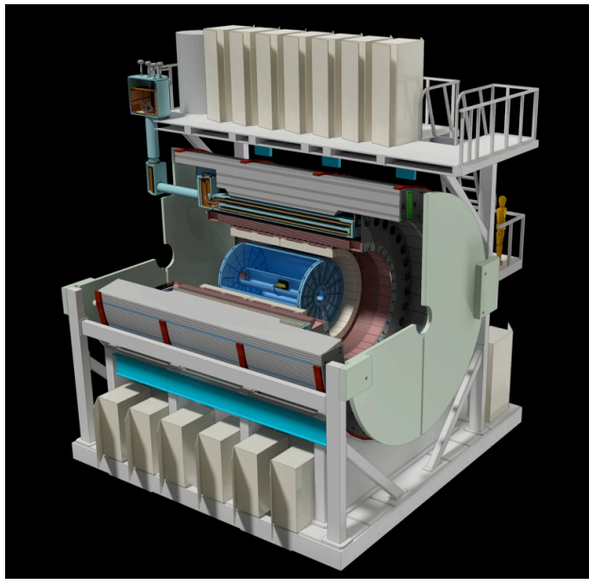


The sPHENIX DAQ System

Martin L. Purschke

BROOKHAVEN NATIONAL LABORATORY

Long Island, NY

RHIC from space

What I'll talk about

I'll give a general overview of the sPHENIX DAQ and point out where is benefits the EIC efforts

There are 2 more sPHENIX detector-related presentations here (TPC/Sampa and MVTX Thursday morning)

I'll also talk a bit about about RCDAQ and where it fits in

Why is sPHENIX important for the EIC?

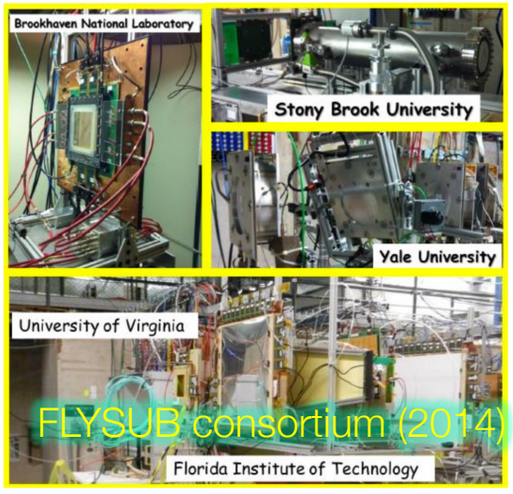
sPHENIX is paving the way for streaming readout in our community

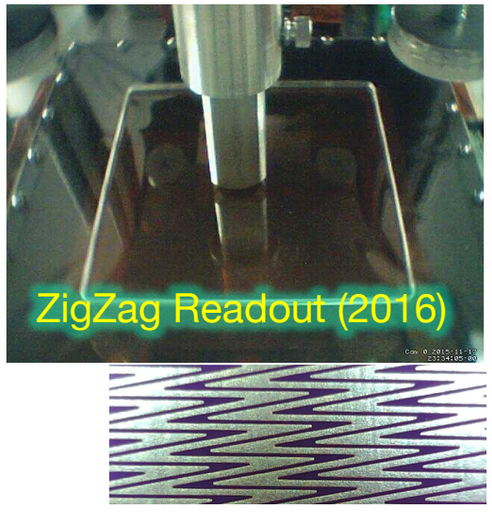
sPHENIX's IP8 location is the envisioned place for the IP8 EIC detector – ECCE that we heard about

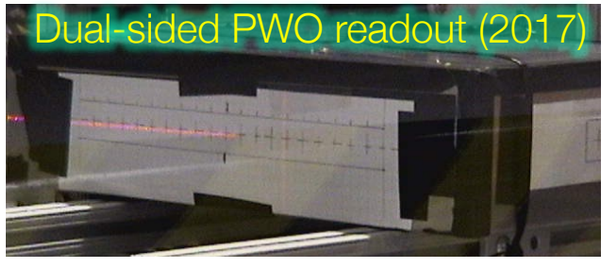
sPHENIX's RCDAQ system has been a pillar of EIC-themed data taking for R&D, test beams etc since 2013 – eRD1, eRD6, LDRDs, …

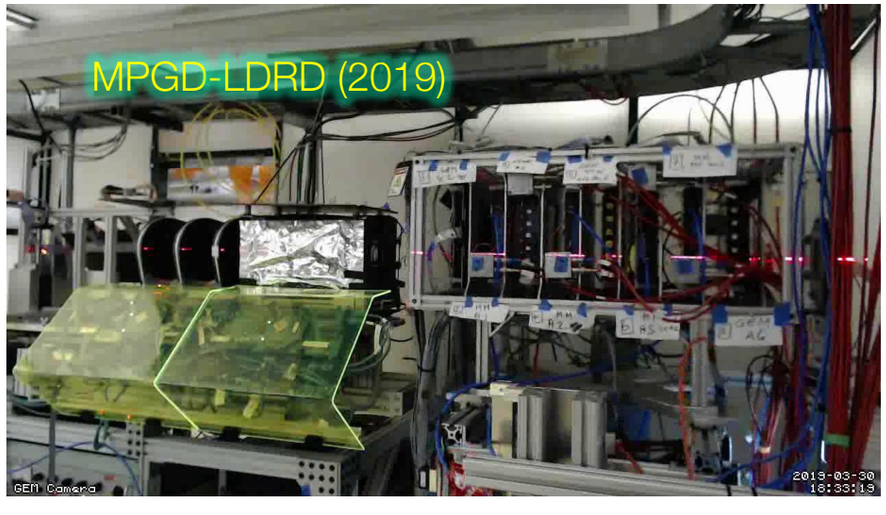
Estimated 25 active RCDAQ installations in the EIC orbit + ~30 elsewhere

Usual entry by ease-of-use for standard devices (DRS, SRS, CAEN, …) and support for fully automated measurement campaigns

