Data Format

G.Gavalian (Jefferson Lab)

Outline

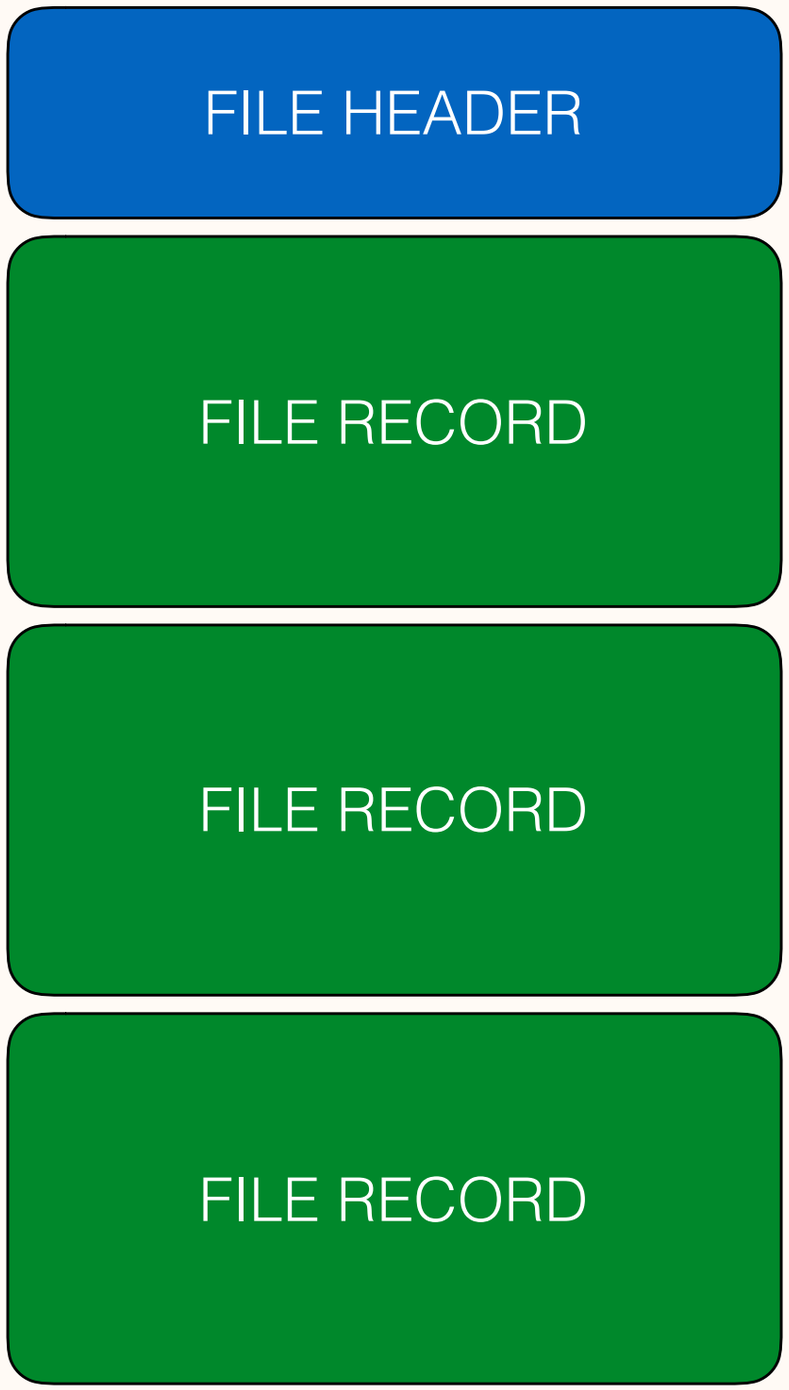
- CLAS12 KPP run data format
- Unified JLAB data format
- Data compression
- Summary

Data Format (in collaboration with CODA group, D. Abbot,C. Timmer)

