ML Acceleration with Heterogenous computing for big data Physics

Philip Harris(MIT)

Fermilab

Burt Holzman
Sergo Jindariani
Benjamin Kreis
Mia Liu
Kevin Pedro
Nhan Tran
Aristeidis Tsaris

University of Toronto

Naif Tarafadar
Paul Chow

UCSD

Javier Duarte

Massachusetts Institute of Technology

Jeff Krupa
Dylan Rankin
Sang Eon Park

University of Washington

Scott Hauck
Shih-Chieh Hsu
Matthew Trahms
Dustin Werran

UIC University of Illinois at Chicago

Zhenbin Wu

Illinois

Mark Neubauer
Markus Crisziani

Microsoft

Suffian Khan
Brandon Perez
Colin Versteeg
Ted W. Way
Andrew Putnam
Kalin Ovatcharov

CERN

Vladimir Loncar
Jennifer Ngadiuba
Maurizio Pierini

Columbia University in the City of New York

Giusppe Di Guglielmo

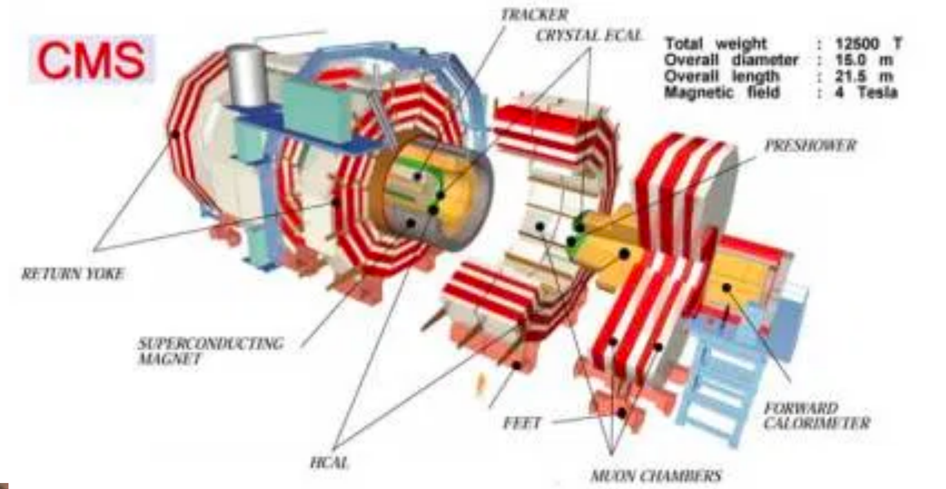
The LHC

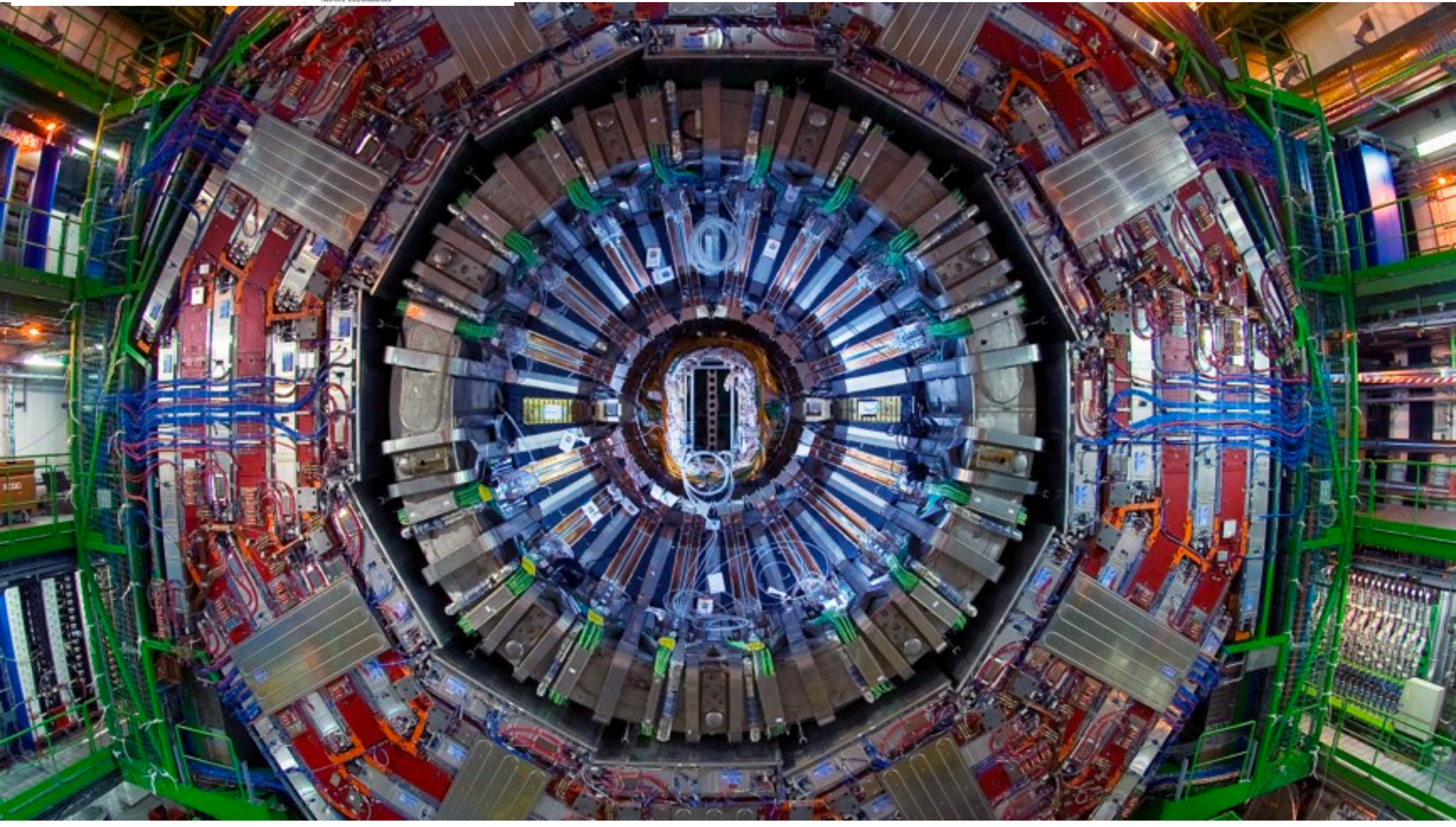

LHCb
ATLAS
CERN Meyrin
CERN Prévessin
PS 628 m
SPS 7 km
SUISSE
FRANCE
CMS
ALICE
LHC 27 km

The LHC

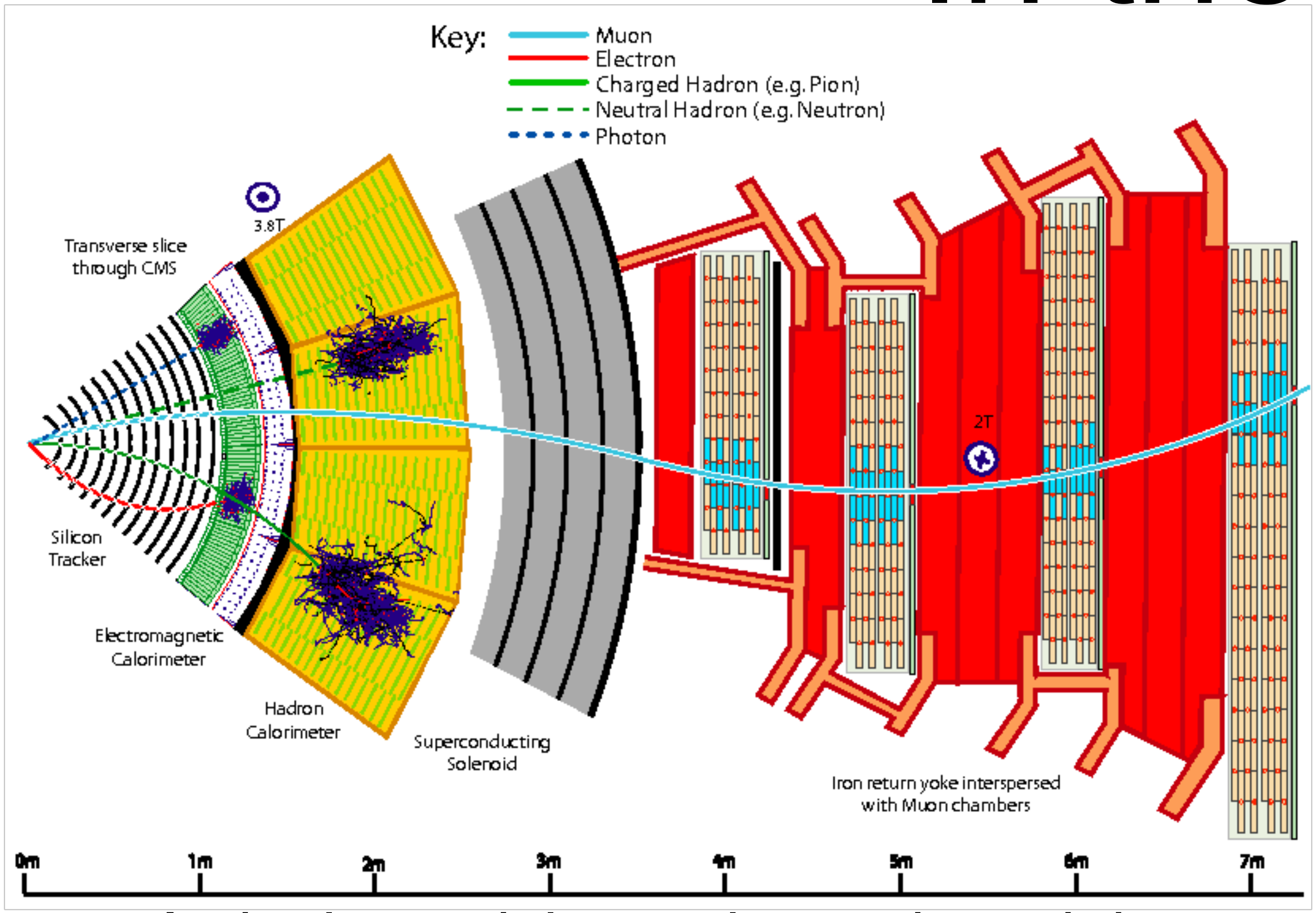

Mt. Blanc
Here is where we take data
LHCb
ATLAS
CERN Meyrin
PS 628 m
CERN Prévessin
SPS 7 km
SUISSE
FRANCE
ALICE
CMS
LHC 27 km

CMS detector





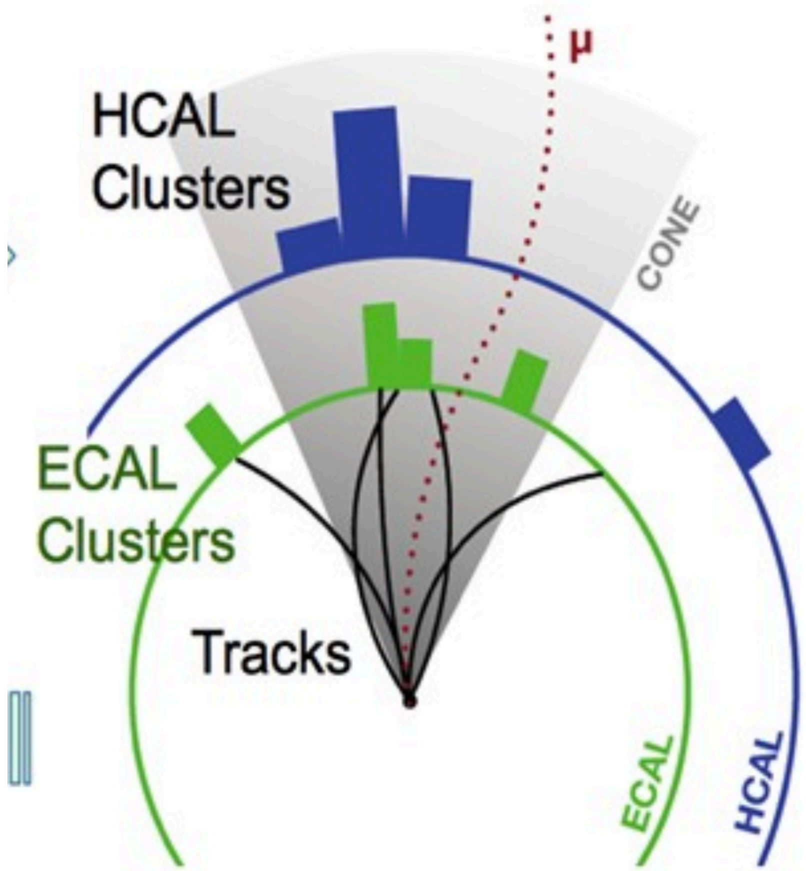
In the detector



All reconstruction is separated on an event by event level

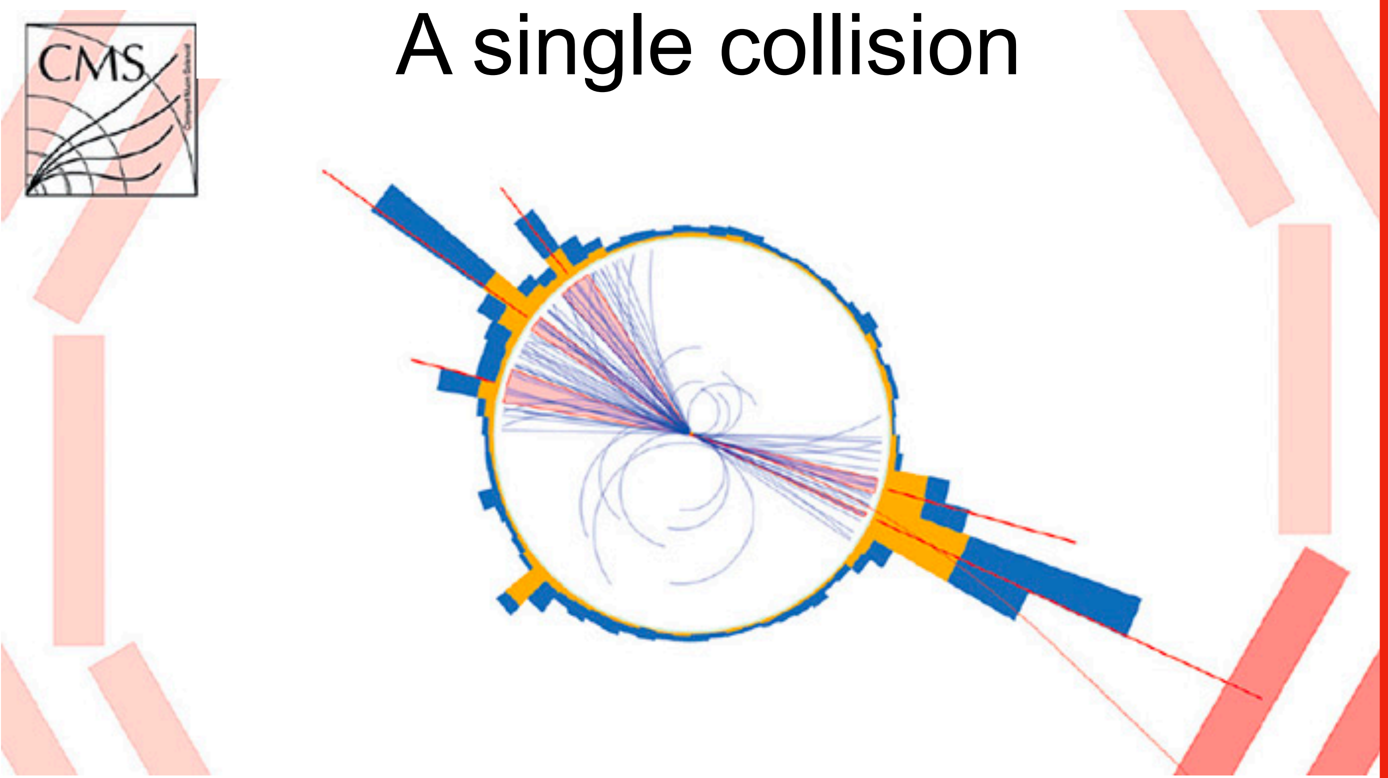
- A single particle can leave deposit in many detectors
  - Each detector deposit a complex and different topology
  - Reconstruction of particles/detectors can be parallelized

Reconstruction Challenge



LHC reconstruction involves combining many different detectors in to particles

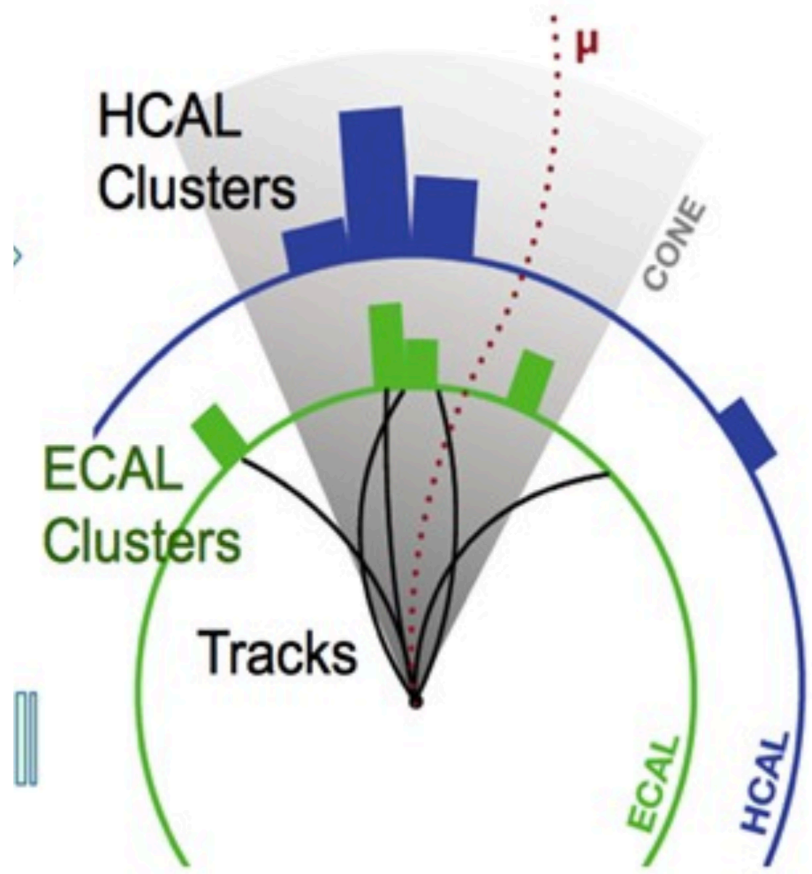
A single collision



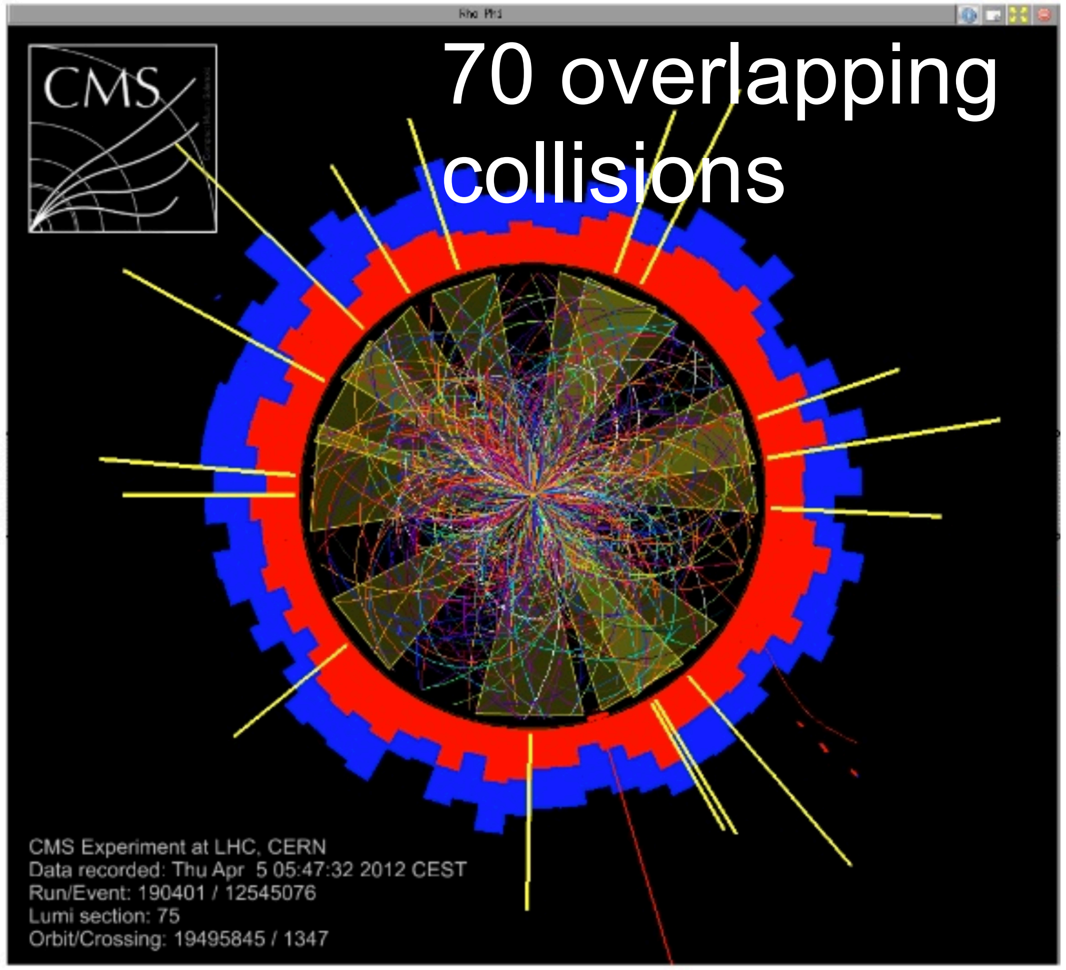
Many particles lie on top of each other making an event

