Using JAS3 for LCD Analysis

Tony Johnson

20th May 2003

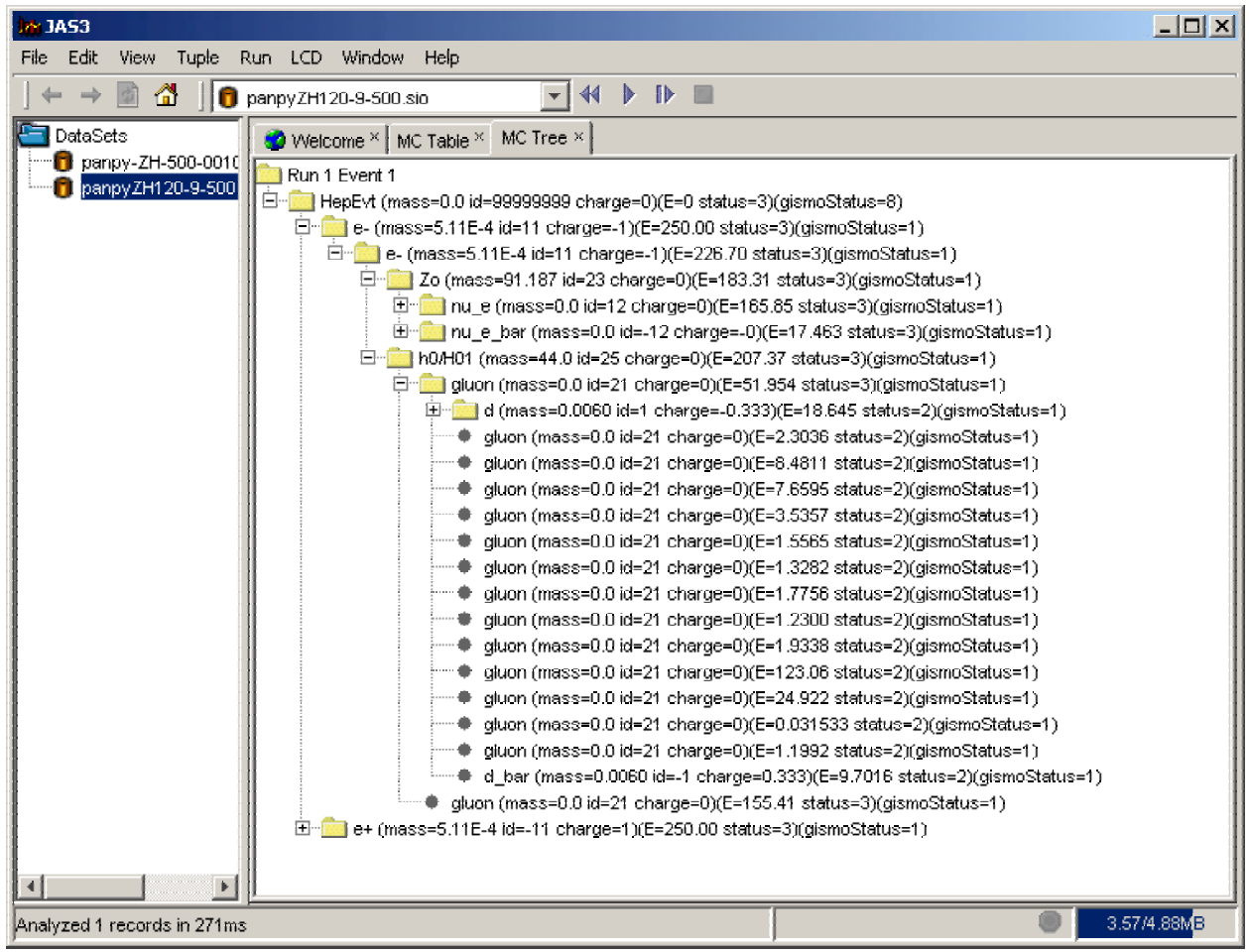
Migration Goals

- Migrate from JAS2 -> JAS3
  - Able to support both in parallel for at least a while (how long)
    - Without user having to make too many changes
    - Without saddling ourselves with too much legacy code.
- Would also like to use some more recent features of Java (in particular collections)
- Would also like to switch to LCIO
  - LCIO has defined its own event classes similar but not the same as our existing hep.lcd.event classes.