- **HIPO Data Format**
  - ✓ Used during KPP, proved to be efficient format data processing
  - ✓ Record structure of file provides fast indexing and random access
  - ✓ Ability to write large files with no penalty on random access
  - ✓ Provides on fly compression (x3 over EVIO)
- **EVIO 6.0 implementation**
  - ✓ HIPO file structure is used to store EVIO events.
  - ✓ Fast compression algorithm provides data reduction up 2 times.
  - ✓ Best compression algorithm provides compression 2.5x-3x times.
  - ✓ Multithreaded data recorder is implemented to keep up with DAQ.
- **Future of Data Formats**
  - ✓ From EVIO 6.0 the file structure for HIPO and EVIO is the same.
  - ✓ Implementations of Java and C++ exist.
  - ✓ HIPO internal bank structure is still faster than EVIO
  - ✓ HIPO API allows modifying banks without decomposing
  - ✓ Reading from the banks is non-copy read (faster, less memory)

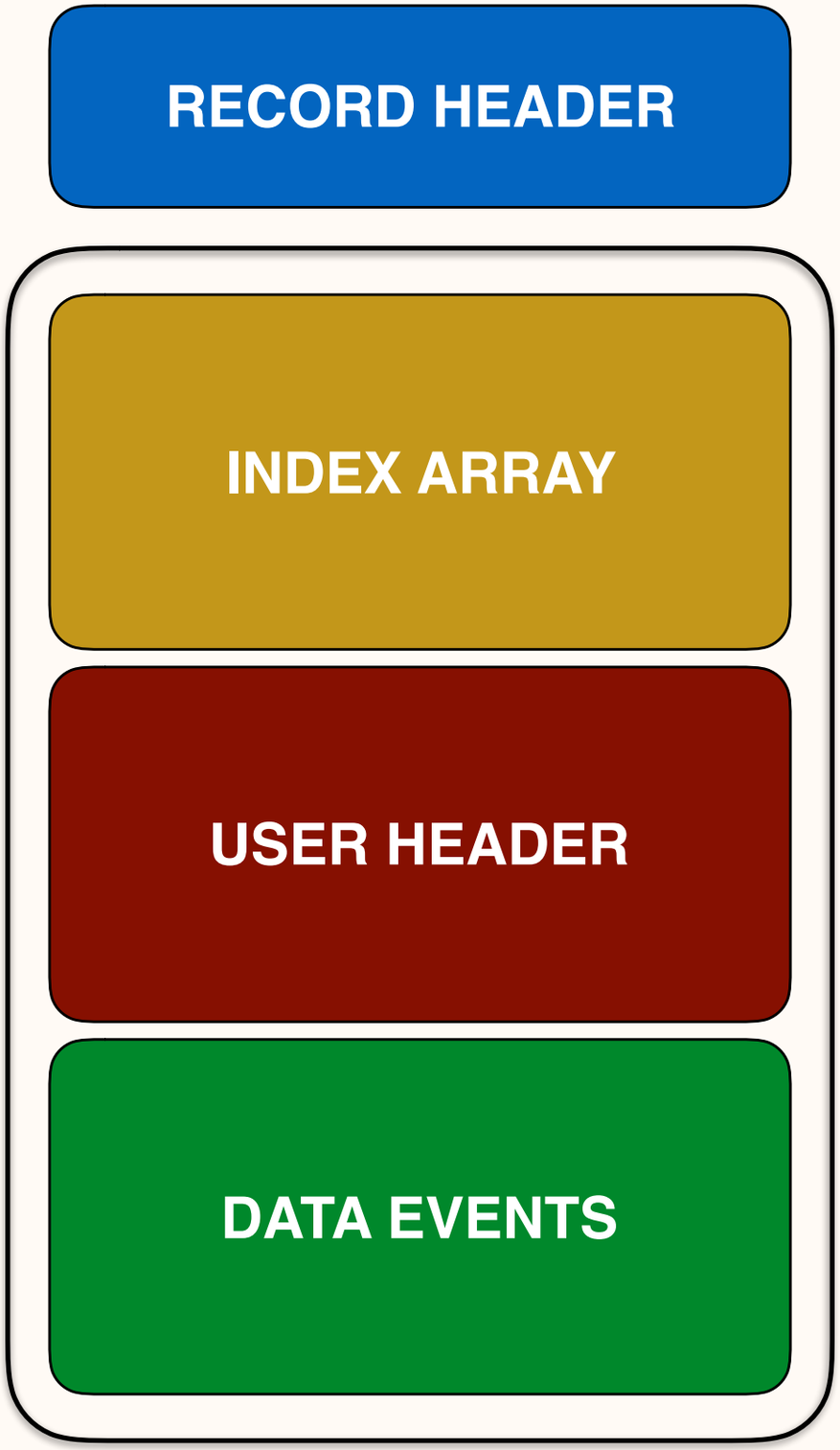
RAW	HIPO FAST	HIPO BEST
2.0 GB	**1.3 GB**	**0.85 GB**