With each collision aim to probe a single event

Reconstruction Challenge



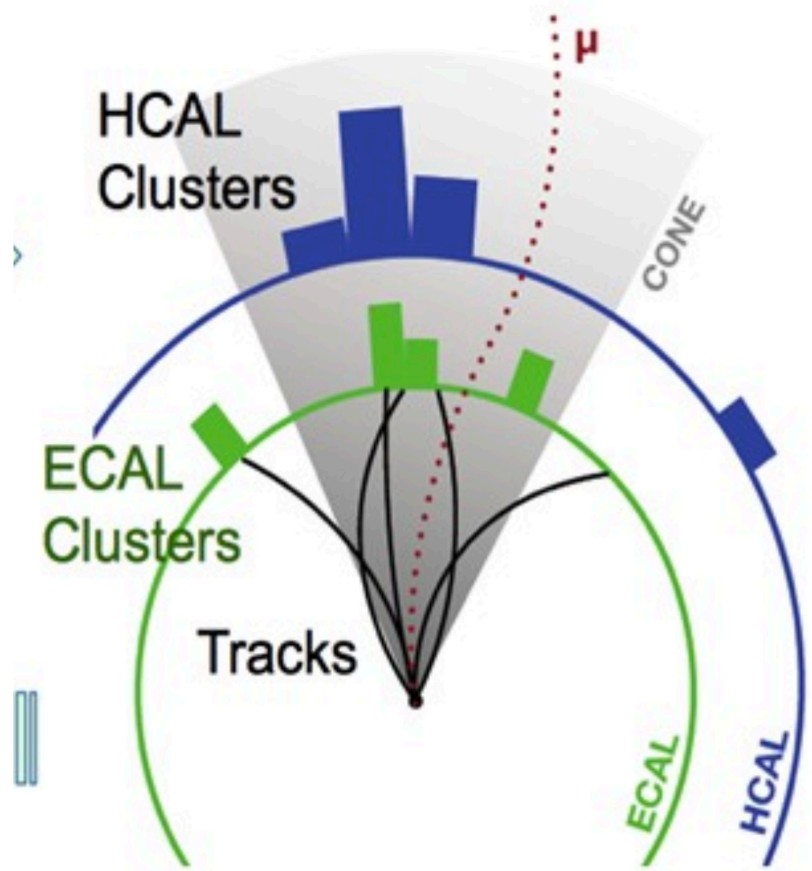
LHC reconstruction involves combining many different detectors in to particles



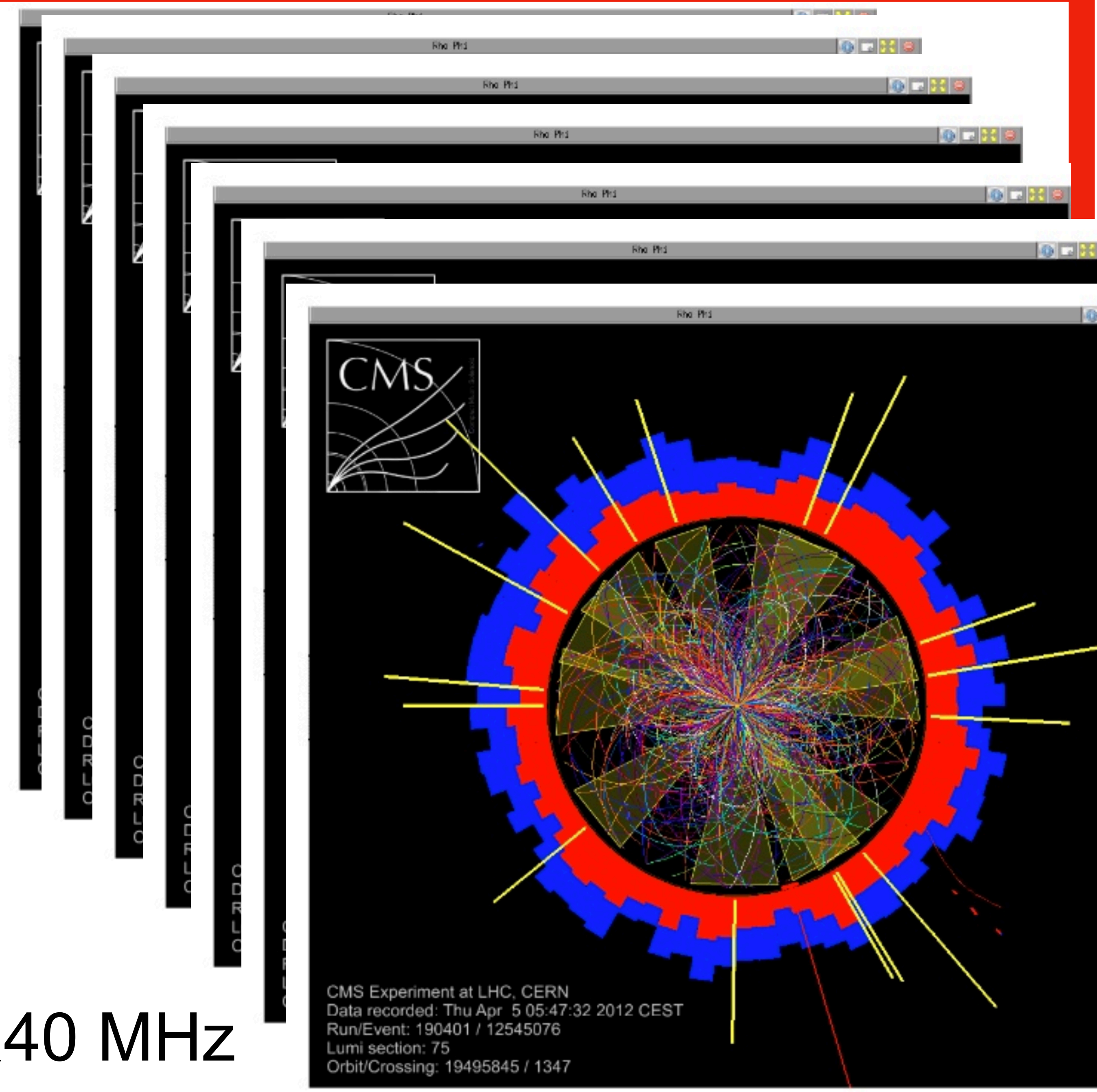
Currently we have 70 collisions lying on top of each other **Event**

In the future will be > 200 collisions

Reconstruction Challenge



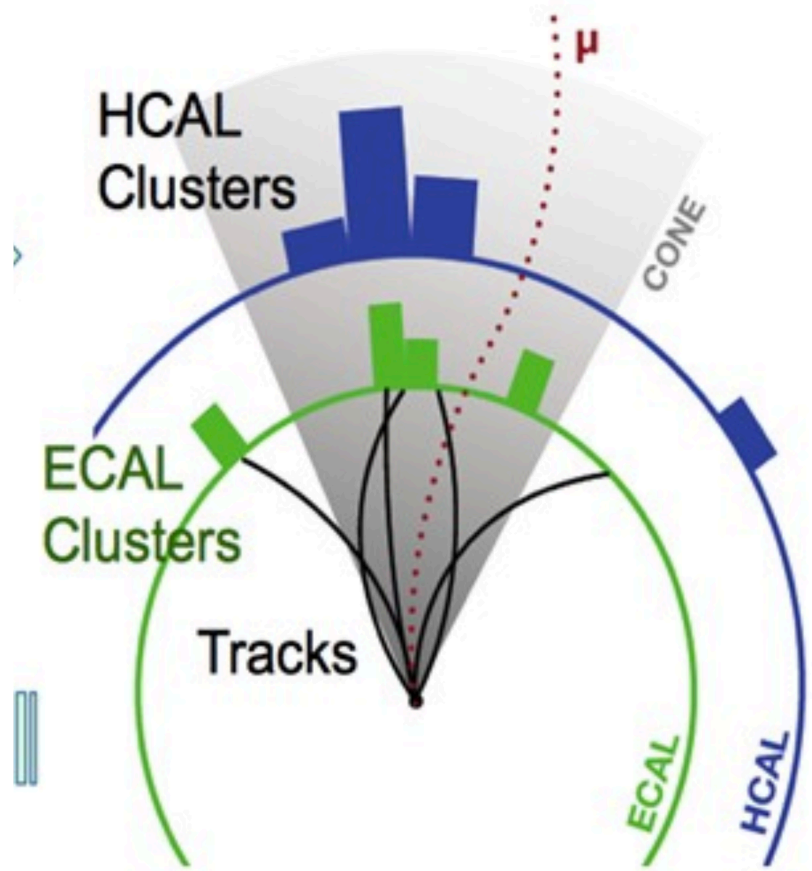
LHC reconstruction involves combining many different detectors in to particles



