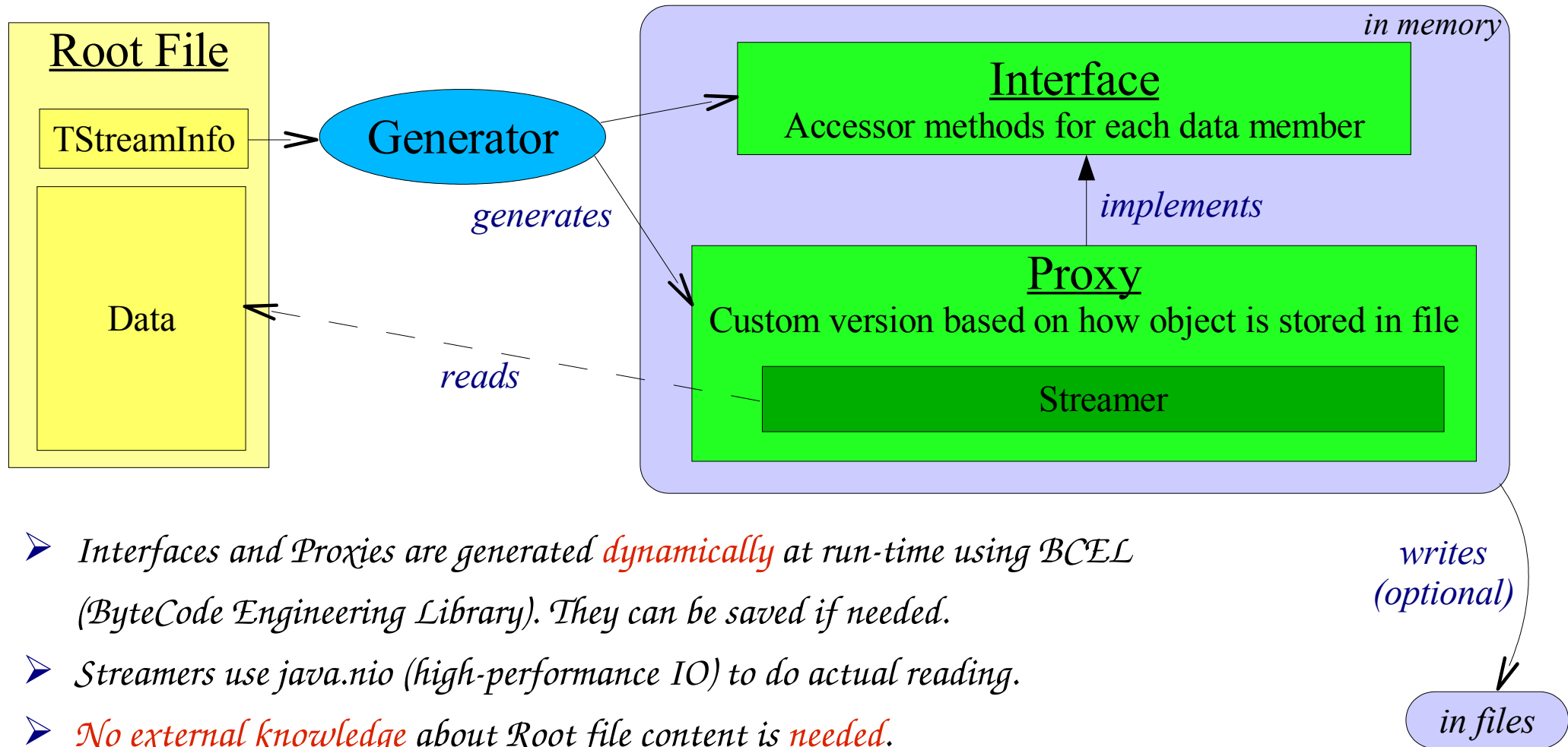
## Made by:

➤ *SLAC: Tony Johnson, Mark  
Donszelmann, Victor Serbo, Max  
Turri*

➤ *LAL: Julius Hrivnac*

➤ *Prague: Petr Vokac*

# Implementation



- Interfaces and Proxies are generated *dynamically* at run-time using *BCEL* (ByteCode Engineering Library). They can be saved if needed.
- Streamers use *java.nio* (high-performance IO) to do actual reading.
- *No external knowledge* about Root file content is *needed*.
- *No C++* is needed.
- Some foundation Root classes are not described in *TStreamInfo*, or are described incorrectly. Their Interfaces and Proxies have been pre-implemented.
- 12k lines of code (Rio: 15k, Root: 200k).

