Data Format
for Physics Analysis

G.Gavalian (JLAB)

Introduction

- ✦ Data File Formats
- ✦ Lessons from previous experience
- ✦ HIPO Data Format
- ✦ Future of Physics Data formats
- ✦ Data Analysis Tools

Lessons (from data-mining)

- **BOS**:
  - format does not have compression, external dictionary
  - file size limit of 2GB
  - sequential read only
- **EVIO**:
  - does not have compression
  - random read is only possible for files <2GB
  - random read initialization very slow
  - big memory foot print for files with small event size
- **HDF5**:
  - event by event reading is slow, designed of keeping large objects
  - unable to write variable length events
  - no native JAVA interface

CLAS12

✦EVIO

- ✦ EVIO is not suited for large data files analysis
- ✦ files with small event size cause big time latency and memory footprint
- ✦ inability to random read large files makes it useless for debugging programs (like CED)
- ✦ no utilities to filter banks or events, split file.

✦HIPO

- ✦ New flexible data format was introduced solve shortcomings of other data formats.
- ✦ Data is on fly compressed with LZ4 (much faster performance compared to GZIP, used by ROOT)
- ✦ Large data file support with random access with smaller latency
- ✦ Internal dictionary describing data structures

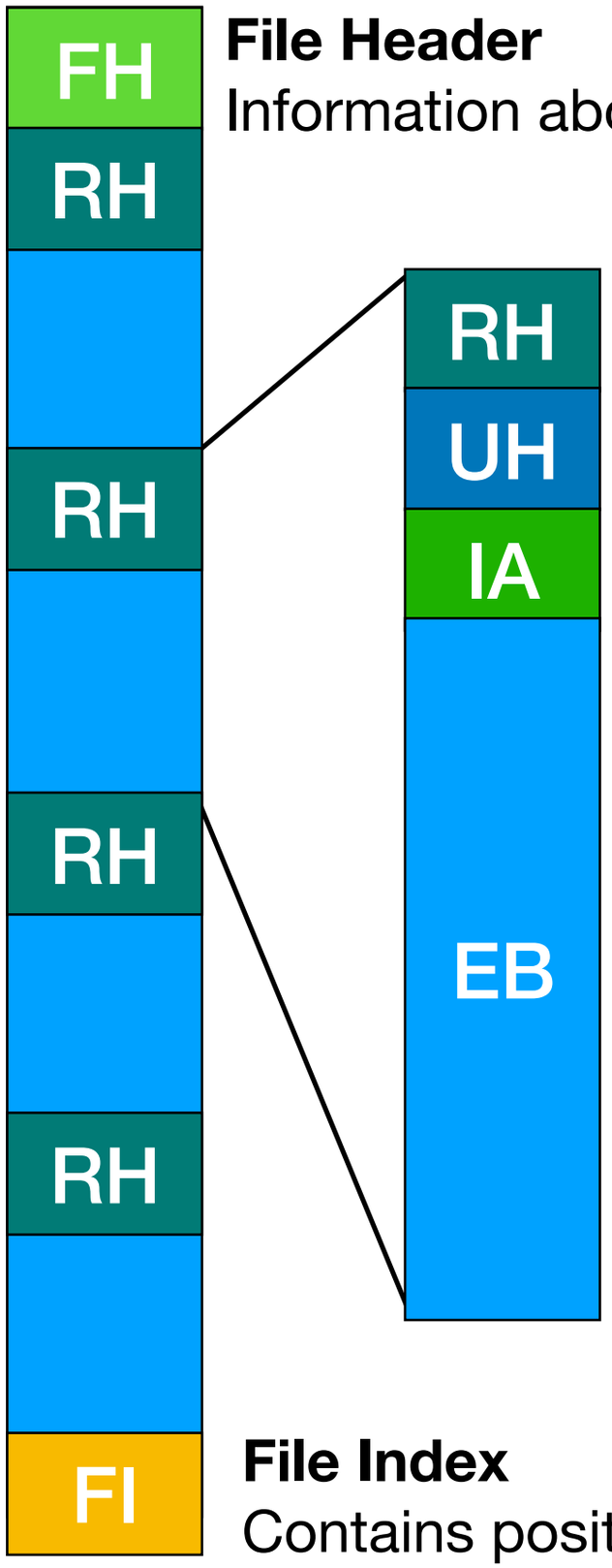
DATA FORMAT

- Good deal of thought have to be given to the data format for storing the physics results from experiment.
- The world is moving towards much larger data sets, fast random access and compression are a must.
- The analysis workflow will depend on the data format.
- Large data sets need different set of tools to browse, skim and condition events for final analysis, tag groups of events.
- Framework will be crucial for reproducibility of results.
- Archiving and tagging of data needs to be done at the first level of creating the data.

DIANA-HEP project

- HEP community is developing new approaches to large data set analysis.
- They employ Apache-Spark for efficient data reading and reducing.
- There is a lot of discussion on how ROOT data format is not well suited for big data sets for efficient filtering and skimming.
- The data is converted to Apache-Avro (from ROOT) for parallel data processing.
- The language used with Apache tools is SCALA (High Level JAVA scripting language, optimized for multi-threaded applications).
- Check it out at: http://diana-hep.org

HIPO File Format



File Header
Information about file

Record Header
Contains record properties (size, compression type, event count). Contains fields for tagging the record (up to 8 tags)

User Header
Contains information about the record dictionary, format. User specified parameters related to conditions of the experiment.

