

HIPO4



Gagik Gavalian (Jefferson Lab)

- ◆ **Better data structures to keep banks of composite data:**

- ◆ Banks are now kept as tables, with associated format (schema)
- ◆ Saves space compared to HIPO3 (~15% smaller)

- ◆ **Improved file indexing optimized for large files:**

- ◆ File index is kept at the end of the file and read once at open operation.
- ◆ File index contains position and tag of each record in the file.

- ◆ **Ability to separate events in baskets depending on event type:**

- ◆ Scaler events from decoding are kept separately in the file
- ◆ Easy to analyze the file for beam trips and helicity by reading a fraction of the file.

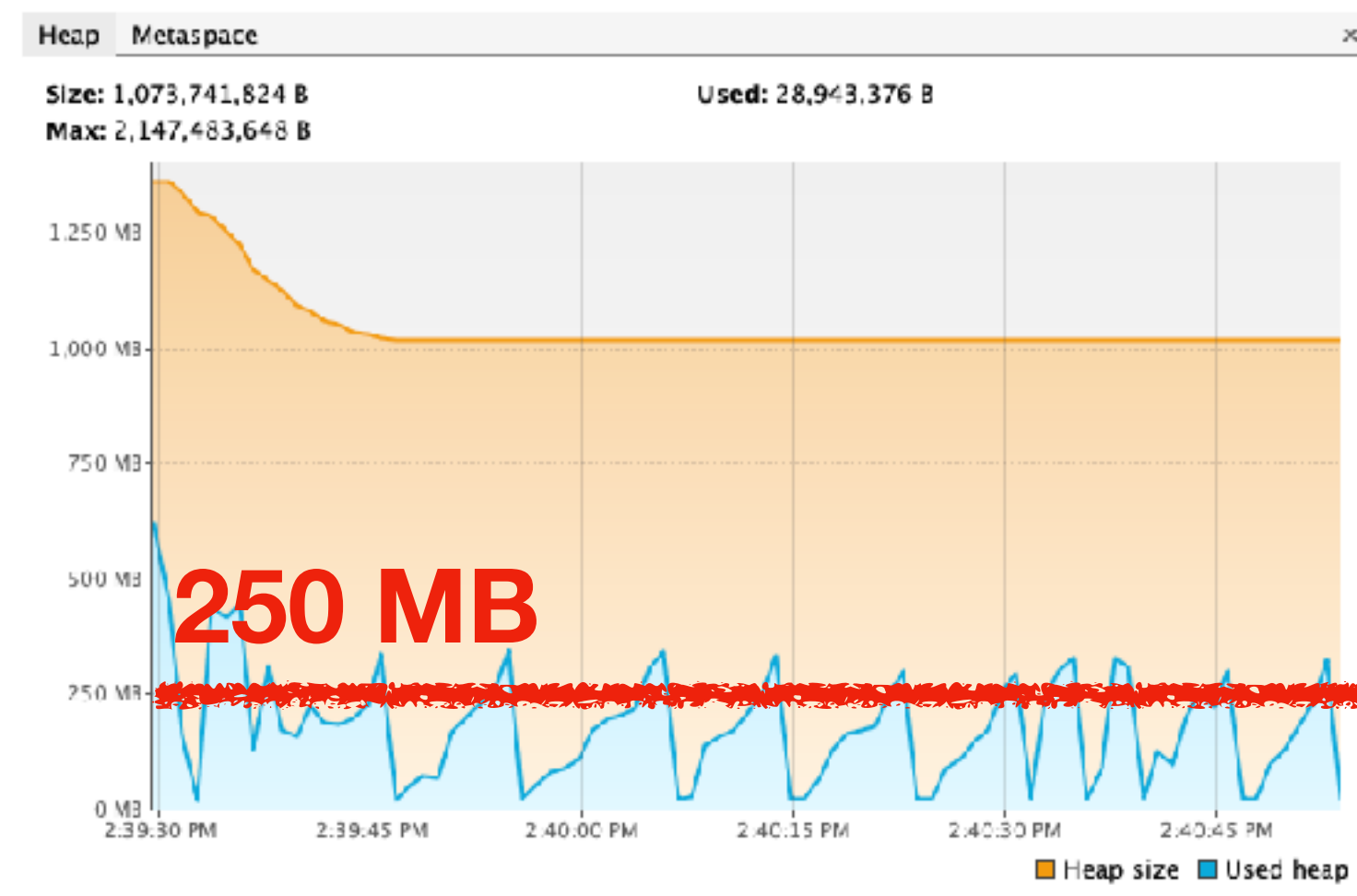
- ◆ **Improved memory footprint:**

- ◆ The memory footprint is significantly improved from previous version (~ 10x)
- ◆ Improved memory footprint also improves reading and parson speed.

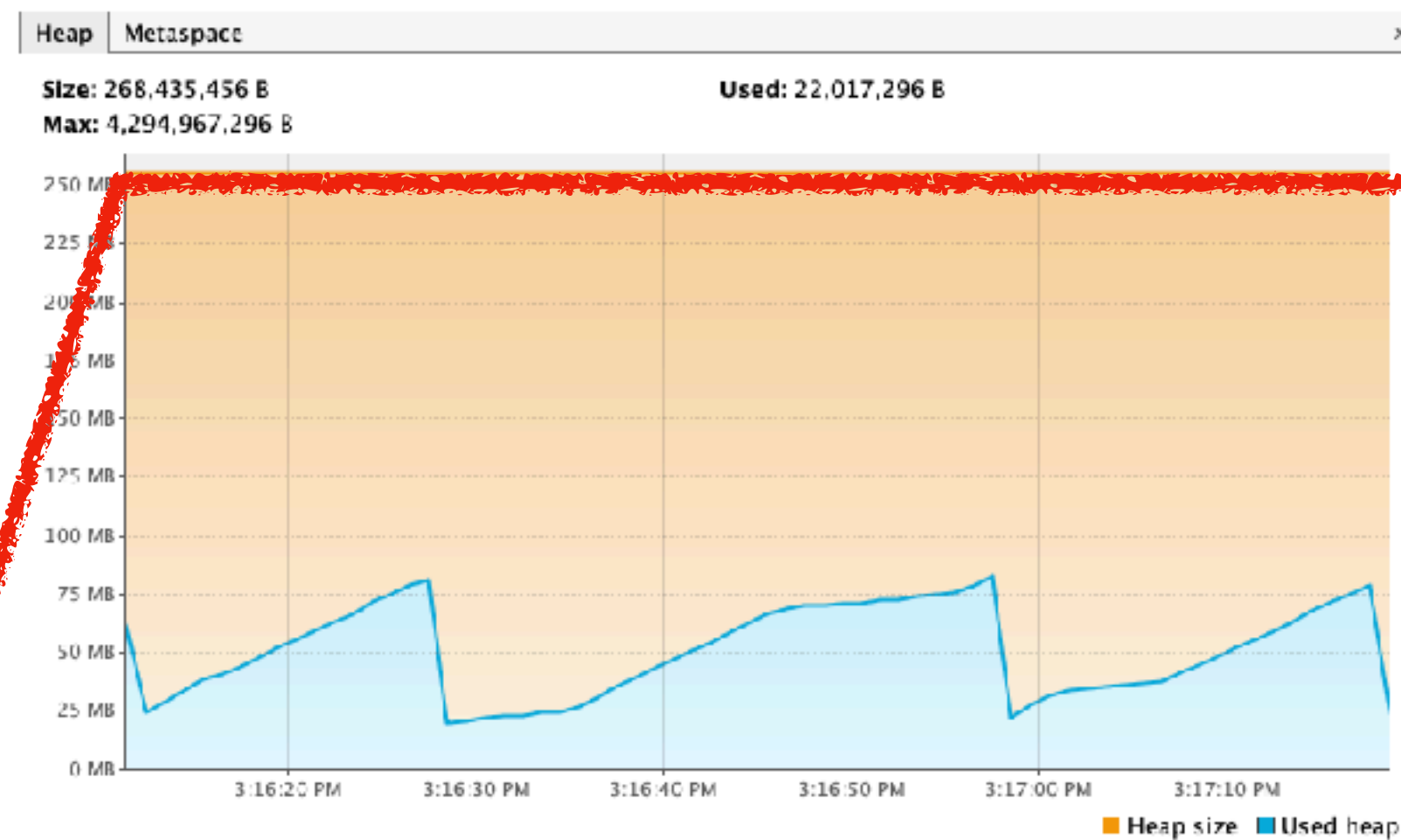
Memory Footprint

Reading production run ~**17GB**, reading **Particle** and **Calorimeter** Banks (~1 min, 34 sec)