HIPO File Structure



- **FILE HEADER**
  - ✓ Data Type (EVIO, HIPO, other ?)
  - ✓ User data (dictionary, user parameters)
- **RECORD**
  - ✓ Each record is chunk of data (compressed)
  - ✓ Data Endianness
  - ✓ Data Type stored
  - ✓ Compression algorithm (LZ4, GZIP)
  - ✓ Record version number
  - ✓ User header (any information user wants to store)
  - ✓ Index Array for each event inside of the record
  - ✓ Typically 8 MB or 16 MB presets (can be user defined)

HIPO Record Structure

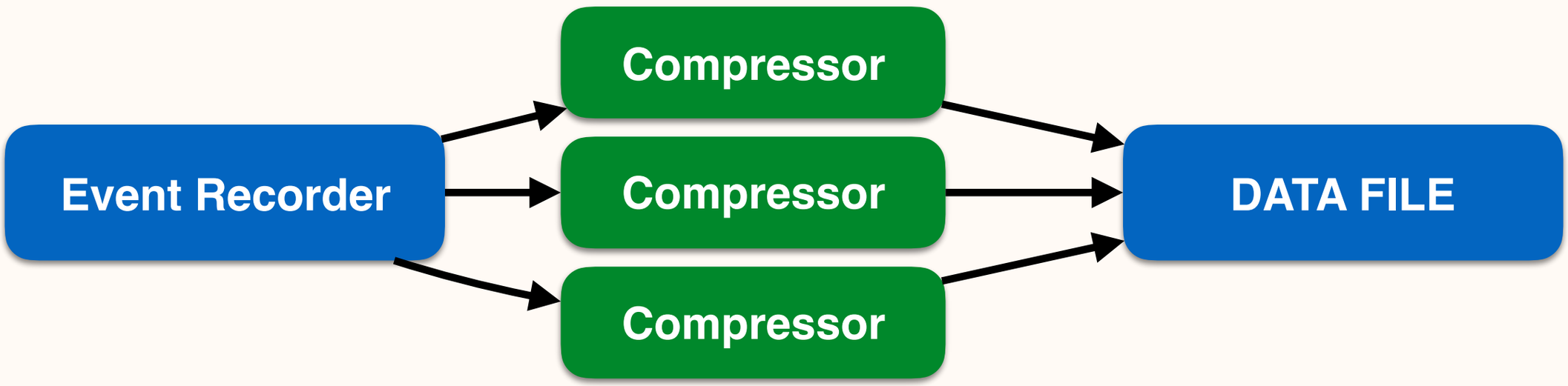


- **RECORD HEADER**
  - ✓ Record Type (EVIO, HIPO, other ?)
  - ✓ Index array Length
  - ✓ User Header Length
  - ✓ Data Length
- **INDEX ARRAY**
  - ✓ Offset of each event in DATA buffer
- **USER HEADER**
  - ✓ User defined byte array containing information in the record. Free form.
- **DATA EVENT**
  - ✓ Data event buffers back to back
  - ✓ disentangled using INDEX Array

Compression challenges

- **Compression rate**
  - ✓ different compression algorithms perform differently
  - ✓ None of them can keep up with data rates
  - ✓ Parallel compression has to be implemented to keep up with DAQ
- **EVIO 6.0 implementation**
  - ✓ JAVA parallel compression package was developed (C.Timmer)
  - ✓ Plans to implement C++ parallel compression library
  - ✓ 4-6 threads can keep up with HALL-D data rates
  - ✓ 2-4 threads can keep up with CLAS12 data rates

LZ4 FAST	LZ4 BEST	GZIP FAST	GZIP BEST
16 sec	117 sec	34 sec	176 sec
134 MB/sec	18 MB/sec	63 MB/sec	12 MB/sec