Index Array
Array of event offsets inside the event buffer. Dynamically creates event random access table.

Event Buffer
Events in the record back to back. To disengage them index array is needed.

File Index
Contains positions of all records in the file

Why is HIPO good for data analysis

- HIPO is dictionary driven data format (backward compatibility)
- Record based data structure allows event grouping by tags:
  - different topologies of events can be separated into different records.
  - record headers can contain scaler information.
  - reading relevant record positions is very fast.
  - skimming files can by desired final state can be done on IO level.
- Well organized file will lead to reproducibility and easy data distribution and archiving.
- Data cuts (fiducial cuts) and correction parameters can be embedded into the data file.
- Record structure of HIPO makes it ideal for Apache-Spark DataFrame implementation.
- Analysis codes can be embedded into the file (as binary JAR)
- It stood test of time. Was successfully used in KPP and ER.

Benchmarks

- For Benchmarking run #2475 was used (entire run size 89 GB)
- Filtered events leaving REC::EVENT and REC::PARTICLE (size 670 MB)
- HIPO file was converted to ROOT file, using HasPec code (Derek)
- Run test on reading the data from the file in both formats
- Second test was done on reconstruction file without filtering (all banks included)
- The all banks test was done on better machine (2.6 GHz i7)

File Format	File Size	Read Time (sec)	Machine
ROOT	810 MB	**32.45**	**1.4 GHz i5**
HIPO	670 MB	**7.12**	**1.4 GHz i5**
ROOT(ALL) STL VECTOR	1.92 GB	**48.43**	**2.6 GHz i7**
ROOT(ALL) ARRAY C-9	1.37 GB	**63.34**	**2.6 GHz i7**
ROOT(ALL) ARRAY C-1	1.58 GB	**65.72**	**2.6 GHz i7**
HIPO(ALL)	1.87 GB	**10.51**	**2.6 GHz i7**
HDDM (HALL-D)	468 MB	**14.87**	**IFARM1401**
HIPO	517 MB	**5.02**	**IFARM1401**

- HIPO file reading is **~5x** faster compared to ROOT
- Performance difference between HDDM and HIPO are largely due to GZIP vs LZ4
- decompression speeds

Benchmark

- I was reminded by many people that ROOT can read only specified branches, which considerably speeds up the reading.
- I spent 1 hour to make each bank to be written into different basket in HIPO allowing only reading specific parts of the file to speed up the process.
- Turned out it was easy to change the read structure to be similar to how ROOT writes the branches.
- Resulting data that is being read is only 5.7 MB, yet again HIPO beats ROOT by large margin.
- In real analysis scenario it is highly unlikely that one will write 1.9 GB file with only 5.7 MB relevant data.
- Surprisingly HIPO reading all the branches is still faster than ROOT reading only two branches (EVENT, and HEADER)