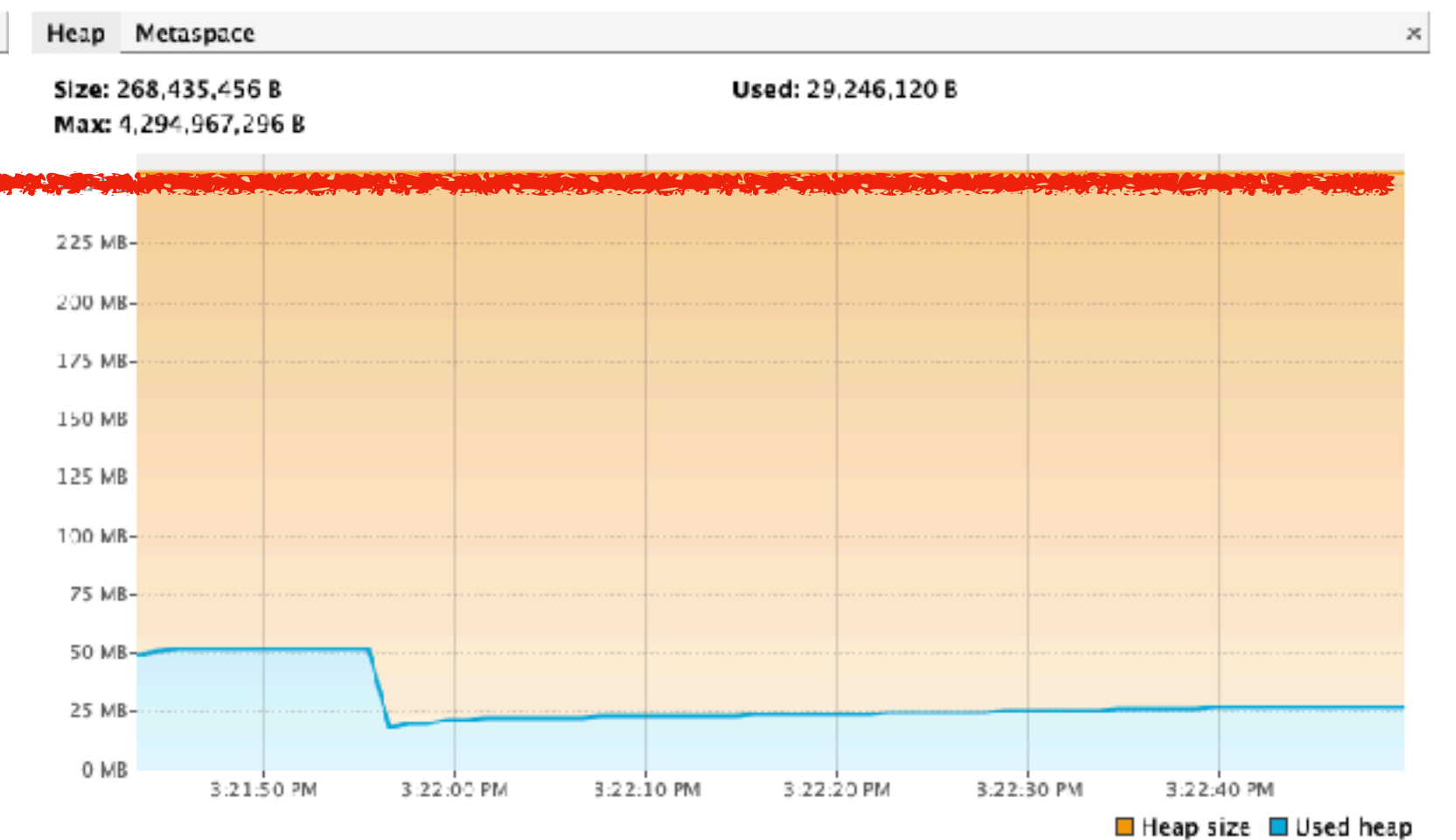
HIPO-3



HIPO-4 EARLY



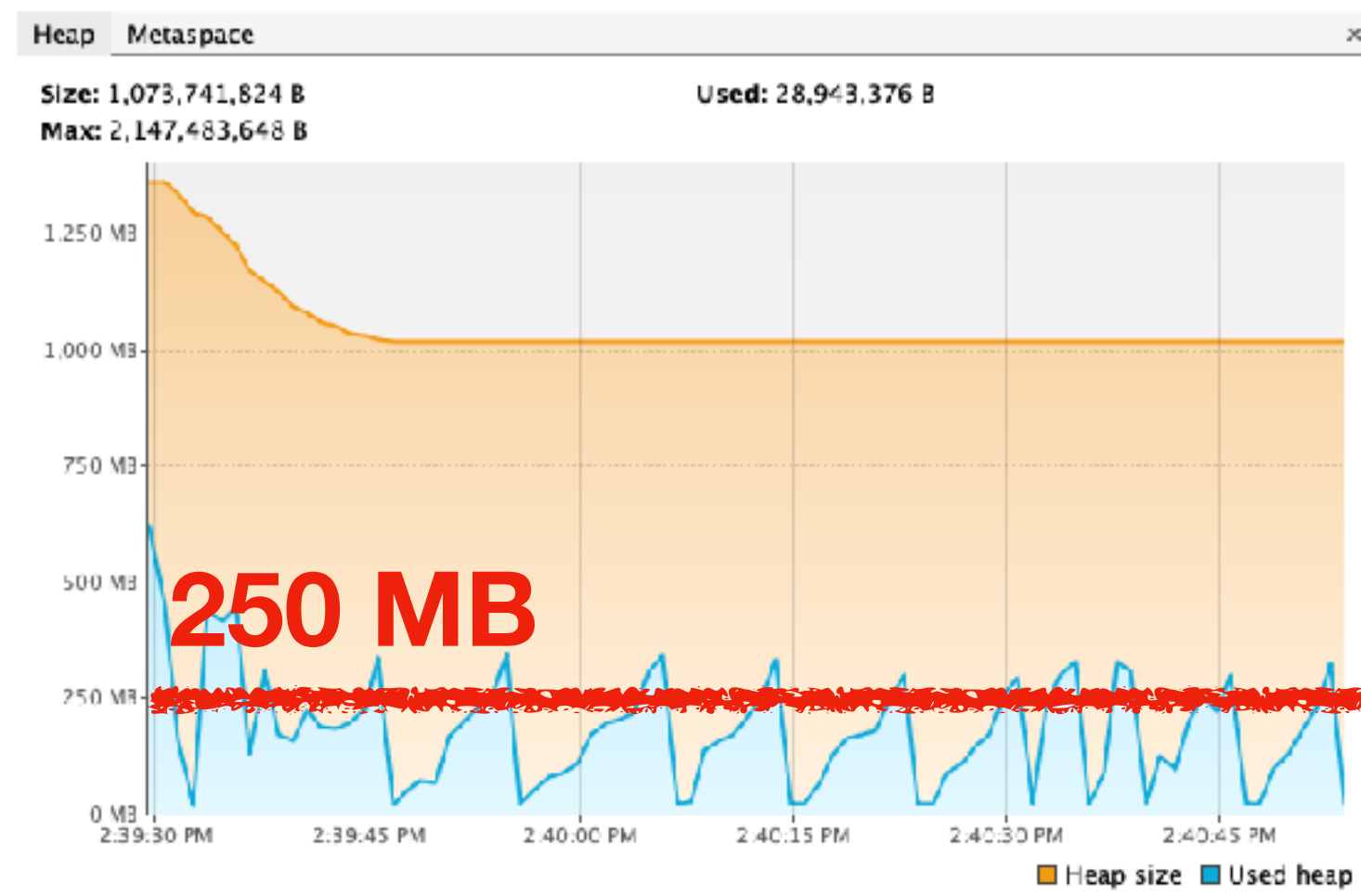
HIPO-4



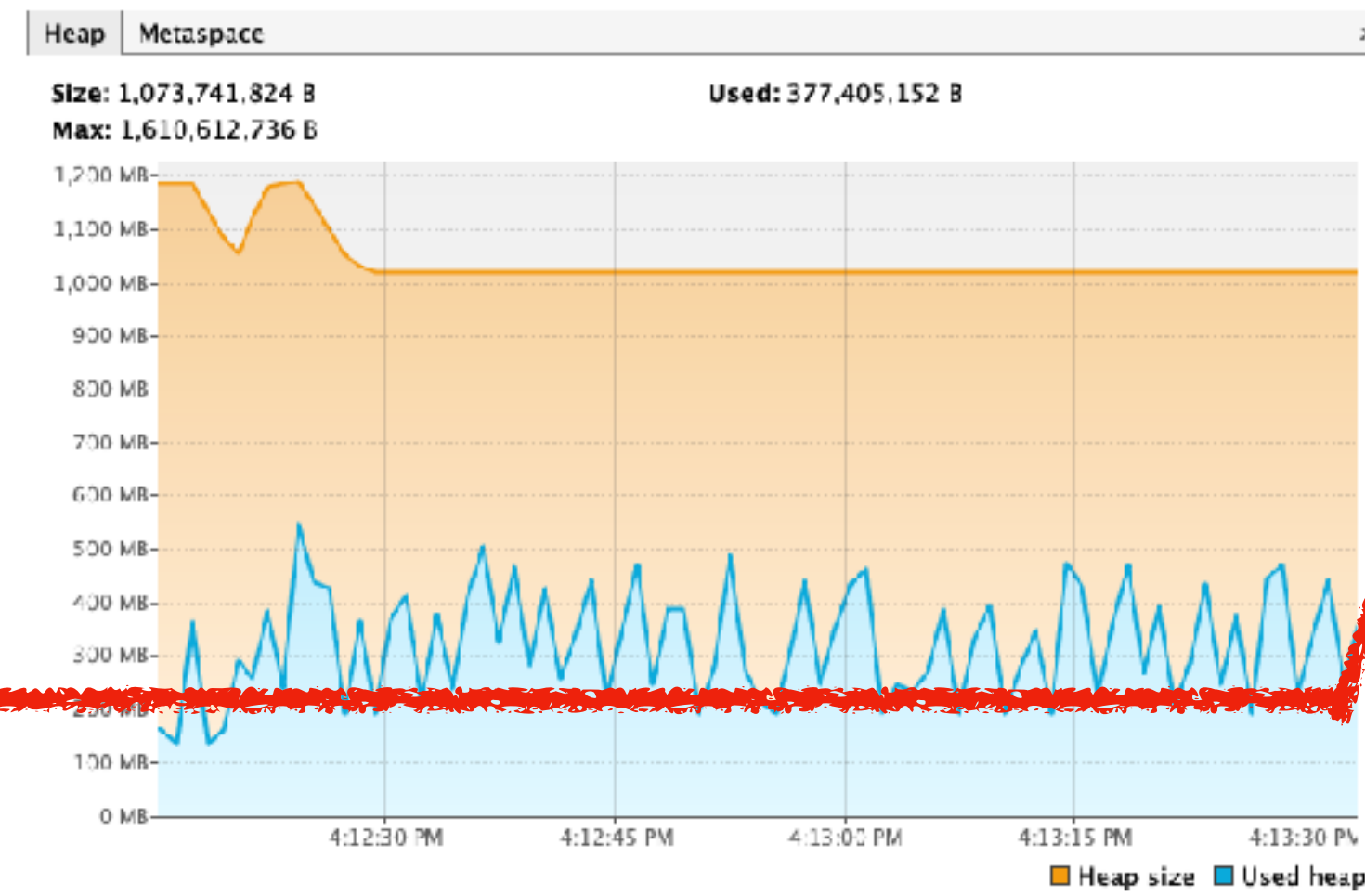
Memory Footprint

Reading production run ~**17GB**, reading **Particle** and **Calorimeter** Banks (~1 min, 34 sec)