# Problems

- *Root file format is completely **undocumented** so a lot of **archaeological** work is needed. Every new Root release brings surprises.*
  - *System is quite robust against changes.*
  - *Each problem is included as a test file. Tests currently exist for Root versions 3.00.06, 3.01.06, 3.02.07, 3.03.07, 3.05.07, 3.10.02, 4.00.08.*
- *LCG Pool has “invented” another private format of streaming information (apparently based on Seal dictionary). This has not yet been decrypted so Pool files are unreadable, except for trivial cases (like `AttributeList`, etc.).*

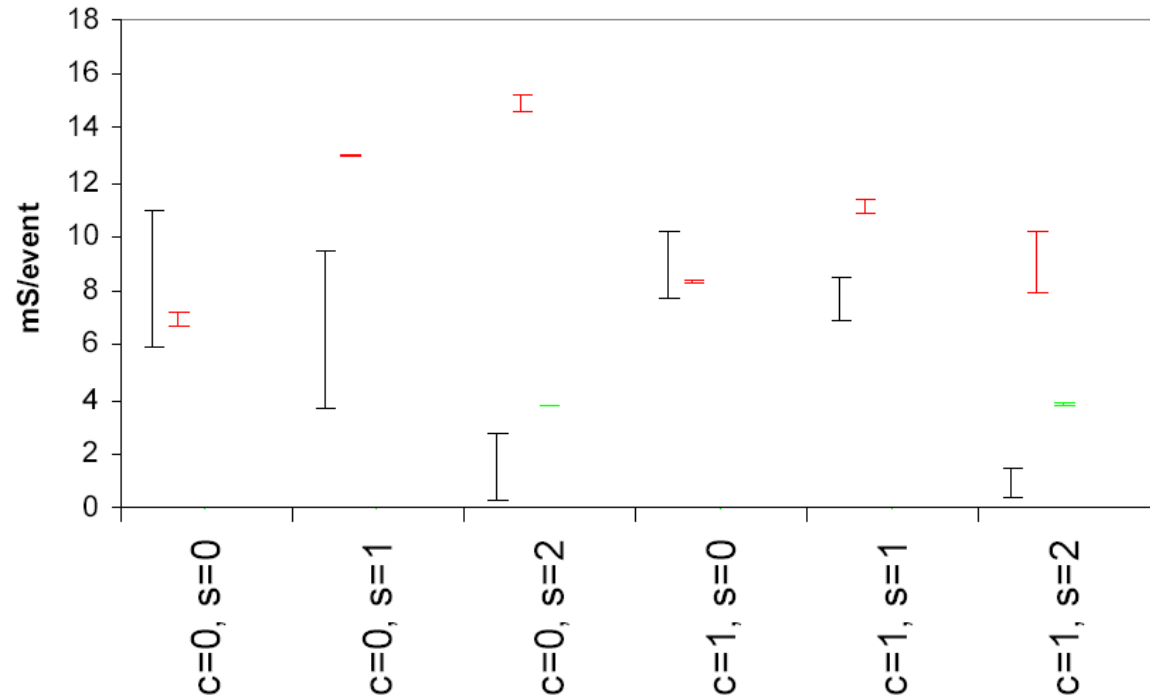
# Example of Use

```
1 import hep.io.root.*;
2 import hep.io.root.interfaces.*;
3 import java.io.IOException;
4 import java.util.*;
5
6 public class EventTest
7 {
8     public static void main (String args[]) throws IOException
9     {
10         RootFileReader reader = new RootFileReader("Event.root");
11         TTree tree = (TTree) reader.get("T");
12         TBranch branch = tree.getBranch("event");
13         int n = branch.getNEntries();
14
15         for (int i=0; i<n; i++)
16         {
17             Event e = (Event) branch.getEntry(i);
18             List l = e.getTracks();
19             System.out.println("NTracks="+e.getNtrack()+" "+l.size());
20             Iterator it = l.iterator();
21             while (it.hasNext())
22             {
23                 Track t = (Track) it.next();
24                 double px = t.getPx();
25                 //etc...
26             }
27         }
28     }
29 }
30
```