JAS3 for LCD Status

- Full support for .stdhep files
- Full support for .sio files
- Initial support for LCIO files
- Tools Ported
  - MC Tree/Table, WIRED, all working
  - Old LCD Event Display not ported
    - Could be done if requested

MC Particle Tree



MC Particle Table

JAS3

File Edit View Tuple Run LCD Window Help

panpyZH120-9-500.sio

DataSets
- panpy-ZH-500-0010
- panpyZH120-9-500

Welcome | MC Table | MC Tree

Run:1 Event: 1 Etot: 390.68

N	Type	Status	Parent	PX	PY	PZ	E
0	HepEvt	Documentati...		0	0	0	0
1	e-	Documentati...	0	0	0	250.00	250.00
2	e-	Documentati...	1	0	0	226.70	226.70
3	Zo	Documentati...	2	155.27	29.223	2.4095	183.31
4	nu_e	Documentati...	3	159.82	42.380	12.954	165.85
5	nu_e	Final State	4	159.82	42.380	12.954	165.85
6	nu_e_bar	Documentati...	3	-4.5439	-13.157	-10.545	17.463
7	nu_e_bar	Final State	6	-4.5439	-13.157	-10.545	17.463
8	h0/H01	Documentati...	2	-155.27	-29.223	60.320	207.37
9	gluon	Documentati...	8	-26.318	-36.387	-26.126	51.954
10	d	Intermediate	9	-13.383	-9.7228	-8.5963	18.645
11	IDontknowino	Intermediate	10	-155.27	-29.223	60.320	207.37
12	rho-	Intermediate	11	-5.5212	-3.9421	-3.9711	7.8982
13	pi-	Final State	12	-1.0293	-0.50015	-0.48886	1.2523
14	pi0	Intermediate	12	-4.4918	-3.4419	-3.4823	6.6459
15	gamma	Final State	14	-3.5031	-2.7531	-2.7510	5.2364
16	e+	Documentati...	15	-2.8628	-2.2499	-2.2482	4.2793
17	e-	Documentati...	15	-0.64030	-0.50321	-0.50283	0.95710
18	gamma	Final State	14	-0.98864	-0.68880	-0.73125	1.4094
19	e+	Documentati...	18	-0.70766	-0.49267	-0.52349	1.0087
20	e-	Documentati...	18	-0.28104	-0.19595	-0.20785	0.40072
21	Deltao	Intermediate	11	-4.6577	-4.4684	-3.8128	7.5933
22	n	Final State	21	-3.6985	-3.2898	-2.8354	5.7814
23	triton	Documentati...	22	-0.15145	0.39345	-0.40765	2.8698
24	p	Documentati...	22	-1.7998	-2.7772	-1.7572	3.8627
25	n	Documentati...	22	-0.57529	-0.50687	-0.64138	1.3718