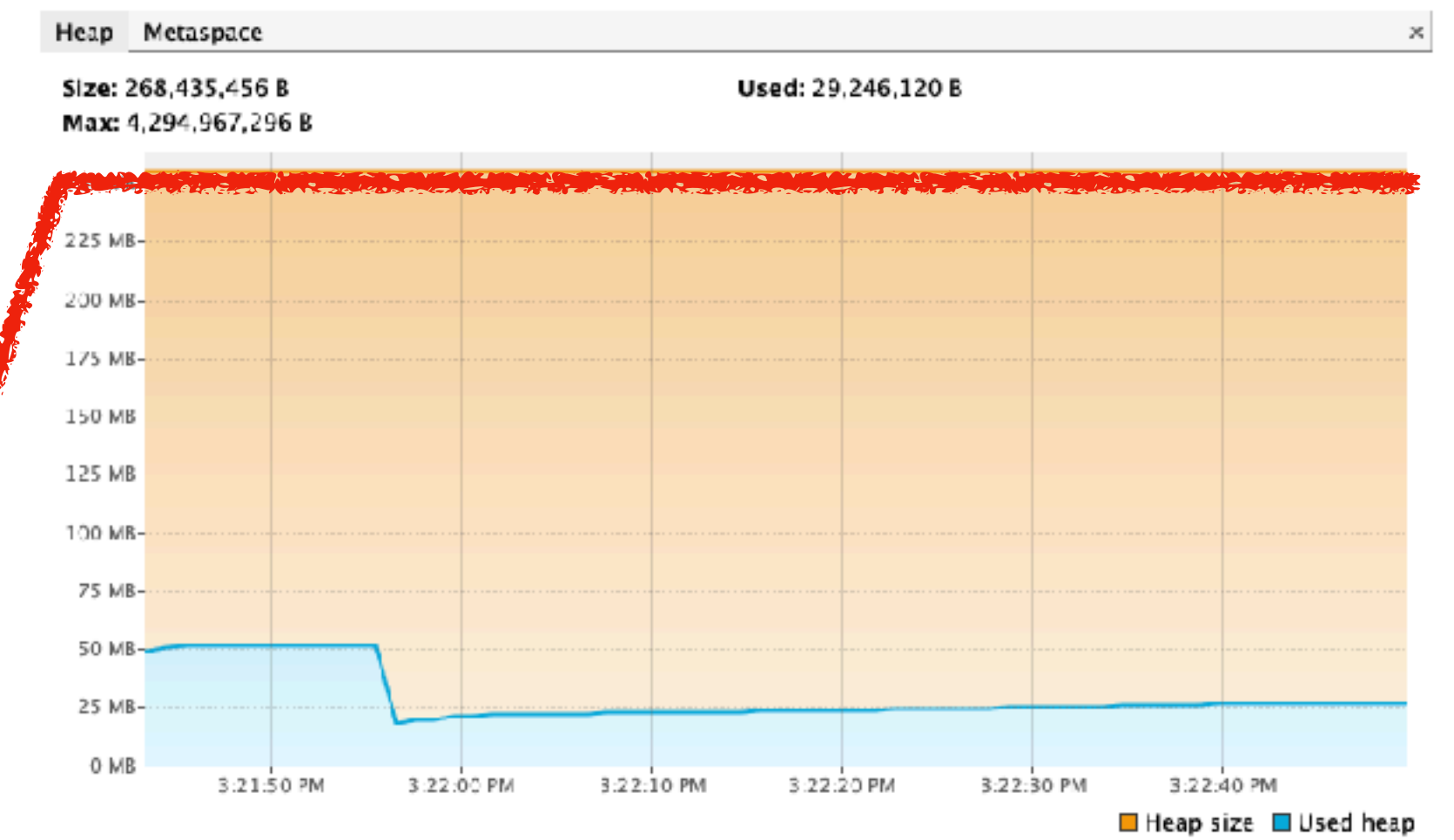
HIPO-3



EVIO Decoder



HIPO-4



Utilities

- ◆ **HIPO utilities** are powerful tool for interacting with HIPO files:
 - ◆ merging files
 - ◆ skimming files
 - ◆ browsing through the file

```
>$COATJAVA/bin/hipo-utils
```

```
Usage : hipo4utils [commands] [options]
```

```
commands:
```

```
  -info : show the information about the file
  -merge : Merge several files together
  -filter : filter banks from the input files
  -stats : prints statistics for the file
  -dump : dump the content of the file on the screen
  -test : reads the file to check for file integrity
```

```
Choose wisely.....
```

```
>$COATJAVA/bin/hipo-utils -filter
```

```
Usage : hipo4utils -b [ filter string separated by (,) ] -o [ output file name ] [input1]
```

Options :

-s : true - will write schemas only for banks that are kept. false - all schemas (default = true)
-t : tag of the events to filter. other tags are written as is. (default = 0)

◆ Bank formats:

- ◆ "REC::Particle,REC::Event,RUN::info" will only keep three banks in the output
- ◆ "RUN::info,REC::*" will keep RUN::info bank and all banks that start with "REC::"
- ◆ "RUN::info,*Track*" will keep RUN::info and all banks that have "Track" in the name.

```
>$COATJAVA/bin/hipo-utils -dump run_004013_full.hipo
```

