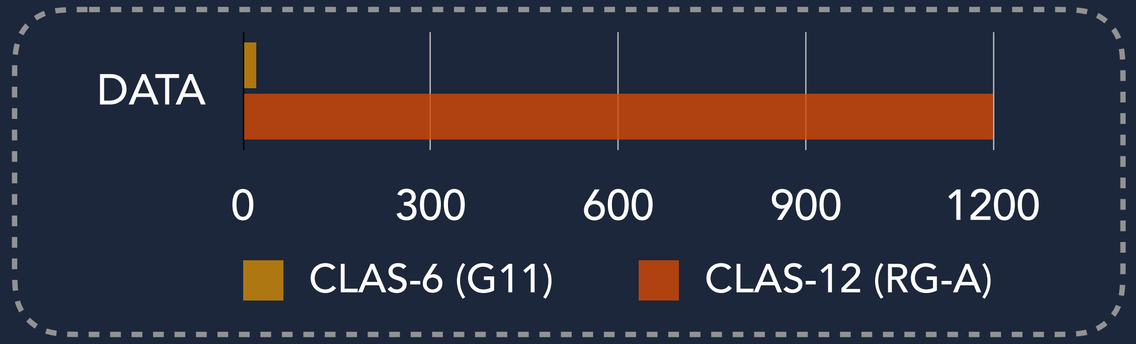
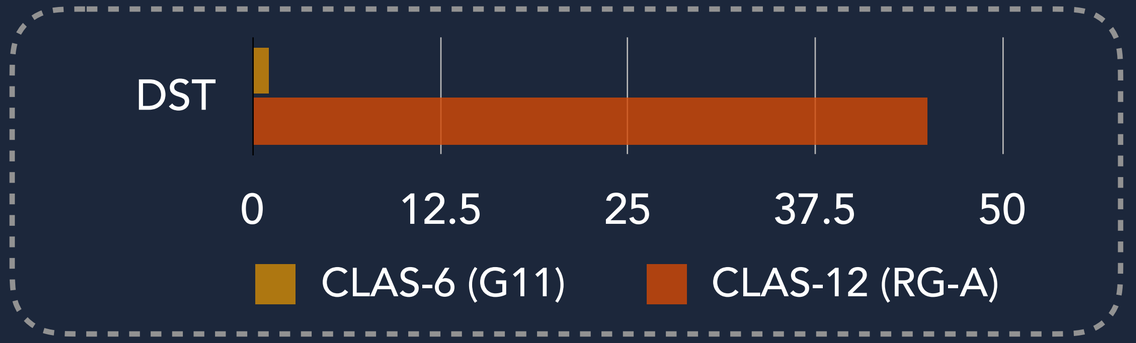
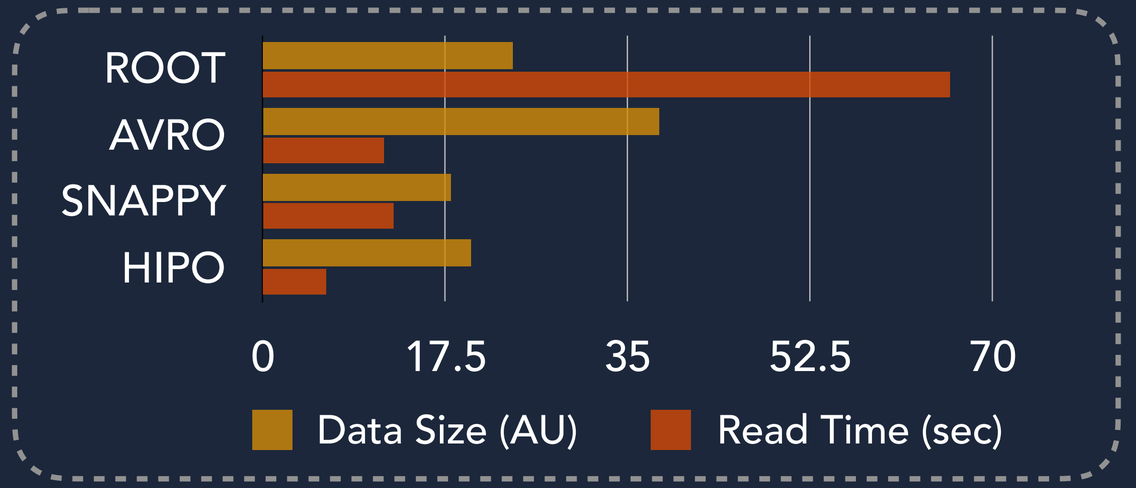
Data Processing for CLAS12

