Data Definitions in EVIO and LCIO

Ebrahim

University of New Hampshire
05/27/2011

What is EVIO?

- A basic data format for writing out raw data
- Extensively tried and tested in Halls A and C
- DOM Tree structure, utilities exist to convert to and from XML

```
evioFileChannel chan("someEvents.dat","w");
chan.open();
for(int i=0; i<N_OF_EVENTS; i++) {
  evioDOMTree event(tag=1, num=0);

  event.addBank(tag=2, num=9,  /*some vector of numbers*/);
  event.addBank(tag=3, num=10, /*some vector of numbers*/);

  chan.write(event);
}
chan.close();
```

What is LCIO?

- A persistency framework that provides a data model for linear collider simulations
- Provides a Java and C++ API
- Provides a common output scheme for international collider studies

```cpp
LCWriter* lcWrt = LCFactory::getInstance()->createLCWriter();

lcWrt->setCompressionLevel(0);
lcWrt->open("somedata.slcio", LCIO::WRITE_NEW);

LCRunHeaderImpl* runHdr = new LCRunHeaderImpl;
runHdr->setRunNumber(0);
runHdr->setDetectorName("HPS Test Run version 394239472");
runHdr->addActiveSubdetector("ECAL");
runHdr->parameters().setValue("beam offset", ".1 mm");

lcWrt->writeRunHeader( runHdr ) ;
```

```cpp
for(int i=0; i<N_OF_EVENTS; i++){
  LCEventImpl*  evt = new LCEventImpl() ;
  evt->setRunNumber(  0   ) ;
  evt->setEventNumber( i ) ;
  LCTime now ;
  evt->setTimeStamp( now.timeStamp()  ) ;
  evt->setDetectorName(ECAL) ;
  evt->parameters().setValue("description","an event");

  LCCollectionVec* mcParts = new LCCollectionVec(LCIO::MCPARTICLE);

  MCParticleImpl* mc_particle = new MCParticleImpl;
  mc_particle->setPDG(1);
  float p0[3] = { 0. , 0. , 1000. };
  mc_particle->setMomentum(p0);
                 :
                 :
                 :
```

EVIO

```
600
   10

        2.23693187839190812838
        2.23693187839190812838
        2.24499056921964097455
        2.24499056921964097455
   20
        1.83538649371907891528
        0.83538649371907891528
        1.02113004828059339424
        0.02113004828059339424
```
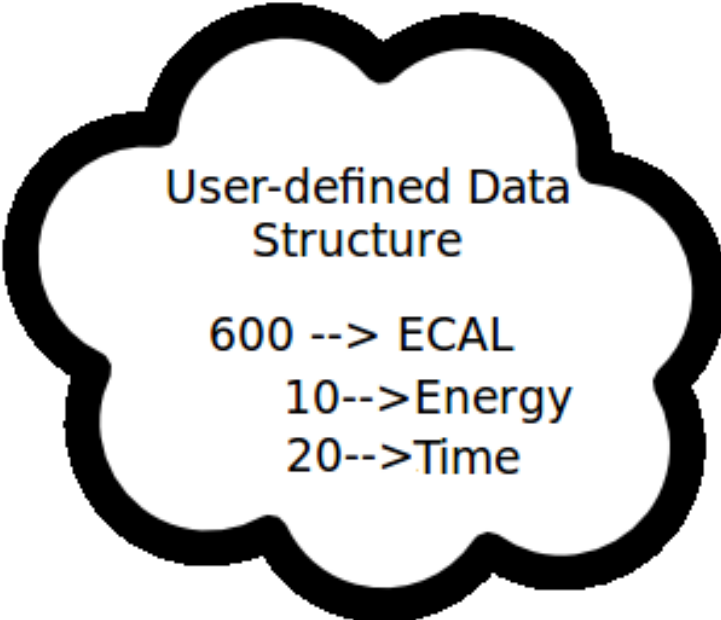
EVIO

600
  10

       2.23693187839190812838
       2.23693187839190812838
       2.24499056921964097455
       2.24499056921964097455

  20

       1.83538649371907891528
       0.83538649371907891528
       1.02113004828059339424
       0.02113004828059339424

User-defined Data Structure

600 --> ECAL
    10-->Energy
    20-->Time

EVIO

600
    10

        2.23693187839190812838
        2.23693187839190812838
        2.24499056921964097455
        2.24499056921964097455
    20
        1.83538649371907891528
        0.83538649371907891528
        1.02113004828059339424
        0.02113004828059339424