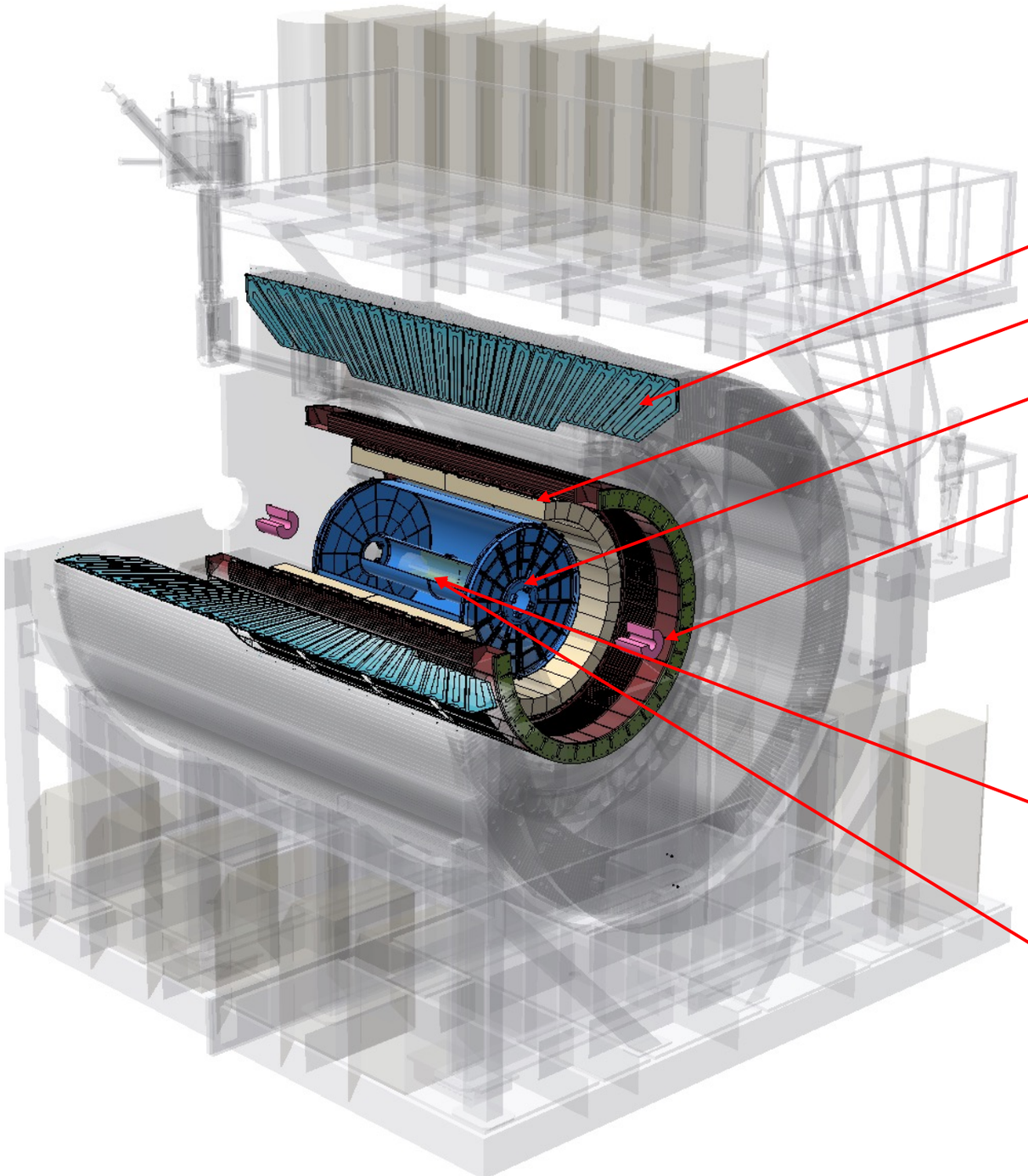




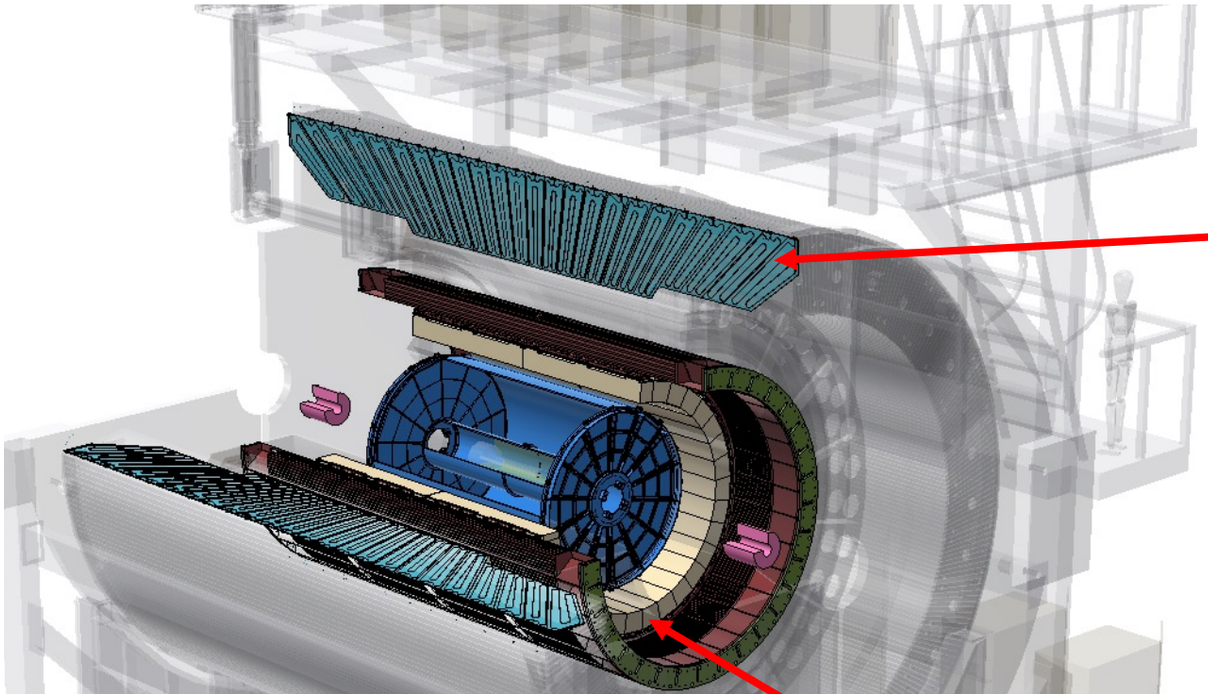


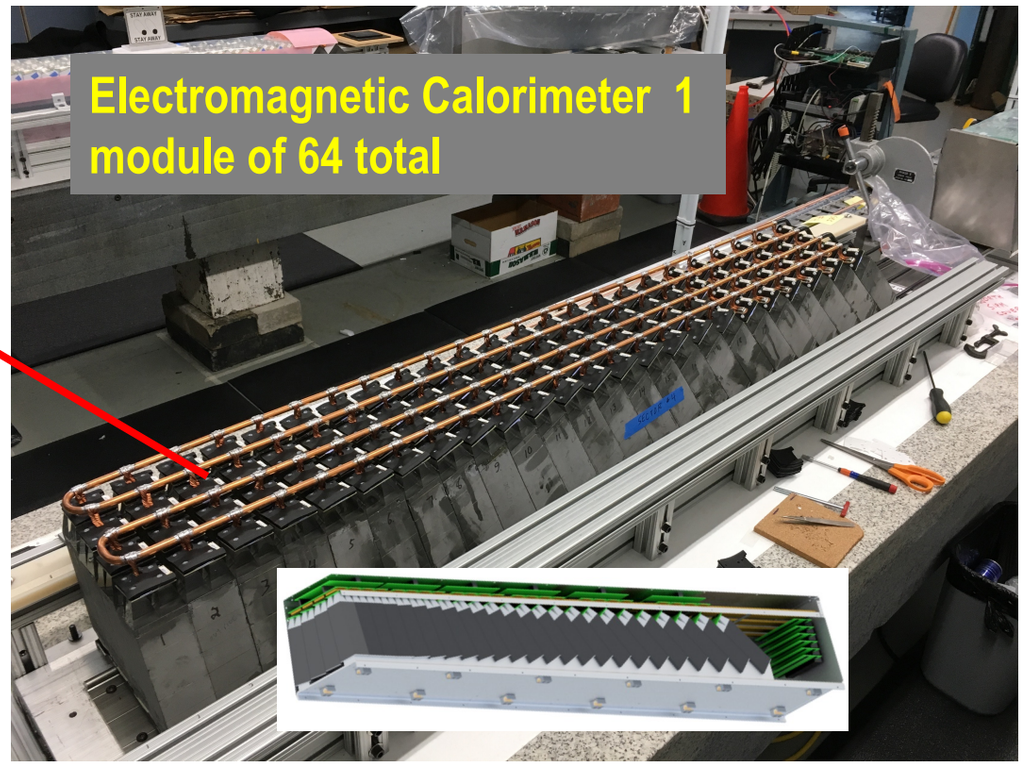


sPHENIX – the Concept

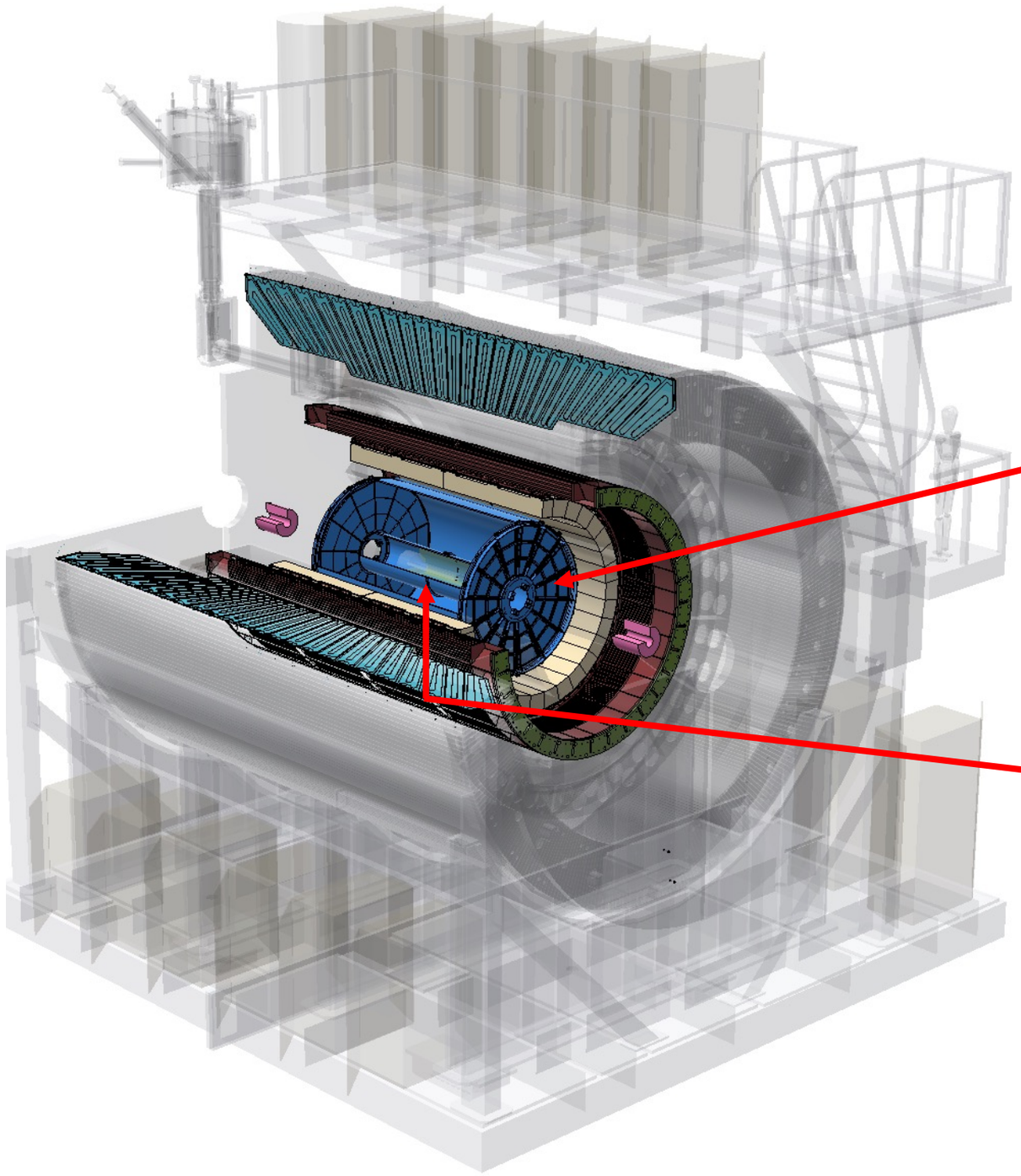


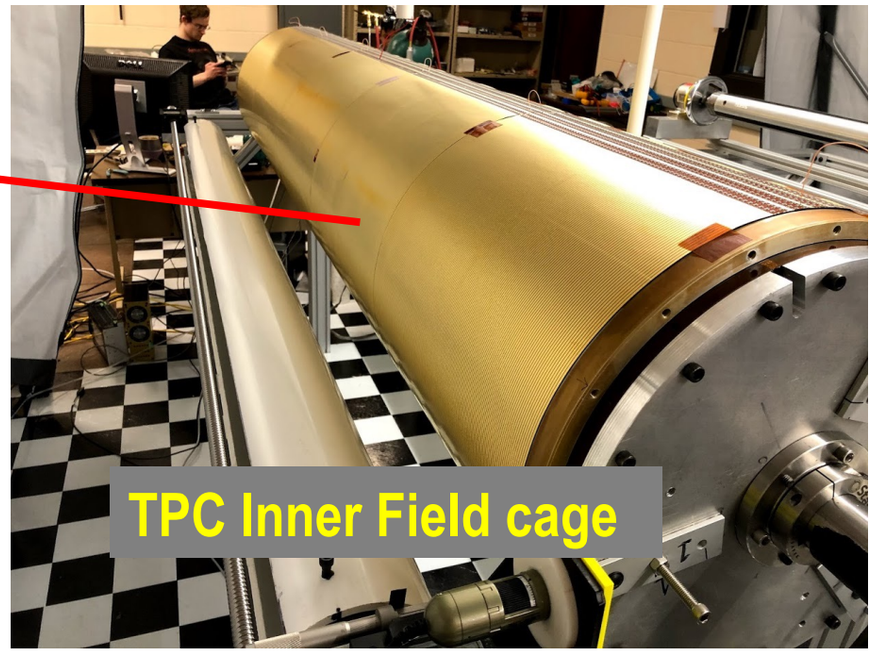
sPHENIX … getting real



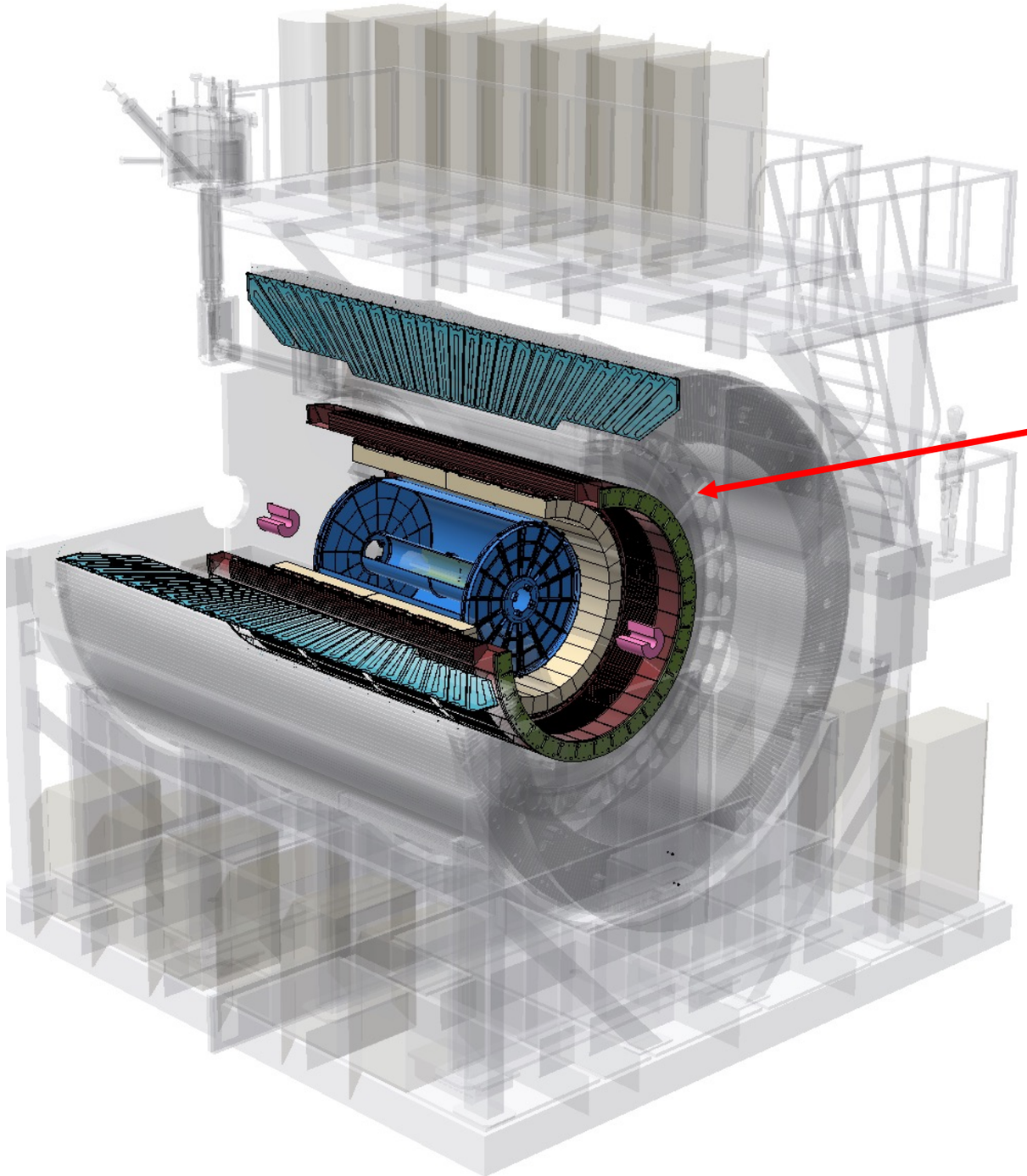


sPHENIX ... getting real





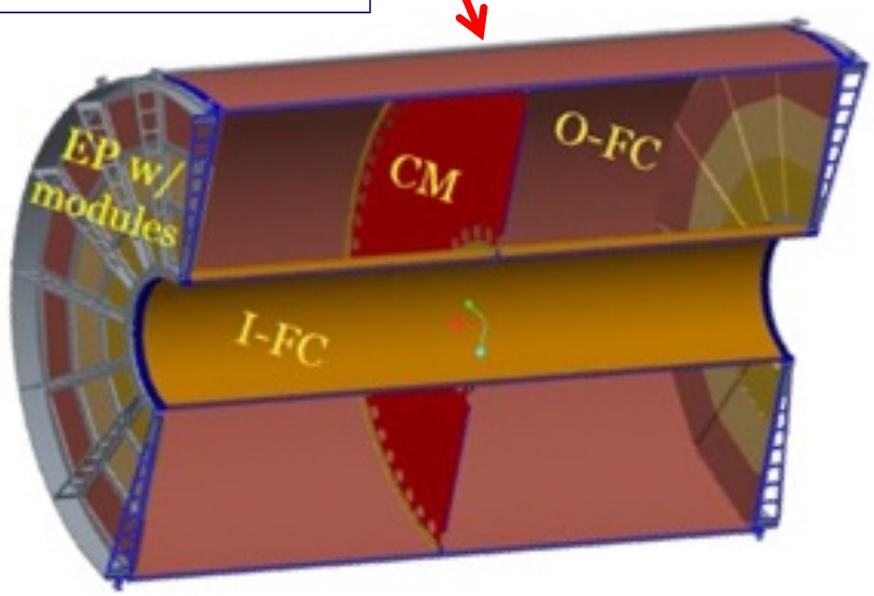
sPHENIX ... getting real



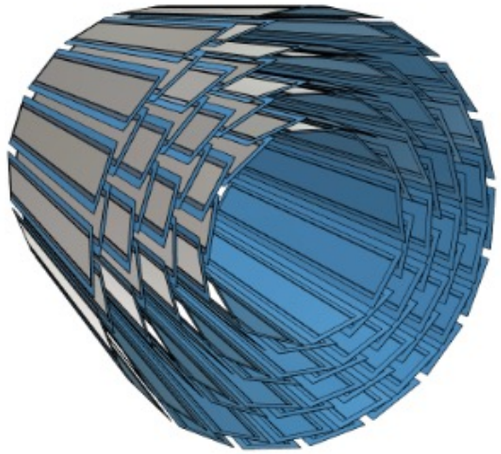
Streaming Readout Detectors

TPC

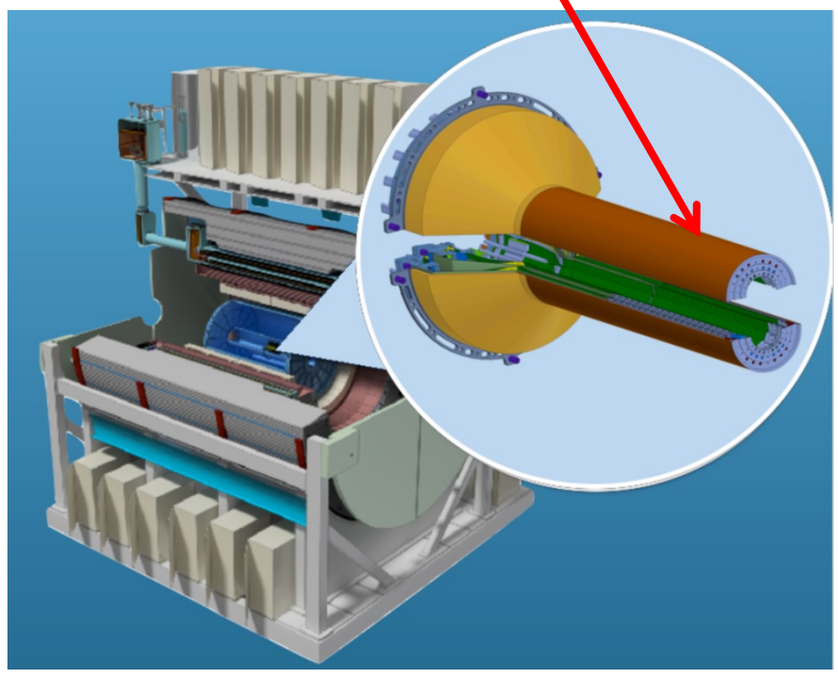
I will mostly concentrate on the TPC.



INTT



MVTX

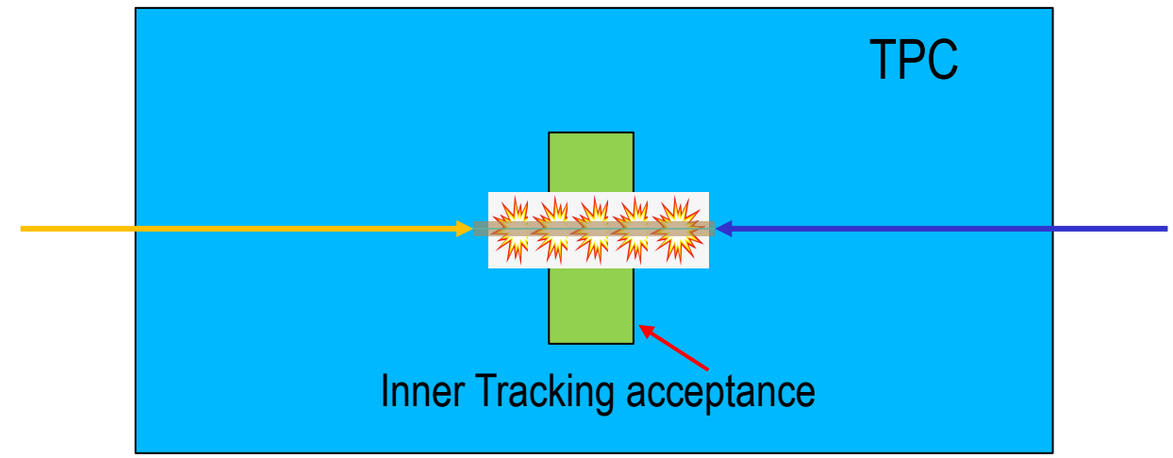


Data volumes, good news! (Updated)

Numbers taken from the beam-use proposal