File Format	File Size	Read Time (sec)	Machine
ROOT (EVENT BRANCH)	1.89 GB	**6.15 (1.03)**	**2.6 GHz i7**
ROOT (ALL BRANCHES)	1.89 GB	**63.34**	**2.6 GHz i7**
HIPO (ALL BRANCHES)	1.92 GB	**7.06**	**2.6 GHz i7**
HIPO (EVENT BRANCH)	1.92 GB	**0.15**	**2.6 GHz i7**

- **to filter 1 PB of data with ROOT ~ 31.7 Years**
- **to filter 1 PB of data with HIPO ~ 0.62 Year**

RUN 3249 (on volatile)

- 2 TB of data with ROOT ~ 23 days
- 2 TB of data with HIPO ~ 0.4 days

Data Analysis Tools

- JAVA Analysis Workstation (JAW) is included in the distribution
- Allows PAW like N-Tuple manipulations with familiar KUIP syntax
- Implements dynamic auto corrective Command Line interface
- Command line history and script running (batch mode coming soon to computers near you)
- Physics analysis module is available for quick data checks
- Modules can be extended to read detector responses as well
- Modular approach to data representation inside JAW allows development of custom modules for specific reactions.

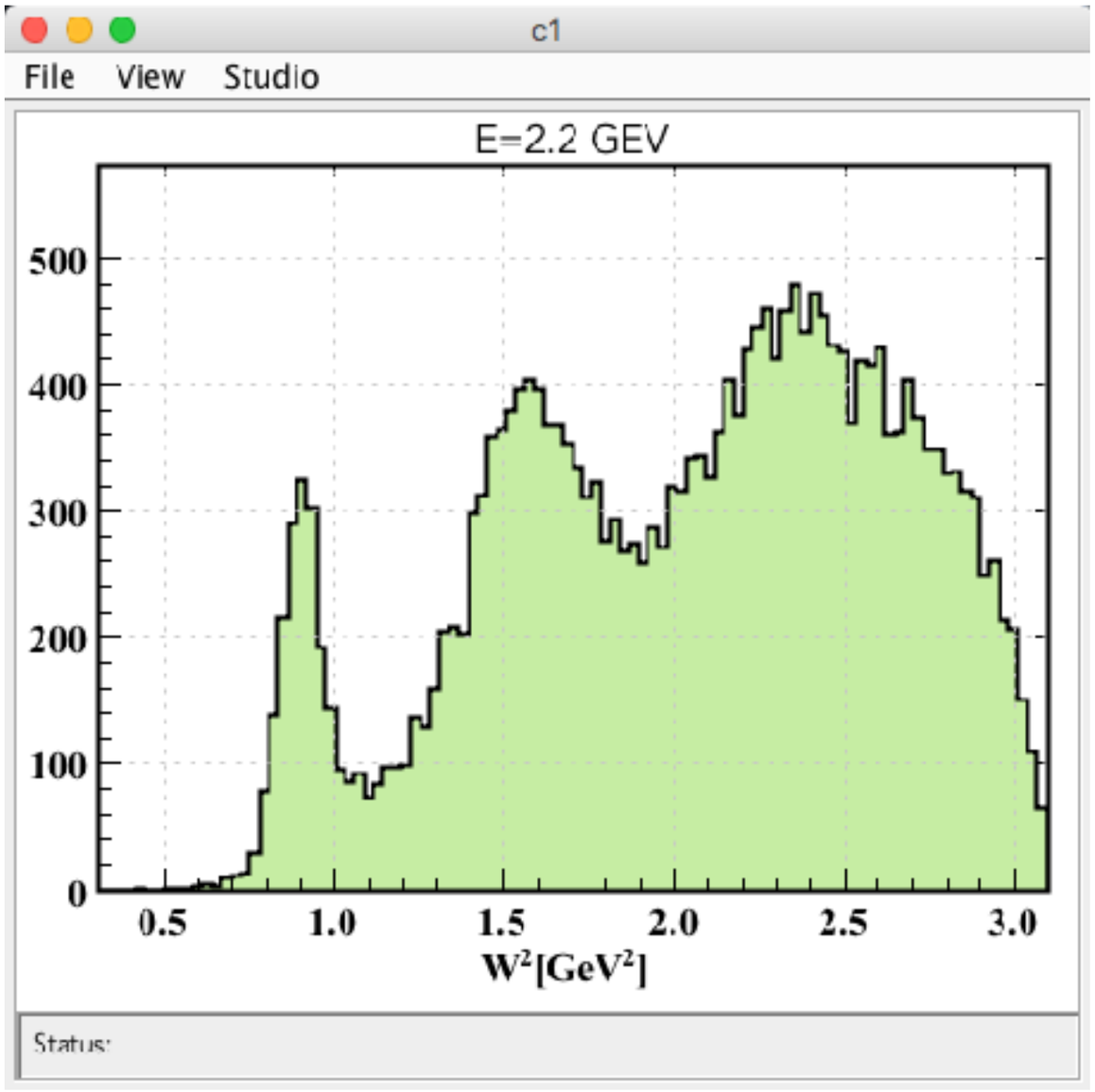
Physics Analysis

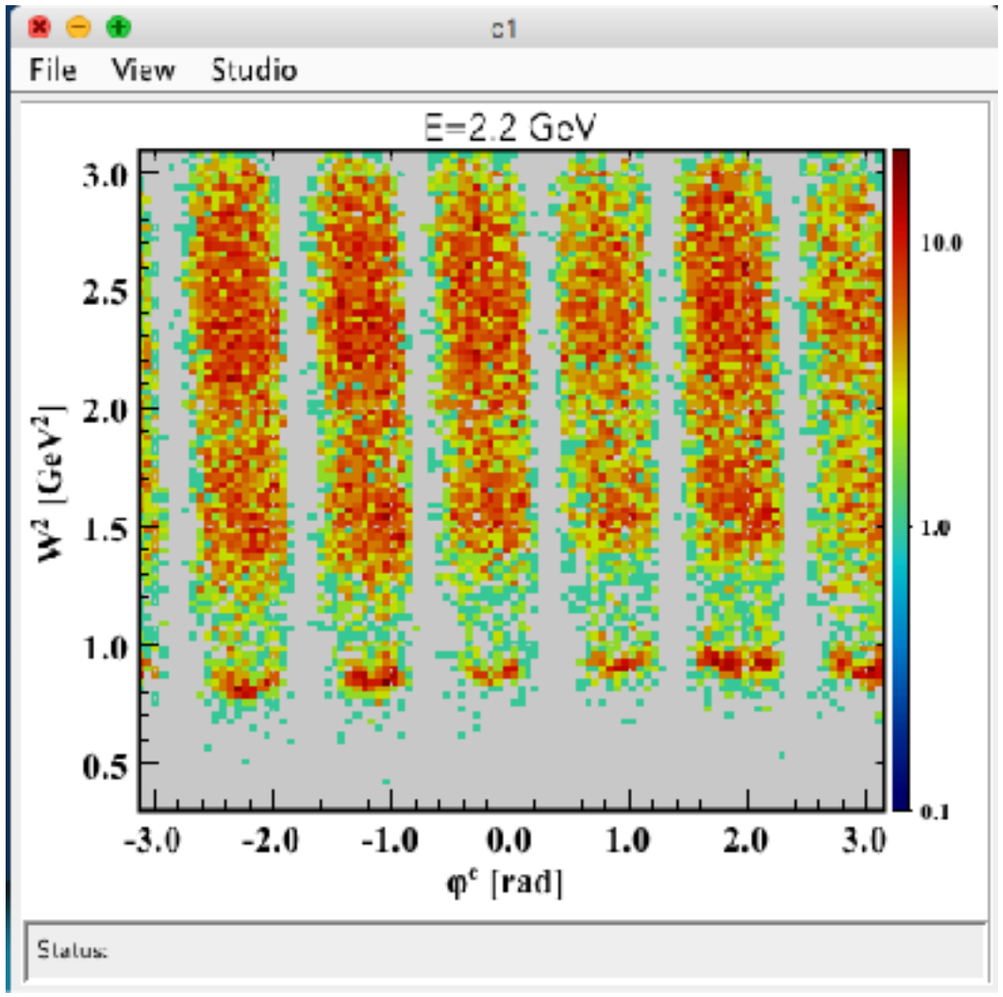
```
reaction/create    10  11:2212:Xn:X+ 2.2
reaction/particle  10  w2 [b]+[t]-[11] mass2
reaction/file      10  clas_run_2475.hipo.0
reaction/plot      10.w2 ! 1000 200
```