LCIO

CalorimeterHit 1

Energy: 2.23693187839190812838
Time: 1.83538649371907891528

CalorimeterHit 2

Energy: 2.23693187839190812838
Time: 0.83538649371907891528

CalorimeterHit 3

Energy: 2.24693187839190812838
Time: 1.83538649371907891528

CalorimeterHit 4

Energy: 2.24693187839190812838
Time: 0.83538649371907891528

EVIO

```
<bank content="bank" data_type="0xe" tag="600" num="200">
   <float64 data_type="0x8" tag="600" num="1">
        1.58693314828655479554e-01    1.51341672774518487543e-01
        2.42506523496802961493e-01    2.60045387153031715144e-01
        1.88592975851489752515e+00    7.19466702526572327026e-02
        1.12154524505383044186e+01    6.32069736165919993098e-02
        1.15866713536064147583e-01    2.02290608881689948362e+00
        1.23556285340076843493e-01
   </float64>
   <float64 data_type="0x8" tag="600" num="2">
        3.54534830226232884343e+02    4.16605932745781331050e+02
       -1.29243959591643687190e+02   -2.48951191828599945666e+02
       -2.36539911279963916968e+02   -1.37297470431418730641e+02
       -2.51772544835338180746e+02   -2.38531700408182302908e+02
       -1.34462110983802062947e+02   -2.31949479311451597141e+02
       -1.68815269638438962829e+02
   </float64>
   <float64 data_type="0x8" tag="600" num="3">
       -3.85836697992915276245e+01   -8.78991751598646686716e+01
       -4.26905904375847455867e+01    4.70915970994837209673e+01
        5.34986207051898787768e+01   -1.99188422882825690863e+01
        1.80436175653222043991e+01   -1.72262162738336179757e+01
        1.75663435029841004109e+01    1.82615316813613297597e+01
       -5.33597832110704786146e+01
   </float64>
   <float64 data_type="0x8" tag="600" num="4">
        1.53062037648387968147e+03    1.52683306621748783982e+03
        1.37782178462894398763e+03    1.48490117371946416824e+03
        1.46730935113335226561e+03    1.37575048708591907598e+03
        1.52015537983158196766e+03    1.46649777598483660768e+03
        1.43873813399454547834e+03    1.48526629633479569748e+03
```
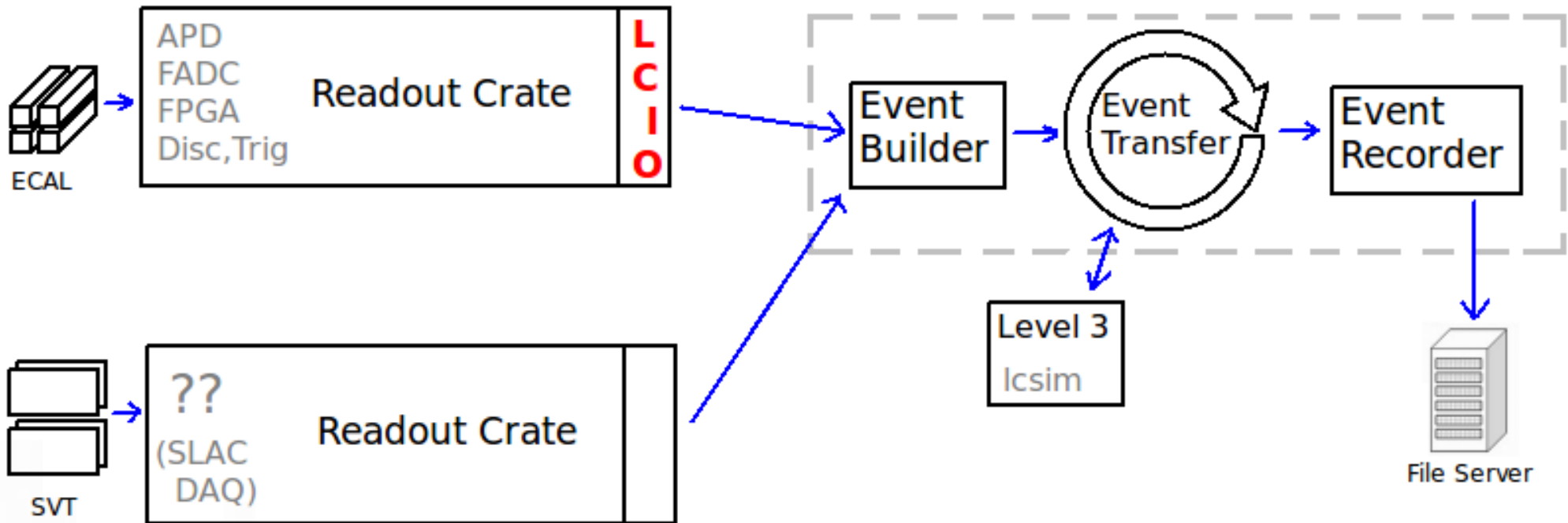
.
.
.