Significantly lower data volumes due to the introduction of a beam crossing angle

fewer useless off-center collisions

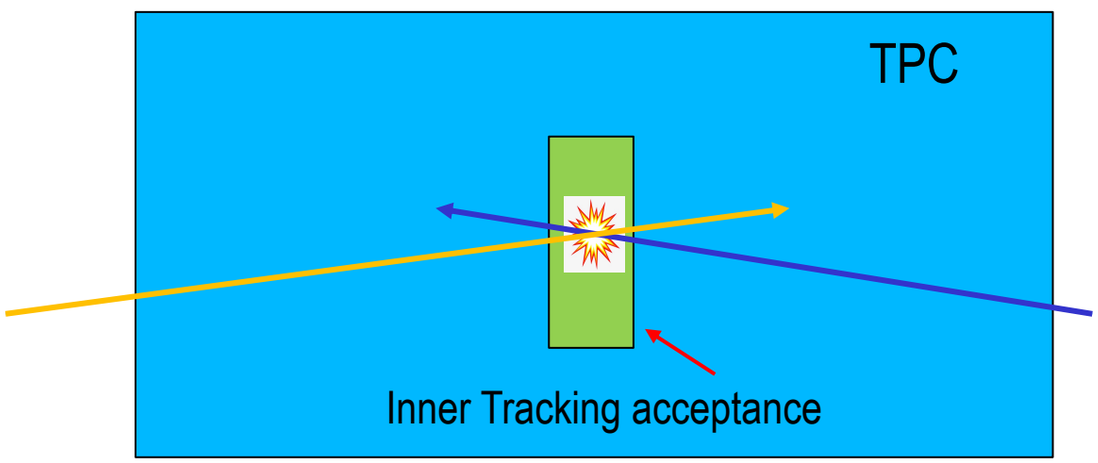


That was what we assumed initially

Lots of off-center vertices

Those still produce charge in the TPC and increases data volume

Largely useless w/o the inner tracking



Introduction of a very small crossing angle

(shown conceptually here, not so so large in reality ☺)

Reduces the amount of off-center collisions dramatically

Reduces TPC charge density and data volume

Data and more data (updated)

With the introduction of a ~2 milliradian crossing angle our old estimated **135GBit/s** average data rate becomes -