# Benchmarks

- *Task: Loop over all events, loop over a tracks, sum px for all tracks. 5000 events files created with "event" test program distributed with Root.*
- *Java 1.4 on MS Windows used.*
- *S = Split level = 0,1,2*
- *C= Compression = 0,1*
- *Each test run 5 times in succession, lin show range in results*
  - *in later runs file is typically cached in memory == faster if task IO limited.*

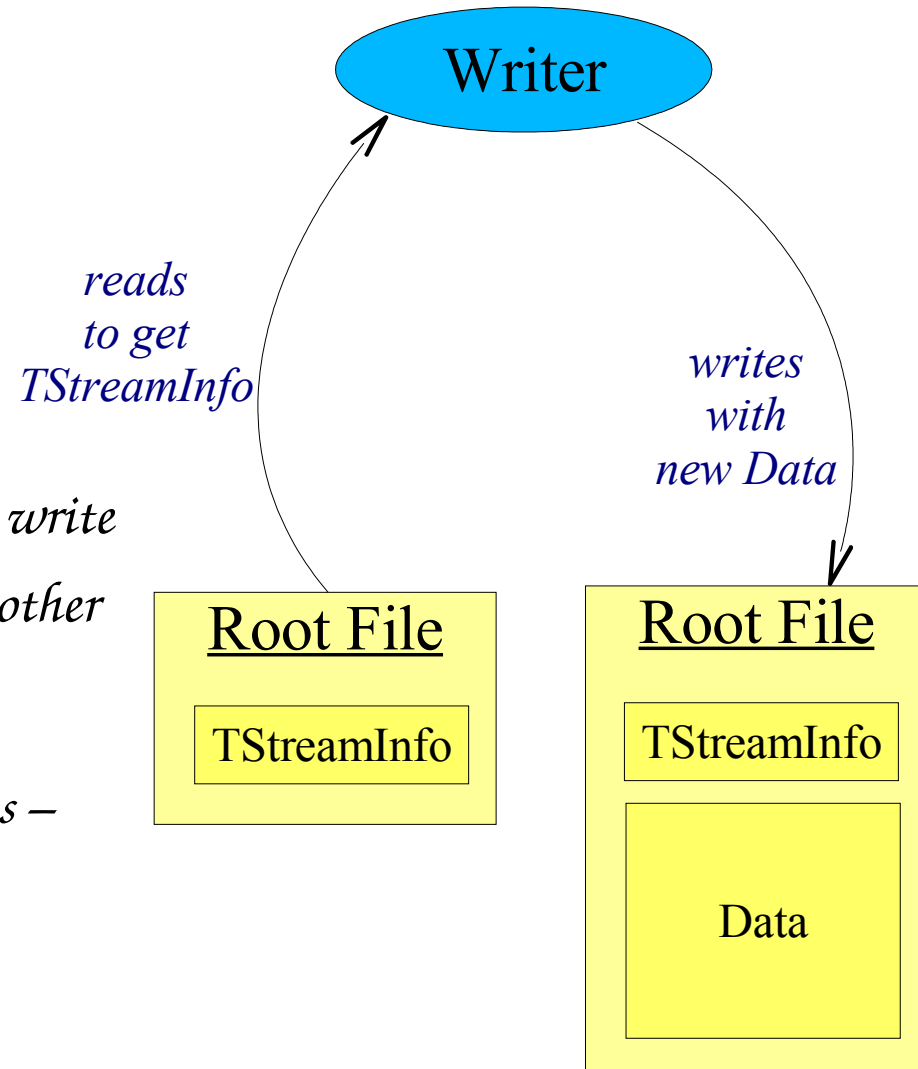
Java Root Benchmark



- *Red = Root*
- *Green = Root (coded for split)*
- *Black = Java*

# Writing

- Standalone application (written by student P.Vokac).
- It will be merged into FreeHEP.
- Limitations:
  - Can't create new StreamInfo entry – can only write objects of classes which has been read from another file.
  - Even if it could create new StreamInfo entries – those may not be acceptable by Root.
  - Has problems with TTrees.
- Good enough for packages like AIDA.