EVIO   LCIO

We need LCIO

- We will eventually have to incorporate it into our DAQ

- It is required for the level 3 trigger

  - The reconstruction depends on the information in an LCIO file

- (Also LCIO is an appropriate data format for the end user)
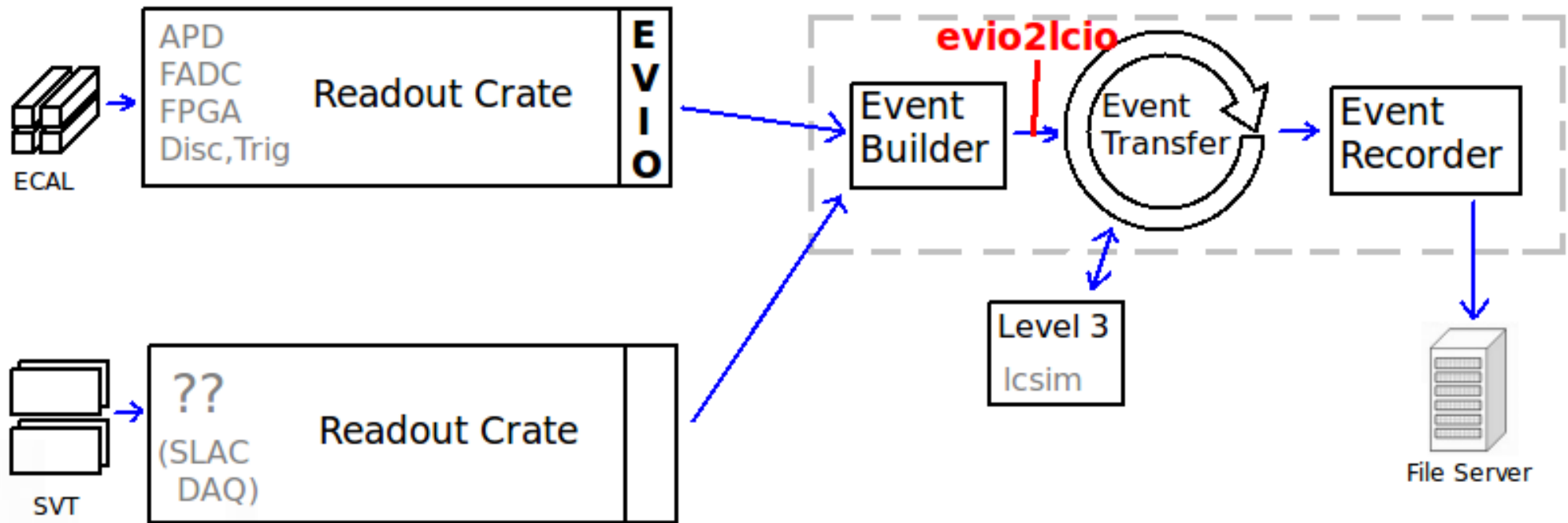
DAQ Overview
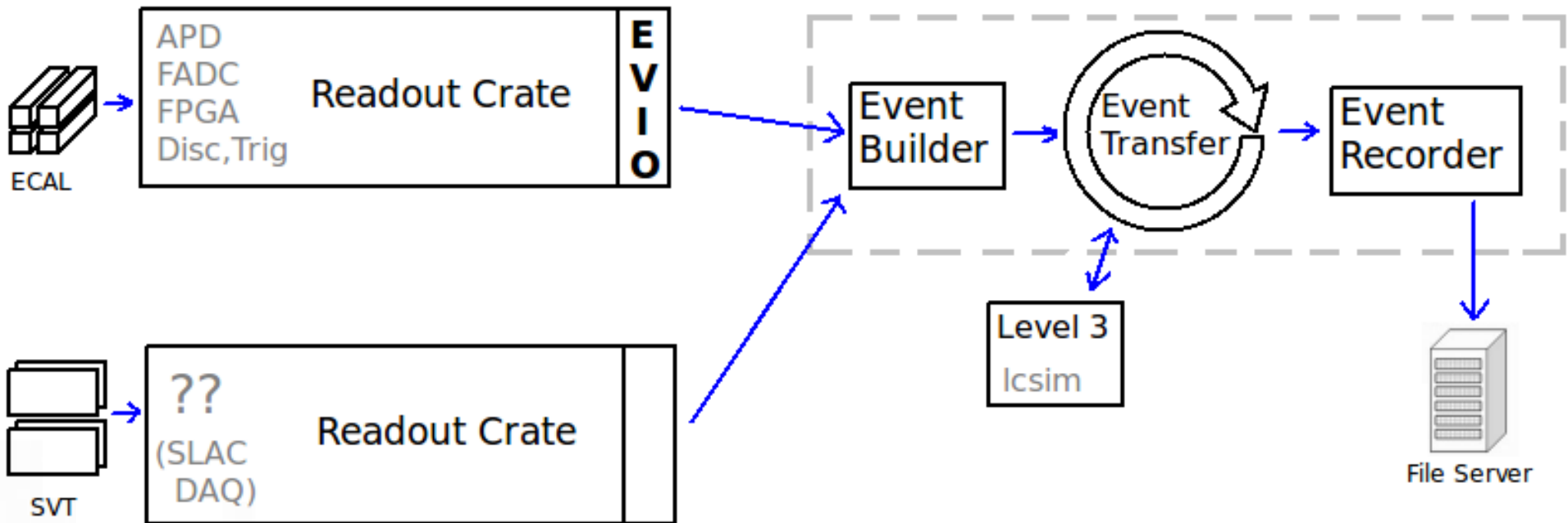
There are a few ways to introduce LCIO into this system...