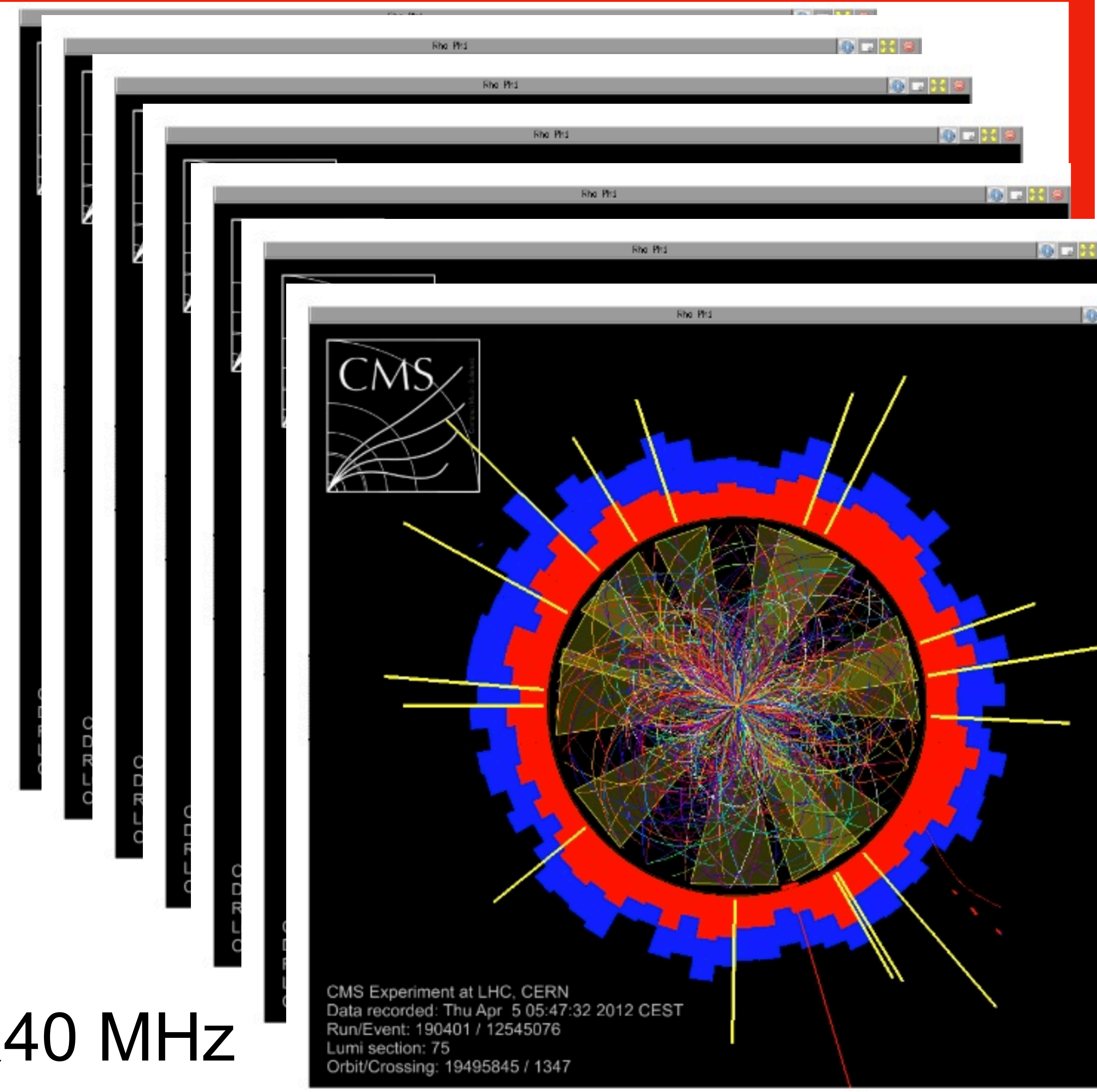
@40 MHz

40 Million times per second

Reconstruction Challenge



LHC reconstruction involves combining many different detectors in to particles



@40 MHz

40 Million times per second

Batch N per particles

Batch 1 per Event

Data Flow in CMS

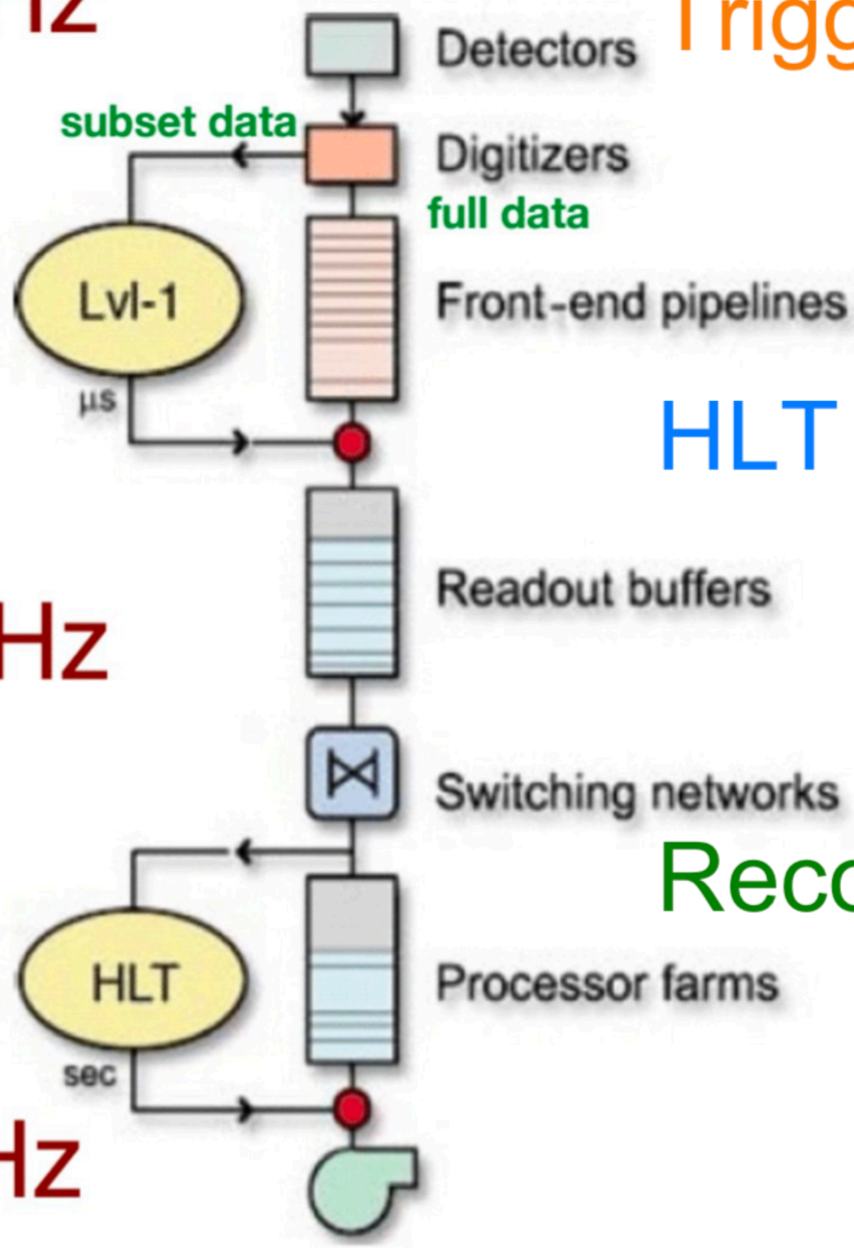
40 MHz

Data reduction

100 kHz

Data reduction

500 Hz

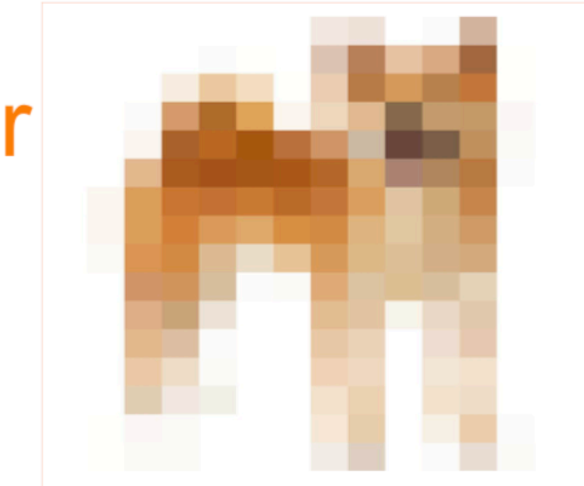


L1 Trigger

HLT

Reco

“Offline Computing”
Grid, O(10) Pb



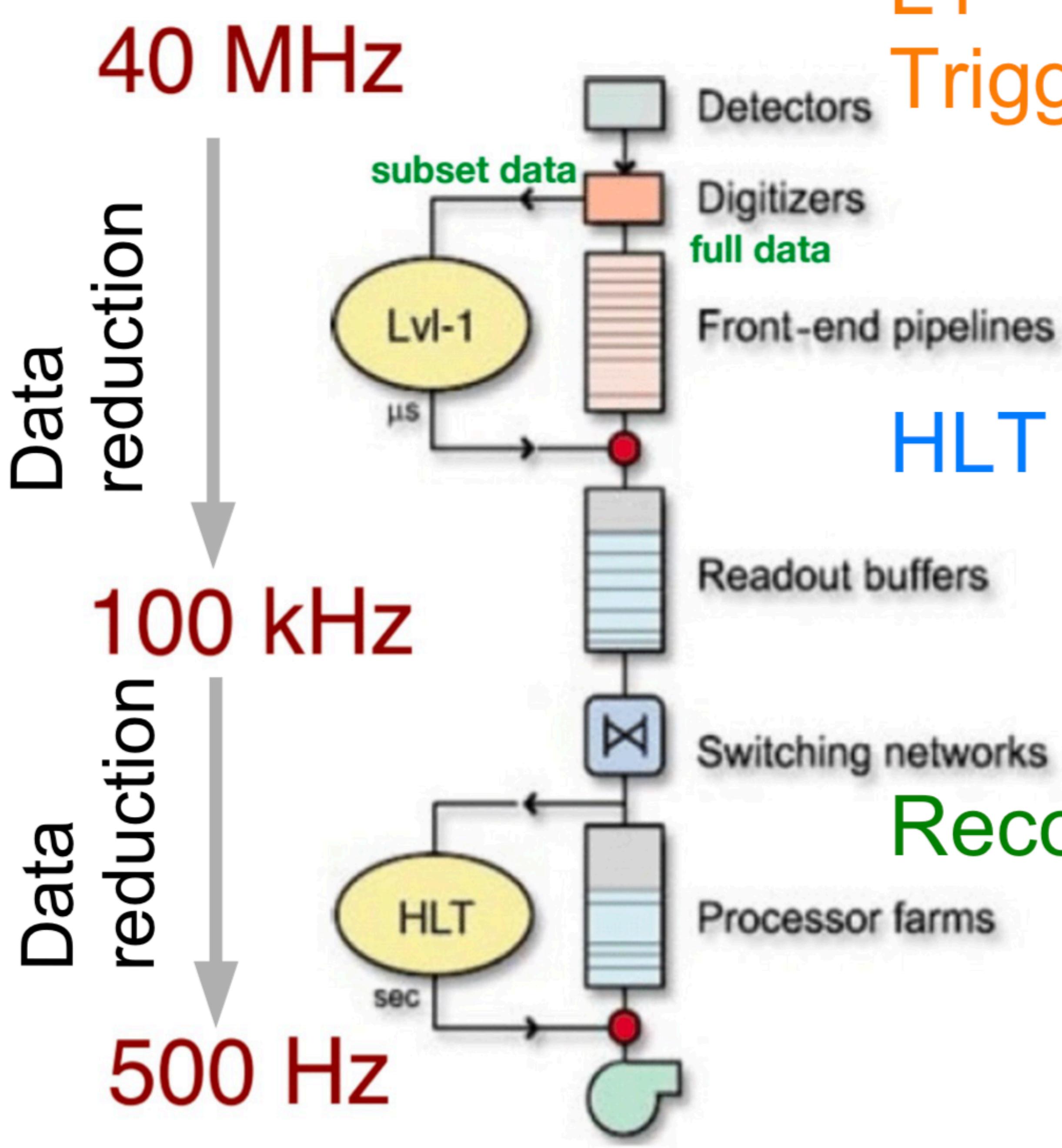
High speed
Low granularity
readout (10μs)

See most collisions like this

And throw away most collisions

Despite the large rate reduction we still store many Petabvtes of data

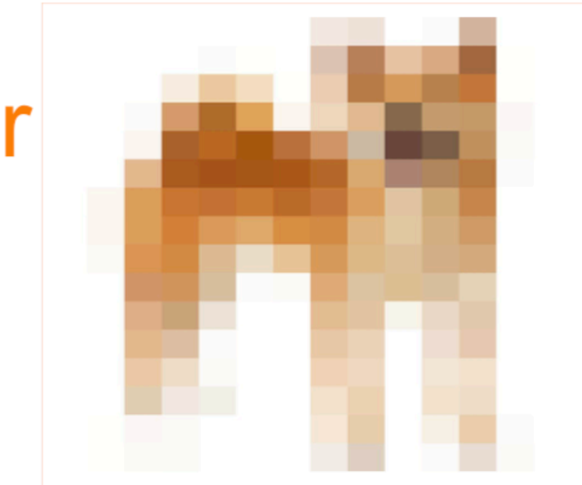
Data Flow in CMS



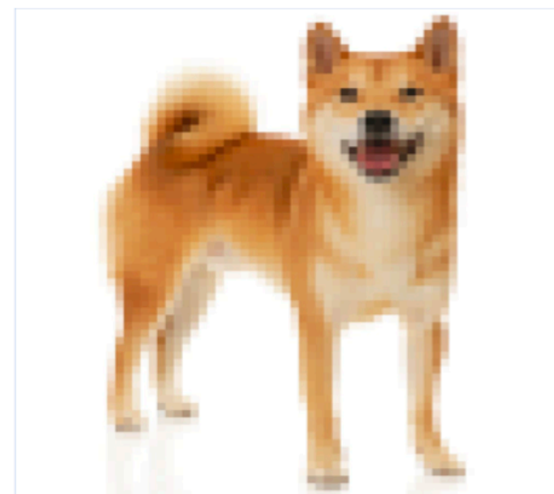
L1 Trigger

HLT

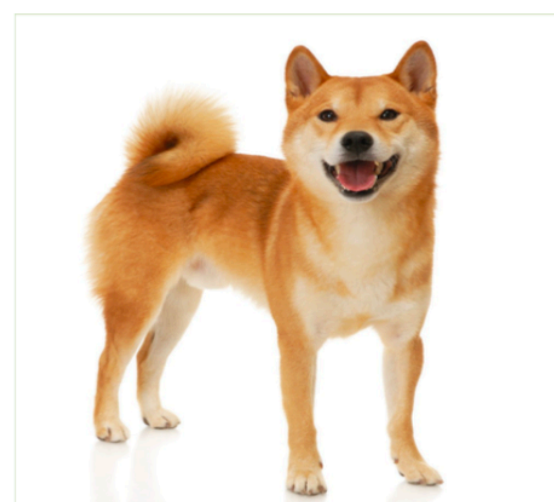
Reco



High speed
Low granularity
readout (10µs)



Intermediate
speed (100ms)
better readout

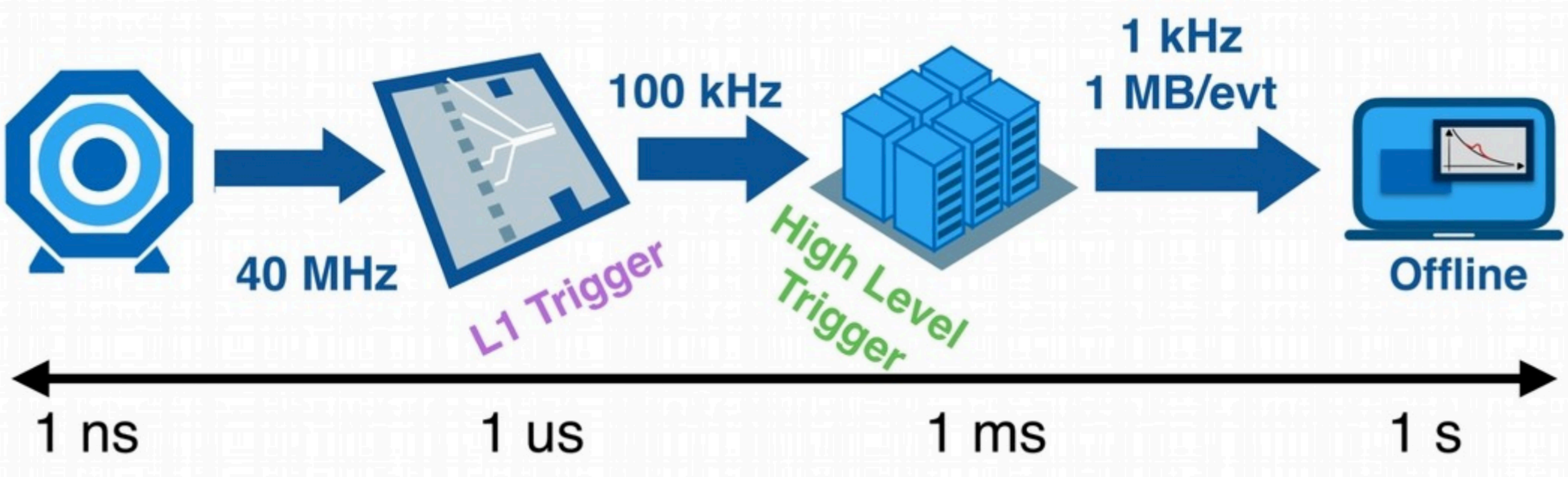


Full data readout
(10s)

"Offline Computing"
Grid, O(10) Pb

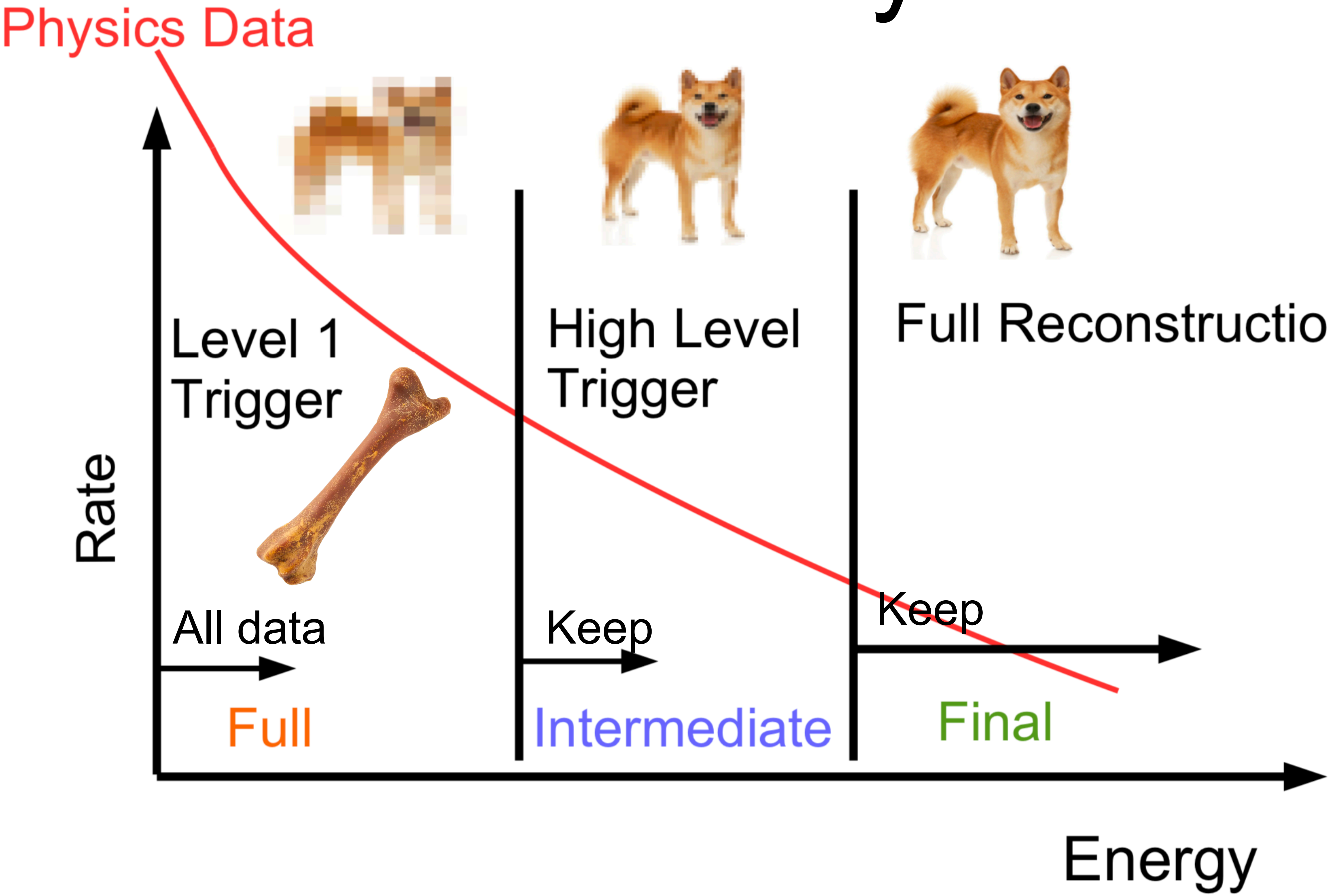
Despite the large rate reduction we still store many Petabytes of data

How do we process data?



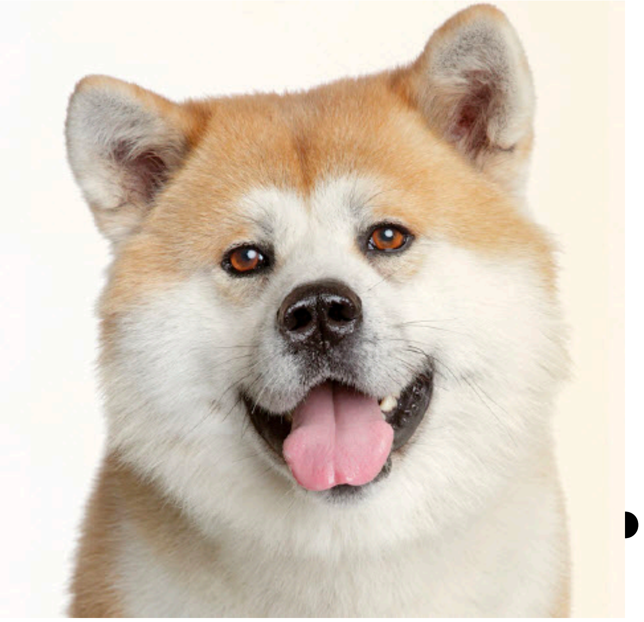
A single event is 1000-2000 particles
thats 8MB after zero suppression

The Physicist View



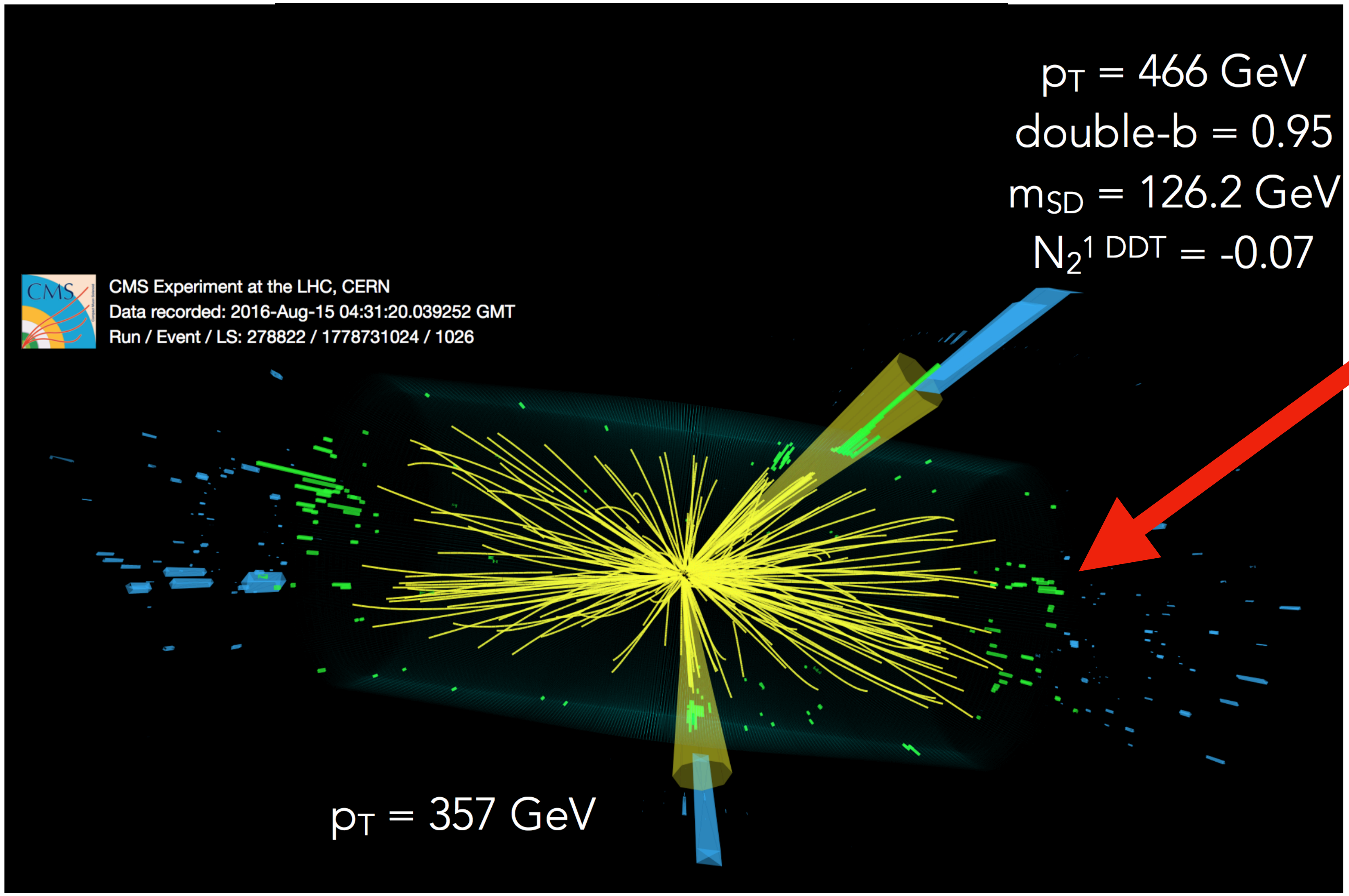
The Physicist View

We know that we are throwing away a lot of good data



Hidden gems?