Run 1: Au+Au: 13 weeks @ 60% RHIC uptime x 60% sPHENIX uptime ⟶ 43 billion events 72 PB **73Gbit/s**

Run 2: p+p, p+A: 21 weeks @ 60% RHIC uptime x 60% sPHENIX uptime ⟶ 69 billion events 78 PB **49Gbit/s**

Run 3: Au+Au: 24.5 weeks @ 60% RHIC uptime x 80% sPHENIX uptime ⟶ 107 billion events 180 PB **97Gbit/s**

These are conservative uptime figures establishing *minimum* sampled luminosity goals

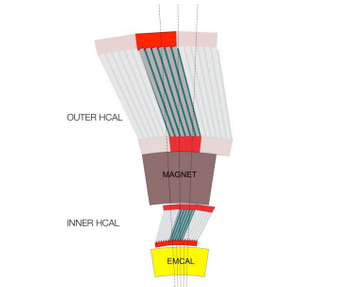
Setup / beam commissioning times rolled into that figure

We want to do a lot better!

Our planning is for the original 135GBit/s and gives a lot of headroom

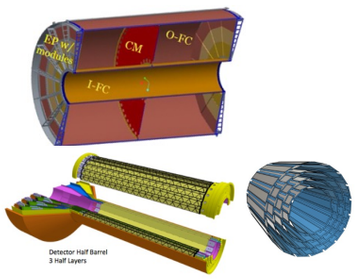
We can write up to 230PB/year if needed, so significantly higher uptimes are ok and expected.