◆ Displays the bank content in the file.

```
Choose (n=next,p=previous, q=quit), Type Bank Name or id : n
*** event *** ::: tag = 0 mask = 0000000000000000000000000000000000000000 size = 191 node count = 0
  0 |          REC::Event |      300 |      30 |      1 |      16 |      42
  1 |          REC::Particle |      300 |      31 |      3 |      66 |     117

Choose (n=next,p=previous, q=quit), Type Bank Name or id : REC::Particle
```

```
* NODE * group =      300, item =   31, type = 11, size =      273
  pid :      -211         0         22      2112      2112         22         22
  px :    -0.4934   -0.3267    0.4570    0.0000    0.1433    0.0514    0.0229
  py :     0.4105   -0.1315   -0.0631    0.0000   -0.7095   -0.0581    0.0452
  pz :     0.6980    0.3263    1.3986    0.0000    1.4397    0.1387    0.0989
  vx :    -1.0308   -0.0623   -1.0308   -1.0308   -1.0308   -1.0308   -1.0308
  vy :    -0.9155   -0.0251   -0.9155   -0.9155   -0.9155   -0.9155   -0.9155
  vz :    63.7825    2.9201    63.7825    63.7825    63.7825    63.7825    63.7825
charge :         -1         1         0         0         0         0         0
  beta :     0.9893  -99.0000    0.9913   -0.1082    0.8639    1.0525    0.9717
chi2pid :   13.8016   99.0000   99.0000   99.0000   99.0000   99.0000   99.0000
status :    -2221     4000     2020     2030     2230     2010     2010

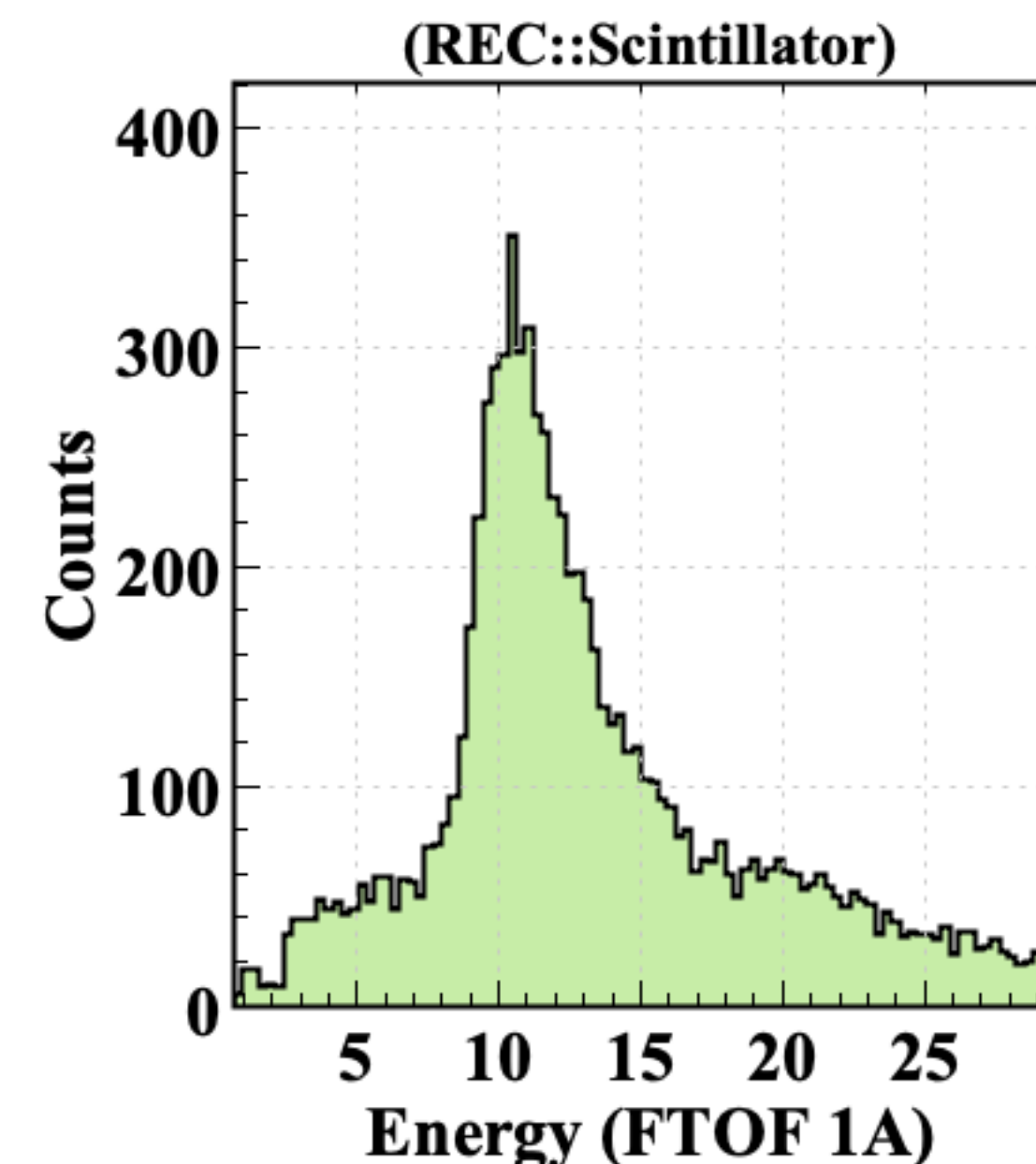
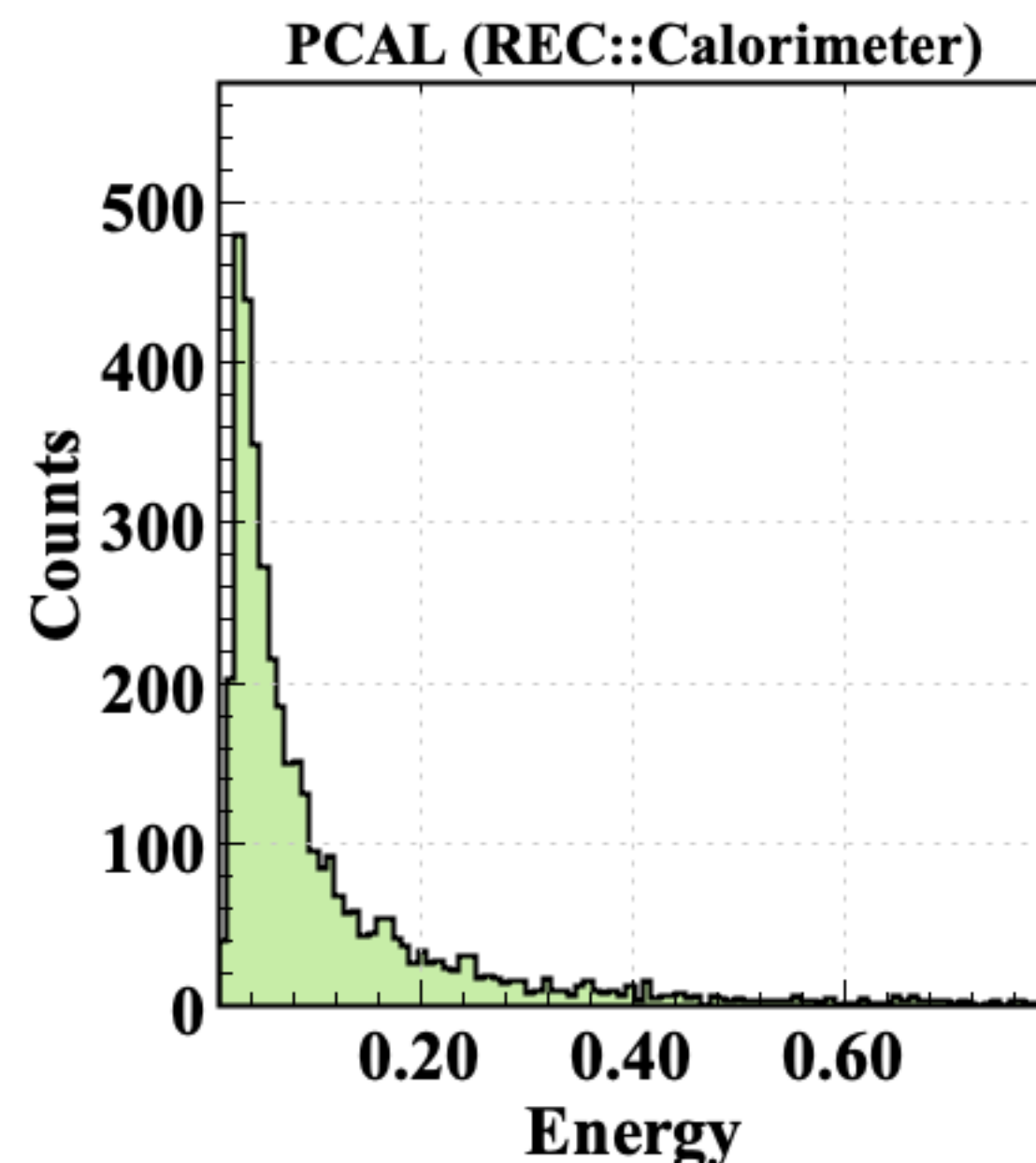
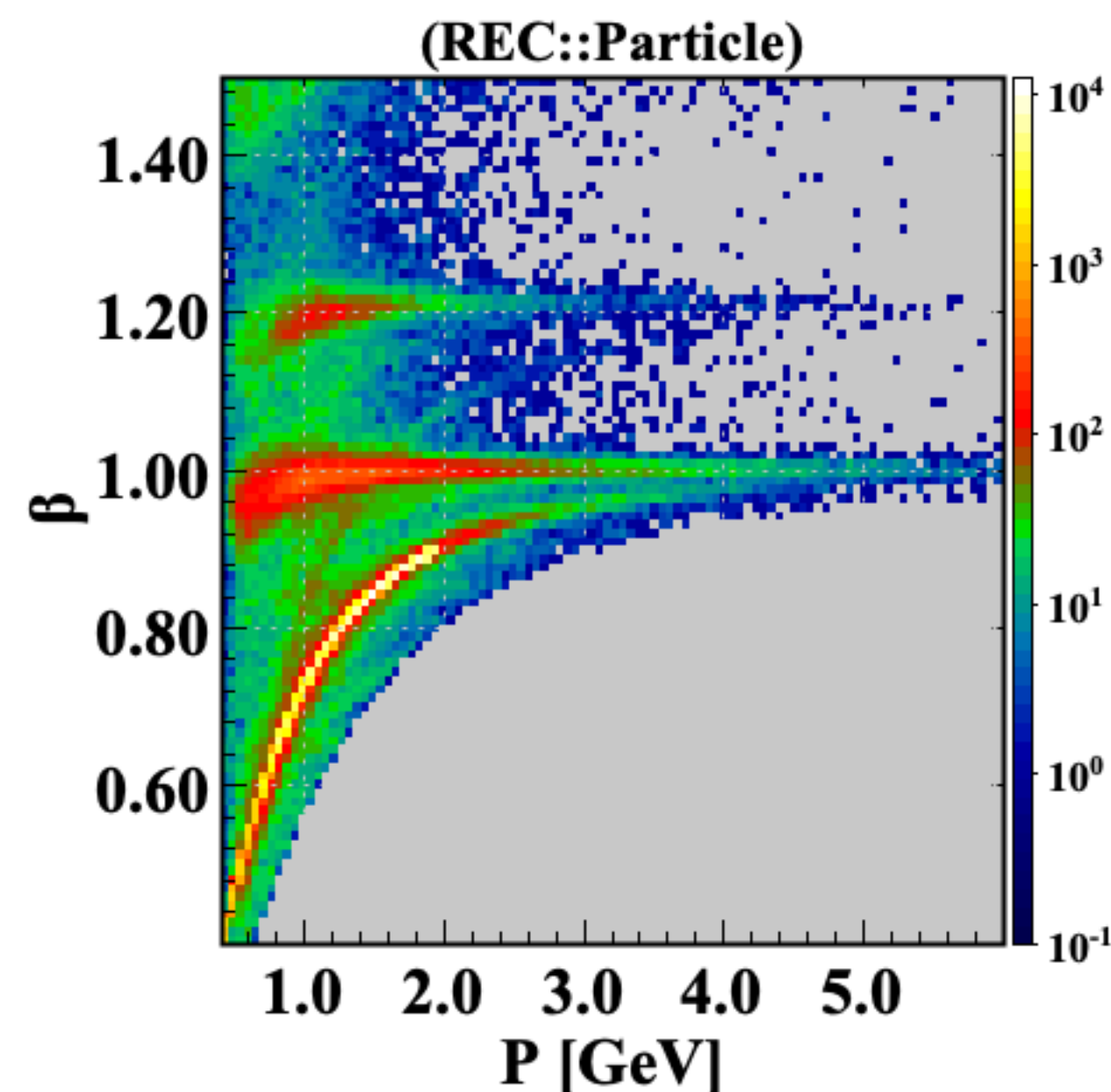
Choose (n=next,p=previous, q=quit), Type Bank Name or id :
```

```
./bin/hipoutils.sh -stats ~/Work/DataSpace/clas12/4013/out_clas_004013.0.hipo -sort 4
```