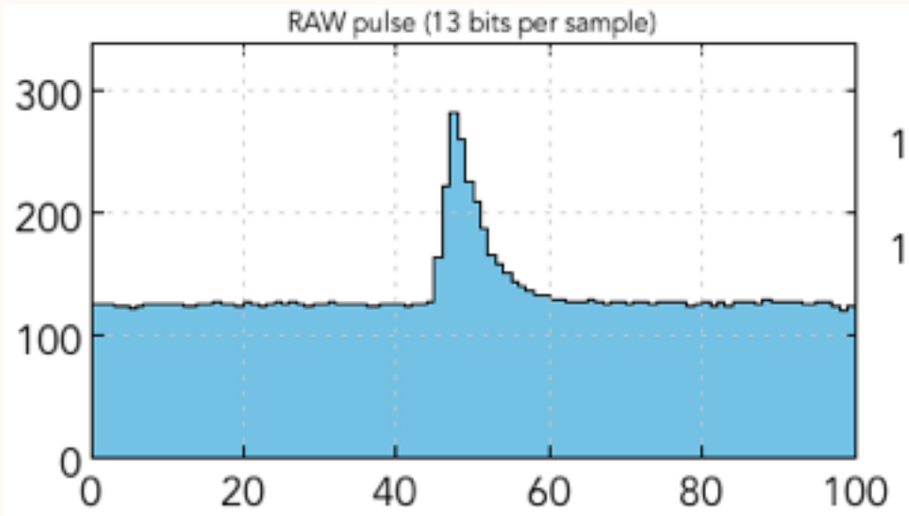
HIPO Wrappers

- **HIPO wrappers**
  - ✓ C++ implementation of HIPO is available
  - ✓ ROOT port exists for reading directly of converting to ROOT tree
  - ✓ FORTRAN wrapper is available for producing NTUPLES
  - ✓ PAW++ tools exist for analysis

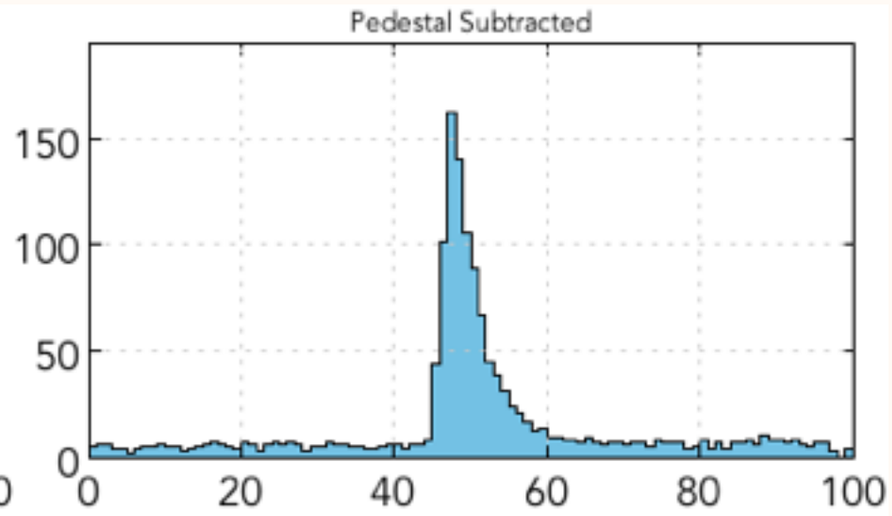
- **Further Data Reduction**
  - ✓ Pulse bit-packing algorithm was developed to reduce the size of the data in raw mode-1
  - ✓ Separation of low bits of the pulse from higher bits and encoding them separately can lead to data reduction of ~3.5x.
  - ✓ Lossy pulse packing can reduce data size by ~7x, with no significant loss of pulse precision.