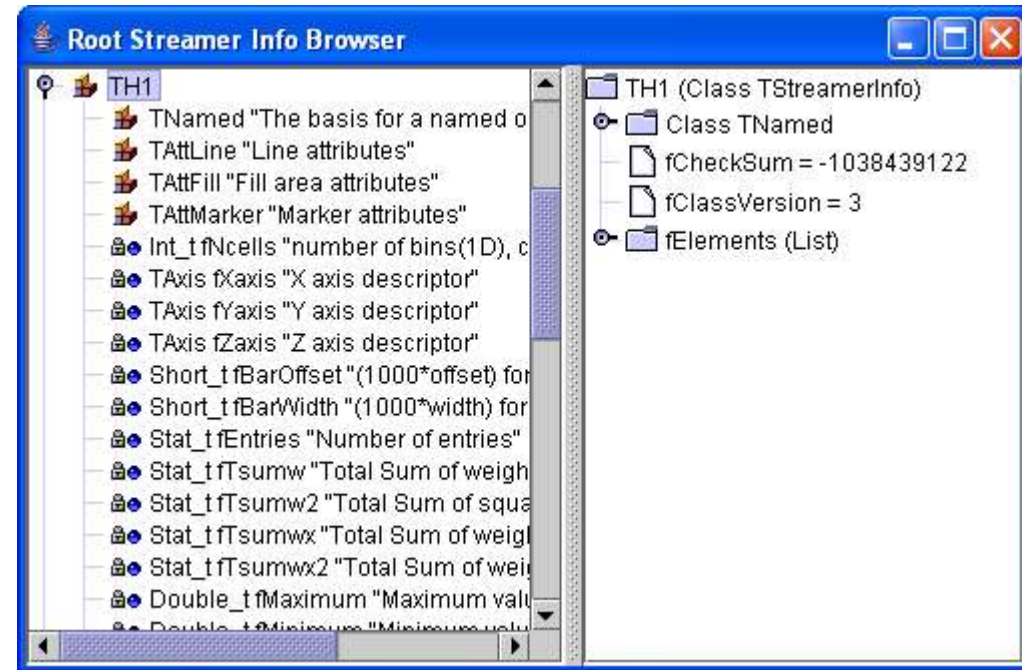
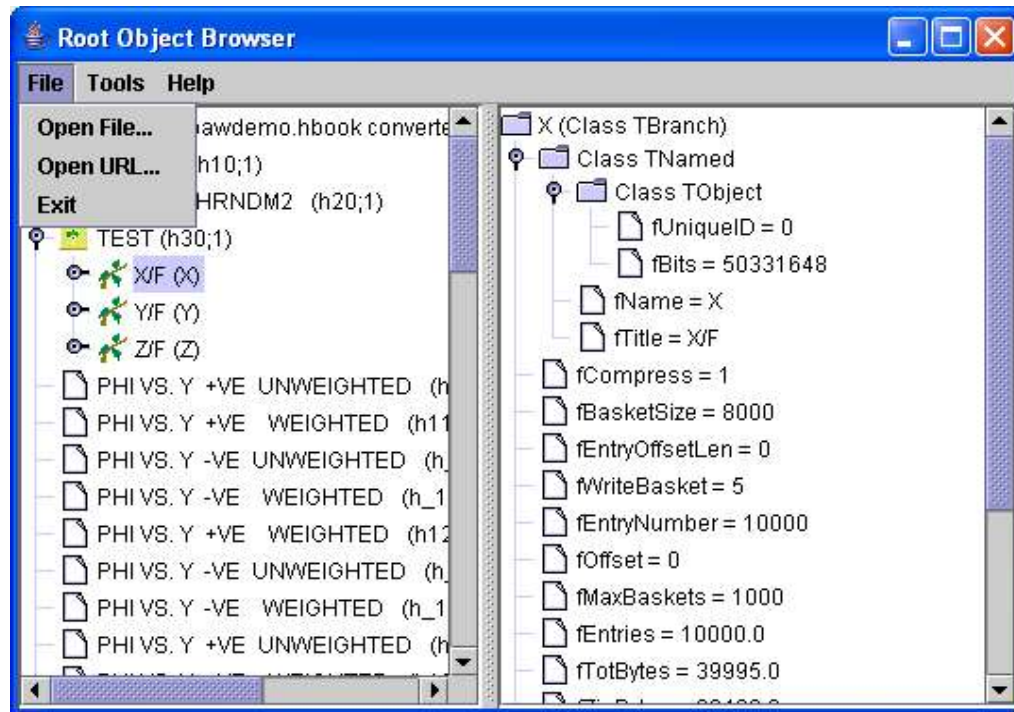


# Xrootd

- *Ability to talk directly to rootd/xrootd server.*
- *Authenticator to authenticate to rootd (not all authentication methods implemented yet).*
- *Extended RootFileReader to accept as URL:*
  - *file:\* to read directly files*
  - *root:\* to access rootd/xrootd (random access)*
  - *http:\* to access http server (use “ranges” if supported by server)*

# Root Object Browser

➤ Allows to visit data as well as StreamInfo.