..... ■										
..... ■ ■										
FTOF::hits		48255		814016		16.87		63,879,288	b	
FTOF::hbhits		48255		814017		16.87		63,879,366	b	
REC::Traj		42844		1888510		44.08		64,552,092	b	
BSTRec::Crosses		48255		1430600		29.65		66,193,640	b	
FTTRK::adc		48068		2526324		52.56		68,595,292	b	
BMTRec::Clusters		48255		1868025		38.71		78,843,090	b	
BMTRec::Crosses		48255		1868025		38.71		86,315,190	b	
FMT::adc		48255		3601986		74.64		97,639,662	b	
BMTRec::Hits		48255		6120356		126.83		122,793,160	b	
BSTRec::Hits		48255		6453981		133.75		129,465,660	b	
TimeBasedTrkg::TBHits		35136		1801184		51.26		138,972,256	b	
HitBasedTrkg::HBClusters		48255		2528673		52.40		139,463,055	b	
BST::adc		48255		6460239		133.88		148,971,537	b	
HitBasedTrkg::HBSegments		48255		2181939		45.22		159,667,587	b	
BMT::adc		48255		6120356		126.83		165,635,652	b	
BSTRec::Clusters		48255		4871619		100.96		204,994,038	b	
DC::tdc		48255		41912176		868.56		377,595,624	b	
HitBasedTrkg::HBHits		48255		17160810		355.63		909,908,970	b	
+-----+										
event count = 48393 , events size = 4,023,336,855										

◆ Java Analysis Workstation (JAW)

- ◆ was developed to provide quick plotting functionality.
- ◆ reading vectors from text files, manipulating and plotting.
- ◆ reading text files into tuples, plotting with cuts and full expression parsing.
- ◆ **NOW ! includes also quick plotting from HIPO4 files.**
- ◆ Powerful interface for extending it for analysis and data checks.
- ◆ autocompletion for commands, and history.
- ◆ help and usage information for commands.



```

canvas/clear
canvas/zone 3 1
hipo/open 10 ../rec_004013_FULL.hipo
hipo/tree 10 REC::Particle
hipo/draw 10 beta%sqrt(px*px+py*py+pz*pz)
sqrt(px*px+py*py+pz*pz)<6.&&status>1500&&status<4000&&sqrt(px*px+py*py+pz*pz)>0.4&&beta>0.4&&beta<1.5&&pid>200
4000000
hipo/open 11 /Users/gavalian/Work/DataSpace/clas12/4013/out_clas_004013.evio.00000-00009.hipo
hipo/tree 11 REC::Calorimeter
hipo/info 11 REC::Calorimeter
hipo/draw 11 energy sector==1&&energy<0.8
hipo/tree 11 REC::Scintillator
hipo/info 11 REC::Scintillator
hipo/draw 11 energy layer==1&&energy<30

```

◆ CLAS12Tools C++ interface to HIPO4

- ◆ C++ HIPO4 api consistent with JAVA API
- ◆ ROOT wrappers for reading HIPO4 files from in ROOT
- ◆ Converter to write ROOT trees from HIPO4 files
- ◆ Plotting library (using RDataFrames) to directly plot from HIPO4 file

◆ Improved file indexing optimized for large files:

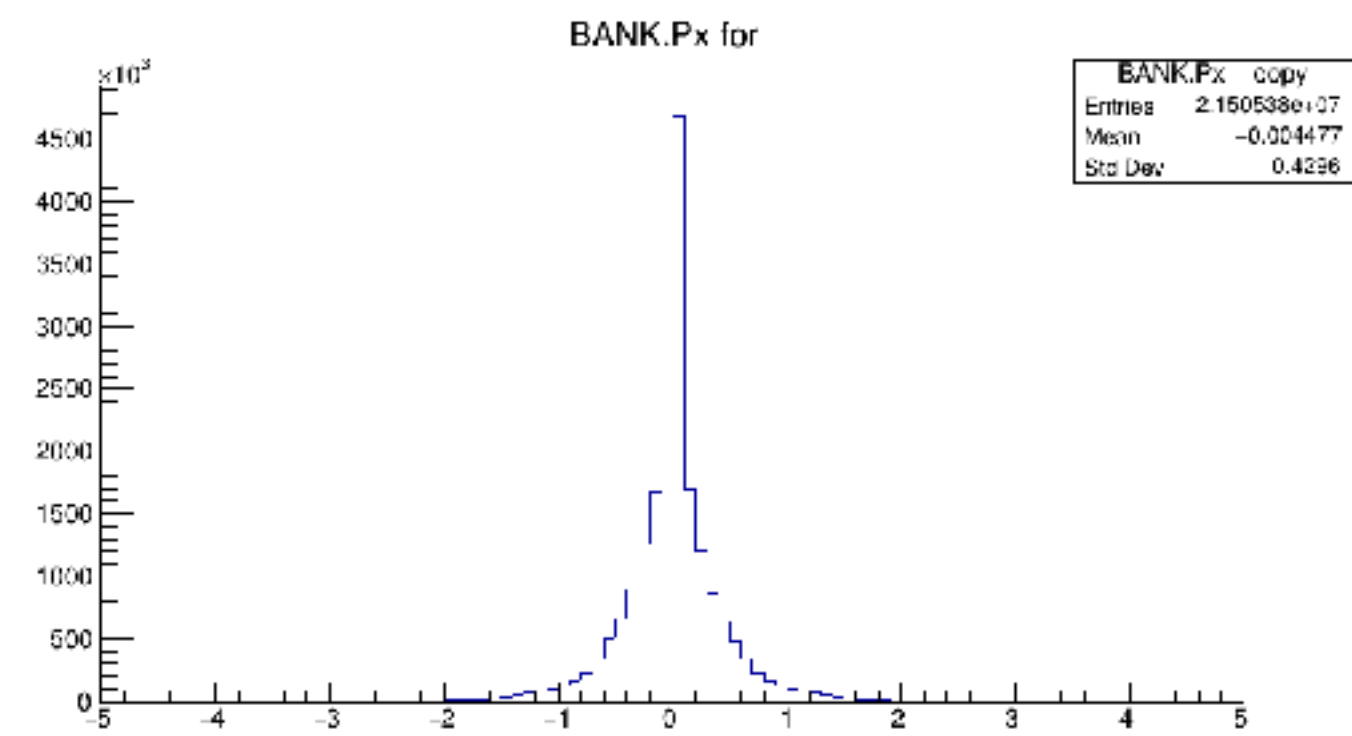
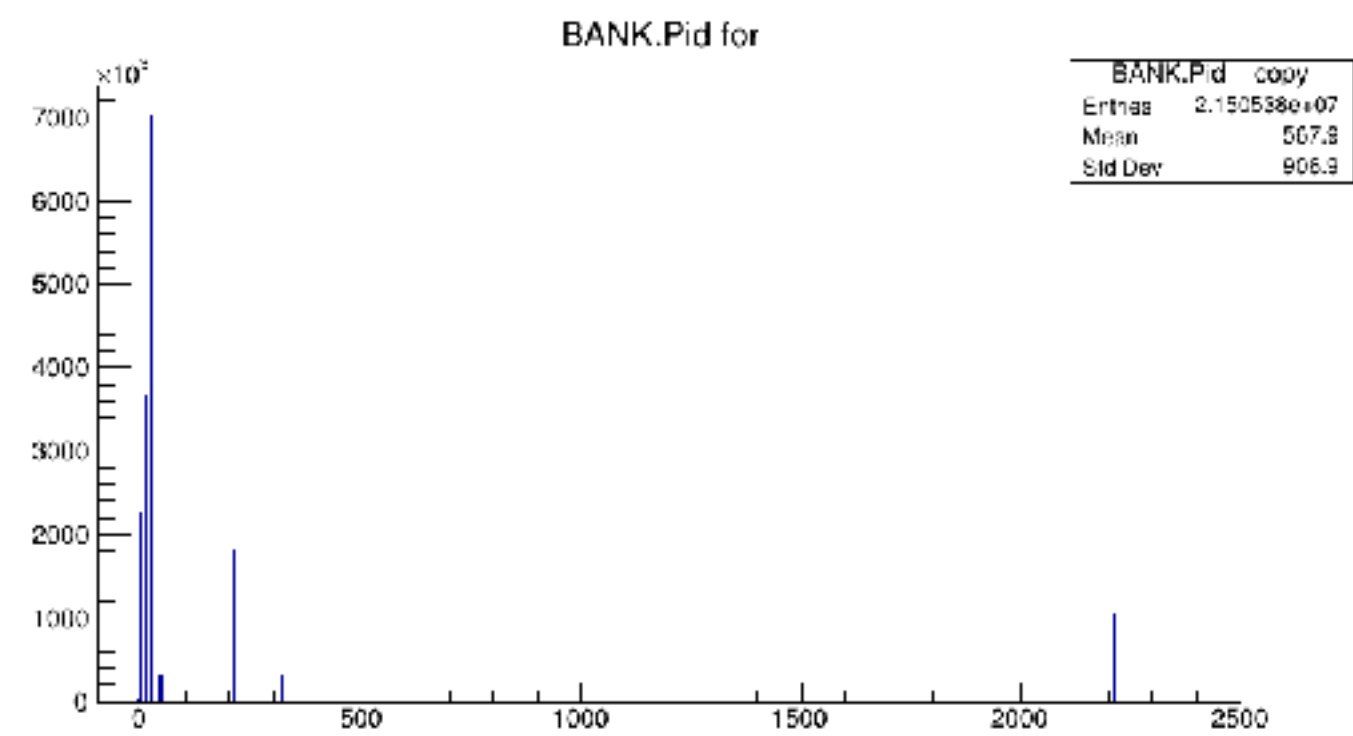
- ◆ File index is kept at the end of the file and read once at open operation.
- ◆ File index contains position and tag of each record in the file.