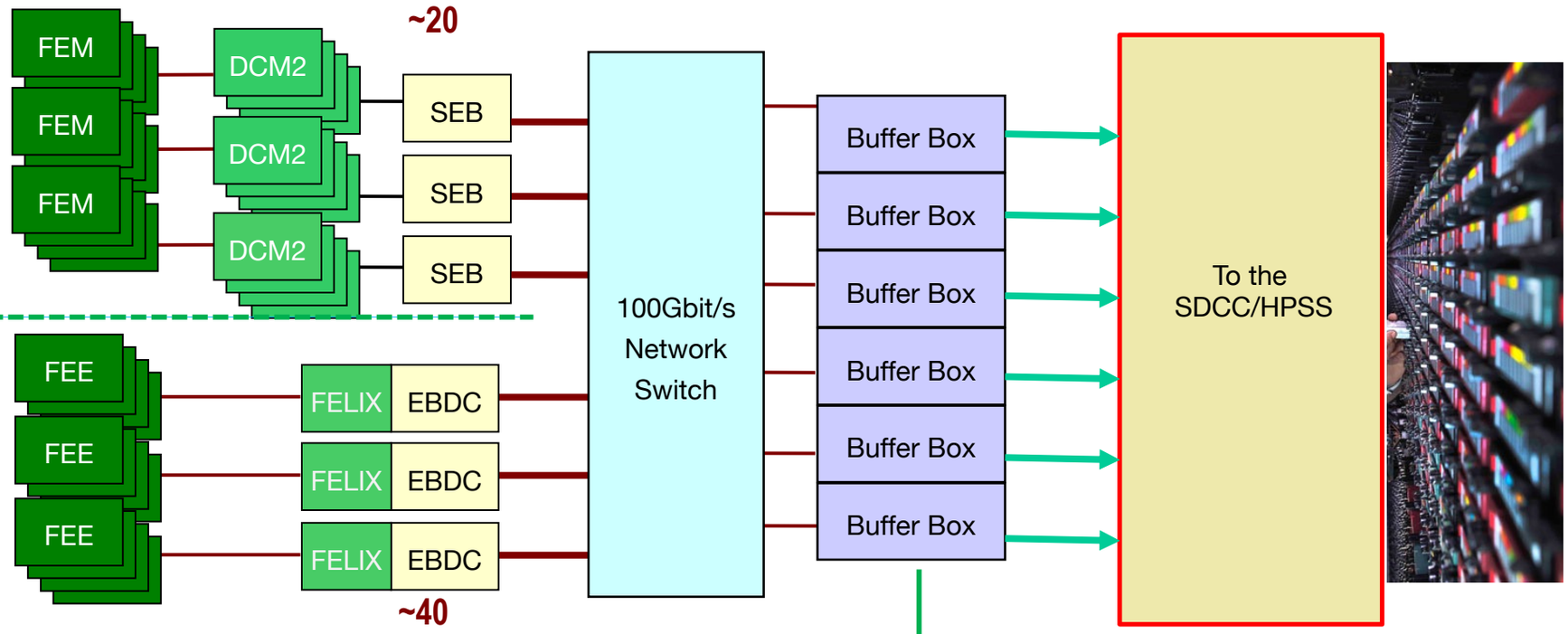
DAQ Overview



Triggered Readout

Streaming Readout

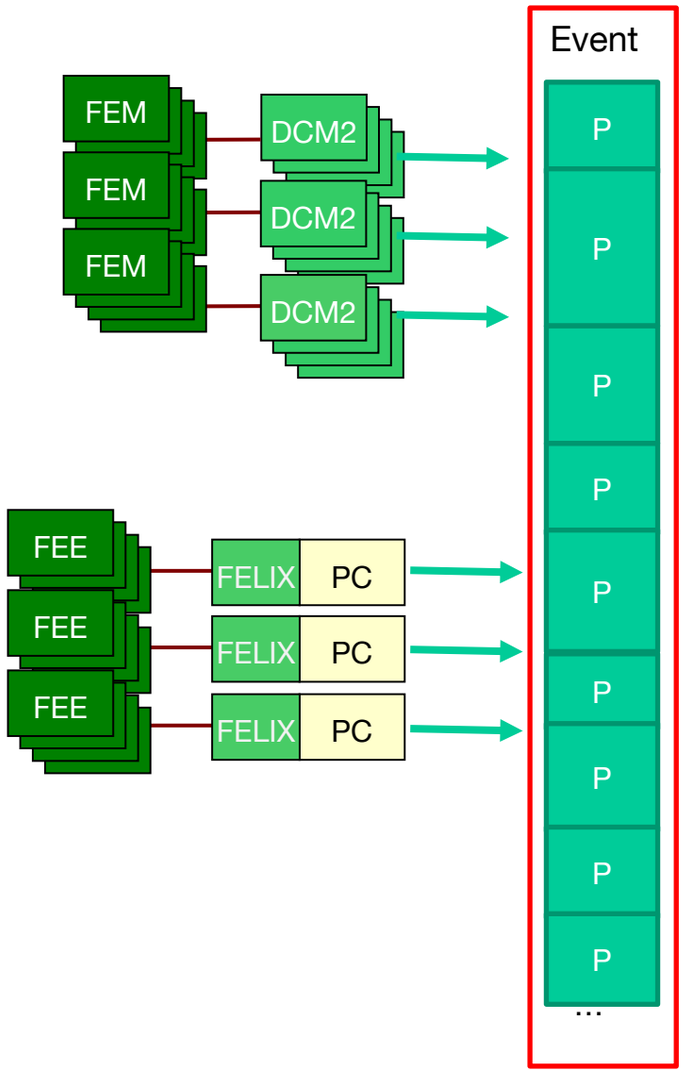




20 of 60 total SEBs / EBDCs

102 14TB disks
1 of 6

Data Structures



Each Front-End Card generally contributes what we call a “Packet” to the overall event structures

A Packet ID uniquely identifies the detector component / front-end card where it comes from

A hitformat field identifies the format of the data, und ultimately selects the decoding algorithm

We can change/improve the binary format and assign a new hitformat for a packet at any time

Insulation of offline software from changes in the online system

About 180-240 such packets in a typical sPHENIX event

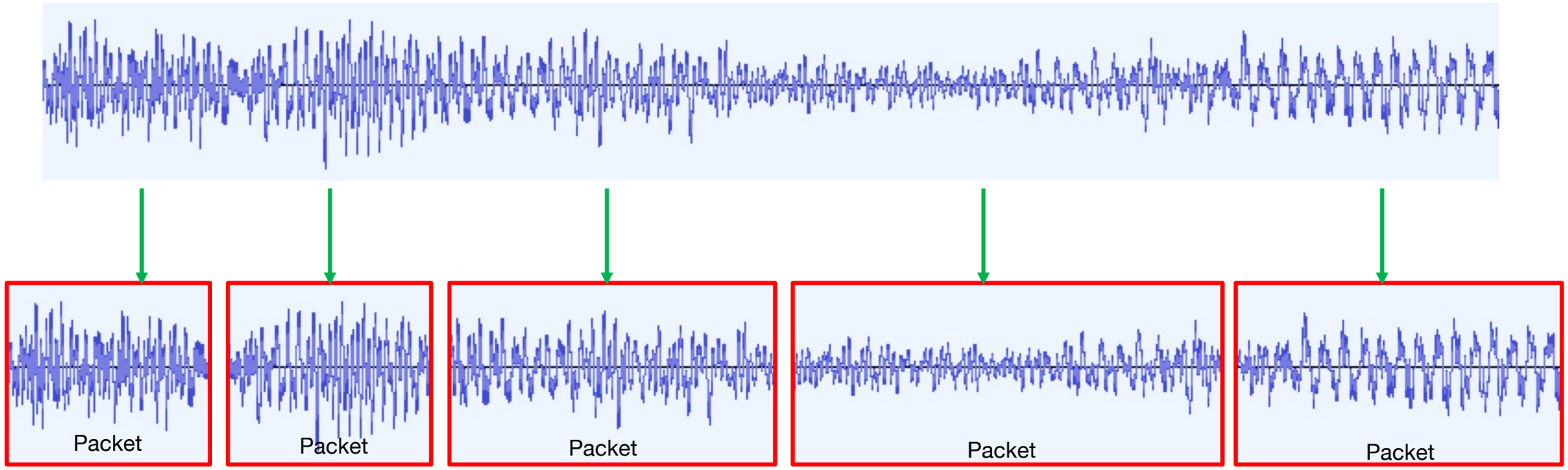
In case of triggered electronics, a packet usually corresponds to a trigger

Similar with a bit different meaning for streaming readout…

Streaming Readout and Packets

For streaming data, the “Packet” paradigm changes its meaning a bit

It becomes like a packet in the Voice-Over-IP sense - VoIP is chopping an audio waveform into conveniently-sized chunks to transfer through a network



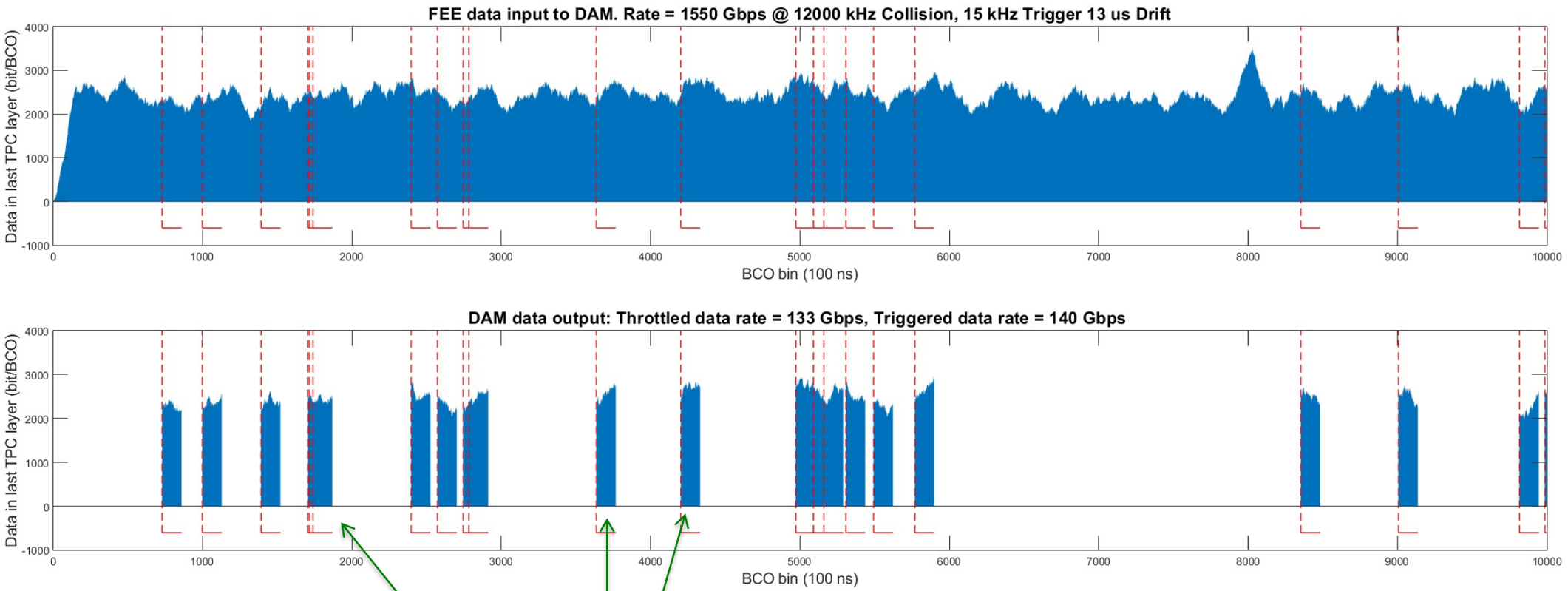
We are chopping the streaming detector data into conveniently-sized packets for storage