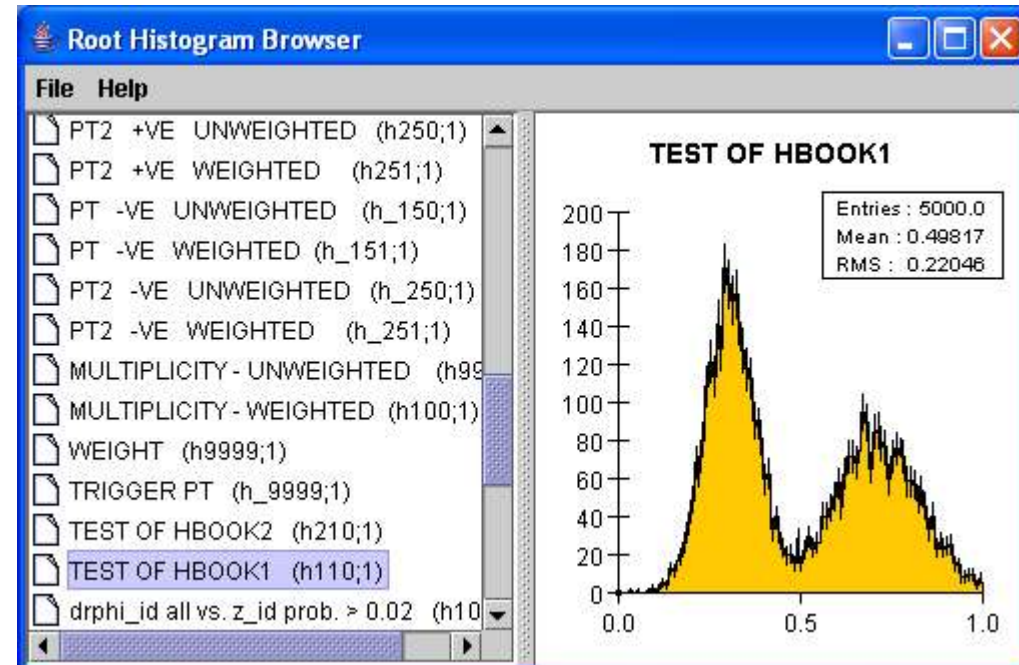




# AIDA on Root Files

- *JAIIDA (FreeHEP implementation of AIDA) provides AIDA interface for Root NTuple/Histogram/... files.*
- *Can be used directly, in **Root Histogram Browser** or within **JAS3**.*

```
1 import hep.aida.*;
2 import java.io.IOException;
3
4 public class AidaExample
5 {
6     public static void main(String[] args) throws IOException
7     {
8         IAnalysisFactory af = IAnalysisFactory.create();
9         ITreeFactory tf = af.createTreeFactory();
10
11         //ITree tree = tf.create("myFile.root", "root");
12         ITree tree = tf.create("root://sldrh2.slac.stanford.edu/pawdemo.root",
13                               "root", true, false, "scheme=anonymous");
14
15         IHistogram1D h100 = (IHistogram1D) tree.find("/h100");
16         IHistogram2D h210 = (IHistogram2D) tree.find("/h210");
17
18         IPlotter plotter = af.createPlotterFactory().create();
19         plotter.createRegions(1,2);
20         plotter.region(0).plot(h100);
21         plotter.region(1).plot(h210);
22         plotter.show();
23     }
24 }
25
```



# AIDATLD

FREE HELP

➤ AIDA Tag Library for use in JSP (Java Server Pages) to put AIDA on the Web.