! - means use default value

CUT **N Events** **N Bins**

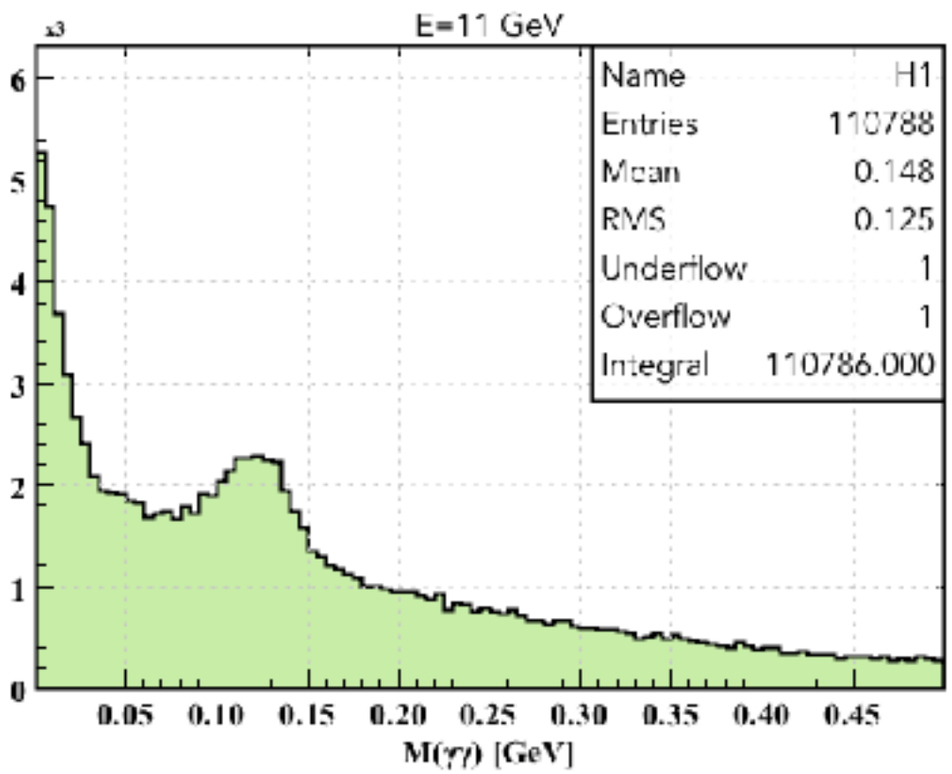
```
reaction/plot      10.w2 ! 1000 200
```