Here: Streaming TPC data at the Fermilab test beam

```
$ dlist  rcdaq-00002343-0000.evt -i
 -- Event     2 Run:  2343 length: 5242872 type:  2 (Streaming Data)  1550500750
Packet  3001 5242864 -1 (sPHENIX Packet)  99 (IDTPCFEEV2)
$
```

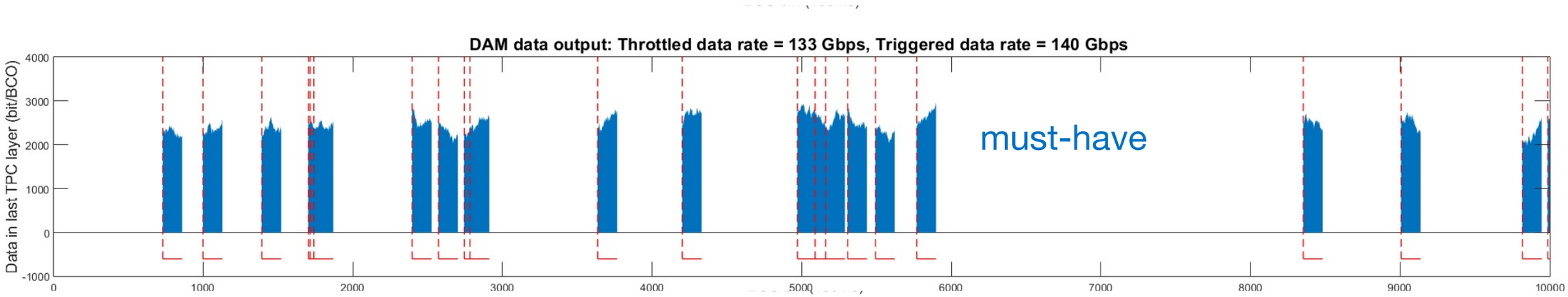
Combining “Triggered” + “Streaming”

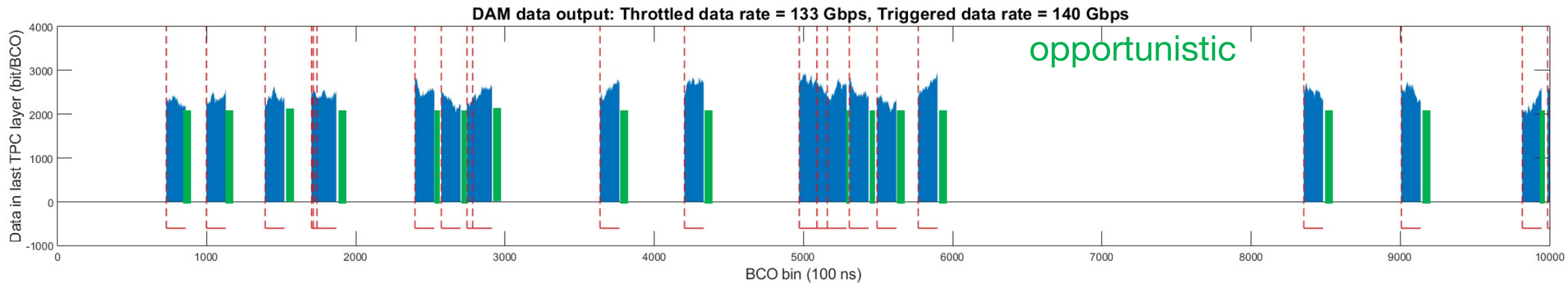
These are the “must-have” data from the entire detector



Chunks correlated with triggered events

…plus “opportunistic” tracking-only data



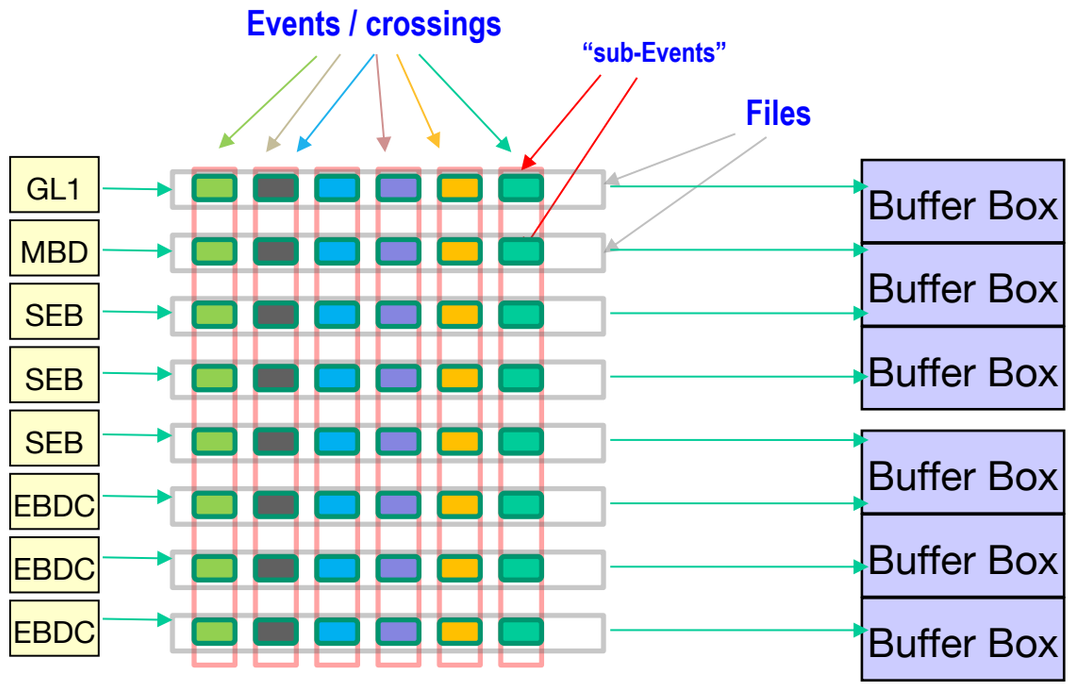


- we extend the “stream time” and add tracking-only events without the calorimeters
- We can tune that addition to “back-fill” our storage limit
- This adds more-than-proportional crossing counts for additional data:
  - ~13us – 13,000ns drift time == “waveform length” for the TPC to cover 1 crossing
  - Add 110ns (1 crossing or 0.9%) length – now you cover 2 crossings – 100% more
  - Each 110ns adds a miniscule amount to the data but an additional crossing coverage

No Event Builder

Each SEB and EBDC writes an individual file to storage, containing “sub-events” from the detector in question (hence the name SEB = “Sub-Event Buffer”)

This results in about 60 files being written concurrently, about 10 files/bufferbox



We have already demonstrated that we can either read 60 streams on the fly, or combine them into a combined file.

Streaming readout speed gain

When you have a “classic” triggered event, you accept the trigger, read out the detectors, done. Next event. One event is a well-contained thing.

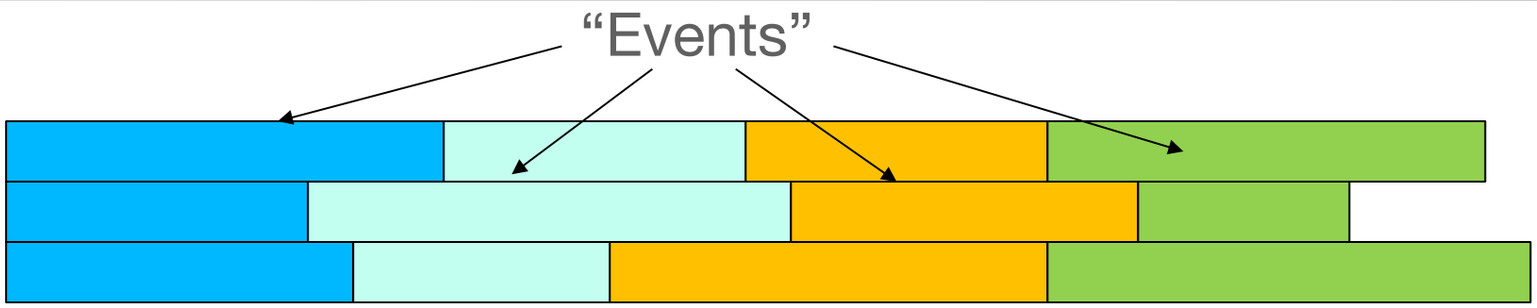
I have come to regard a particular feature of SRO as the defining property, even if you ultimately trigger your front-end:

There is no synchronized end to a given event!