- There is a plethora of physics that we throw out



Higgs boson right on the cusp of being thrown out

Higgs boson discover at CERN
2013 Nobel Prize

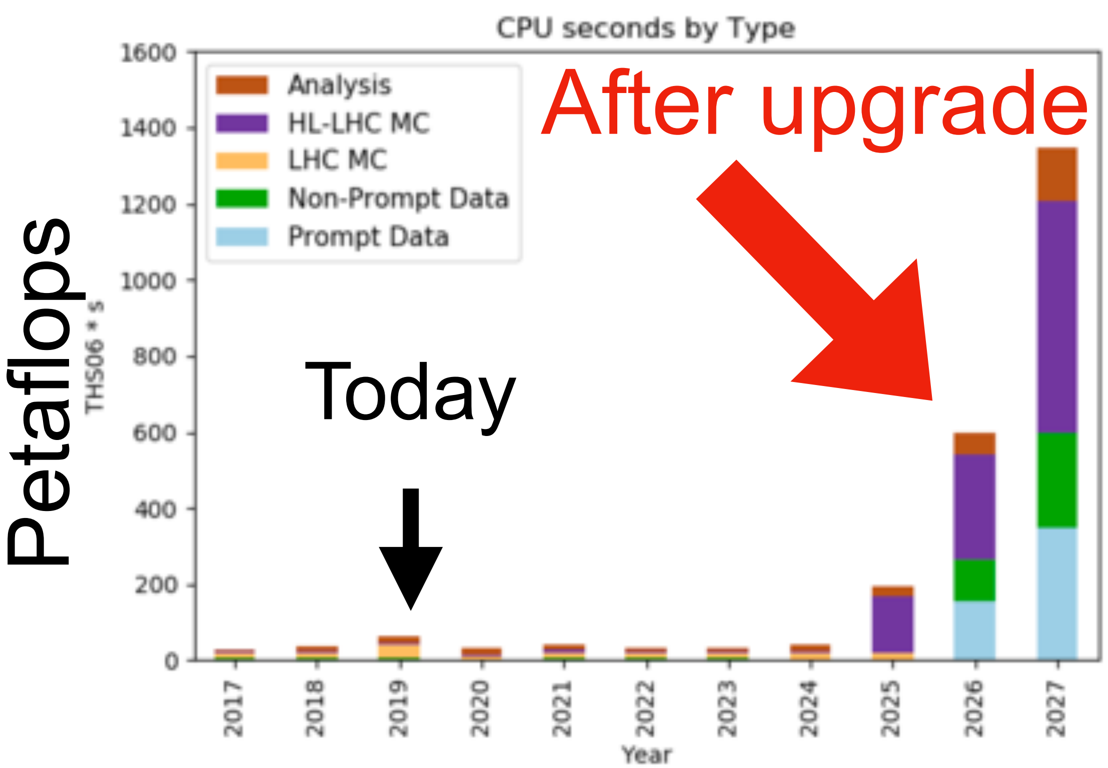
The dream

- At the moment:
  - We only get a full data of one in 100,000 collisions
  - There is interesting physics that we have to throw away
- We would like to analyze every collision at the LHC
  - To deal with this we need to increase our throughput
  - Ultimately this means going to 100s of Tb/s

The Challenge

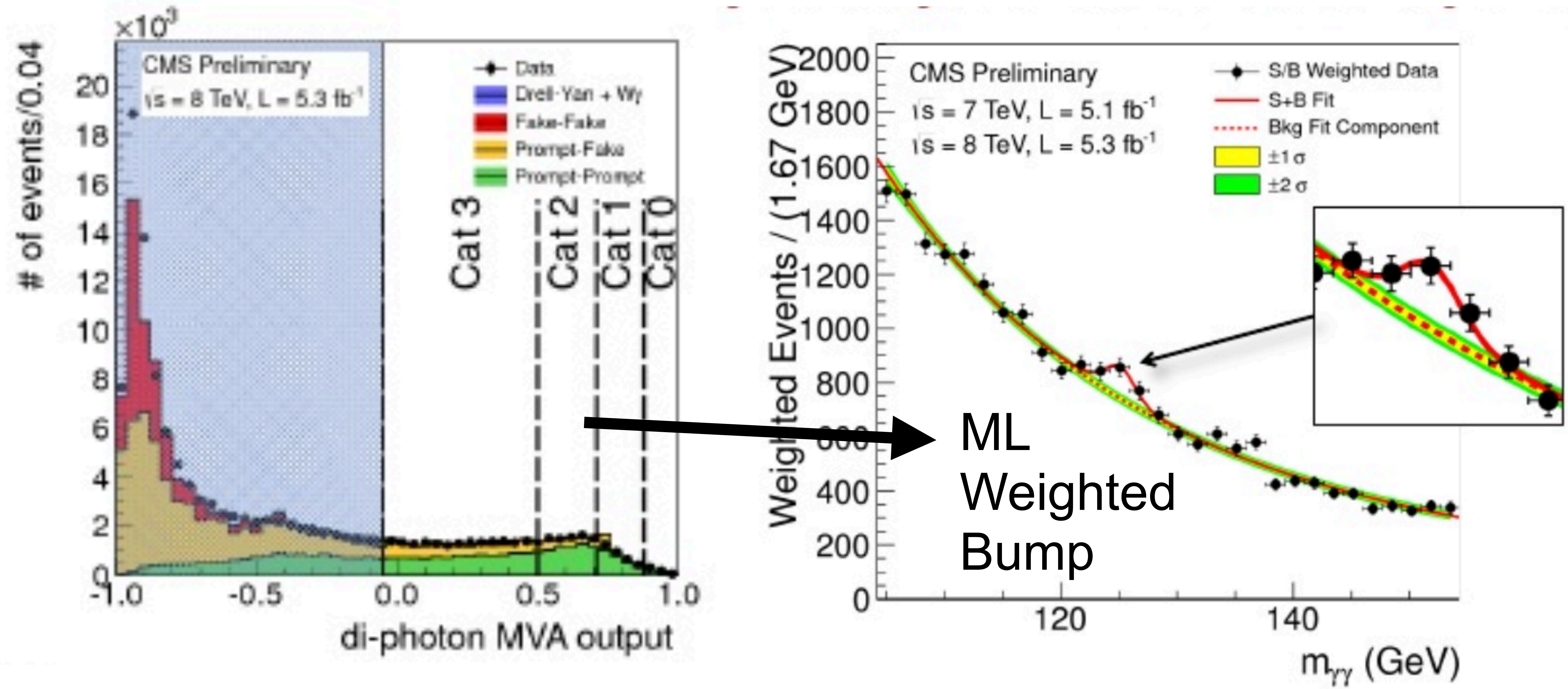
- We are upgrading the system
  - Our event size will be 10 times larger
  - And we have to take data at 5x the rate
    - Need this just to preserve our existing physics
- 10s of years of processing without modifying system

End of Dennard Scaling is about to hit us hard



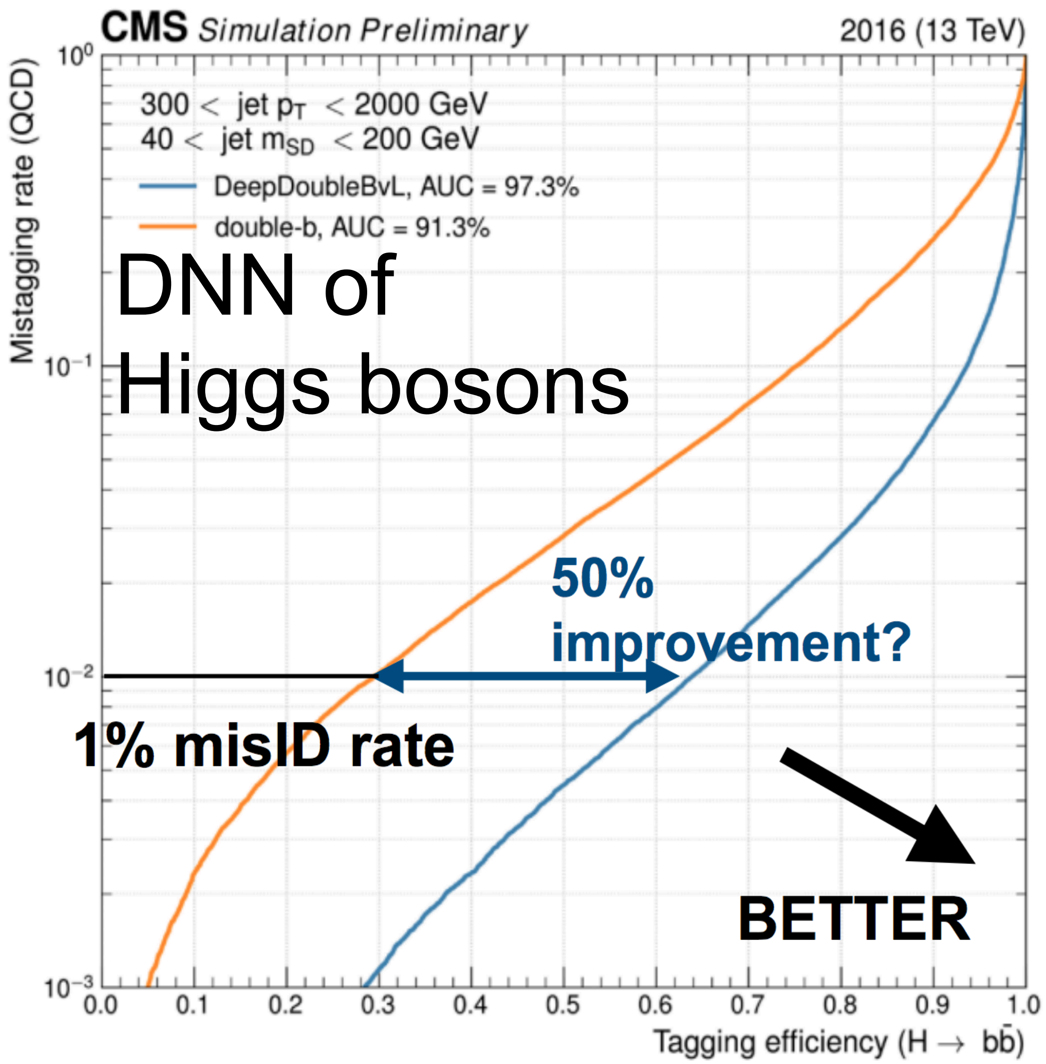
ML in HEP

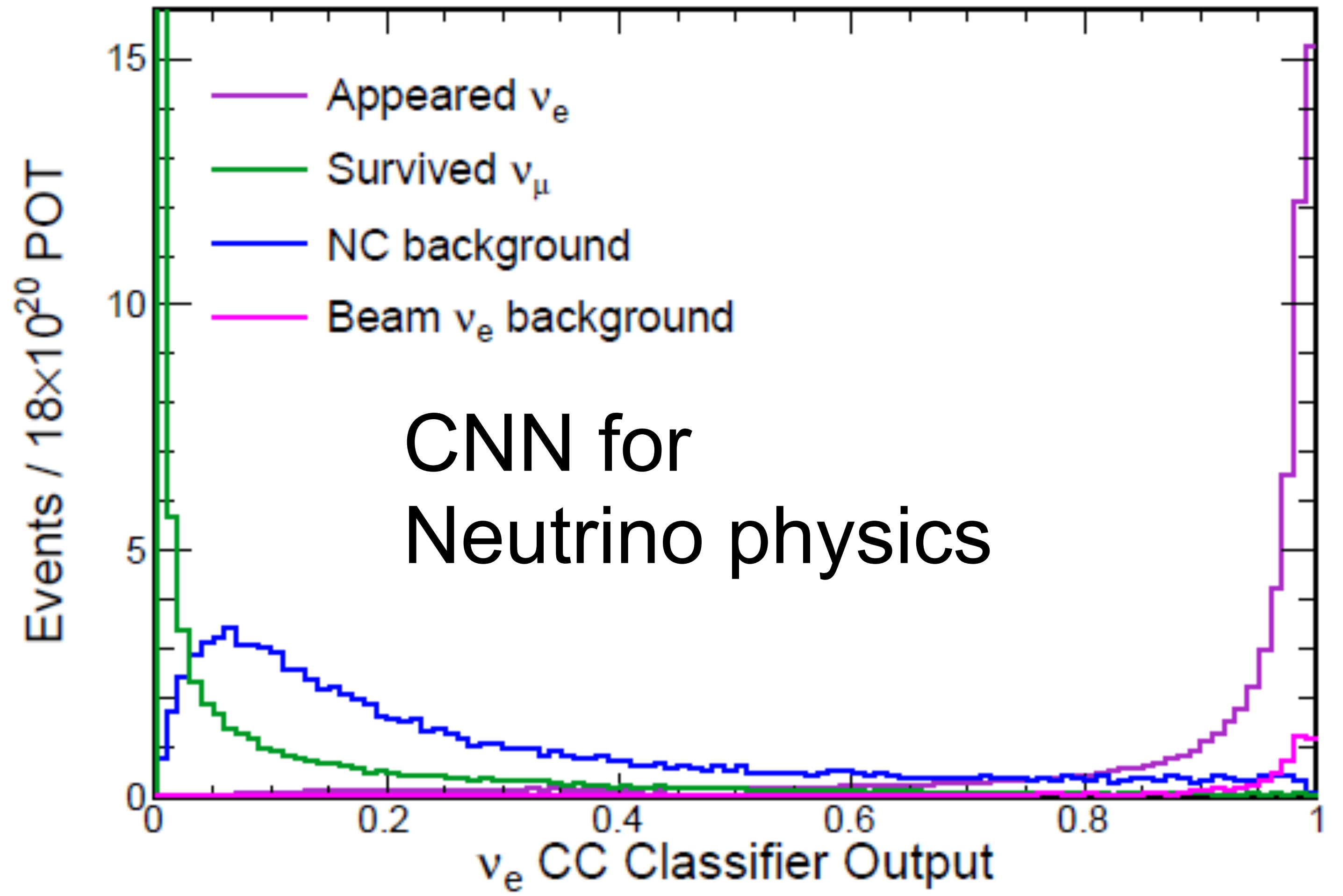
- High Energy Physics has been quick to adopt ML

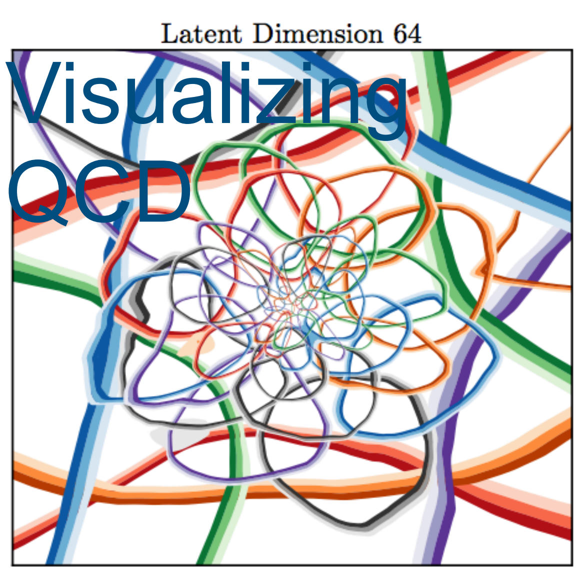


Higgs Discovery had Machine Learning all over it

Deep Learning in HEP





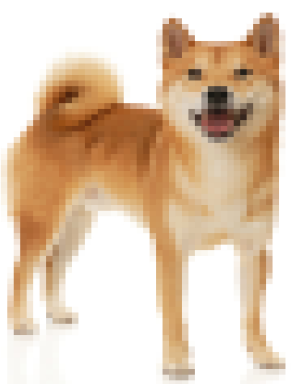


With rise of deep learning we are quickly coming up with new ways to interpret the data and improve our Physics data analysis

Rest of This talk

- Going to look at what we are doing to improve data rates

Ultra low latency high throughput processing
FPGA+ASIC Based system
<10μs latency



One site accelerated processing of the data
Accelerated based system
< 500ms latency

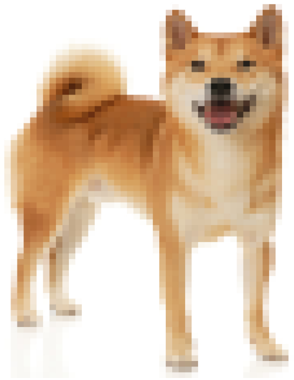
Distributed processing of the data
Cloud based system
< 30s latency

- After this we will look at how this applies to Physics (#trending)

Rest of This talk

- Going to look at what we are doing to improve data rates


Custom Hardware


Edge

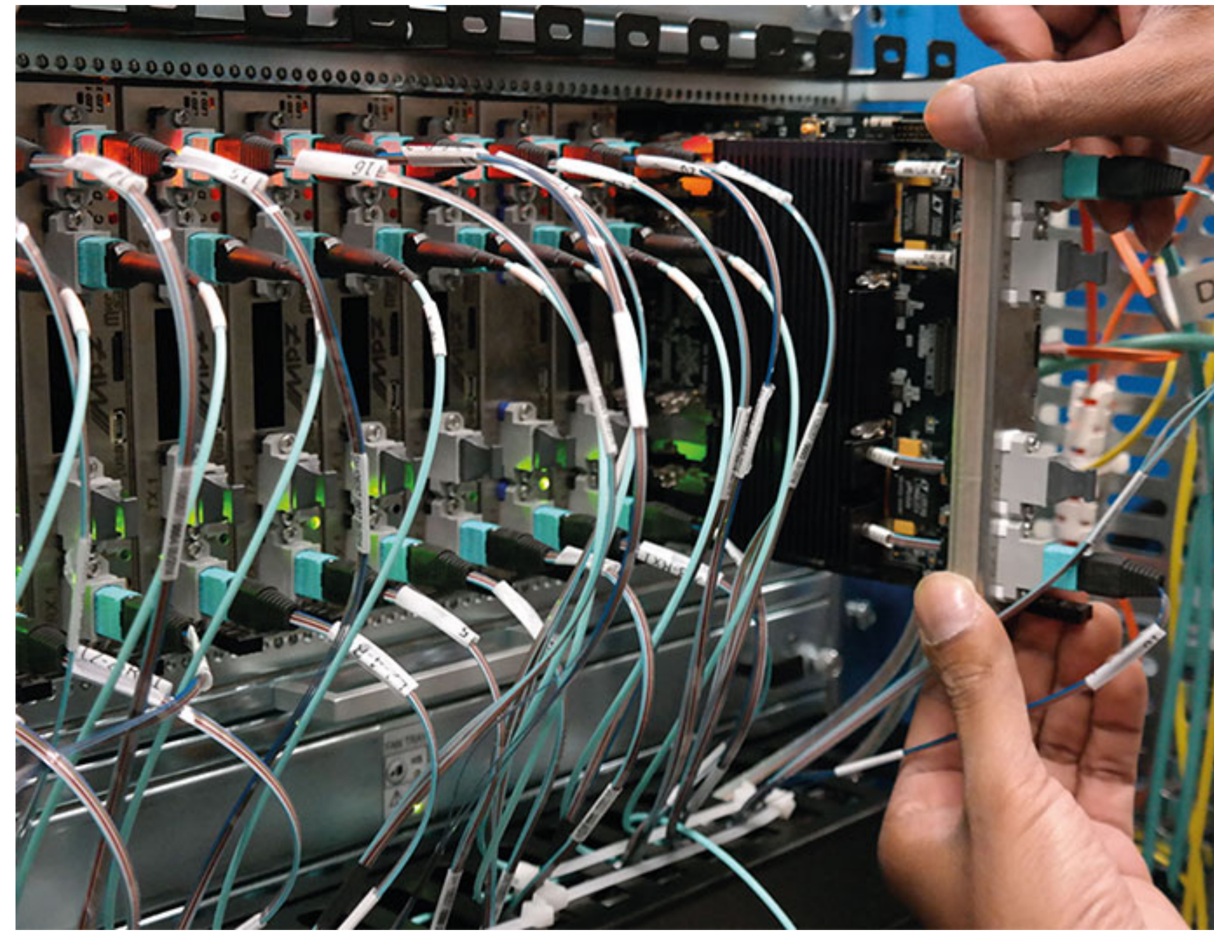

Cloud

All of these can or do use FPGAs

- After this we will look at how this applies to Physics (#trending)