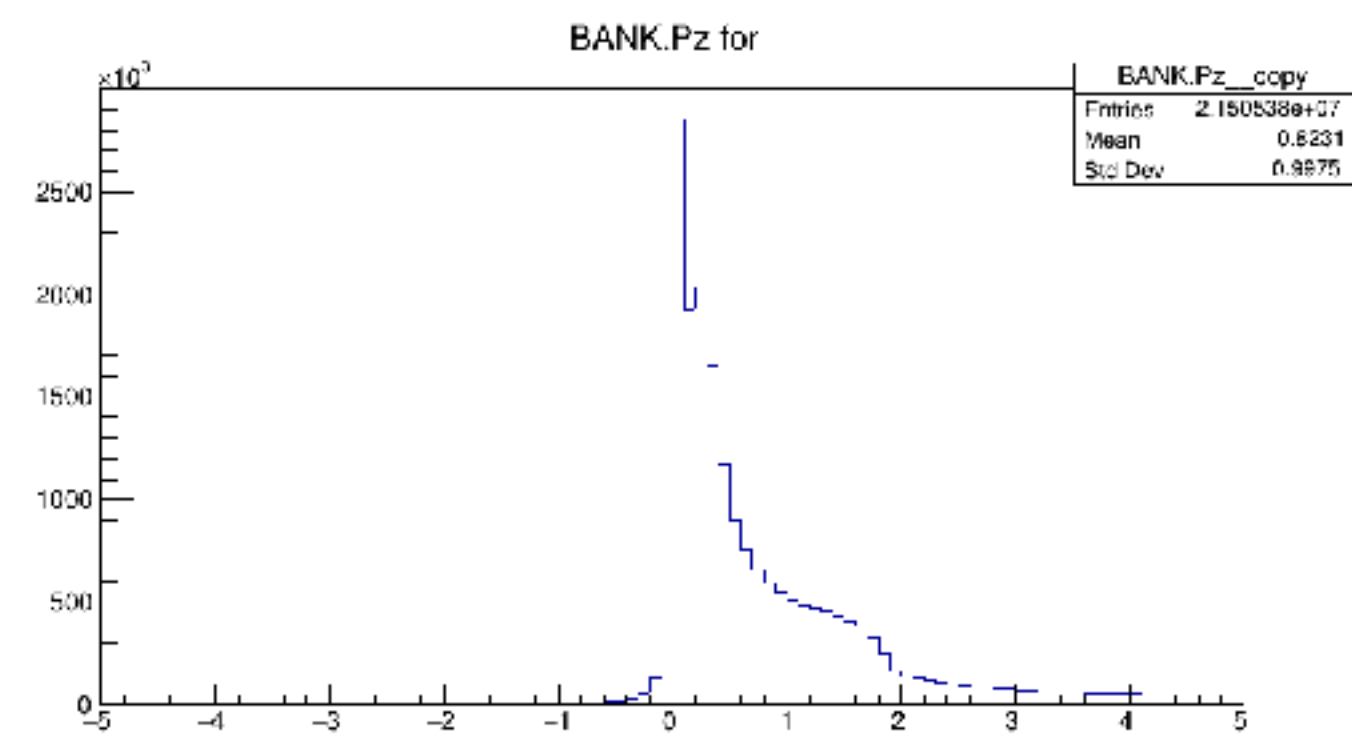
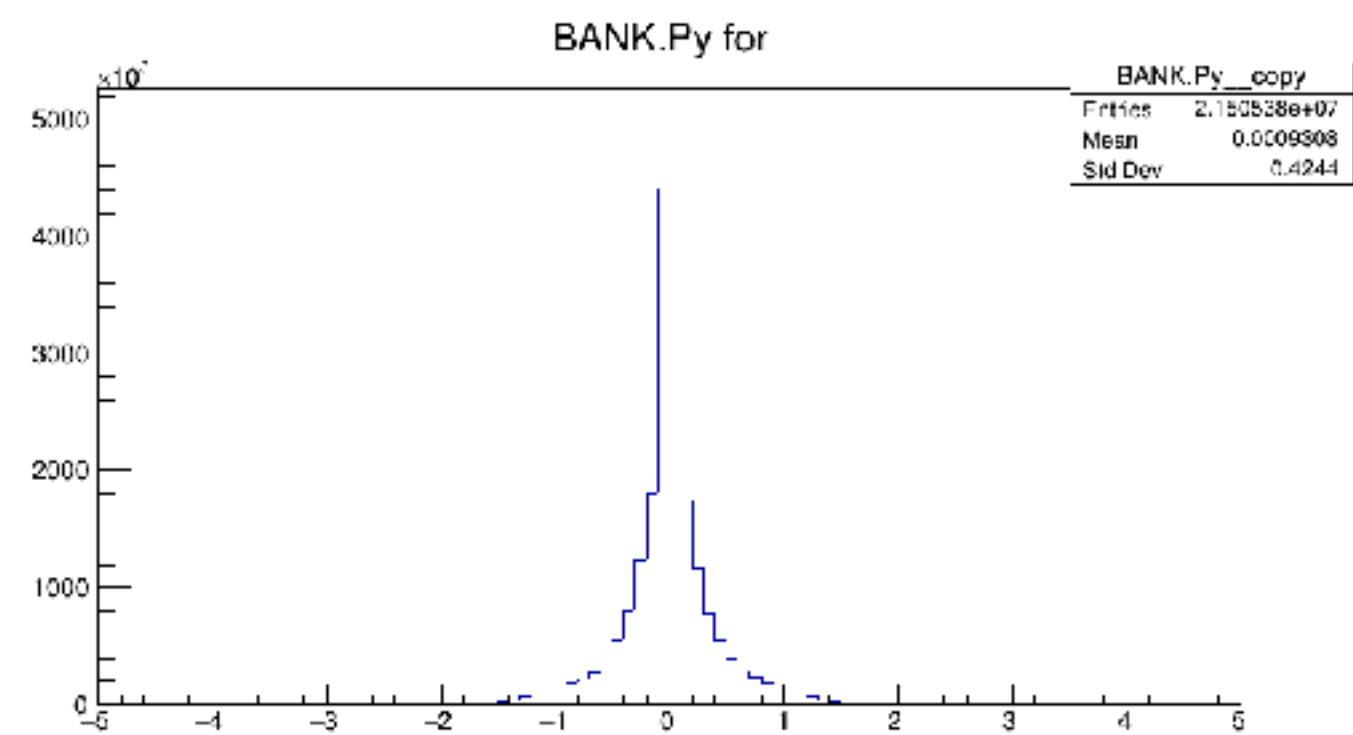
```
BankHist bankDraw("/work/jlab/clas12data/dst_skim4_5038.hipo");
bankDraw.Hist1D("REC::Event::StartTime",100,0,200,"")->Draw();

bankDraw.Hist1D("REC::Particle::Pid",1000,-100,2500,"");
bankDraw.Hist1D("REC::Particle::Px",100,-5,5,"");
bankDraw.Hist1D("REC::Particle::Py",100,-5,5,"");
bankDraw.Hist1D("REC::Particle::Pz",100,-5,5,"")->Draw("(2x2)");
```