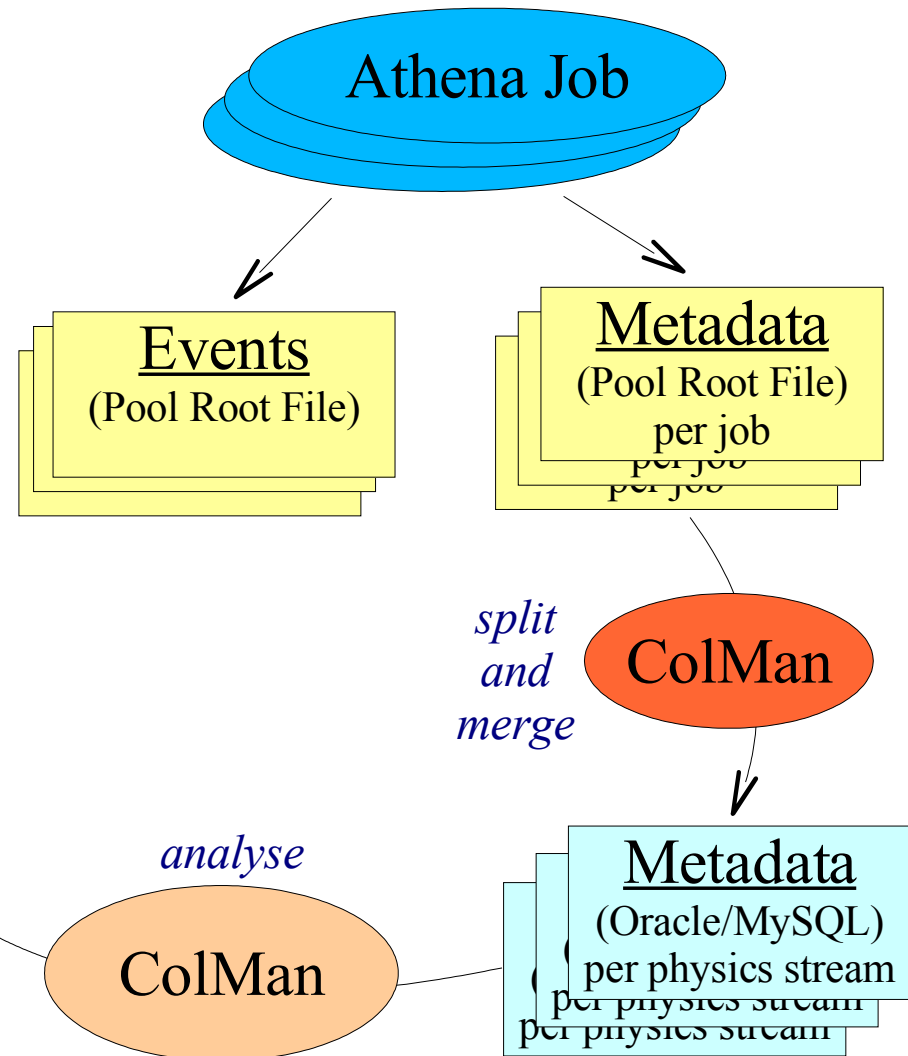
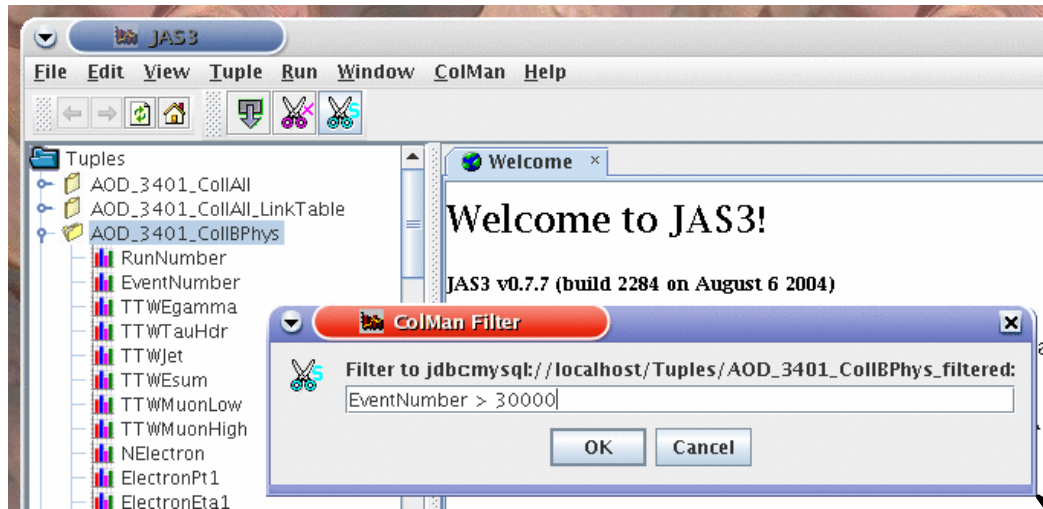
The screenshot displays the AIDA Tag Library interface within a Microsoft Internet Explorer browser window. The interface is divided into several sections:

- Tree View:** A hierarchical tree on the left lists test categories such as AllGamma, MC Stuff, Tracker, Tracker 2, Tracker 3, Trig+Cal, ACD Stuff, Cal Elayer, Cal Nlayer, CAL, CAL Recon, More CAL, Position Hits, Integrating H, IM Variables, Event Variabl, VerticalGamma1, VerticalGamma1, VerticalGamma1, and VerticalGamma1.
- Plots:** Three plots are shown in the center-left area:
  - Quality - Track Quality:** A plot showing track quality versus track energy, with data points for v4r4 (red) and v4r3 (blue).
  - Track Hit Plane Z - Track Hit Plane Z:** A plot showing track hit plane Z versus track hit plane Z, with data points for v4r4 (red) and v4r3 (blue).
  - Track Energy - Track Energy:** A plot showing track energy versus track energy, with data points for v4r4 (red) and v4r3 (blue).
- Table:** A table on the right displays statistical results for various tests. The table has columns for AndersonDarling, KolmogorovSmirnov, FiszCramerVonMises, Tiku, chi2, and Goodman. The row for TKRTRKQUALITY is highlighted in red.

	AndersonDarling	KolmogorovSmirnov	FiszCramerVonMises	Tiku	chi2	Goodman
PARTCOUNTMC	0	1.0000	0	0.55289	1.0000	1.0000
MCX	0	1.0000	0	0.55301	1.0000	1.0000
MCY	0	1.0000	0	0.55301	1.0000	1.0000
ENERGYMC	0	1.0000	0	1.0000	1.0000	1.0000
MCTERMZ	0	1.0000	0	0.76768	1.0000	1.0000
TKRTRKQUALITY	29.228	9.4367E-9	4.8933	1.7390E12	1.0355E4	2.3905E8
TKRTRKSTARTY	6.1811E-6	1.0000	1.4904E-6	0.55297	1.0000	1.0000
TKRHITPLANEZ	0	1.0000	0	1.0000	1.0000	1.0000
ENERGYMC	0	1.0000	0	1.0000	1.0000	1.0000
TKRTRKENERGY	0.055949	1.0000	0.055388	0.87548	0.99999	0.98377
TKRNUMTRACKS	0	1.0000	0	0.55289	1.0000	1.0000
TKRTRKNHITS	0	1.0000	0	0.55293	1.0000	1.0000
TKRTRKSLOPEY	1.2197E-3	1.0000	3.6317E-4	0.55462	1.0000	0.99991
TKRTRKSLOPEX	9.4535E-4	1.0000	2.7238E-4	0.55420	1.0000	0.99996
TKRTRKSTARTZ	0	1.0000	0	0.55293	1.0000	1.0000
TKRTRKSTARTX	6.4997E-6	1.0000	3.0538E-6	0.55298	1.0000	1.0000
TKRNUMVERTICES	6.9749E-3	1.0000	3.8423E-3	0.57118	1.0000	0.99993
TKRVTXQUALITY	0.070245	1.0000	0.14438	0.39140	0.99986	0.98658
TKRVTXNUMTRK	0.41459	1.0000	0.044087	0.86842	0.79470	0.96836
TKRVTXSTARTZ	4.4159E-3	1.0000	3.3697E-3	0.56890	1.0000	0.99951

# ColMan

- Allows management of (LCG Pool) *AttributeLists* (Tag databases – Event Metadata) between SQL tables (using *SQLTuple*), AIDA XML files and Pool Root files.
- Can be used as JAS3 Plugin.
- Has been used in *Atlas DC2*.



# More Info

- *FreeHEP Java Library:*
  - <http://java.freehep.org>
- *FreeHEP Root IO:*
  - <http://java.freehep.org/lib/freehep/doc/root>
- *JAI DA:*
  - <http://java.freehep.orgt/jaida>
- *AIDATLD:*
  - <http://aidatld.freehep.org>
- *JAS3:*
  - <http://jas.freehep.org/jas3>
- *ColMan:*
  - <http://home.cern.ch/hrivnac/Activities/Packages/ColMan>