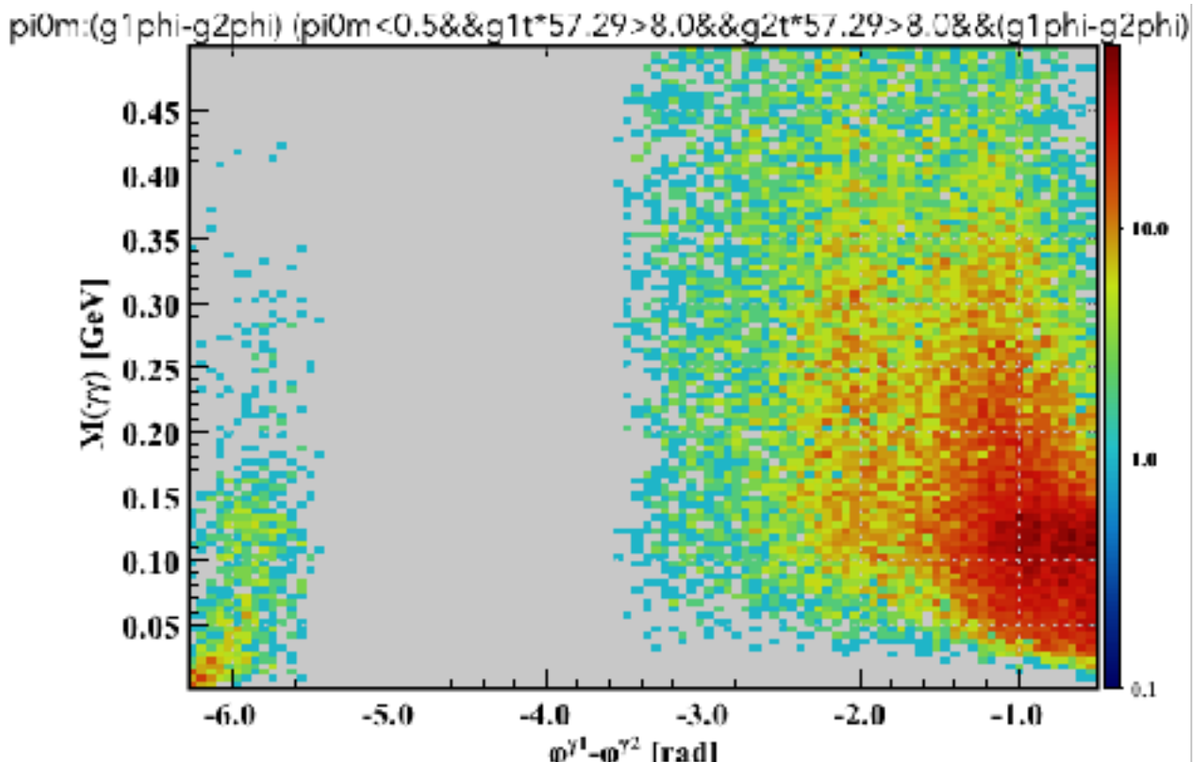





```
reaction/particle  10  ephi [11] phi
reaction/plot      10.w2%ephi
```

Physics Analysis





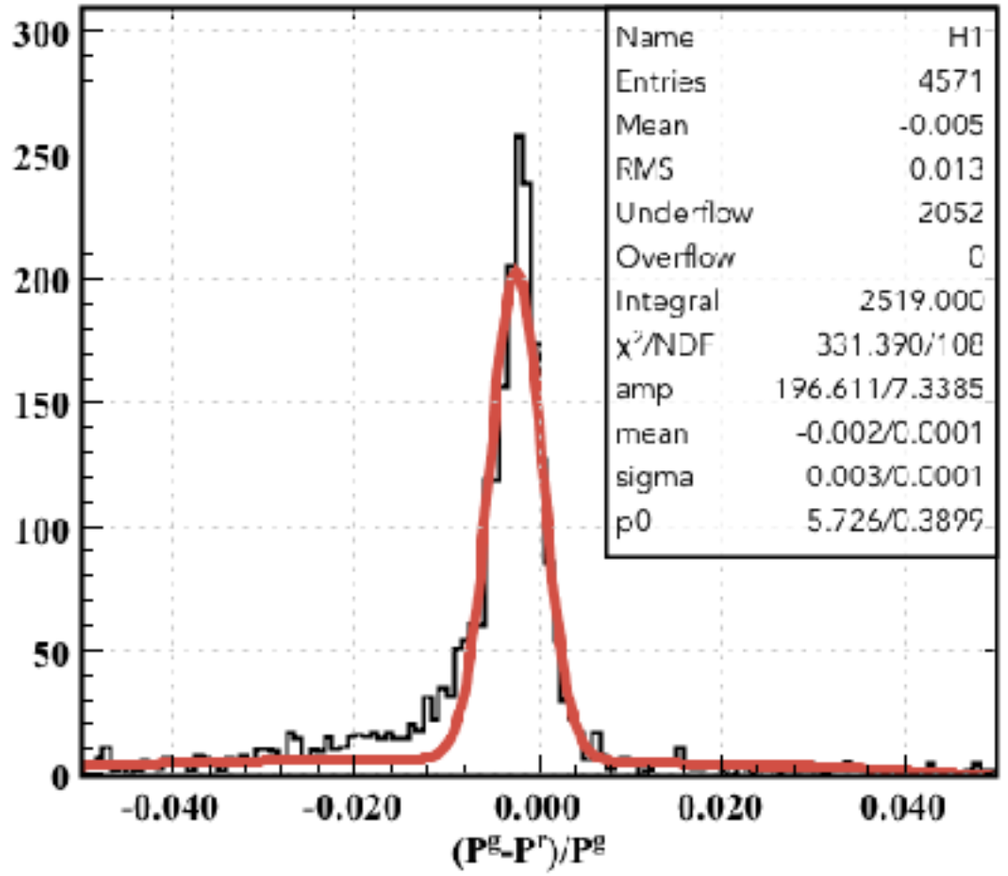

```
canvas/zone 2 1
reaction/create   10 11:22:22:X+:X-:Xn 2.2
reaction/file     10 run_02861_0_20.hipo
reaction/particle 10 g1p    [22,0] p
reaction/particle 10 g2p    [22,1] p
reaction/particle 10 g1t    [22,0] theta
reaction/particle 10 g2t    [22,1] theta
reaction/particle 10 g1phi  [22,0] phi
reaction/particle 10 g2phi  [22,1] phi
reaction/particle 10 pi0m   [22,0]+[22,1] mass
reaction/plot     10.pi0m pi0m<0.5
reaction/plot     10.pi0m%(g1phi-g2phi) pi0m<0.5&&g1t*57.29>8.0&&g2t*57.29>8.0&&(g1phi-g2phi)<-0.5
```