Gagik Gavalian (Jefferson National Laboratory)

CLAS 12 Detector







DATA SIZES:

- CLAS12 First Experiment collected ~50x more data than longest experiment in CLAS6 era (RGA~1.12 PB)
- Data Summary tapes are ~100x larger (45 TB) than G11 experiment (on the graph compressed size is shown)

HIGH PERFORMANCE OUTPUT (HIPO):

- HIPO data format was implemented for CLAS12
- Dictionary driven data format with Schema evolution.
- Fast compression algorithm (LZ4)
- Chunked data frame implementation to support multithreaded applications.
- Event chunk tagging and indexing to allow reading selective types of events from Data Summary Tapes.
- Deserialization performance better than most commonly used data formats in Nuclear Physics.
- Faster than industry standard data types.

Data Formats (Column Store vs Row Store)

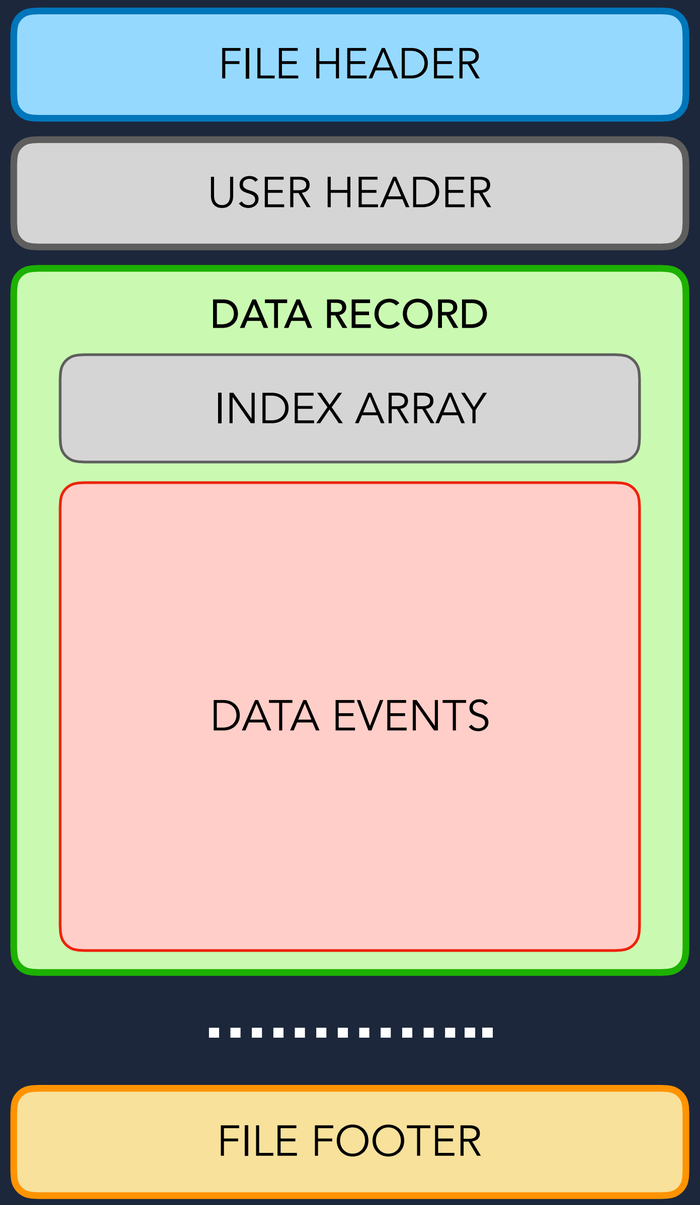
- In Row store data are stored on the disk tuple by tuple (Apache Avro, port Buffer)
- In Column store data are stored on the disk column by column (ROOT, Apache Parquet)

- Most of the queries does not process all the attributes of a particular relation.
- For example the query

```
Select c.name and c.address
From CUSTOMER as c
Where c.region=Mumbai;
```

- Only process three attributes of the relation CUSTOMER. But the customer relation can have more than three attributes.
- Column-stores are more I/O efficient for read-only queries as they read, only those attributes which are accessed by a query.

HIPO Data Format (File Format)



User Header

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

Data Record

Compressed buffer of data consisting of events and index. Record header provides number of events and the TAG for the record. Data records are typically ~8 MB.