Option 1

This would leave everything as LCIO with no trace of EVIO. That sounds really nice, but it's also very risky because there may not be sufficient time to test code by the time our event data model is pinned down.

Option 2

Although bugs are equally dangerous here, this conversion is much easier to debug. The downside is that JLab monitoring tools that already work with EVIO need to have a layer of LCIO input to be used.

Option 3
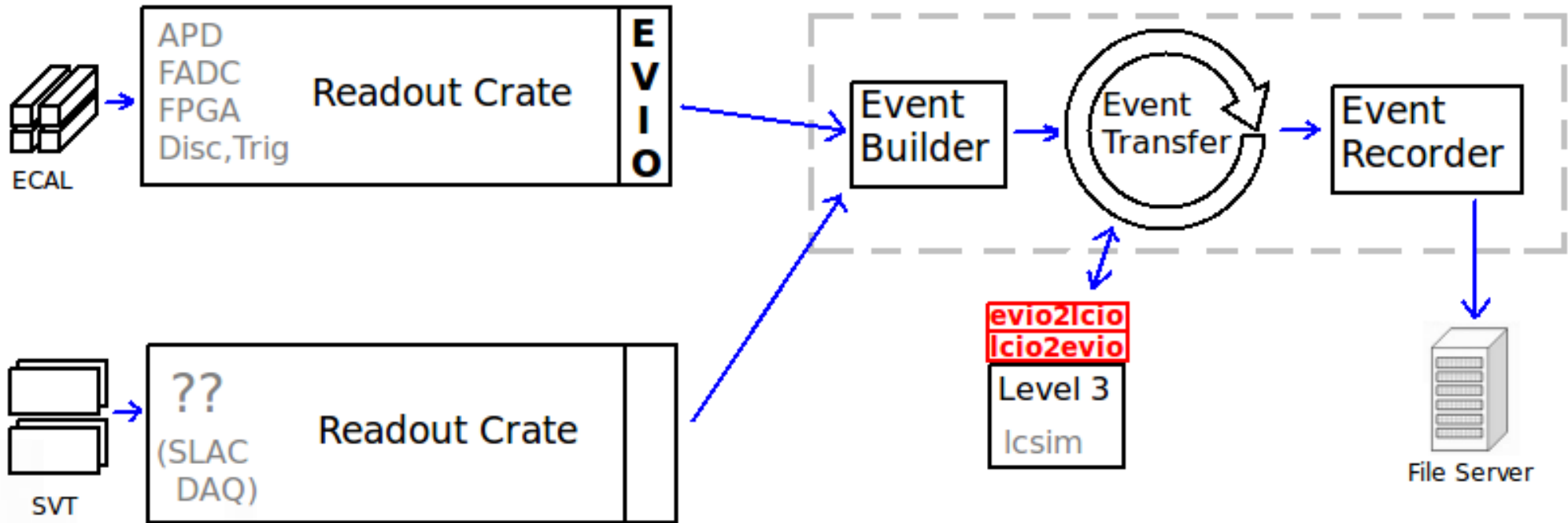
This is a minimally intrusive and probably the safest option.

Decisions

- Clearly some decisions need to be made about what layer of DAQ will implement LCIO.

- The decision depends on how quickly we can get critical DAQ parts to test (like FADC), and how quickly we can pin down a data model.

- The fall-back solution should probably be the safest, for the HPS Test Run at least.