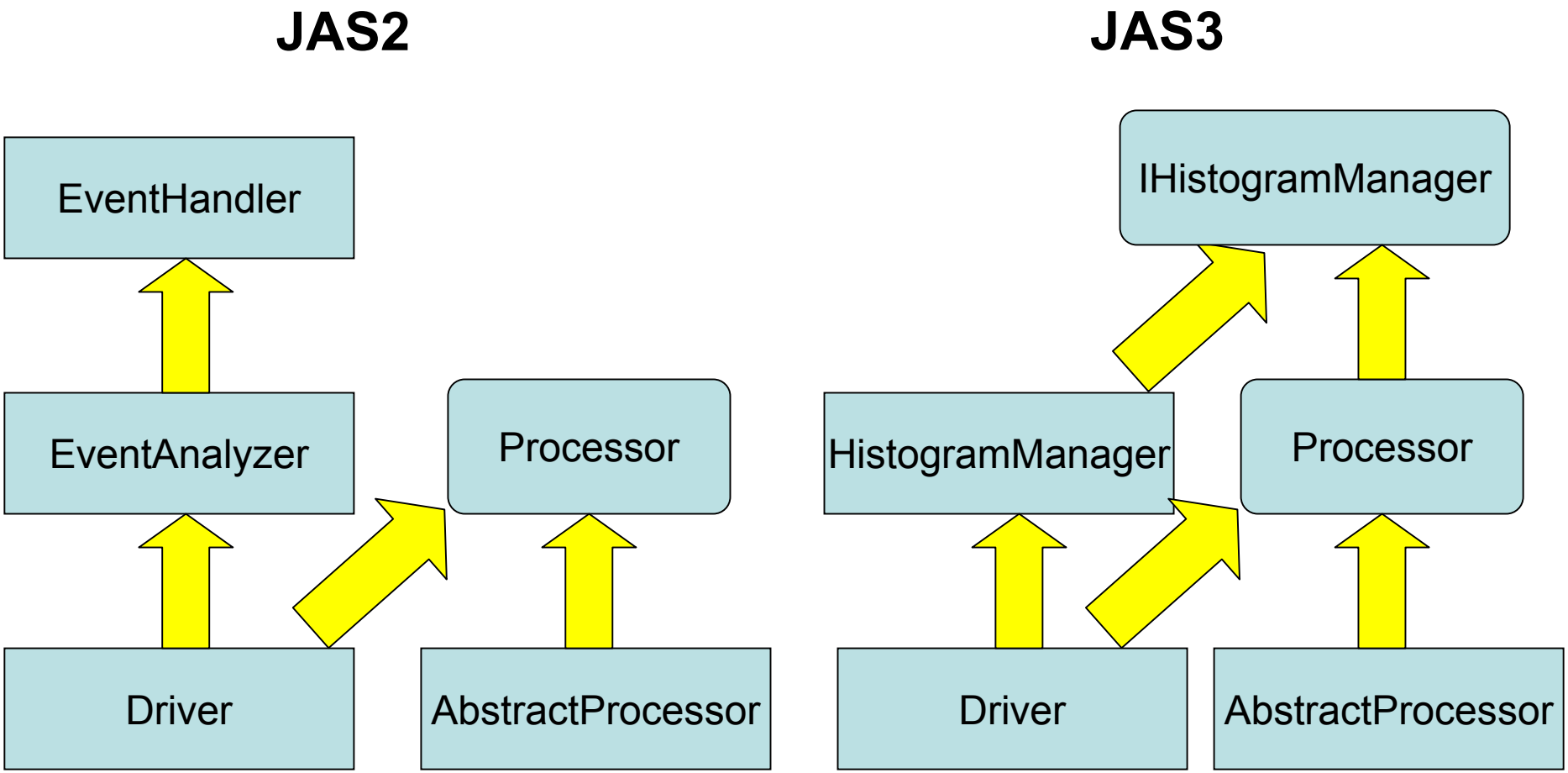
Analyzed 1 records in 271ms

3.54/4.88MB

Migrating LCD Code to JAS3

- LCD code makes extensive use of
  - hep.physics -- Basic 3,4 vectors, particle properties
  - hep.physics.jets -- Jet finders *etc*.
    - Neither of these packages are included in JAS3 by default
      - Will probably have a “HEP” plugin in future but would like to take opportunity to upgrade to latest Java collections style
      - For now migrated these classes almost unchanged into the LCD repository – can migrate to newer HEP classes later
  - hep.analysis -- Combination of Event Loop and Histograming
    - Replaced by AIDA and JAS3 “Record Loop” respectively
      - Fortunately most LCD code does not use these classes directly, most LCD code extends Driver or Processor and LCDEvent