Index Array

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

FILE FOOTER

Contains positions of every record in the file with number of events for fast random access. Also has tags for each Data Record.

Data Formats

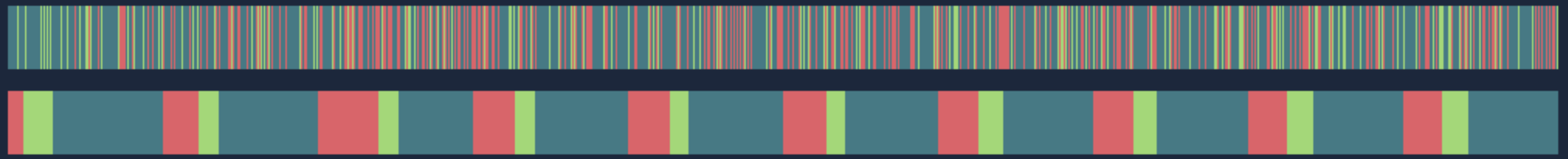
CONVENTIONAL APPROACH

- Conventional data formats write data event after event as they come out from reconstruction.
- Each event has different topology of final state.
- Not all events are useful for given analysis.
- User has to read entire data set to determine which events are useful for his analysis.
- Not suitable for big data.

NEW APPROACH

- HIPO data format is record based format with full indexed file structure.
- Records are tagged according user given criteria.
- Reading specific tags from data file is possible.
- Filtering data does not require parsing events in the records, so each skimming operation is on the level of disk IO.

EVENT TOPOLOGY FROM CLAS12 (RUN #3856)



59.7% Trigger particle is not an electron.
No electron Forward Tagger.

25.6% Electron trigger.
Forward Detector

14.7% Forward Tagger
No Electron in ECAL

Data Formats

HIPO Utilities

- HIPO comes packed with utilities program:
  - merging/splitting files
  - filtering banks for DST and Calibration set
  - prints statistics of the file
  - display content of the file event by event

- HIPO-4 will include:
  - data frame support
  - faster bank reading
  - relational bank parsing
  - conditional event filtering

    event::particle.pid==11&&event::particle.$entries>3

STATISTICS: EVENT COUNT = 1475224

name	count	rows	freq	row freq	bank size
REC::ForwardTagger	149738	288877	0.10	0.20	30.64 MB
RAW::scaler	4905	111837	0.00	0.08	1.29 MB
REC::Track	1475224	3091284	1.00	2.10	312.74 MB
REC::Event	1475224	1475224	1.00	1.00	216.66 MB
REC::Particle	1475224	9310049	1.00	6.31	470.08 MB
REC::Cherenkov	1475224	2194856	1.00	1.49	288.93 MB
REC::Traj	1475224	48895906	1.00	33.14	1.71 GB
RUN::config	1475224	1475224	1.00	1.00	154.76 MB
REC::Scintillator	1475224	8802820	1.00	5.97	619.48 MB
REC::Calorimeter	1475224	10625717	1.00	7.20	1.27 GB
REC::CovMat	1475224	3091284	1.00	2.10	380.01 MB

TOTAL SIZE = 5.40 GB

HIPO (C++)

C++/ROOT

- HIPO is readable from C++/FORTRAN
  - convertors to produce ROOT files (HBOOK files)
  - api to read from C++ code
  - analysis frameworks in C++/Python/Jupyter
  - analysis framework in ROOT (MesonX,HASPECT)