40 MHz (10μs)

L1 Trigger



A new event every 25ns

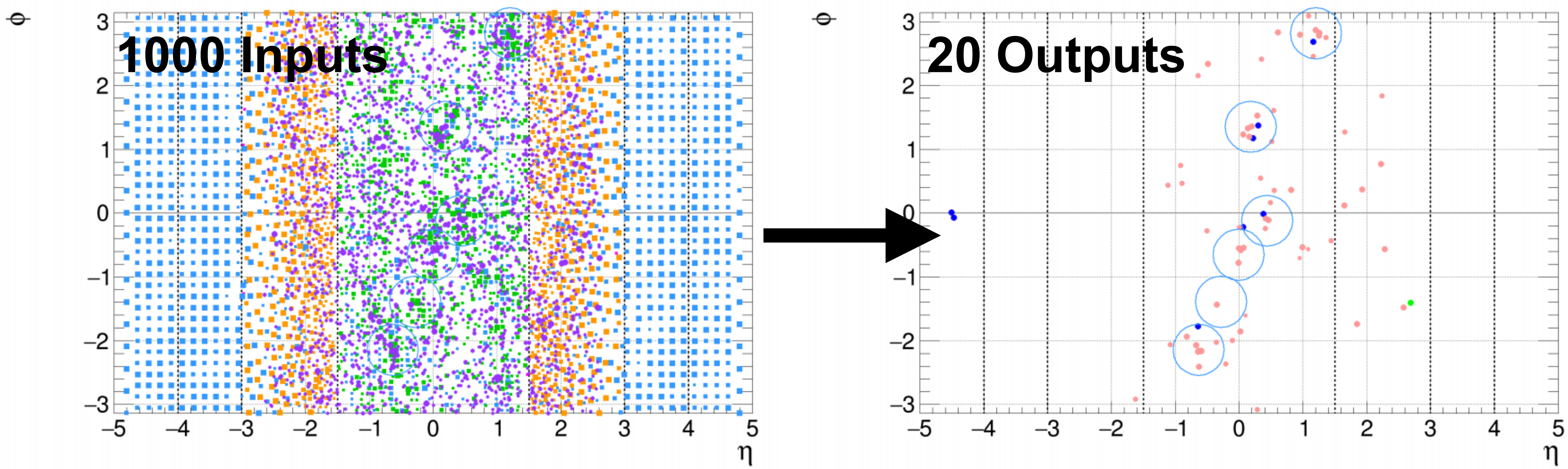
Interconnected FPGAs
Optical links between the chips
48-112 Links per chip
Links run at 10-25 Gbps
Full system is O(1000) FPGAs

- We have at MOST 1μs to run an algorithm
  - We aim for algorithms that are in the 100ns range
- **Want to make the fastest possible algorithm**
- **Want to have the smallest initiation interval**
  - We apply algorithms to multiple subsets of total event

40 MHz (10μs)

Capabilities

Before After



Particle level reconstruction and event cleanup is a first step

- Above algorithm takes roughly 700ns* on a Xilinx VU9P
- Parallelize algorithm amongst regions

As physicists we wrote most of this in High Level Synthesis (HLS)

- C-based compiler makes our code readable

*tested on AWS f1 instances

Deep Learning

hls4ml

ML compiler targeting low latency

Can we run deep learning in our system?

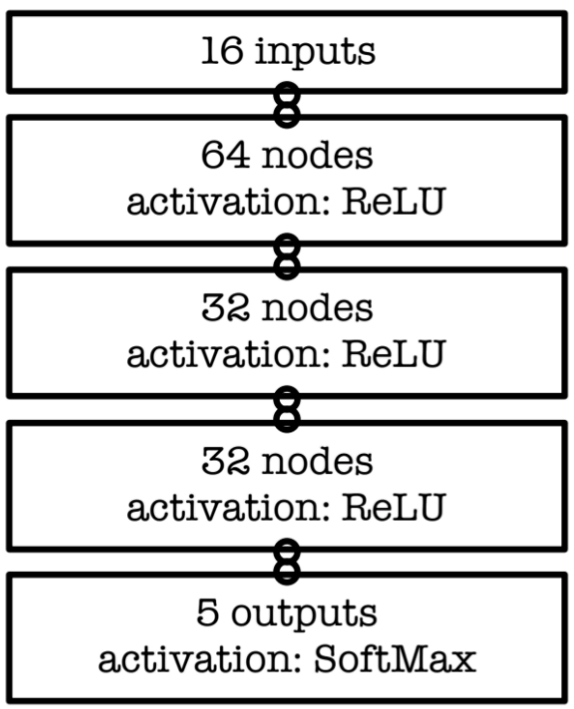
Yes

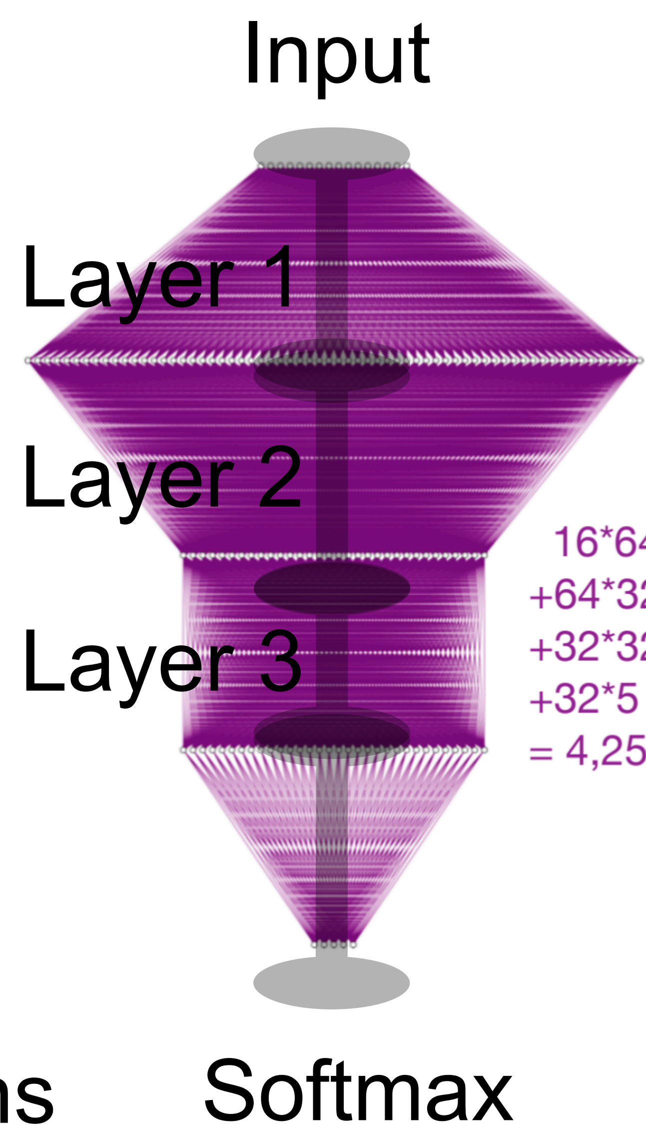
As physicists……
used HLS for our setup
Targeted low latency

Quickly being adopted:
Anomalies(Autoencoder)
Muon reconstruction
Tau Lepton reconstruction
Quark/Gluon showers
Many more

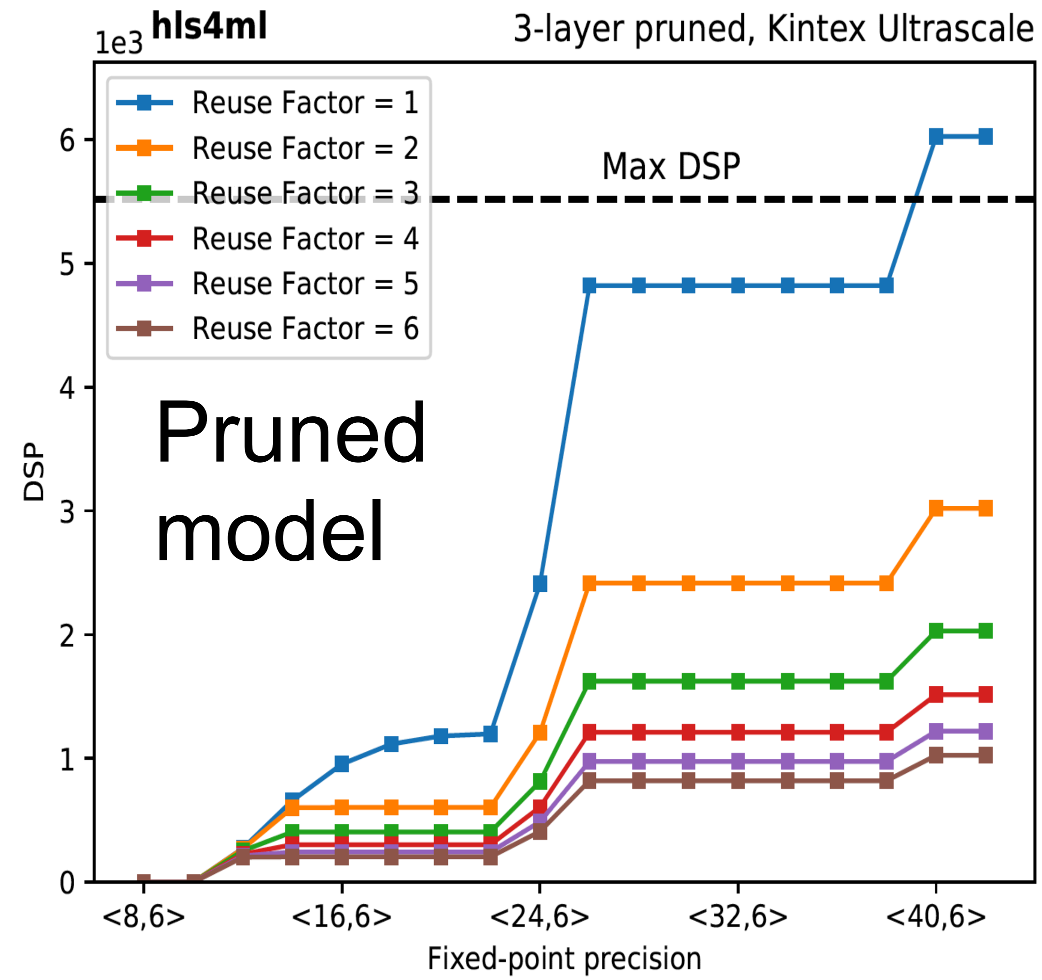
What can we run?

Case Study:
Particle Jet classifier





16*64
+64*32
+32*32
+32*5
= 4,256 syr

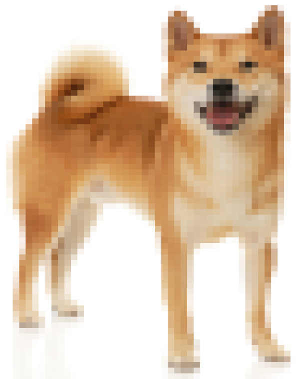


75ns latency
new input every 5ns
fits in a VU9P

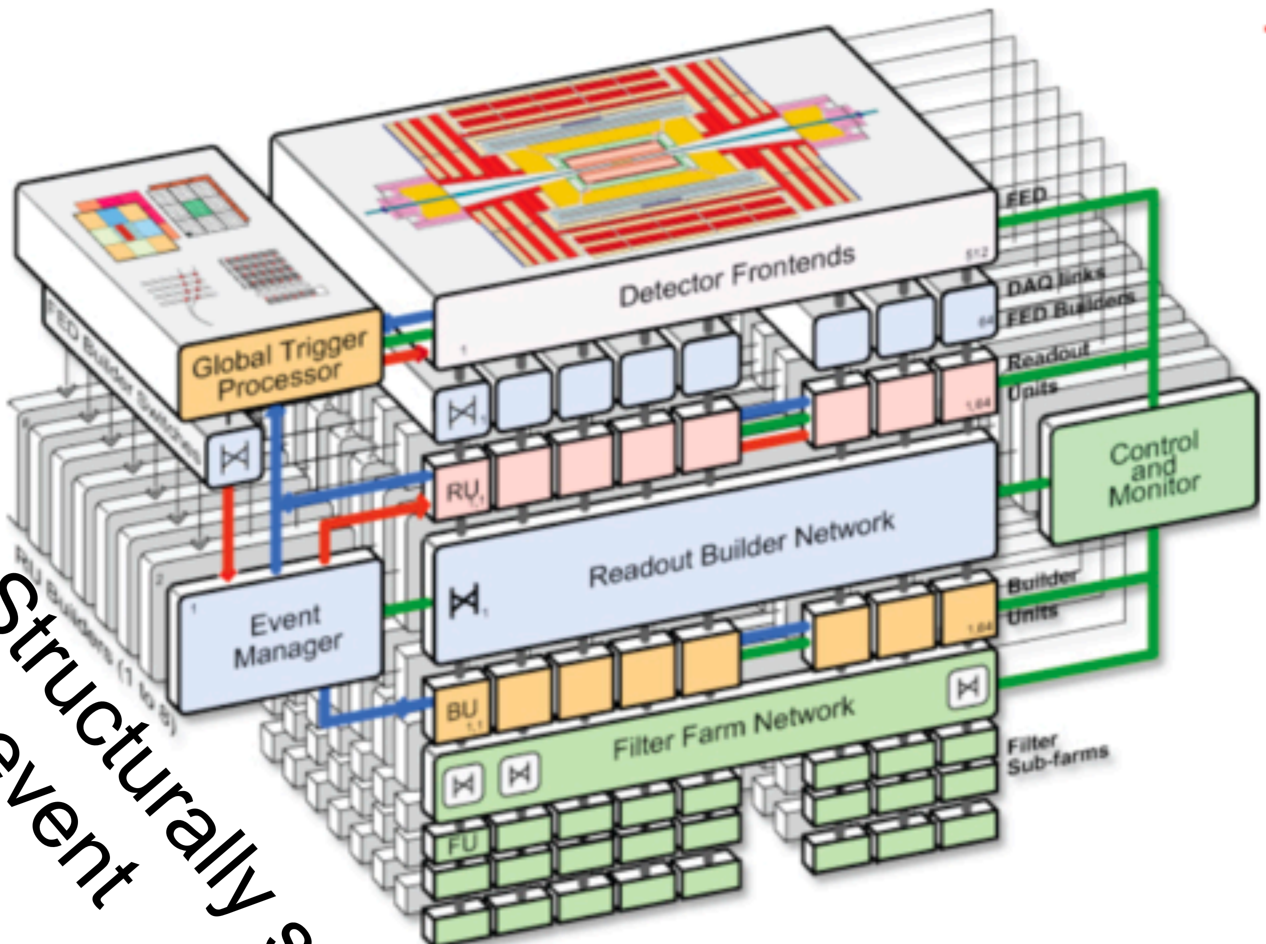
Low latency support in HLS4ML for
MLPs,CNNs,Binary/Tenary NNs,BDTs,Graph NNs,LSTM/GRUs

Takeaways

- LHC has a unique role to play when processing data
  - With the insanely large data rates
  - Low latency+high throughput demands specialized system
    - Our system will always be ASIC+FPGA-only
    - Working to bring ML and complex algorithms to the system
- As part of this work we developed HLS4ML
  - Quickly becoming a staple for L1 trigger development



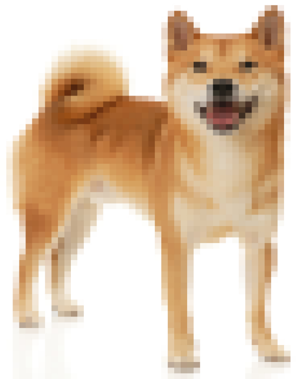
100 kHz (500ms) High Level Trigger



Structurally split by event

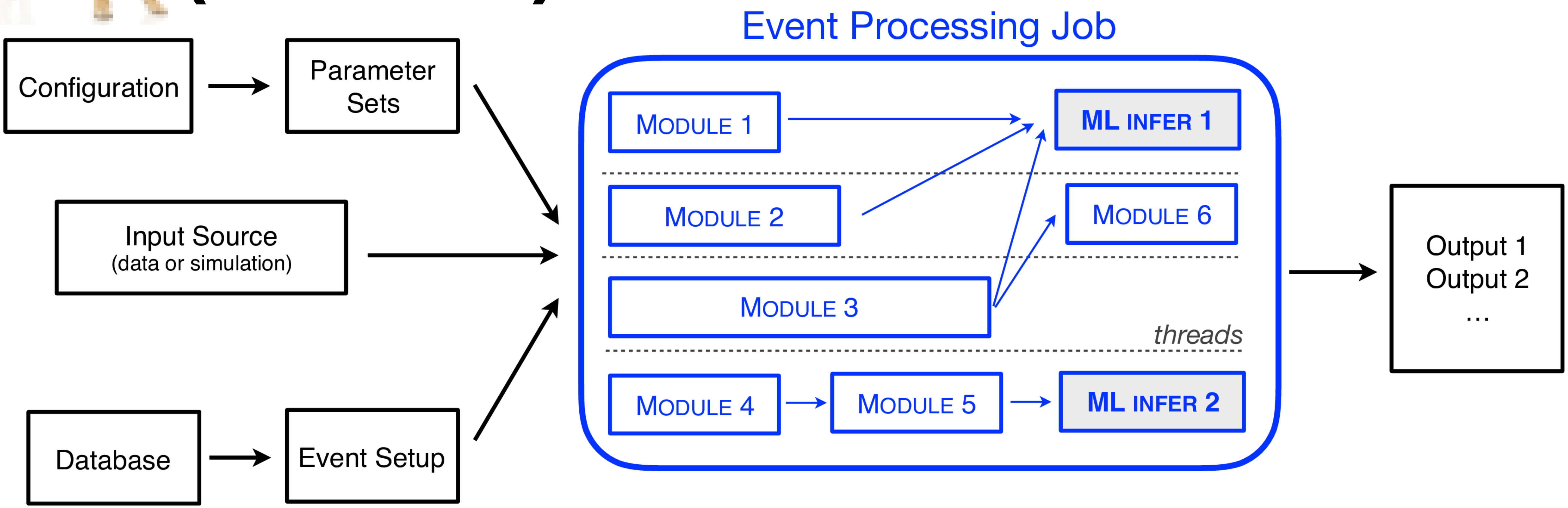
- 100 kHz of collisions in
- 1kHz of collisions out
- <500ms to analyze collision
- Currently
  - A local computing cluster
  - System is all CPUs