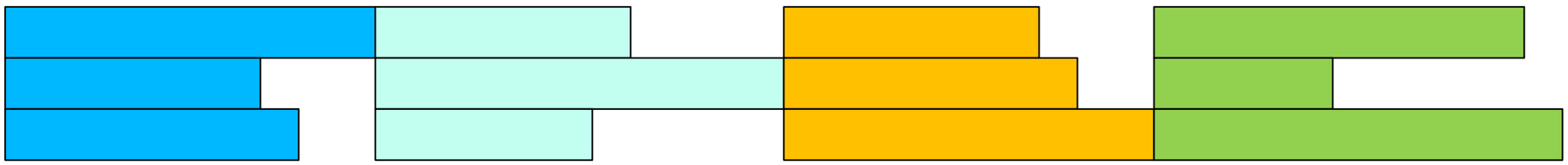
While “event” *n* is streaming, in other places, event *n-1* (or *-2*) isn’t finished yet

And that’s where the speed increase can be significant even for “classic” systems

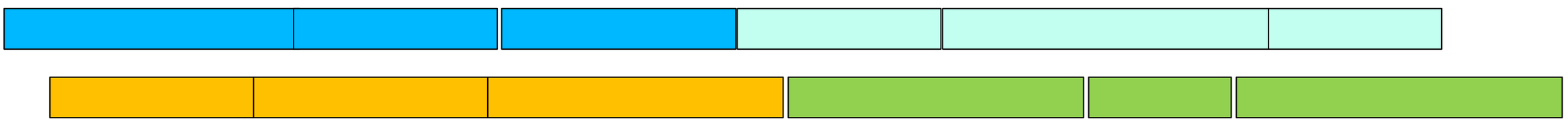
How do you deal with that?



You could throttle your event rate with busies to not let that happen:



Or, if you insist on “event boundaries” in your data, you could buffer those event fragments in DAQ memory, assemble them, then write out

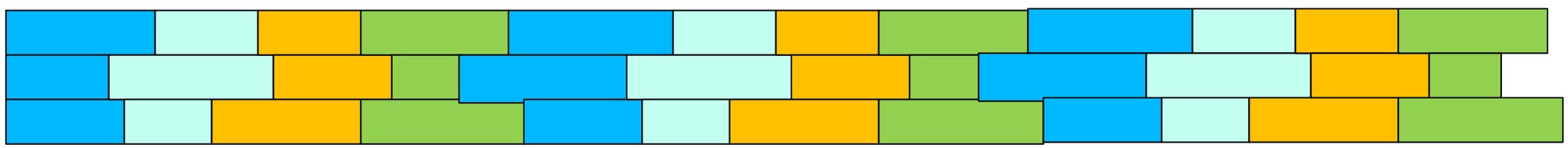


Remember that you can often not “ask” a device to give you its data when it’s convenient for you, you need to be ready to catch them as they come (e.g. network)

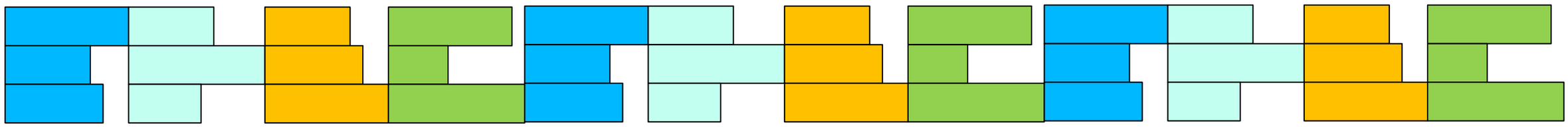
You have brought the event builder back, have to do it online, have to do it right the first time…

Offline sorting streaming data into events/crossings

A “streaming data” offline **pool** holds a number of crossings worth of data (like 1000) and sorts them by crossing number (beam clock value)



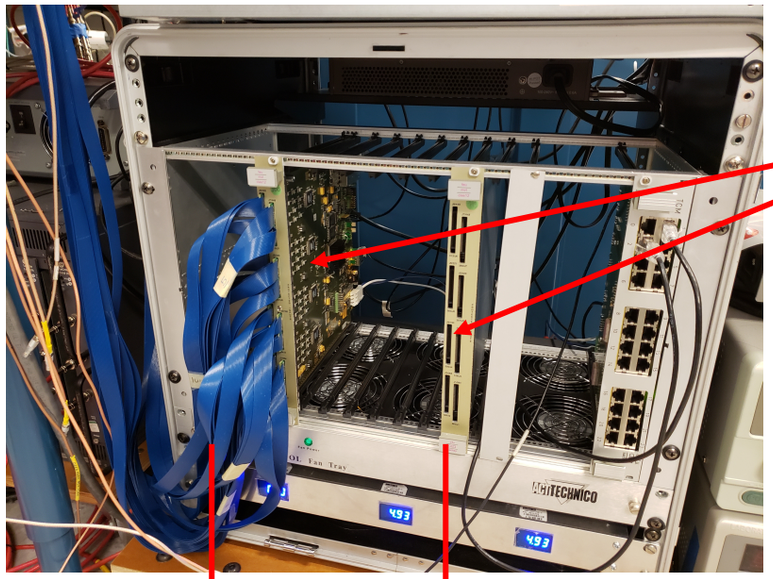
It then hands out per-crossing data:



As processed data (oldest crossings) get discarded, new data are inserted (high-and low-water marks)

By doing that you relieve the DAQ from a lot of bookkeeping tasks. Faster!

A good example: Saclay “DREAM” electronics



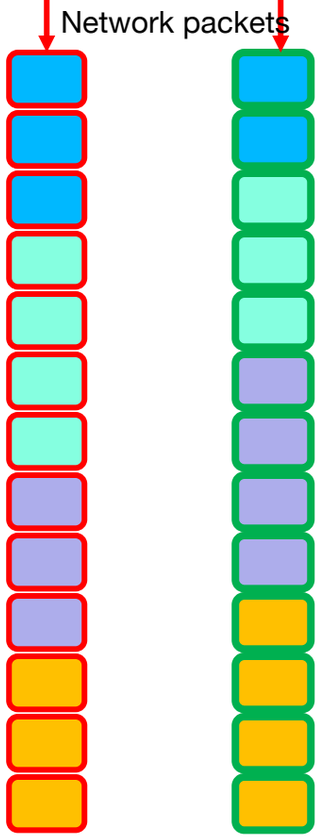
DREAM ASIC to read out MPGDs etc

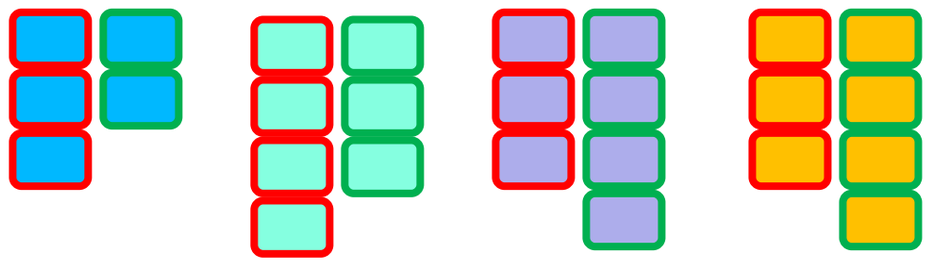
“Front-End-Unit” FEUs

The FEUs send data packets on the network

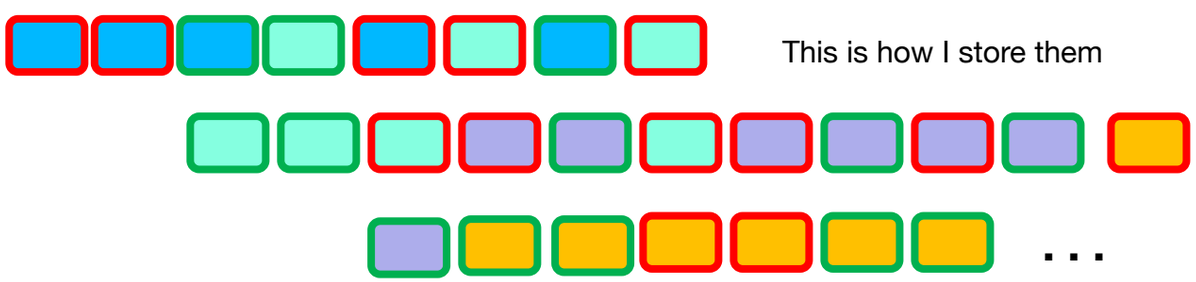
You need to catch them as they come (two sources shown here, but there can be many more)

The reference readout implementation stores the packets from different FEUs in different files and so implements simple “streaming”





This is what belongs together



By just “storing them as they come” I save all the decision-making where what data needs to go
All sorting done offline

Of utmost importance – the correct clock value

A “trigger number” by and large uses its meaning with SRO

The clock number (crossing number, what have you) is what really matters

We had to demonstrate the “we can do it”.

Problem: our legacy calorimeter digitizers don’t receive/know an absolute beam clock value, they count independently

Ok, you can see that they are consistent among themselves… but how about the rest of the system?