HASPECT:
https://github.com/dglazier/HASPECT6

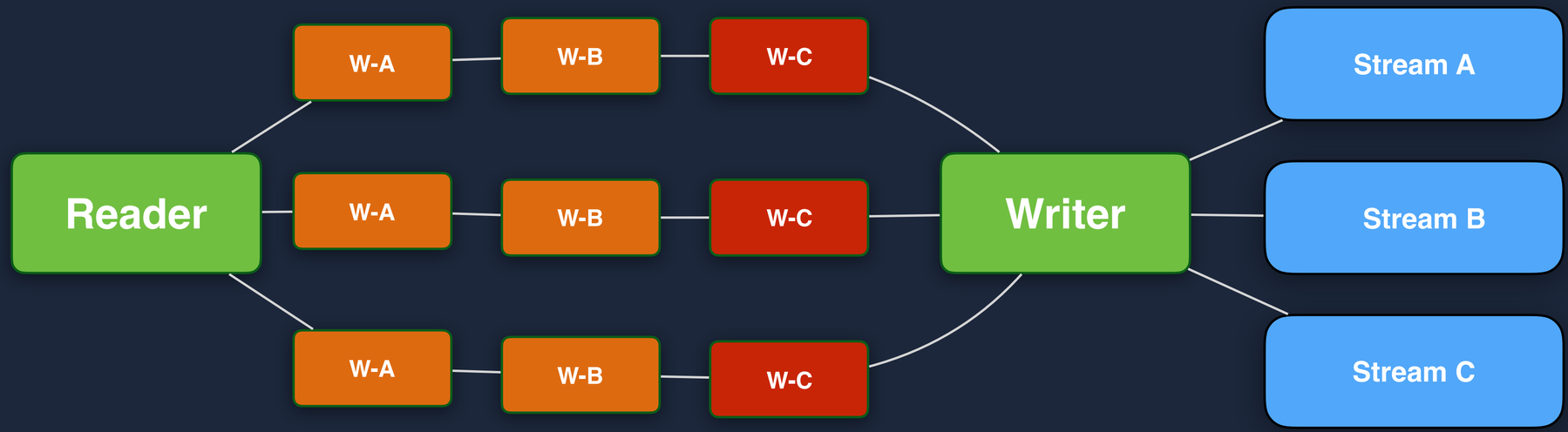
HIPOTOOLS:
https://github.com/JeffersonLab/hipo_tools

COBRA:
https://github.com/JeffersonLab/cobra12

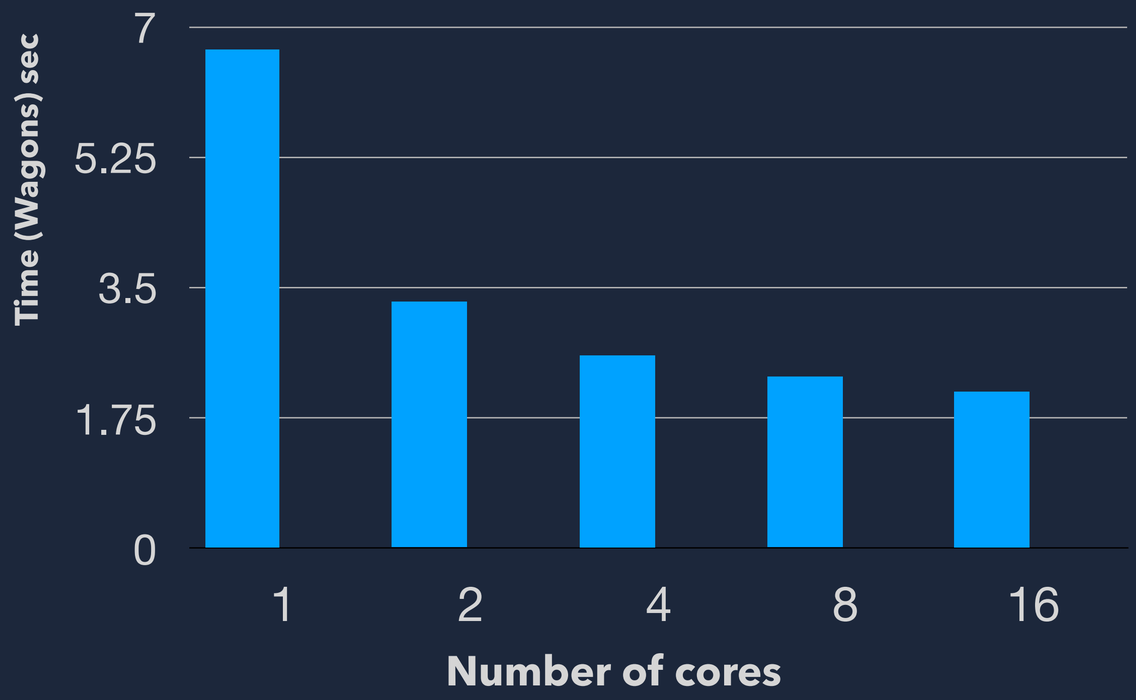
CLAS12TOOL:
https://github.com/gavalian/Clas12Tool

Data Trains

- Reader Service from CLARA Reconstruction is used
- Writer Service was rewritten to output multiple streams
- Generic Wagon service was implemented for validating event
- Standard Wagon was implemented to make particle selections



Data Trains



- With increasing number of cores performance does not scale.
- Message passing has 1ms latency on entire chain.
- HIPO reader reads and parses 300K event/s second.
- Introduces service latency when services are really fast.