Particle Pseudo-Code

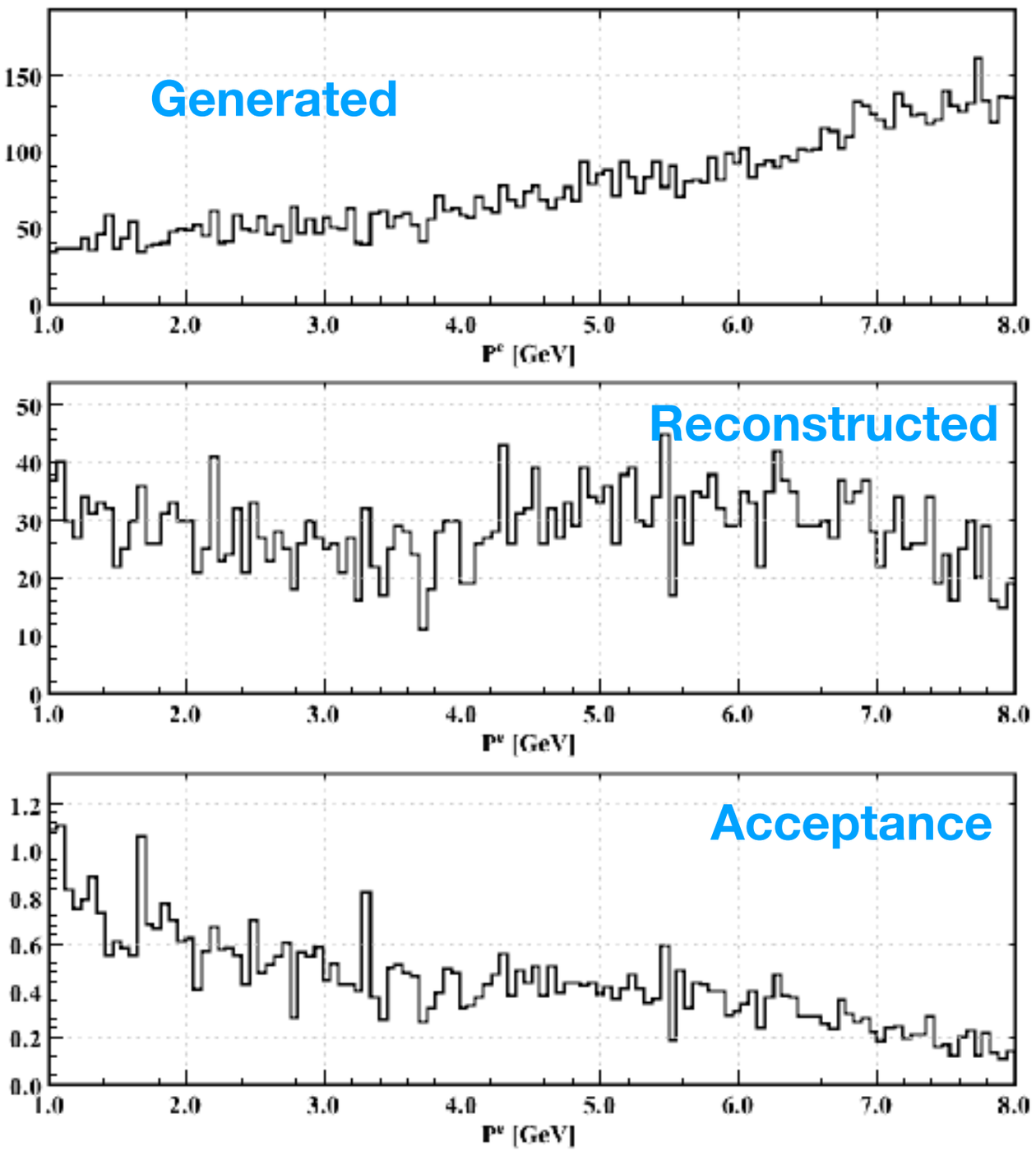
- Event Filters:
  - **11:X+:X-:Xn** at least one electron, any number of positive, negative and neutral particles
  - **11:2212:22:22:X+:X-** at least one electron, at least one proton, only two photons, and any number of charge particles to follow
  - **11:211:22** - only one electron, one pion and only one photon
- Event Particles:
  - **[11,0]** - electron from the reconstructed event (first one)
  - **(11,0)** - electron from generated event
  - **{11,0}** - a negative particle from the reconstructed event that matches first generated electron
  - **[22] or [22,0]** - first photon in the event
  - **[22,1]** - second photon from the event
  - **(22,2)** - third generated photon
  - **[211,0,0.938]** - first positive pion from the event (assign proton mass to the particle)

Resolution

```
reaction/create    10 11:X+:X-:Xn 11.0
reaction/mcfilter  10 11:X+:X-:Xn
reaction/file      10 generated_sidis.hipo
reaction/particle  10 ep  [11] p
reaction/particle  10 egp (11) p
reaction/plot      10.(ep-egp)/egp ! ! 120 -0.2 0.2 220
```



Easy Acceptance Calculations




```
reaction/create   10 X+:X-:Xn 11.0
reaction/mcfilter 10 11:X+:X-:Xn
reaction/file     10 generated_sidis.hipo
reaction/particle 10 epp (11) p
reaction/create   11 11:X+:X-:Xn 11.0
reaction/file     11 generated_sidis.hipo
reaction/particle 11 erp [11] p
canvas/zone 1 3
reaction/plot     10.epp ! ! 120 1.0 8.0 220
reaction/plot     11.erp ! ! 120 1.0 8.0 221
histogram/divide  222 221 220
histogram/plot    222
```

Summary

- **HIPO data format was developed for CLAS12 reconstruction**
  - Flexible data format (dictionary driven, backward compatible).
  - Compression (LZ4) and random access.
  - Outperforms all the other formats tested.
  - Ideal for streaming applications due to internal structure.
  - Can be used as DST format, provides ability to sort internal content.
  - Utilities to tag individual data records for fast reading.
- **Physics Analysis Framework**
  - Analysis framework was developed as a prove of concept.
  - The analysis framework is being extended to be used with HADOOP and Apache-Spark, for fast data skimming and plotting through distributed file systems.
  - Modular design of this framework allows implementations of sharable analysis codes (like fiducial cuts and corrections) that can be standardized.
  - **Documentation** - http://clasweb.jlab.org/jhep/docs/jaw/html/