7.4GB
HDD 75s
Cached 5s



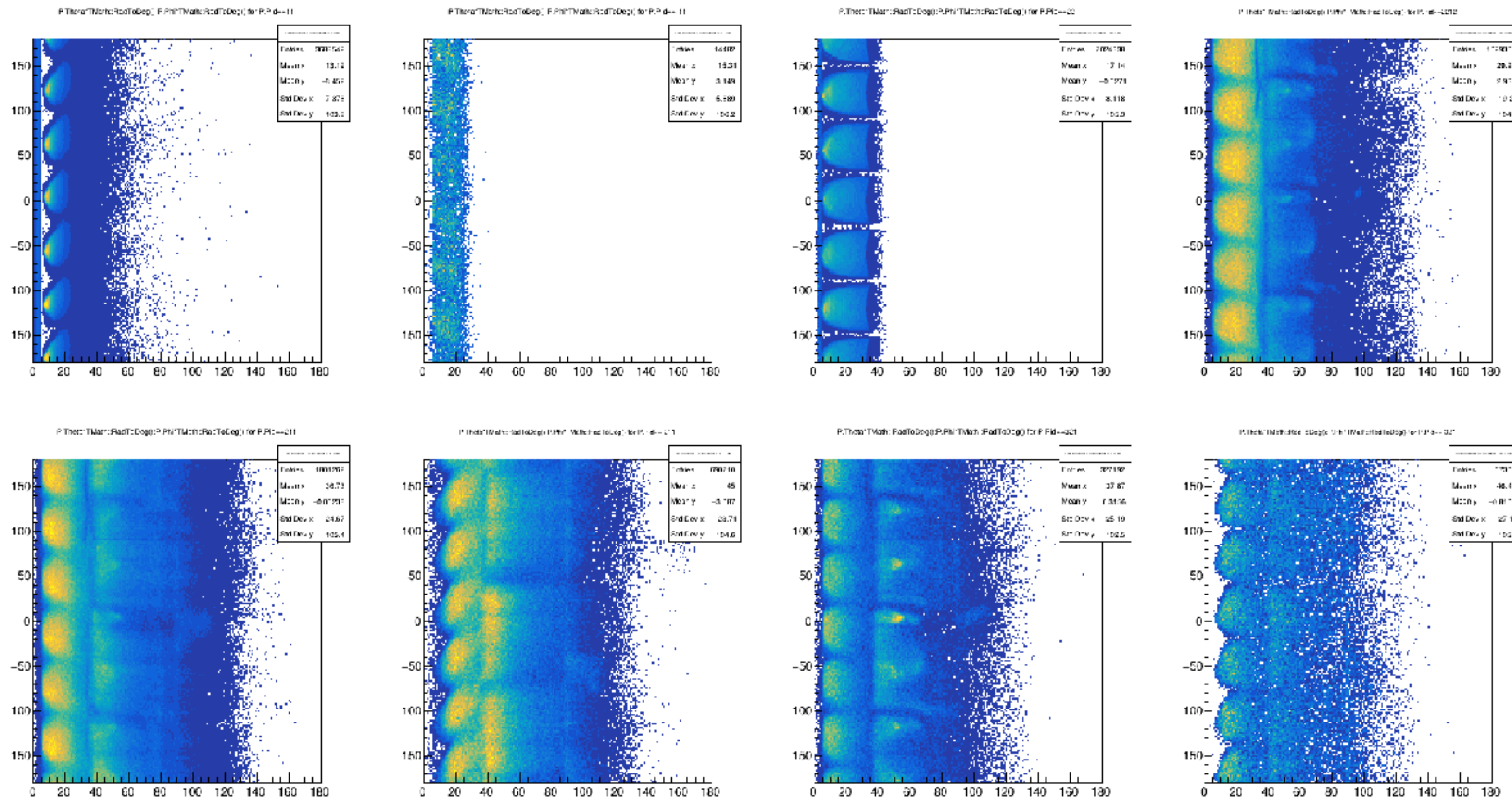
3.1M Events



HIPO/C++

```
hists.Hist2D("P.Theta*TMath::RadToDeg():P.Phi*TMath::RadToDeg()",180,0,180,180,-180,180,"P.Pid==11");  
hists.Hist2D("P.Theta*TMath::RadToDeg():P.Phi*TMath::RadToDeg()",180,0,180,180,-180,180,"P.Pid==-11");  
hists.Hist2D("P.Theta*TMath::RadToDeg():P.Phi*TMath::RadToDeg()",180,0,180,180,-180,180,"P.Pid==22");  
hists.Hist2D("P.Theta*TMath::RadToDeg():P.Phi*TMath::RadToDeg()",180,0,180,180,-180,180,"P.Pid==2212");  
hists.Hist2D("P.Theta*TMath::RadToDeg():P.Phi*TMath::RadToDeg()",180,0,180,180,-180,180,"P.Pid==211");  
hists.Hist2D("P.Theta*TMath::RadToDeg():P.Phi*TMath::RadToDeg()",180,0,180,180,-180,180,"P.Pid==-211");  
hists.Hist2D("P.Theta*TMath::RadToDeg():P.Phi*TMath::RadToDeg()",180,0,180,180,-180,180,"P.Pid==321");  
hists.Hist2D("P.Theta*TMath::RadToDeg():P.Phi*TMath::RadToDeg()",180,0,180,180,-180,180,"P.Pid==-321")->Draw("(4x2)col1");
```

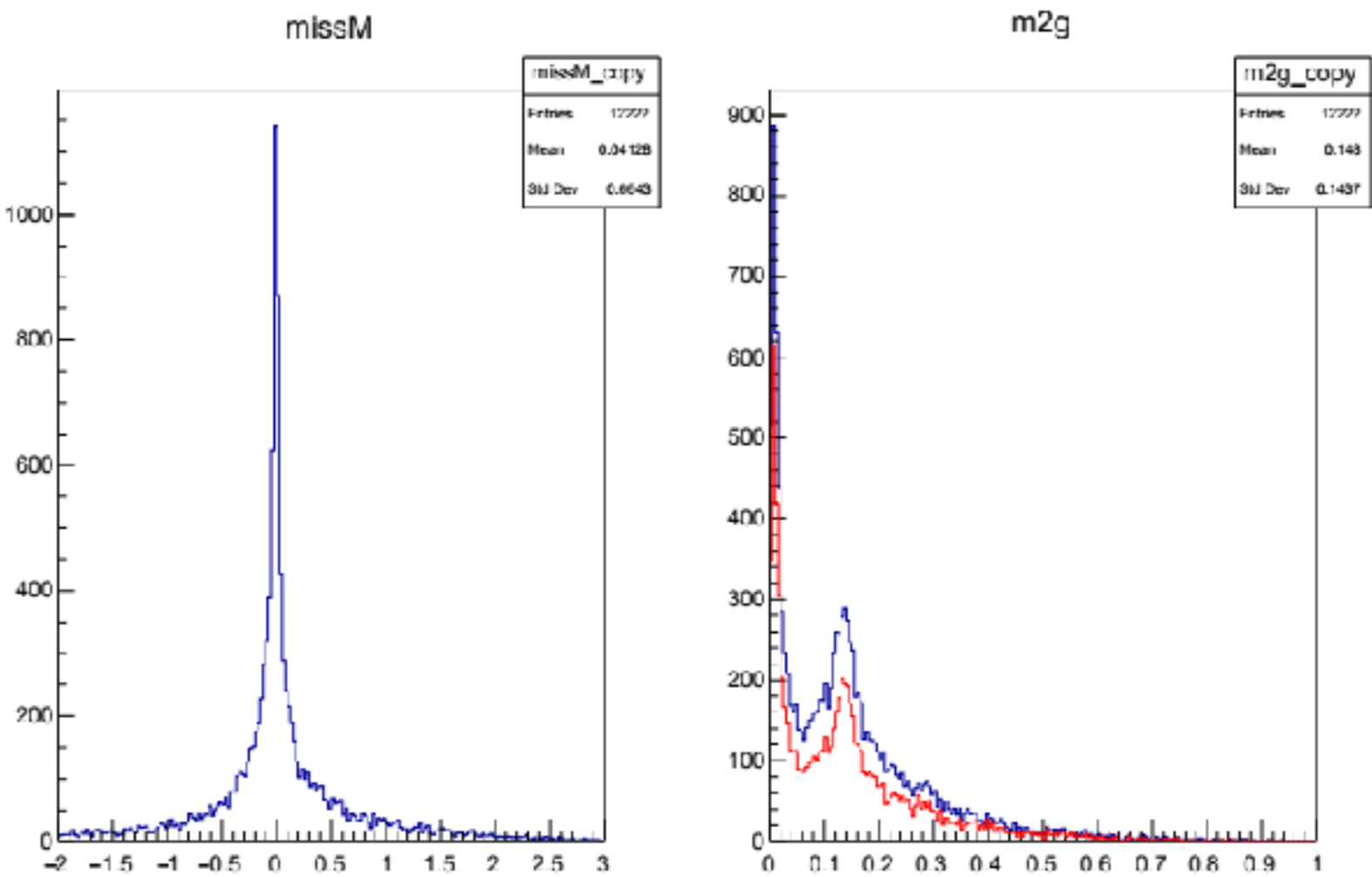
7.4GB
15s



Test 3 pi analysis
HipoSelector
TestSelector.C

7.4Gb
3.1M events

Read from HDD to cache = 75s



Workers	1	2	4
Full HDD	86 s	93 s	93 s
With Selection HDD	75 s	92 s	92 s
Full file cached	13 s	7 s	4 s
With Selection cached	4 s	3 s	3 s

Clas12Tool

Data Analysis Tools for HIPO data format.

Seperate code is provided for hipo3 and hipo4 format. The corresponding directories, libraries and binaries have an additional 3 or 4 to distinguish. Users are responsible for using the correct format for their hipo files.

Examples are given for running in interactive ROOT sessions and ROOT-Jupyter notebooks.

HipoX -> Clas12BanksX -> Clas12Root

The Hipo c++ reader library can be used independent of specific banks and ROOT, but depends on Hipo.

The Clas12Banks implementation can be used independent of ROOT, although currently ROOT dictionaries are created for the classes via cmake (this could be removed). This defines the specific CLAS12 DST banks and provides an interface to the data.

For actual Clas12Banks definitions see [HIPO4 DSTs](#)

The Clas12Root package depends on both Hipo and Clas12Banks. This provides ROOT-like analysis tools for operating on clas12 hipo DSTs.

```
HipoDraw  
HipoTreeMaker  
HipoProof
```

```
//create the event reader
clas12reader c12("file.hipo");

//Add some event Pid based selections
//////////c12.AddAtLeastPid(211,1); //at least 1 pi+
c12.addExactPid(11,1);    //exactly 1 electron
c12.addExactPid(211,1);   //exactly 1 pi+
c12.addExactPid(-211,1);  //exactly 1 pi-
c12.addExactPid(2212,1);  //exactly 1 proton
c12.addExactPid(22,2);    //exactly 2 gamma
////////c12.addZeroOfRestPid(); //nothing else
////////c12.useFTBased(); //and use the Pids from RECFT

while(c12.next()==true){
    c12.event()->getStartTime(); //hipo4
    // c12.head()->getStartTime(); //hipo3
    //Loop over all particles to see how to access detector info.

    for(auto& p : c12.getDetParticles()){
        // get predefined selected information
        p->getTime();
        p->getDetEnergy();
        p->getDeltaEnergy();
    }
}
```



```
//create the clas12 event reader
clas12reader c12event(inputFile);

while(c12event.next()==true){//loop over all events
    //loop over particles
    for(auto& p : c12event.getDetParticles()){

        int    pid = p->par()->getPid();
        float  px  = p->par()->getPx();
        float  py  = p->par()->getPx();
        float  pz  = p->par()->getPx();
    }
}
```

```
// get particles by type
auto electrons=c12.getByID(11);
auto gammas=c12.getByID(22);
auto protons=c12.getByID(2212);
auto pips=c12.getByID(211);
auto pims=c12.getByID(-211);

if(electrons.size()==1 && gammas.size()==2 && protons.size()==1 &&
    pips.size()==1 && pims.size() == 1){

    // set the particle momentum
    SetLorentzVector(el,electrons[0]);
    SetLorentzVector(pr,protons[0]);
    SetLorentzVector(g1,gammas[0]);
    SetLorentzVector(g2,gammas[1]);
    SetLorentzVector(pip,pips[0]);
    SetLorentzVector(pim,pims[0]);

    TLorentzVector miss=beam+target-el-pr-g1-g2-pip-pim;
    hmiss->Fill(miss.M2());
    TLorentzVector pi0 = g1+g2;
    hm2g->Fill(pi0.M());
    if(TMath::Abs(miss.M2())<0.5)hm2gCut->Fill(pi0.M());

    //could also get particle time etc. here too
    //Double_t eTime=electrons[0]->sci(FTOF1A)->getTime();
}
```



♦ Jupyter Notebooks:

- ♦ jdk9 (jshell) made jupyter java kernel possible.
- ♦ docker container is created with CLAS12 Java Software.
- ♦ GROOT was modified to be able to produce plots for Jupyter
- ♦ examples of reading/writing HIPO4 files in Java are moved to Jupyter notebooks.
- ♦ Clas12Tools (C++) package also has Jupyter notebooks.