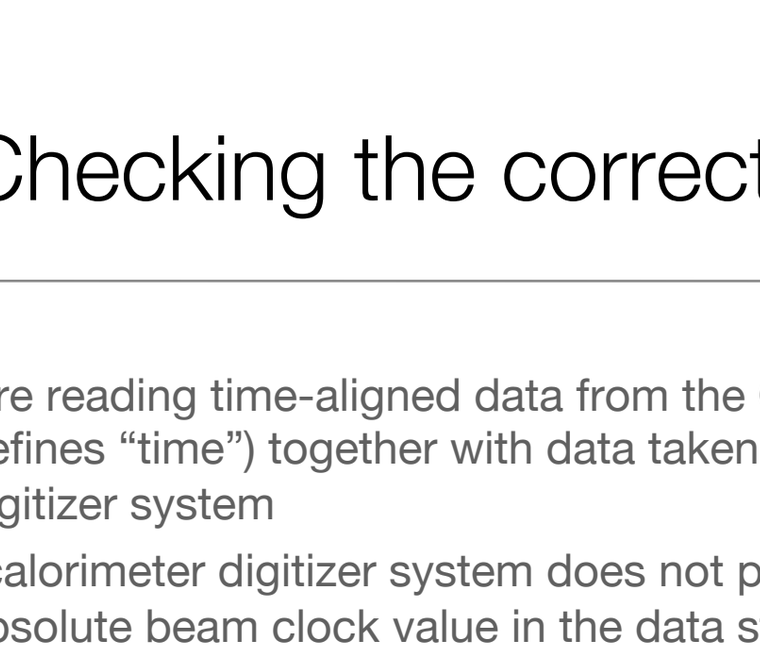
Here’s what we did -

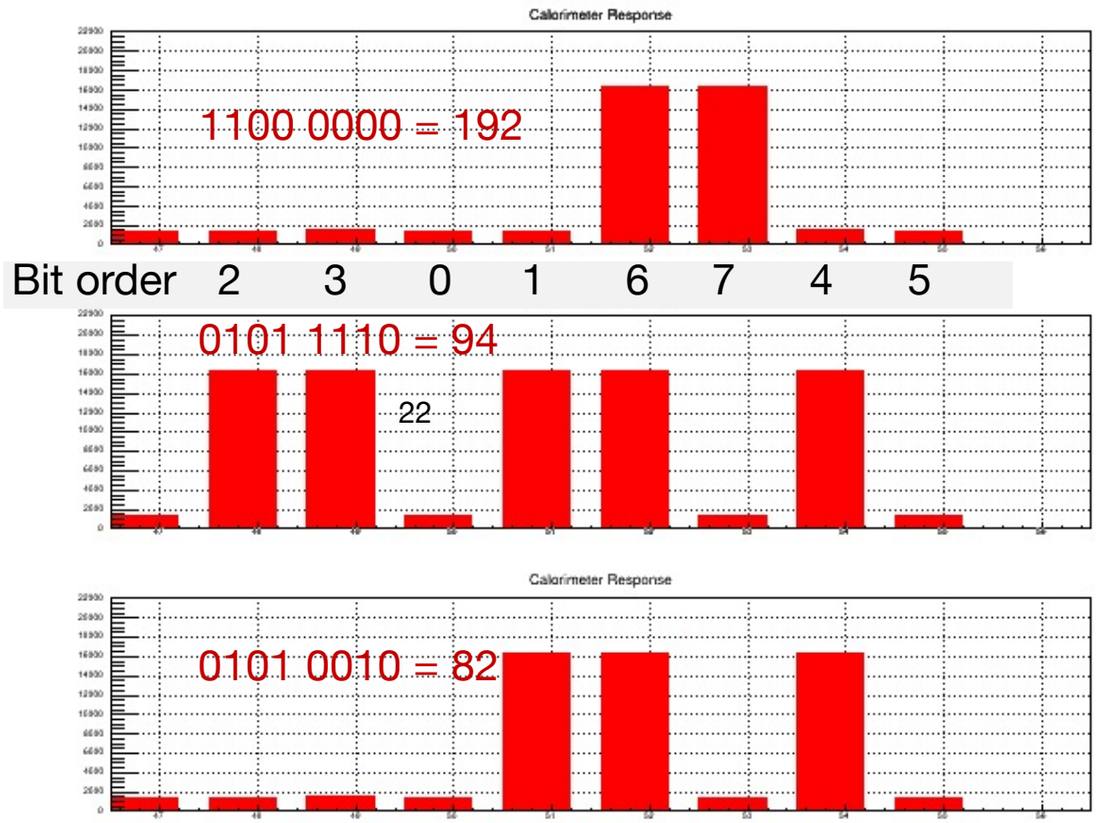
Checking the correct clock value

We are reading time-aligned data from the GL1 (that actually defines “time”) together with data taken with the calorimeter digitizer system

The calorimeter digitizer system does not provide the original absolute beam clock value in the data stream

We enlist 8 digitizer channels to digitize 8 bits of that counter, brought out on a GL1 header, connected to digitizer inputs





With 8 bits we get 0…255 to compare with the full counter.

You will see the 192 - 94 - 82 sequence on the next slide

Checking the correct clock value

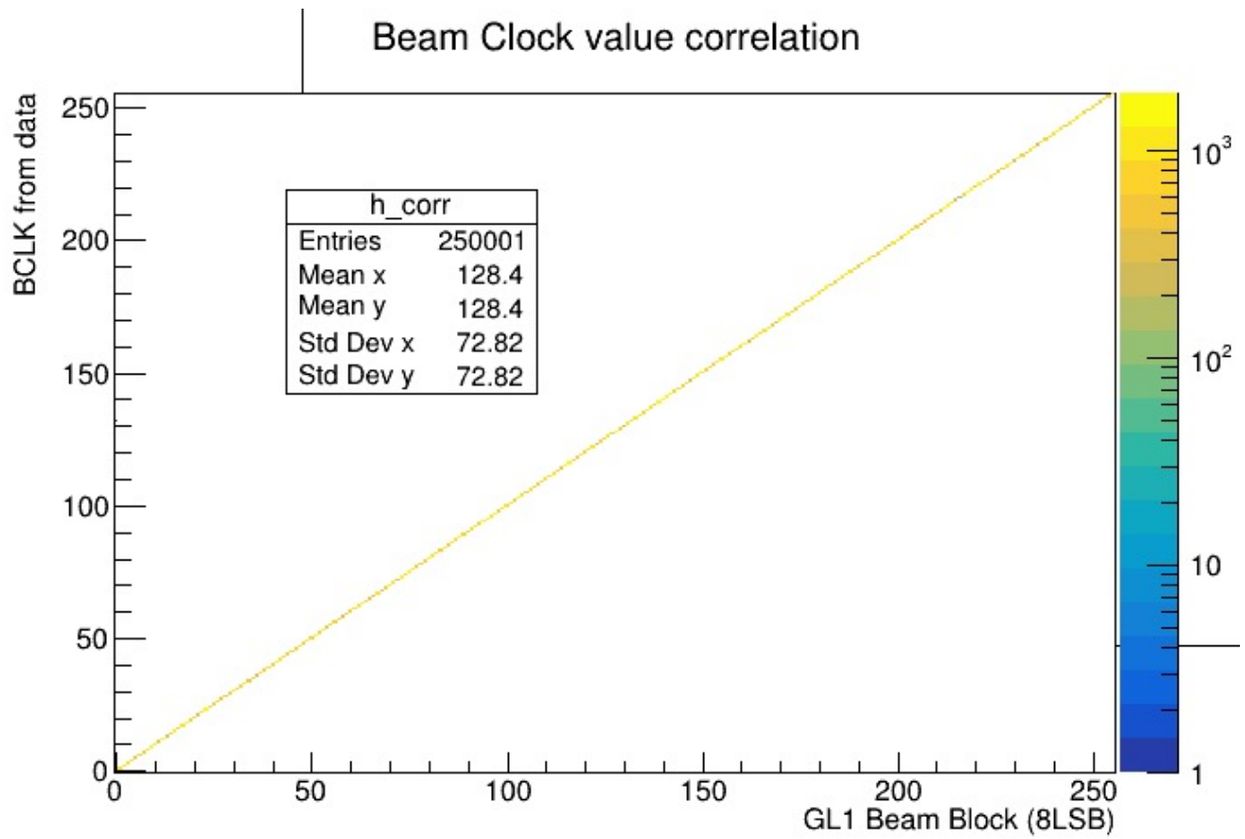
We are reading the 2 systems' data together and compare the numbers (within the 8 bit limit)

Beam Clock from GL1

evt nr	Bclk
2	192
3	94
4	82
5	242
6	66
7	94
8	181
9	41
10	71
11	12
12	57
13	119
14	88
15	84
16	191
17	158
18	237
19	235
20	175
. . .	

Beam Clock from data

evt nr	Bclk
2	192
3	94
4	82
5	242
6	66
7	94
8	181
9	41
10	71
11	12
12	57
13	119
14	88
15	84
16	191
17	158
18	237
19	235
20	175
. . .	



Summary

sPHENIX is on a good track to taking data in 2023

We got a jump start taking streaming data last year, good

We have demonstrated that we can properly align and combine the data

We have a good concept combining triggered and streaming readout in the experiment.

Thank you!

This page is intentionally left blank

(The End)