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

<table>
<thead>
<tr>
<th>N</th>
<th>Type</th>
<th>Status</th>
<th>Parent</th>
<th>FX</th>
<th>FY</th>
<th>FZ</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>e-</td>
<td>Documented</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>250.00</td>
<td>250.00</td>
</tr>
<tr>
<td>2</td>
<td>e-</td>
<td>Documented</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>226.70</td>
<td>226.70</td>
</tr>
<tr>
<td>3</td>
<td>Z0</td>
<td>Documented</td>
<td>2</td>
<td>155.27</td>
<td>29.223</td>
<td>2.4085</td>
<td>163.31</td>
</tr>
<tr>
<td>4</td>
<td>nu_e</td>
<td>Final State</td>
<td>3</td>
<td>155.27</td>
<td>42.280</td>
<td>12.354</td>
<td>165.85</td>
</tr>
<tr>
<td>5</td>
<td>nu_e</td>
<td>Final State</td>
<td>4</td>
<td>155.27</td>
<td>42.280</td>
<td>12.354</td>
<td>165.85</td>
</tr>
<tr>
<td>6</td>
<td>nu_e_bar</td>
<td>Documented</td>
<td>5</td>
<td>-4.5439</td>
<td>-13.137</td>
<td>-10.045</td>
<td>17.400</td>
</tr>
<tr>
<td>7</td>
<td>nu_e_bar</td>
<td>Final State</td>
<td>6</td>
<td>-4.5430</td>
<td>-13.137</td>
<td>-10.045</td>
<td>17.463</td>
</tr>
<tr>
<td>8</td>
<td>h0H01</td>
<td>Documented</td>
<td>7</td>
<td>-165.27</td>
<td>-20.223</td>
<td>69.320</td>
<td>207.37</td>
</tr>
<tr>
<td>9</td>
<td>gluon</td>
<td>Documented</td>
<td>8</td>
<td>-26.318</td>
<td>-36.387</td>
<td>-26.126</td>
<td>51.054</td>
</tr>
<tr>
<td>11</td>
<td>D0_charm</td>
<td>Intermediate</td>
<td>10</td>
<td>-165.27</td>
<td>-29.223</td>
<td>60.320</td>
<td>207.37</td>
</tr>
<tr>
<td>12</td>
<td>rhn</td>
<td>Intermediate</td>
<td>11</td>
<td>-5.5717</td>
<td>-3.9421</td>
<td>-0.9711</td>
<td>3.8500</td>
</tr>
<tr>
<td>13</td>
<td>p</td>
<td>Final State</td>
<td>12</td>
<td>-1.0743</td>
<td>0.15111</td>
<td>0.49998</td>
<td>1.1752</td>
</tr>
<tr>
<td>15</td>
<td>gamma</td>
<td>Final State</td>
<td>14</td>
<td>-3.5031</td>
<td>-2.7531</td>
<td>-2.7510</td>
<td>5.2564</td>
</tr>
<tr>
<td>16</td>
<td>e+</td>
<td>Documented</td>
<td>15</td>
<td>-2.6626</td>
<td>-2.2489</td>
<td>-2.2542</td>
<td>4.2793</td>
</tr>
<tr>
<td>17</td>
<td>e+</td>
<td>Documented</td>
<td>16</td>
<td>-0.84030</td>
<td>-0.50321</td>
<td>-0.50283</td>
<td>0.85710</td>
</tr>
<tr>
<td>18</td>
<td>gamma</td>
<td>Final State</td>
<td>17</td>
<td>-0.90006</td>
<td>-0.00000</td>
<td>-0.73125</td>
<td>1.4094</td>
</tr>
<tr>
<td>19</td>
<td>e+</td>
<td>Documented</td>
<td>18</td>
<td>-0.70760</td>
<td>-0.43207</td>
<td>-0.52045</td>
<td>1.0007</td>
</tr>
<tr>
<td>20</td>
<td>e</td>
<td>Documented</td>
<td>19</td>
<td>0.28104</td>
<td>0.19505</td>
<td>0.20786</td>
<td>0.10072</td>
</tr>
<tr>
<td>21</td>
<td>Delta_c</td>
<td>Intermediate</td>
<td>20</td>
<td>1.6857</td>
<td>4.4584</td>
<td>3.8128</td>
<td>7.5933</td>
</tr>
<tr>
<td>22</td>
<td>n</td>
<td>Final State</td>
<td>21</td>
<td>3.6985</td>
<td>3.2805</td>
<td>2.8354</td>
<td>5.7811</td>
</tr>
<tr>
<td>23</td>
<td>h0H01</td>
<td>Documented</td>
<td>22</td>
<td>1.01145</td>
<td>0.33345</td>
<td>-0.41765</td>
<td>2.9868</td>
</tr>
<tr>
<td>24</td>
<td>p</td>
<td>Documented</td>
<td>23</td>
<td>-1.7998</td>
<td>-2.7772</td>
<td>-1.7572</td>
<td>3.8627</td>
</tr>
<tr>
<td>25</td>
<td>n</td>
<td>Documented</td>
<td>24</td>
<td>-0.35725</td>
<td>-0.06431</td>
<td>-0.84138</td>
<td>11.3713</td>
</tr>
</tbody>
</table>

Analyzed 1 records in 271 ms
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
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
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);
}
Migration of User Code

• Bottom line, most LCD code can remain the same, except for histogramming, 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.physic.*;
import hep.lcd.util.driver.*;
import hep.lcd.event.*;
import java.util.*;

public class HxAnalysis extends Driver {
    public void process(LCDEvent event) {
        // Called by the framework to process each 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
            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);
        }
    }
}
```
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
import org.freehep.record.loop.event.*;
import hep.event.*;
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()=
        "+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 151 ms
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