- Experiments are considering GPU/CPU system for 2022

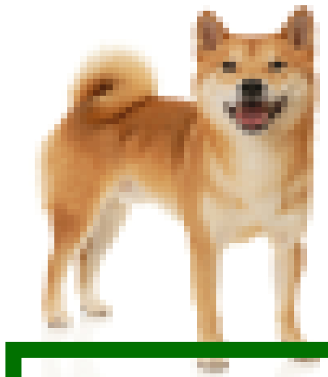


100 kHz (500ms)

Reco Strategy



- Complicated scheme of modules
  - While some parts are parallelizeable
  - Collision level analysis built in by construction **(Batch 1)**



Improving Performance

Idea #1

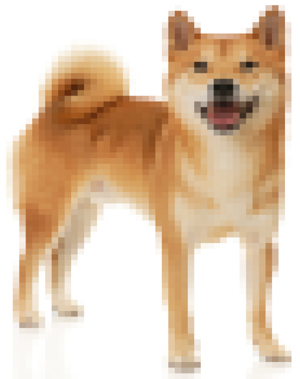
- Buy a GPU/FPGA card for each node
  - Pro: Can be done now   Con: Massive code rewrite

Idea #2

- Do onsite as-a-service processing
  - Pro: build up system over time Con: Networking

Idea #3

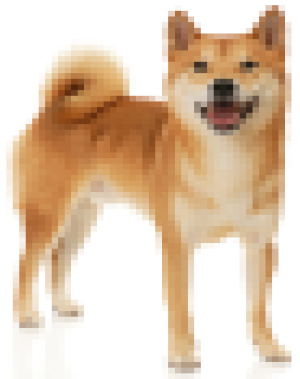
- Port what we can to ML and rely on existing/new tools
  - Pro: We like ML   Con: Redisgn algorithms can be hard



Future Strategies

Incorporating Heterogenous systems(GPU/FGPA)

	Idea #0 Port Existing Algos	Idea #3 Upgrade to ML Algos
Idea #1 Investigate onboard GPU/FPGA	Rewrite all of our code in CUDA/Kokkos HLS/RTL/???	Tools exist TF/Pytorch/TRT Xilinx ML Suite Brainwave….
Idea #2 Outsource GPU/FPGA to a service	Write specialized interface	Tools exist: TRT-server Brainwave **and in cloud!**



Future Strategies

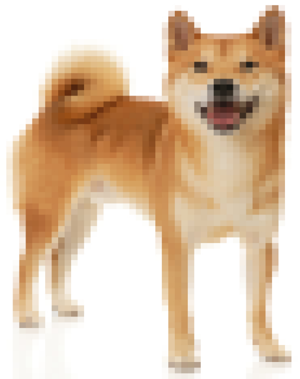
Incorporating Heterogenous systems(GPU/FGPA)

	Idea #0 Port Existing Algos	Idea #3 Upgrade to ML Algos
Idea #1 Investigate onboard GPU/FPGA	Rewrite all of our code in CUDA/Kokkos HLS/RTL/???	Tools exist TF/Pytorch/TRT Xilinx ML Suite Brainwave….
Idea #2 Outsource GPU/FPGA to a service	Write specialized interface	Tools exist: TRT-server Brainwave **and in cloud!**

Focus of this talk

(see backup for others)

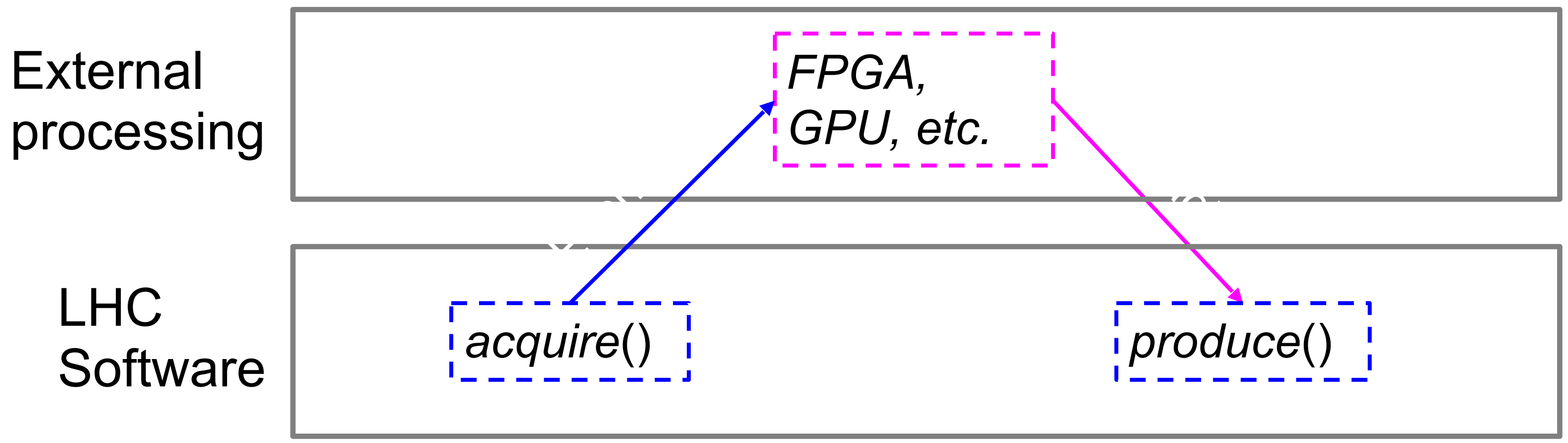
ML is highly parallelizeable→Big speed ups



Idea #1:External

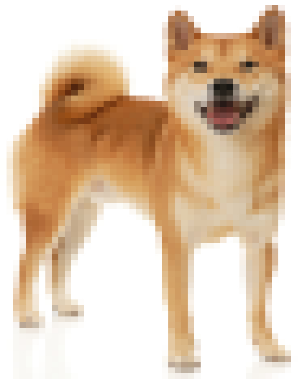
- To run these algorithms within our software

Asynchronous task based processing



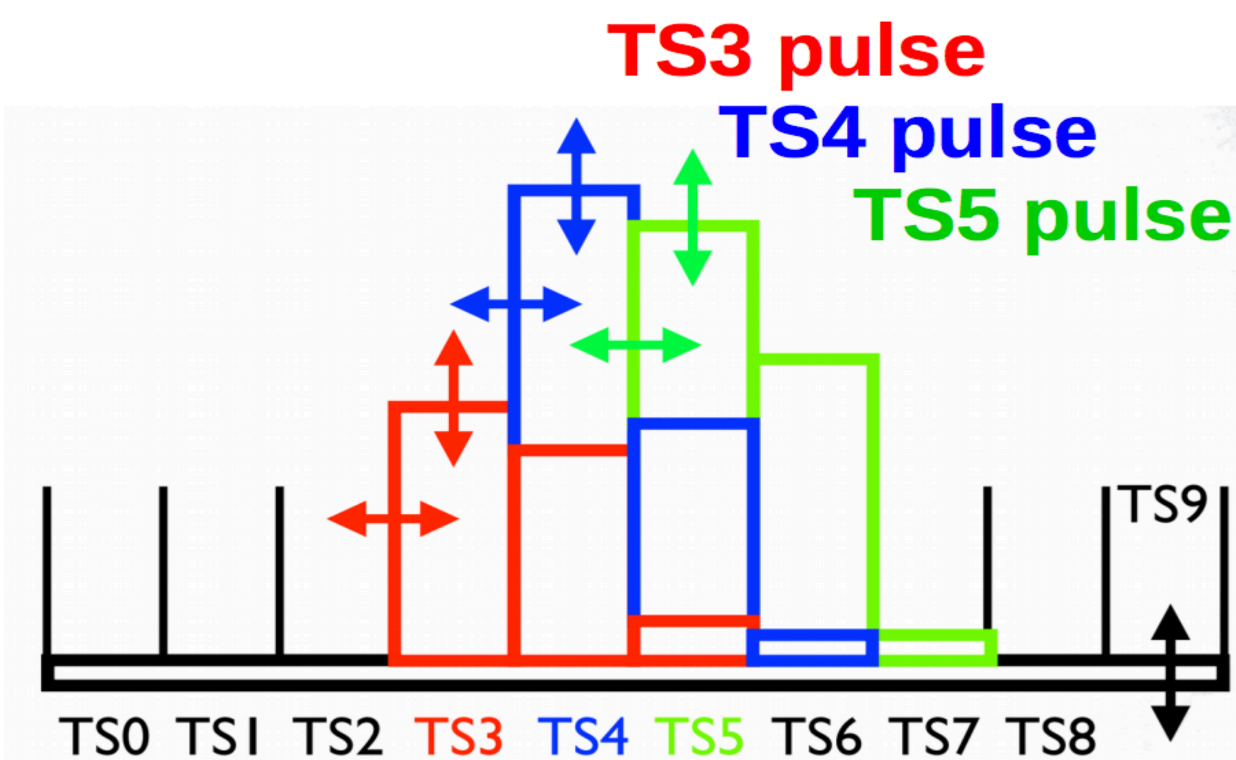
Non-blocking: schedule other tasks while waiting

- Our Strategy
  - Pick benchmark ML examples+put them on FPGAs/GPUs
  - Observe what level speed up we get over CPUs and how



Idea #1

Benchmark #1



Energy reconstruction of Hadronic showers

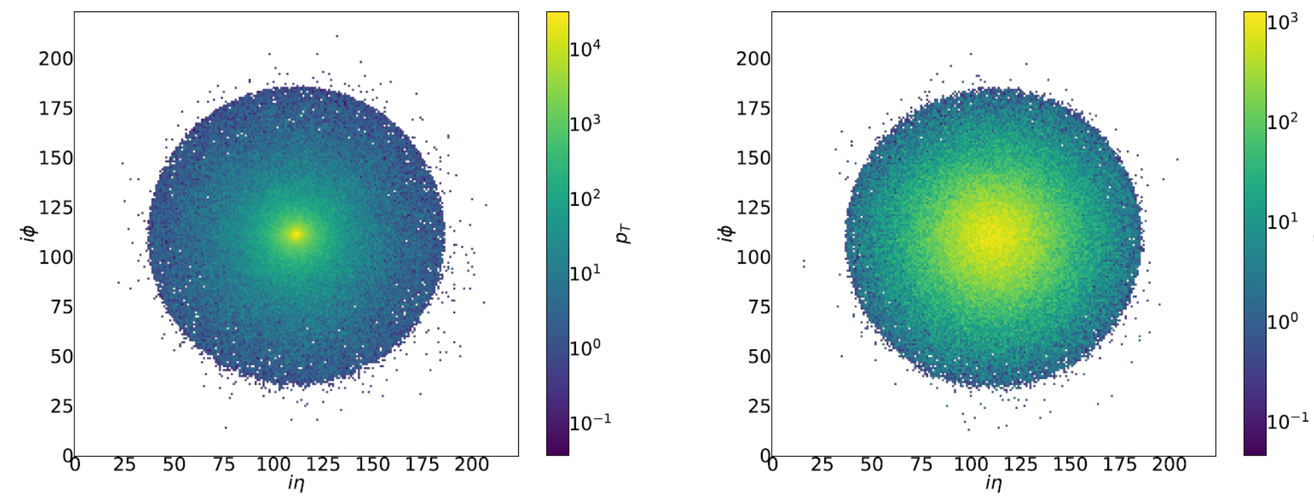
Simple energy regression

16000 times per collision

Batch N per particles

Benchmark #2

QCD, averaged over 5k jets   top, averaged over 5k jets



Top quark identification

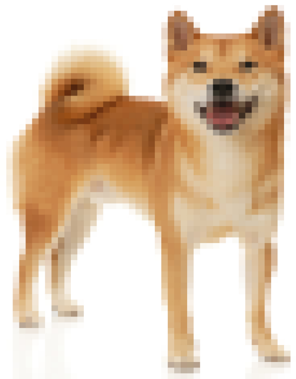
Here we use Resnet50 as benchmark

Complicated identification

Many inputs

1-2 times per collision

Batch 1 per Event



Idea #1

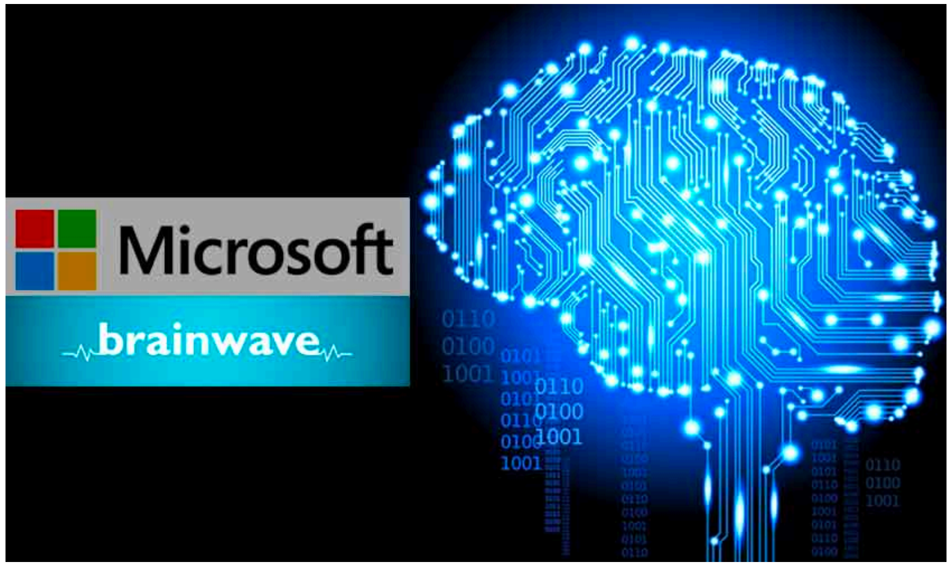
Benchmark #1

XILINX ALL PROGRAMMABLE™

hls4ml

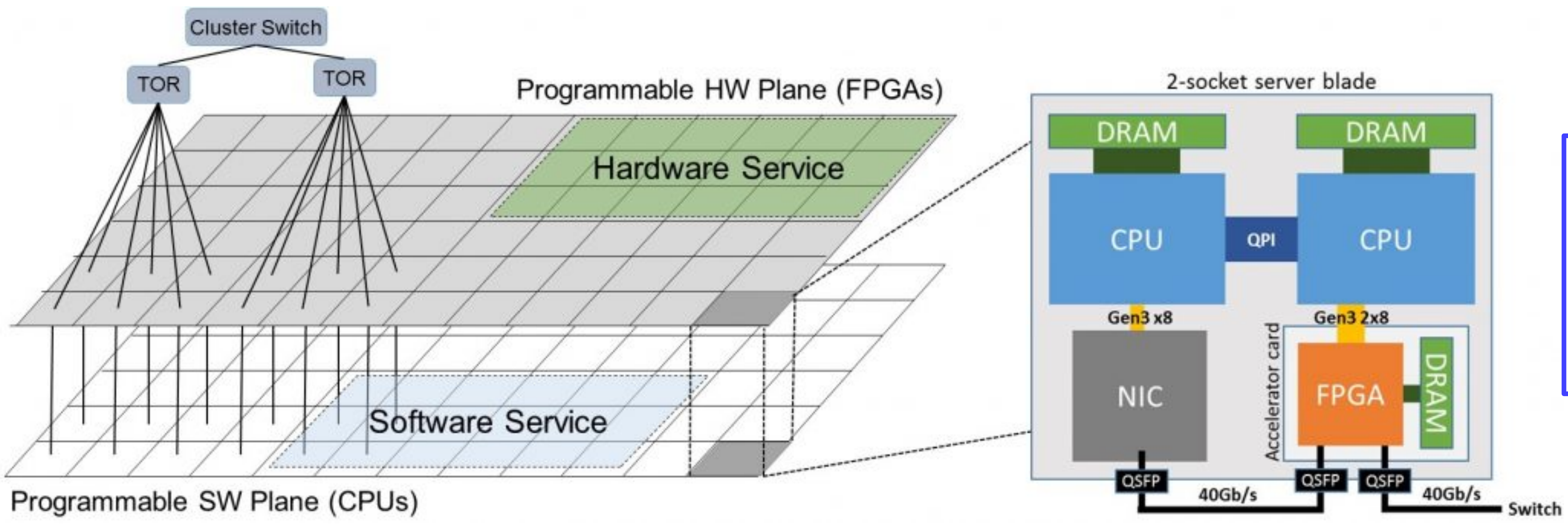
Benchmark #2

intel STRATIX 10 inside



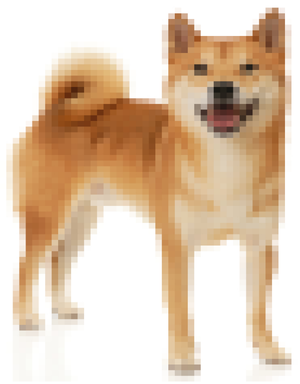
*Also investigating Xilinx ML suite(see backup) + Intel Open Vino

Microsoft Brainwave



Brainwave supports:
- ResNet50
- ResNet152
- DenseNet121
- VGGNet16

- Full FPGA interconnected fabric setup-as-a-service
  - Capable of running many different NN architectures
  - Relying on the NPU framework for ML compilation
  - (Very) optimized use of ML on the FPGA