Data Compression

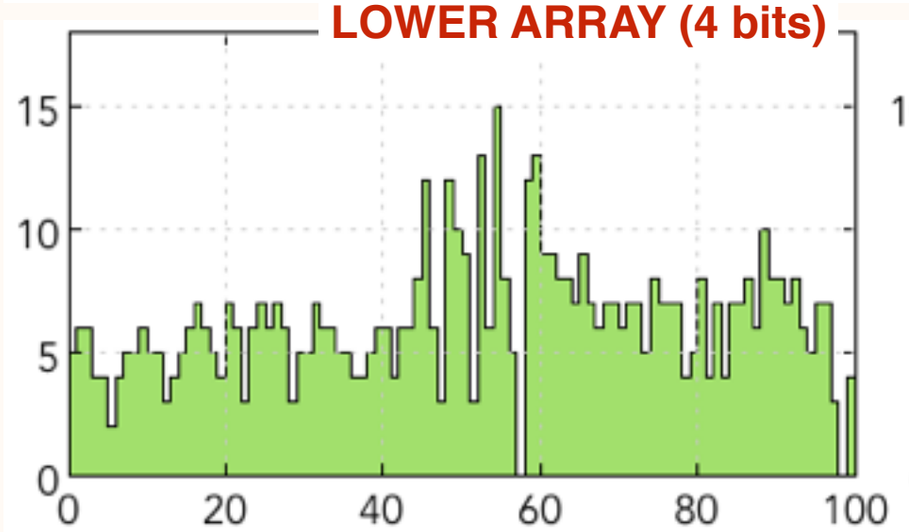
RAW PULSE (200 Bytes)



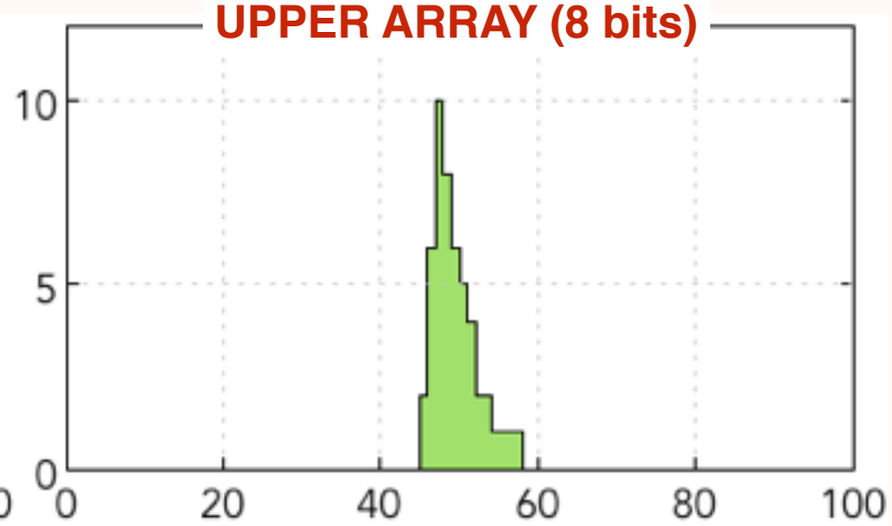
PEDESTAL SUBTRACTED



LOWER ARRAY (4 bits)



UPPER ARRAY (8 bits)



- **Pulse bit packing**
  - ✓ separating lower 4 bits and upper 8 bits of the pulse
  - ✓ encode lower 4 bits into 50 byte array (or 25 bit for lossy)
  - ✓ append non-zero elements of the upper 8 bit array to the pulse
  - ✓ further compression (LZ4) reduces data size even further

CLAS NOTE 2017-007

Method	File Size	Ratio
Raw	1.87 MB	1.00
Raw (LZ4)	0.81 MB	0.43
Raw (GZIP)	0.71 MB	0.37
Bit Packed	0.63 MB	0.34
Bit Packed (LZ4)	0.55 MB	0.29
Bit Packed (GZIP)	0.48 MB	0.26
Bit Packed Lossy	0.39 MB	0.21
Bit Packed Lossy (LZ4)	0.30 MB	0.16
Bit Packed Lossy (GZIP)	0.27 MB	0.14

- **Bit packing efficiency**
  - ✓ lossless packing provides data compression **~1/3** (**~1/5** with compression)
  - ✓ lossy packing provides compression ratio of **~1/5** (**~1/7** with compression)
  - ✓ lossy packing for average sized pulse introduces a loss of **~0.2%-0.6%** of total integral
  - ✓ storing only surrounding area of the pulse (not all 100 bins) reduces data **~1/14**
  - ✓ mode 7 stores data in **12 bytes**, lossless partial waveform **16 bytes** (**38-42** bytes for lossy)

Summary

- Common Data Format (HIPO/EVIO) for JLAB was developed for all Halls to use
- Provides compression and on fly indexing for fast random access
- Implementations of JAVA, C++ and FORTRAN exist.

- Pulse packing was investigated for future improvements on data compression.
- Can be used in CLAS for taking data in MODE-1