- Passing single event is not efficient for fast data selection algorithms
- There is need to implement new data transport concepts for multi-threaded applications.
- Implementation is on-going, meanwhile trains are being tested on current cooked data.

Data Trains Benchmark

- The skimming speed is currently low because we feed services on event by event basis.
- The speed has nothing to do with CLARA, we have to change the way the application works.
- Several tests with HIPO4 were done to establish real operating speed for data trains
- and YES, there is such a thing as HIPO4

Skimming 16 GB DST file with two services

```
2018-07-25 18:49:20.604: Average processing time  =    0.15 ms
2018-07-25 18:49:20.604: Total processing time    = 12640.17 s
2018-07-25 18:49:20.605: Total orchestrator time  = 12644.96 s
```

3.4 Hours

Running 16 GB DST file through DUMMY service

```
2018-07-25 21:44:57.523: Average processing time  =    0.11 ms
2018-07-25 21:44:57.523: Total processing time    = 9055.96 s
2018-07-25 21:44:57.523: Total orchestrator time  = 9060.64 s
```

2.5 Hours

Data Trains Benchmark (HIPO 4)

- HIPO4 introduced new concept of Data Frames which allows speedy reading of cluster of events.
- Data Frames are individually indexed and events can be decoupled inside services.
- Data Frames will also be useful in ONLINE data rings, since it can be configured to different buffer sizes, and benchmarks show that sending/receiving data frames is more efficient
- Reader function is just to channel through the data frames to the services threads.
- Data Frames also provide non invasive modifications, such as marking events, and tagging them for output

Running 16 GB DST file through DUMMY service

```
2018-07-25 19:10:11.075: Average processing time  =    0.27 ms
2018-07-25 19:10:11.075: Total processing time    =  229.71 s
2018-07-25 19:10:11.076: Total orchestrator time  =  234.85 s
```

~4 min

Data Trains (filtering)

- Services can be configured separately for each part of the detector
- Or overall filter can be set to accept particles from entire CLAS12

```
configuration:
  services:
    PIONS:
      id: 1
      tagger: 11:X+:X-:Xn
      forward: 211:-211:2212
    GAMMAS:
      id: 2
      filter: 11:2212:22:22:X+:X-:Xn
mime-types:
```

Require electron in forward tagger
and proton and two pions in forward detector

electron proton and two photons
wherever they are detected

```
SETTING FILTER [CENTRAL] : X+:X-:Xn
SETTING FILTER [FORWARD] : 211:-211:2212
SETTING FILTER [ TAGGER] : 11:X+:X-:Xn
SETTING FILTER [OVERALL] : X+:X-:Xn
```

Full Description of filters syntax:

https://userweb.jlab.org/~gavalian/docs/sphinx/hipo/html/chapters/analysis_train.html

Path forward

- Further development of HIPO data format (requires person power).
- Development of Data Analysis Tools in JAVA for skimming.
- Improve data skimming services with flexible final state identification.

- Convert skimmed DST's to ROOT at the end of trains.
- Users have standard tools to use TTreeSelector to work with DST's
- More familiar environment for doing physics analysis and calibration.

- Collaboration has to participate in development of analysis tools.
- Will speed up development of software needed to produce publications.

Path forward

- Analysis Tools and Framework do not depend on Data Format used for DSTS.
- Good example id ClasTool (ROOT based) that was used for Nuclear Target Experiments
  - Implemented data analysis environment.
  - Abstract reader class with implementations for:
    - **BOS data format**
    - **ROOT clones array classes**
    - **NTuple 10 reader**
- We need developers who can develop analysis environment in C++, where we can plug in different data sources and have coherent analysis for all run groups.

Conclusion

DATA FORMATS

- HIPO data format provides many tools to work easily with files, filtering, merging and statistics
- HIPO-4 is being developed to address missing functionality for data streaming and indexed tagging of the file content for fast read and fast skimming.
- Big progress has been made in C++/ROOT/FORTRAN interface to HIPO, many convertors to ROOT exist, users can choose.

DATA TRAINS

- Data trains were developed and tested for the first time for spring data of RG-A
- HIPO-4 development considerably sped up the data skimming process.
- The data analysis library is being developed to improve data skimming process, combinatorics final state identification is being added, also simple cuts on kinematic variables will be implemented to further reduce skimmed data size and make all data useful for analysis.

BACKUP SLIDES