Driver



- Changes are mostly transparent to LCD Code
- One big exception is the use of the histogram method, and Histogram classes from hep.analysis package

IHistogramManager

- IHistogramManager interface
  - Hides details of AIDA analysis/histogram factory
  - Maintains ability in LCD code to add a histogram with a single line of code:
    - cloud1D(“Etot”).fill(etot);
    - cloud2D(“x vs y”).fill(x,y);
    - histogram1D(“Etot”,50,0.,100.).fill(etot);
    - etc.
  - Provides convenient access to AIDA IAnalysisFactory and ITree

IHistogramManager details

```
public interface IHistogramManager
{
    ITree tree();
    IAnalysisFactory analysisFactory();
    IHistogramFactory histogramFactory();

    ICloud1D cloud1D(String path);
    ICloud2D cloud2D(String path);
    ICloud3D cloud3D(String path);
    ICloud1D cloud1D(String path, int nMax);
    ICloud2D cloud2D(String path, int nMax);
    ICloud3D cloud3D(String path, int nMax);

    IHistogram1D histogram1D(String path);
    IHistogram2D histogram2D(String path);
    IHistogram3D histogram3D(String path);
    IHistogram1D histogram1D(String path, int bins, double lowerEdge, double upperEdge);
    IHistogram2D histogram2D(String path, int nBinsX, double lowerEdgeX, double upperEdgeX,
                                          int nBinsY, double lowerEdgeY, double upperEdgeY);
    IHistogram3D histogram3D(String path, int nBinsX, double lowerEdgeX, double upperEdgeX,
                                          int nBinsY, double lowerEdgeY, double upperEdgeY,
                                          int nBinsZ, double lowerEdgeZ, double upperEdgeZ);

    IProfile1D profile1D(String path);
    IProfile2D profile2D(String path);
    IProfile1D profile1D(String path, int nBins, double lowerEdge, double upperEdge);
    IProfile2D profile2D(String path, int nBinsX, double lowerEdgeX, double upperEdgeX,
                                      int nBinsY, double lowerEdgeY, double upperEdgeY);
}
```

http://jas.freehep.org/jas3/Tutorial/Day2/api/hep/lcd/util/aida/IHistogramManager.html

Migration of User Code

- Bottom line, most LCD code can remain the same, except for histograming, which has to be migrated from JAS2 style -> AIDA style.
  - Using the fact that Driver and Processor both implement IHistogramManager makes this quite straightforward.
  - Access to IAnalysisManager makes available other AIDA features, such as tuples, fitting etc