CLAS12 Data Analysis in Jupyter

Recent developments with Java CLI introduced new JAVA kernel in Jupyter, which allows using Notebooks for data analysis in JAVA. For CLAS12 a docked container was created for easy Jupyter notebook environment for CLAS12 JAVA data analysis framework.

Getting Docker image

To run the Docker image on local host use the command:


```
docker run -p 8888:8888 gavalian/jnp_docker
```

Once docker is downloaded and runs, the message will be displayed with token information to access the notebook. It will look something like this:

```
Or copy and paste one of these URLs:  
Blah-Blah-Blah:8888/?token=74158f2adef2087ba9eb3bcf1152f67432391752942ac91f
```

The "Blah-Blah-Blah" replaces the string that people kept clicking on before reading rest of the instructions.

Open a browser and type in the URL:

```
http://localhost:8888 
```

You'll be asked for a token to start the notebook, copy/paste the token only (not entire URL) into the box, and that will start the notebooks.

NOTE ! your token will be different.


```
In [5]: //---- imports for HIPO4 library
import org.jlab.jnp.hipo4.io.*;
import org.jlab.jnp.hipo4.data.*;
//---- imports for CROOT library
import org.jlab.groot.data.*;
import org.jlab.groot.graphics.*;
//---- imports for PHYSICS library
import org.jlab.jnp.physics.*;

HipoReader reader = new HipoReader(); // Create a reader object
reader.open("../run_004013_00_2c.hipo"); // open a file
```

```
reader:: *****>>> opening file : ../run_004013_00_2c.hipo
reader:: ***** dictionary entries :      2
scanning trailer : # bytes = 120
reader:: ***** number of records :      3
reader:: ***** number of events :    47210
```

Now we create an Event class and a bank instance, in this case for particle bank. The event instance is used to read events from the file, and Bank instance is used to read particle bank from each event. NOTE ! the file must be opened first in order to initialize Bank object since it takes the schema for the bank from file dictionary.

```
In [6]: Event    event = new Event();
Bank    particles = new Bank(reader.getSchemaFactory().getSchema("REC::Particle"));
```

Now we can loop over the events in the file and read particle bank and check for electron in the first row, if there is one we can create lorentz vector for the particle and calculate W2 and Q2, and plot it. First we will declare histogram objects and canvas object:

```
In [7]: H1F hW = new H1F("hW", 100, 0.5, 4.0);
H1F hQ2 = new H1F("hQ2", 100, 0.1, 4.0);
hW.setTitleX("W [GeV]");
hQ2.setTitleX("Q^2 [GeV/c^2]");

EmbeddedCanvas ec = new EmbeddedCanvas(800,400);

[SystemFonts] ---> set size = 25, available 2
```

Now we can loop over the events and count how many events we have where electron is detected.

```
In [8]: int counter = 0;
int elec = 0;

while(reader.hasNext()==true){
    reader.nextEvent(event);
    event.read(particles);
    if(particles.getRows()>0){
        int pid = particles.getInt("pid",0);
        if(pid==11){
            elec++;
        }
    }
    counter++;
}
System.out.println("processed # " + counter + " , electrons : " + elec);

processed # 47210 , electrons : 4013
```

```
In [9]: LorentzVector vBeam = new LorentzVector(0.0,0.0,10.6,10.6);
LorentzVector vTarget = new LorentzVector(0.0,0.0,0.0,0.938);
LorentzVector electron = new LorentzVector();
LorentzVector vW = new LorentzVector();
LorentzVector vQ2 = new LorentzVector();

reader.getEvent(event,0); // Reads the first event and resets to the begining of the file

while(reader.hasNext()==true){
    reader.nextEvent(event);
    event.read(particles);
    if(particles.getRows()>0){
        int pid = particles.getInt("pid",0);
        if(pid==11){
            electron.setPxPyPzM(
                particles.getFloat("px",0),
                particles.getFloat("py",0),
                particles.getFloat("pz",0),
                0.0005
            );

            vW.copy(vBeam);
            vW.add(vTarget).sub(electron);

            vQ2.copy(vBeam);
            vQ2.sub(electron);

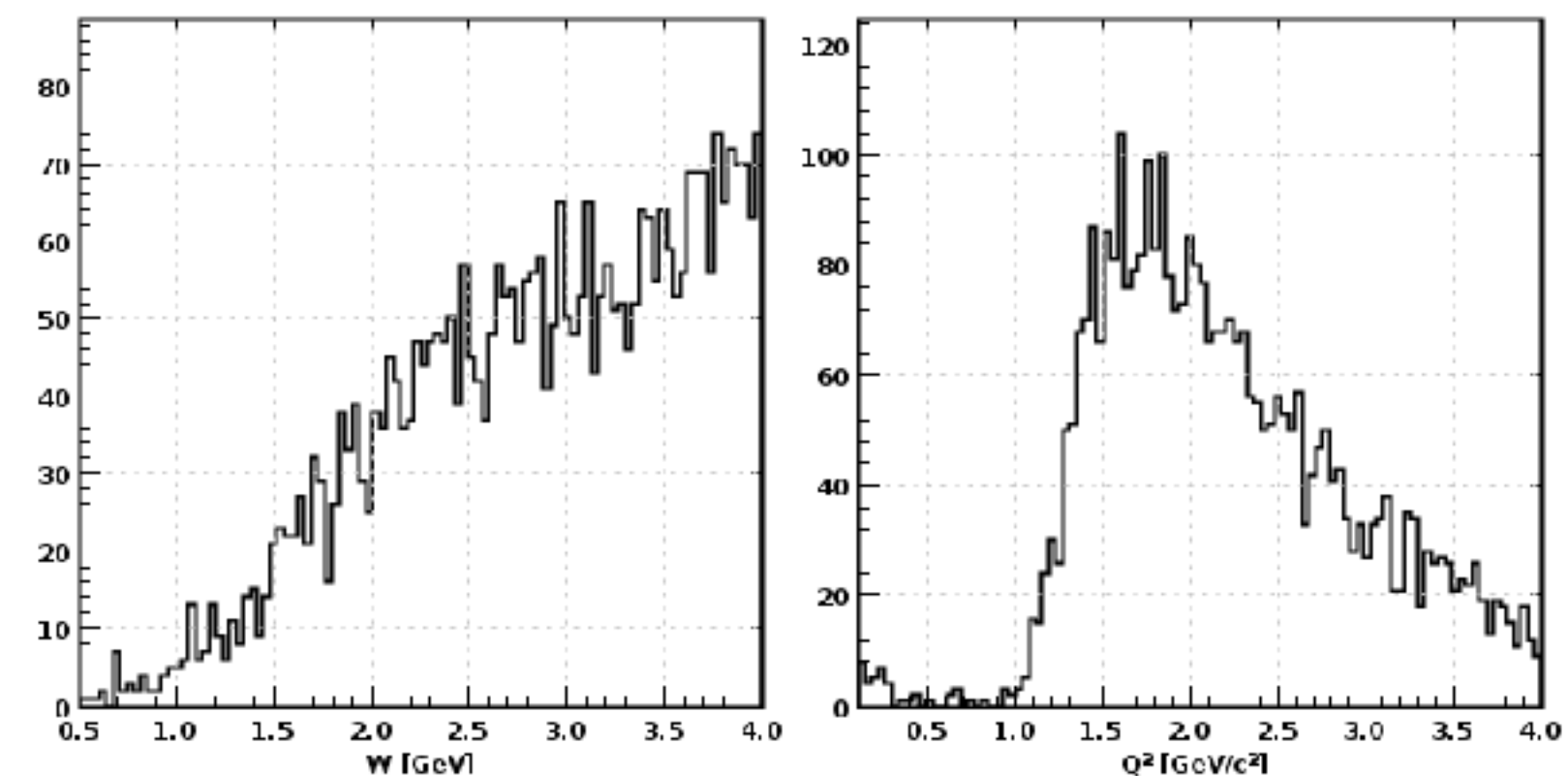
            hW.fill(vW.mass());
            hQ2.fill(-vQ2.mass2());
        }
    }
}

ec.divide(2,1);

ec.cd(0).draw(hW);
ec.cd(1).draw(hQ2);

ec.getScreenShot();
```

Out[9]:



Summary

- ◆ HIPO4 transition is now complete, COATJAVA versions 6.X.X.
- ◆ utilities program provides most of the functionality needed day-by-day
- ◆ JAW was extended to plot content of HIPO4 for quick checks (for developers only)
- ◆ C++ interface is now complete (Clas12Tools):
 - ◆ makes it possible to make quick plots in ROOT.
 - ◆ nice interface to loop over particles and do analysis
 - ◆ convertor to write ROOT files from HIPO
- ◆ Jupyter Notebook examples on how to read/write HIPO files in JAVA/Groovy
 - ◆ Live demo will be shown during software workshop



BACKUP SLIDES