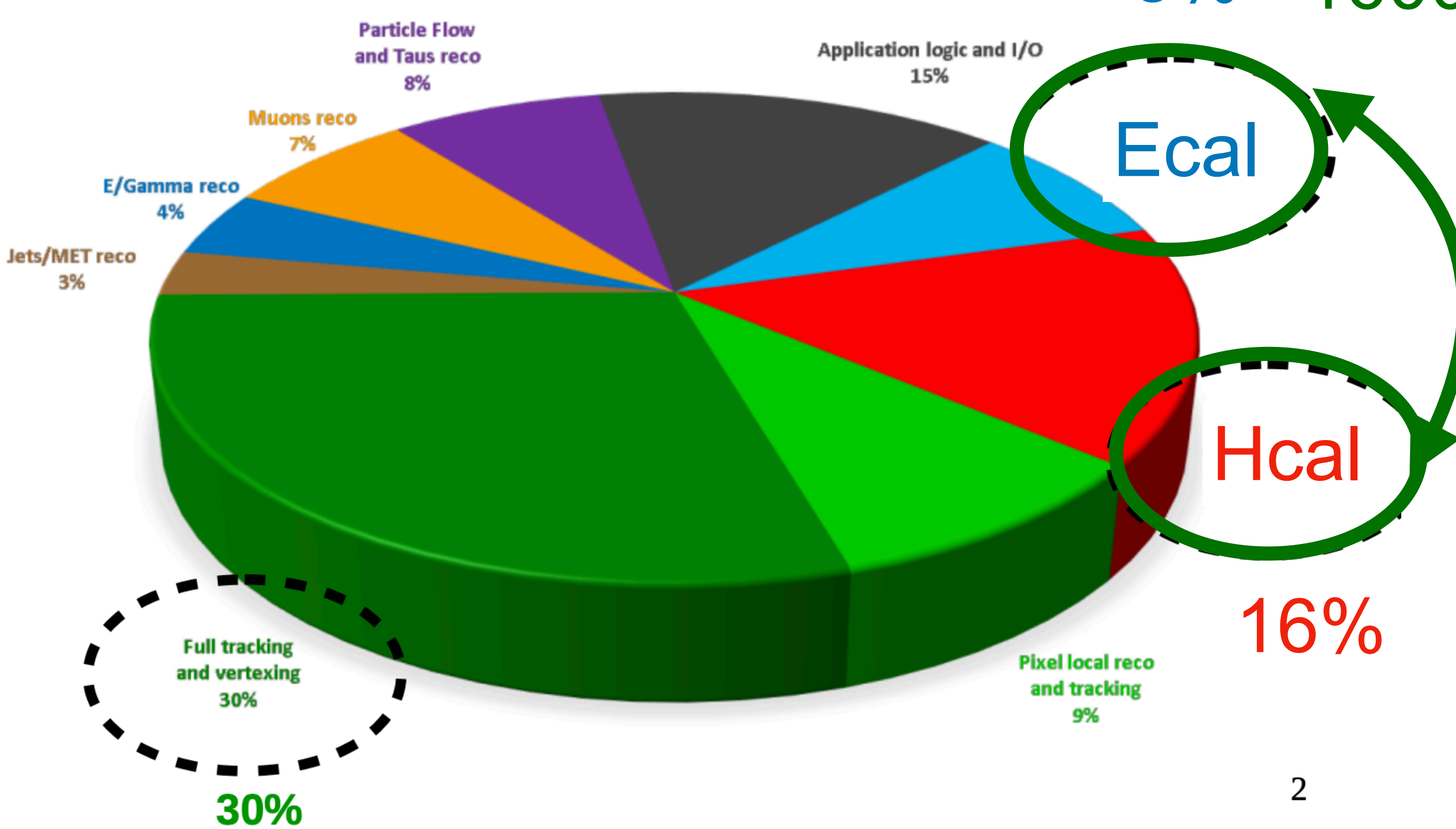


Benchmark #1

Time budget per algorithm

Energy reconstruction of
Hadronic showers
Simple energy regression
16000 times per collision

8%

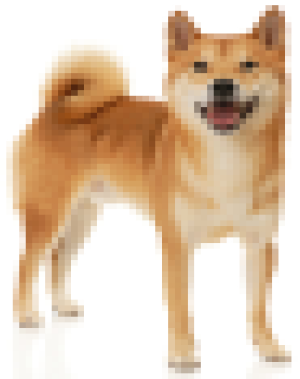


Roughly 25% of our
computing budget

16%

Network Arch
4 Layer MLP
2000 weights
Easy to put on an FPGA
7% of a Xilinx VU9P

Already developed algo w/good performance

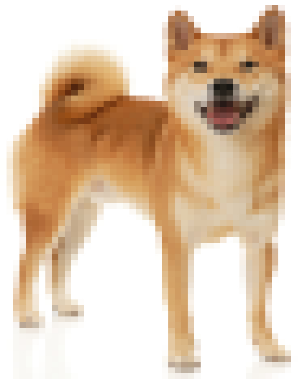


How Fast is it?

- Unroll network on the FPGA with hls4ml+SDAccel
- Actual network runs in 70ns on an FPGA with II of 5ns
    - For 16000 channels this equates to 80μs total
    - Transfer back and forth on PCIe is 700μs each way
- Current non-ML-based algorithm takes 50ms

Algo	Per Event
Old	50ms
NN CPU	15ms
NN GPU(1080 Ti)	3ms (prelim)
NN FPGA	2ms

Significant speed ups



Benchmark #2

- Resnet50 on Azure FPGA cluster with <**2ms/inference**

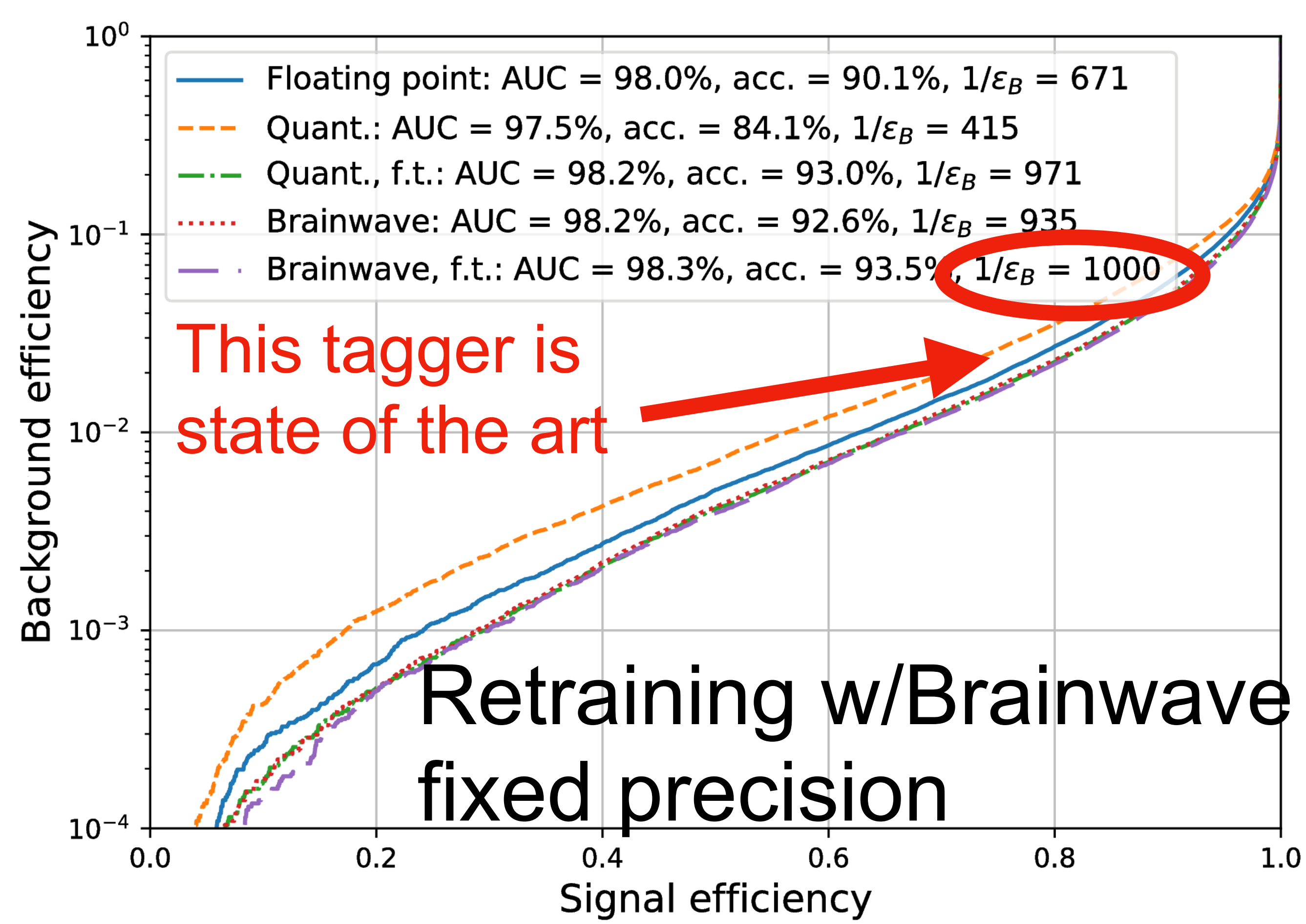
- A standard ML benchmark: Top Tagging (resnet50 for physicists)

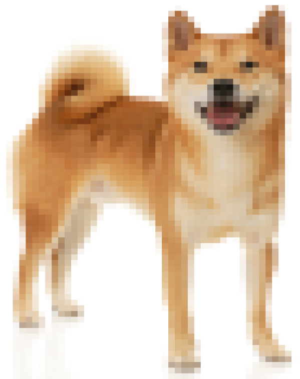
Many different Top Tagging attempts

Approach	AUC	Acc.	1/eB (@ eS=0.3)	Contact	Comments
LoLa	0.980	0.928	680	GK / SImon Leiss	Preliminary number, based on LoLa
LBN	0.981	0.931	863	Marcel Rieger	Preliminary number
CNN	0.981	0.93	780	David Shih	Model from *Pulling Out All the Tops with Computer Vision and Deep Learning (1803.00107)*
P-CNN (1D CNN)	0.980	0.930	782	Huilin Qu, Loukas Gouskos	Preliminary, use kinematic info only (https://indico.physics.lbl.gov/indico/event/546/contributions/1270/)
6-body N-subjettiness (+mass and pT) NN	0.979	0.922	856	Karl Nordstrom	Based on 1807.04769 (*Reports of My Demise Are Greatly Exaggerated: N-subjettiness Taggers Take On Jet Images*)
8-body N-subjettiness (+mass and pT) NN	0.980	0.928	795	Karl Nordstrom	Based on 1807.04769 (*Reports of My Demise Are Greatly Exaggerated: N-subjettiness Taggers Take On Jet Images*)
Linear EFPs	0.980	0.932	380	Patrick Komiske, Eric Metodiev	d<= 7, chi <= 3 EFPs with FLD. Based on 1712.07124: *Energy Flow Polynomials: A complete linear basis for jet substructure.*
Particle Flow Network (PFN)	0.982	0.932	888	Patrick Komiske, Eric Metodiev	Median over ten trainings. Based on Table 5 in 1810.05165: *Energy Flow Networks: Deep Sets for Particle Jets.*
Energy Flow Network (EFN)	0.979	0.927	619	Patrick Komiske, Eric Metodiev	Median over ten trainings. Based on Table 5 in 1810.05165: *Energy Flow Networks: Deep Sets for Particle Jets.*
2D CNN [ResNeXt50]	[illegible]	[illegible]	[illegible]	Huilin Qu, Loukas Gouskos	Preliminary from indico.cern.ch/event/745718/contributions/3202526
DGCNN	0.984	0.937	1160	Huilin Qu, Loukas Gouskos	Preliminary from indico.cern.ch/event/745718/contributions/3202526

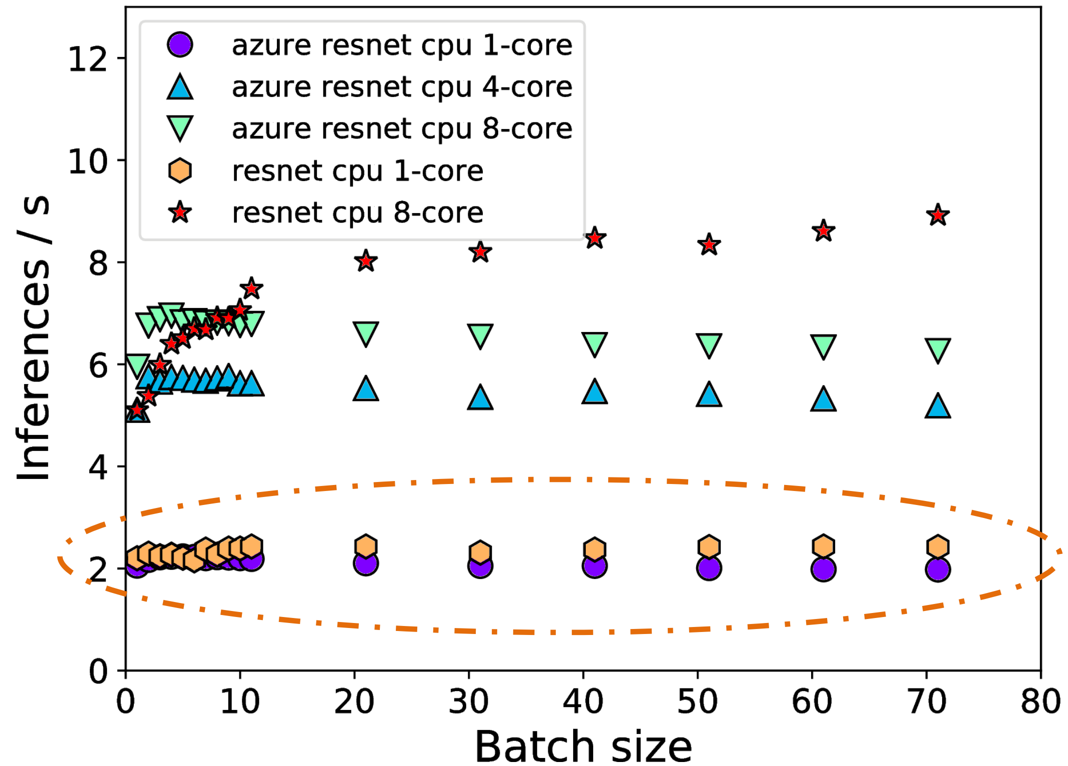
Worlds Best Tagger:
AUC=98.4% acc.=93.7% $1/\varepsilon_B = 1160$

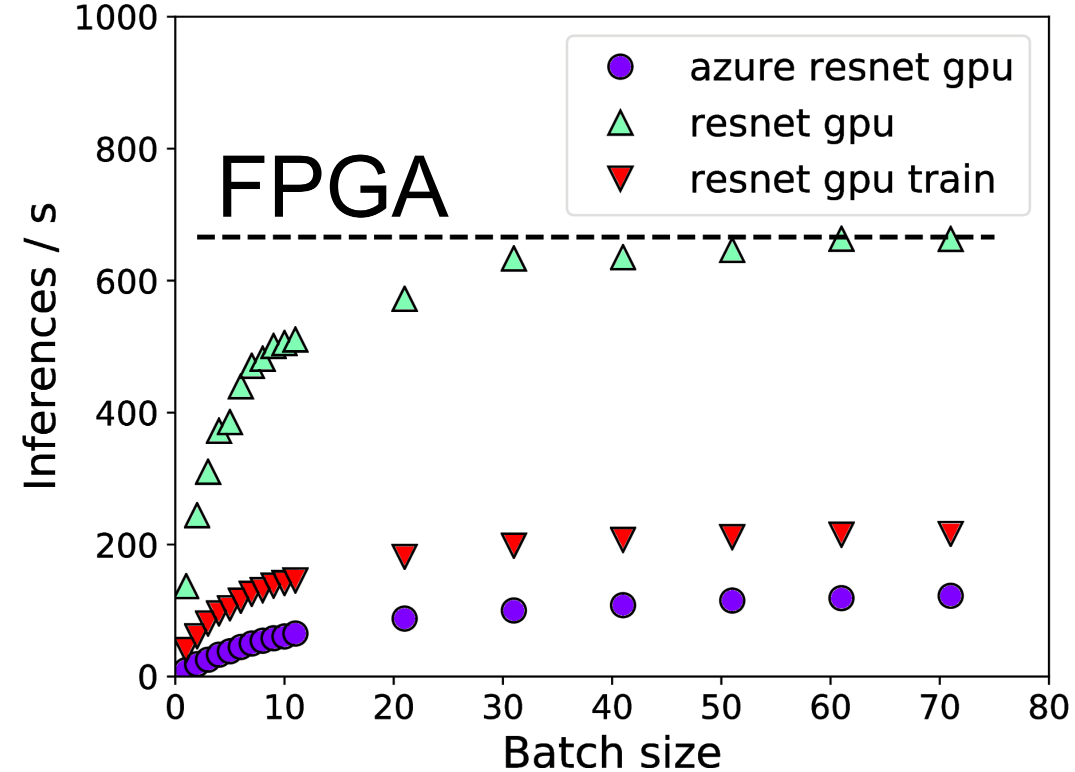
Our Tagger:
AUC=98.3% acc.=93.5% $1/\varepsilon_B = 1000$





How Fast is It?

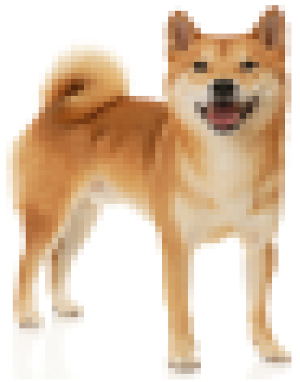




Algo	Per Event
CPU	1.75s
GPU Batch 1	7ms
GPU Batch 32	2ms
FPGA	1.7ms

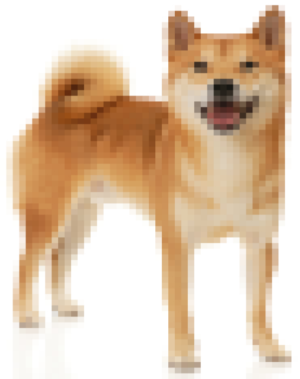
With an FPGA can get 1.7ms inference time at batch 1

With a GPU can get 2ms/img time at batch 70



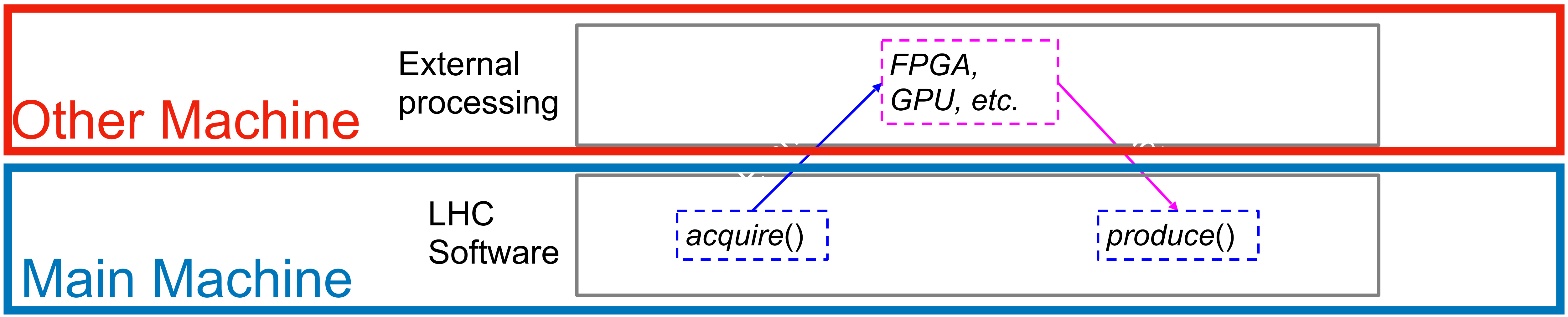
Accelerators Takeaway

- FPGAs and GPUs both work FPGAs better(low batch)/as good
- Benchmark #1
  - Latency lowest on FPGA despite a large batch process
  - Limited by I/O considerations with PCIe
- Benchmark #2
  - FPGA dominates at batch 1
  - With large throughput GPU can start to compete