Changes to User Code

```java
import hep.analysis.*;
import hep.physics.*;
import hep.lcd.util.driver.*;
import hep.lcd.event.*;
import java.util.*;

public class MyAnalysis extends Driver
{
   // Called by the framework to process each event
   public void process(LCDEvent event)
   {
      // Loop over the MC particles
      ParticleVector list = event.getMCParticles();
      Enumeration e = list.particles();

      double etot = 0;
      while (e.hasMoreElements())
      {
         MCParticle mc = (MCParticle) e.nextElement();

         // histogram particle energy
         double energy = mc.getEnergy();
         histogram("Particle Energy").fill(energy);

         // reject non final state particles
         histogram("Particle Status").fill(mc.getStatusCode());
         if (mc.getStatusCode() != mc.FINALSTATE) continue;

         // reject neutral particles
         ParticleType type = mc.getType();
         int charge = (int) type.getCharge();
         histogram("Particle Charge").fill(charge);
         if (charge == 0) continue;

         // Some more histograms
         String name = type.getName();
         histogram("Particle Type").fill(name);
         // Create a folder for each particle type
         HistogramFolder.setDefaultFolder("/"+name);
         histogram("Energy").fill(energy);
         HistogramFolder.setDefaultFolder("/");

         etot += energy;
      }
      histogram("Etot").fill(etot);
   }
}
```

```java
import hep.physics.*;
import hep.lcd.util.driver.*;
import hep.lcd.event.*;
import java.util.*;
import hep.aida.*;

public class MyAnalysis extends Driver
{
   // Called by the framework to process each event
   public void process(LCDEvent event)
   {
      System.out.println(event);
      // Loop over the MC particles
      ParticleVector list = event.getMCParticles();
      cloud1D("nMC").fill(list.size());
      Enumeration e = list.particles();

      double etot = 0;
      while (e.hasMoreElements())
      {
         MCParticle mc = (MCParticle) e.nextElement();

         // histogram particle energy
         double energy = mc.getEnergy();
         cloud1D("Particle Energy").fill(energy);

         // reject non final state particles
         cloud1D("Particle Status").fill(mc.getStatusCode()).
         if (mc.getStatusCode() != mc.FINALSTATE) continue;

         // reject neutral particles
         ParticleType type = mc.getType();
         int charge = (int) type.getCharge();
         cloud1D("Particle Charge").fill(charge);
         if (charge == 0) continue;

         // Some more histograms
         String name = type.getName();
         cloud1D(name + "-Energy").fill(energy);

         etot += energy;
      }
      cloud1D("etot").fill(etot);
   }
}
```

Changes to User Code

- We have gone through about 1/3 of the Snowmass tutorial code and changed it to use AIDA/JAS3
  - FastMC, Jet Finder *etc* tested
  - Full reconstruction not yet tested, but should be no major problems
  - Some rough edges still remain to be fixed
    - Histograms are not cleared when data is “rewound”
    - Extra “empty” AIDA trees are created

LCIO

JAS3

File Edit View Tuple Run LCD Window Help

z0.lcio

DataSets
- z0.lcio

Programs
- LCIOAnalysis

tree-1
- R

Welcome | LCIOAnalysis.java

```
import org.freehep.record.loop.event.*;
import hep.aida.*;
import hep.lcio.event.*;

public class LCIOAnalysis extends RecordAdapter
{
   public void recordSupplied(RecordSuppliedEvent e)
   {
      LCEvent event = (LCEvent) e.getRecord();
      System.out.println("run "+event.getRunNumber()+" "+event.getEventNumber()+
                         " Detector="+event.getDetectorName());
   }
}
```

```
run 0 9
run 0 0 Detector=SDMAR01
run 0 1 Detector=SDMAR01
run 0 2 Detector=SDMAR01
run 0 3 Detector=SDMAR01
run 0 4 Detector=SDMAR01
run 0 5 Detector=SDMAR01
run 0 6 Detector=SDMAR01
run 0 7 Detector=SDMAR01
run 0 8 Detector=SDMAR01
run 0 9 Detector=SDMAR01
```

Compiler | Record Loop

Analyzed 10 records in 451ms

13.3/14.4MB

LCIO Status

- Files can be read and analyzed in JAS3
  - LCIO defines its own hep.lcio.event classes
  - Currently no interface to LCD code
    - Two possible paths:
      - Write a layer which converts the LCIO data to our current hep.lcd.event representation
        - » Quite easy, all existing tools and software would then work with LCIO.
        - » We would then be “isolated” from LCIO format
      - Change all our code to use hep.lcio.event
        - » Would give better path for future support of LCIO
        - » More work, do we have anyone to do it?
        - » hep.lcio.event classes would need some work or extension to be more “user friendly”
        - » Could convert existing SIO files to LCIO format for comparison purposes.

Conclusions

- First release of jas3lcd.jar is available
  - Allows for use of lcd code with JAS3 without breaking existing compatibility with JAS2
  - Still some bugs, rough edges, that need to be worked out
- Significant fraction of Snowmass Tutorial has been updated for JAS3.
  - Forms basis of this afternoon's tutorial