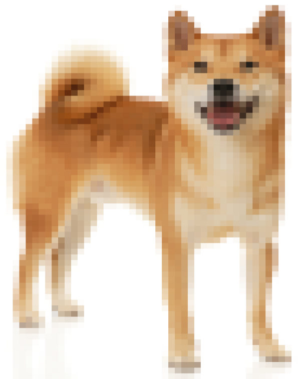
Idea #2: Services

- To run these algorithms within our software



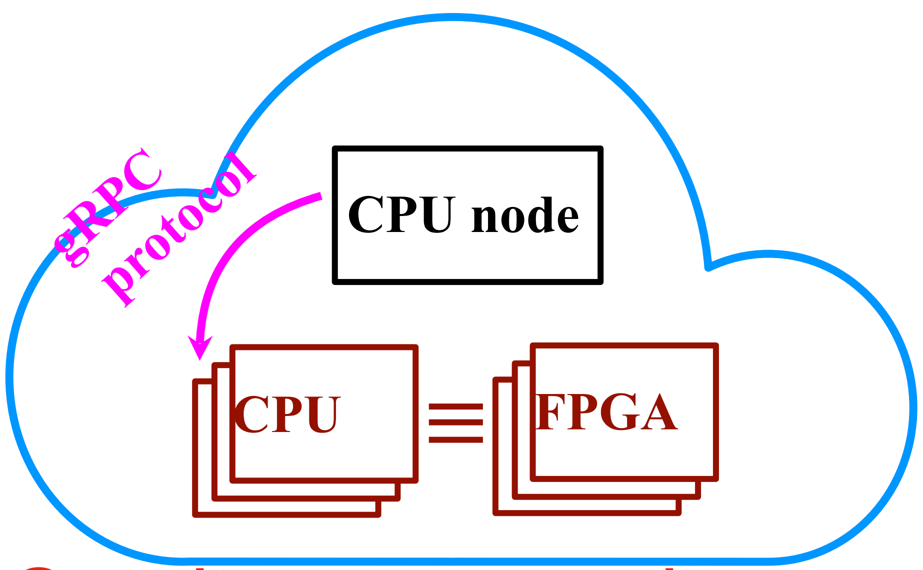
- SONIC : Services for Optimized Network Inference on Coprocessors
- Strategy
  - Use the same benchmarks as before
  - Now wrap these with gRPC protocol between different machines

Service Options

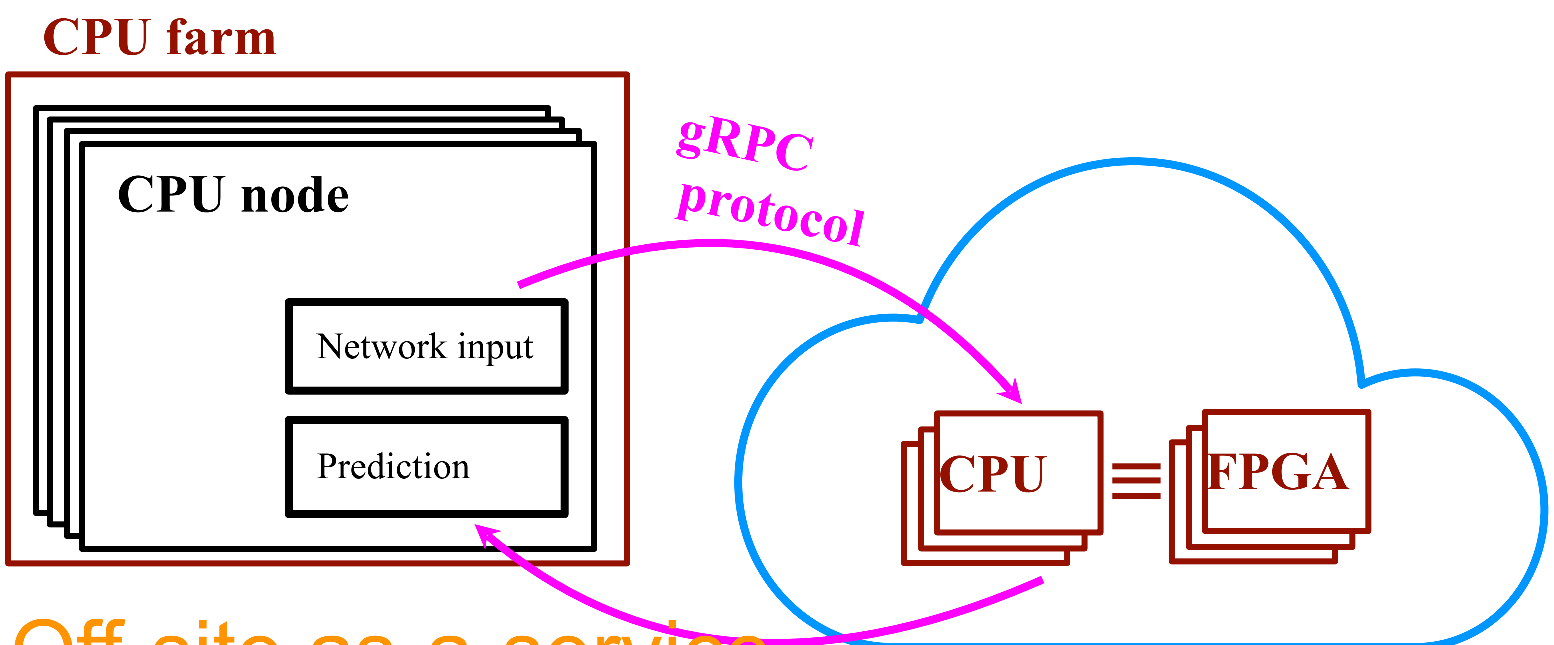


Low latency Triggering (previous slides)

Larger latency but still large throughput (future slides)

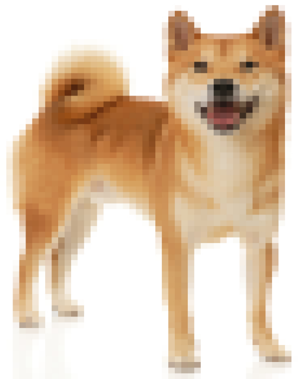


On-site as-a-service



Off-site as-a-service

When latency not critical element : can go off-site to the cloud
Here latency needs to be < 500ms (consider just on the premises)



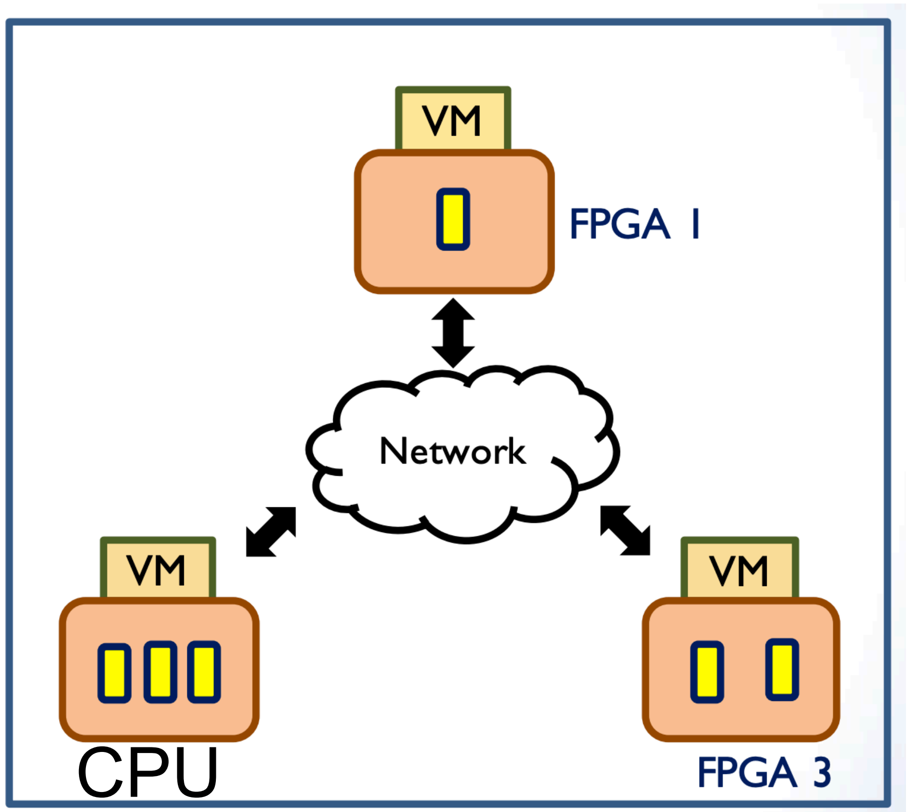
Benchmark #1

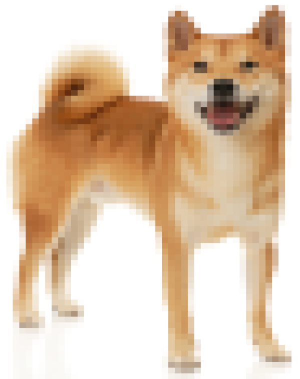
- GPU as a service
  - Using tensor-rt-server
    - Industry standard
  - Latency : 16ms

- FPGA as a service
  - Numbers TBD (<10ms)
  - Using Galapagos
    Naif Tarafdar+Paul Chow
    - Heterogenous middleware

Algo	Per Event	+On-site aaS
Old	50ms	N/A
NN CPU	15ms	N/A
NN GPU(1080 Ti)	3ms (prelim)	16ms
NN FPGA	2ms	TBD(<10ms)

8ms/event w/concurrent calls





Benchmark #2

- Three Options considered : all from computer in same cluster

GPU as a service

From local CPU
to GPU service

Batch 1 latency: 23ms
Batch 32 latency: 230ms

Azure Cluster

From local CPU
to Brainwave

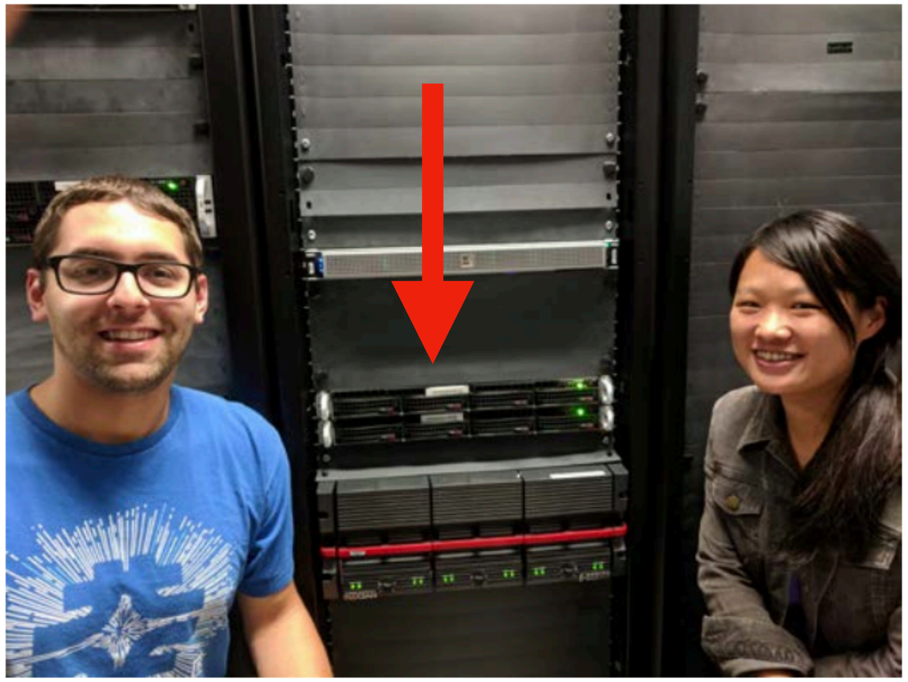
Batch 1 latency: 15ms

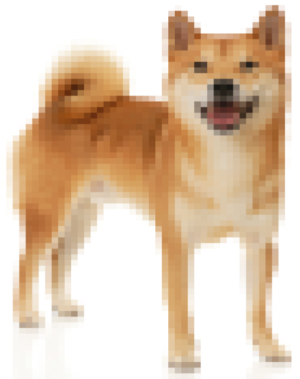
Microsoft Databox Edge

From local CPU
to FPGA system at FNAL

Batch 1 latency: 20ms

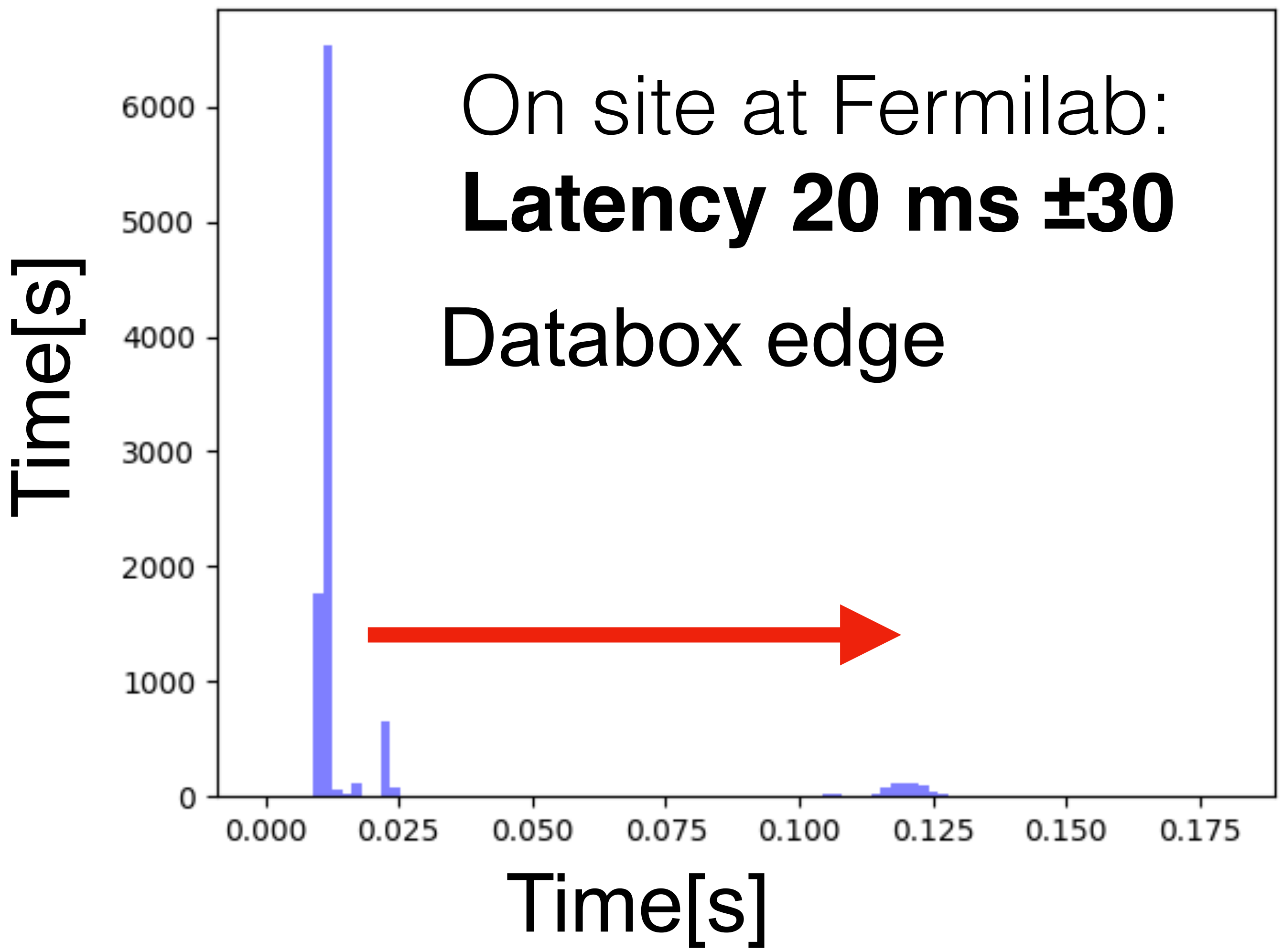
Algo	Per Event	+On-site aaS
CPU	1.75s	N/A
GPU Batch 1	7ms	23ms
GPU Batch 32	2ms	230ms
FPGA	1.7ms	15ms

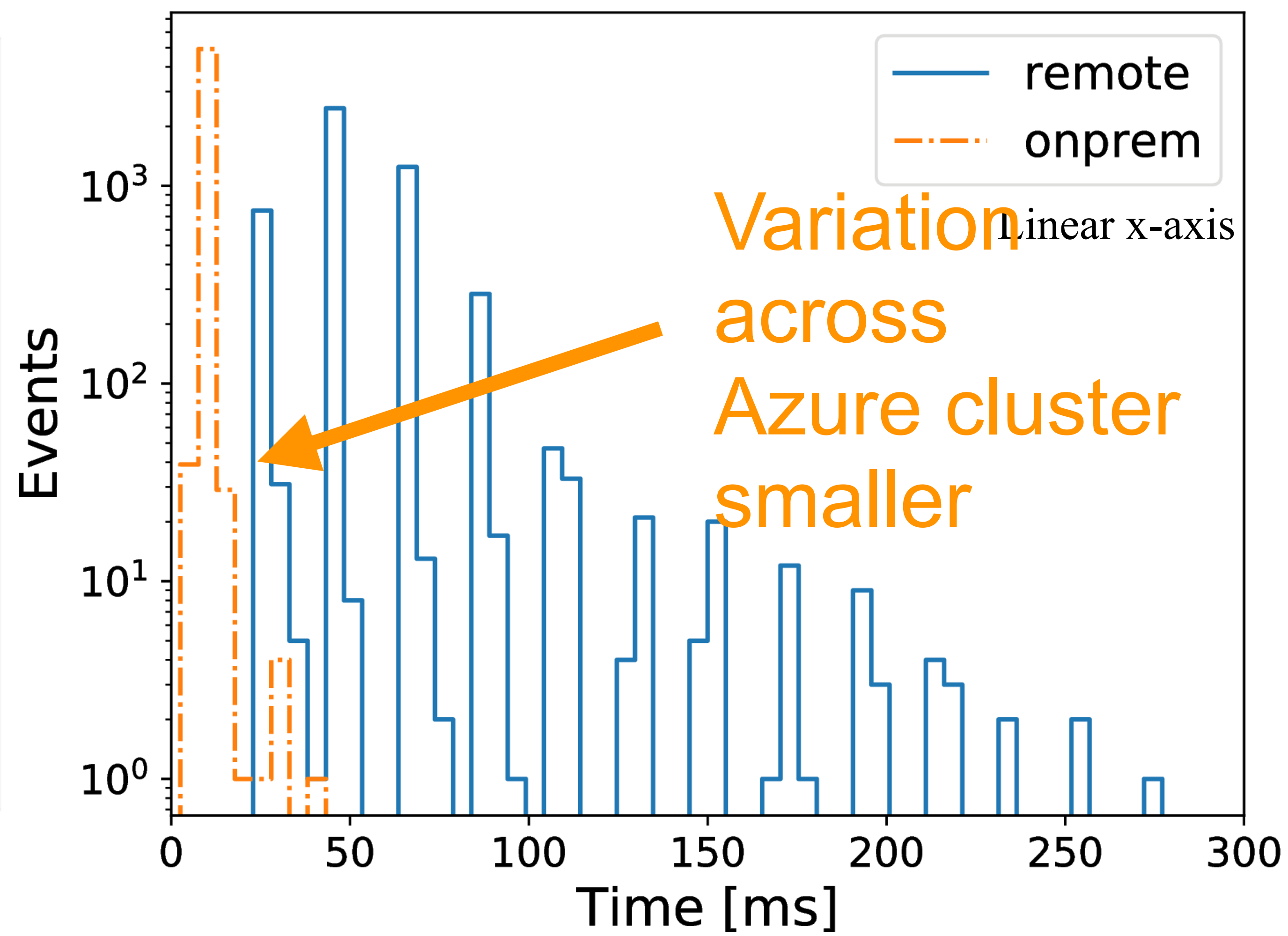


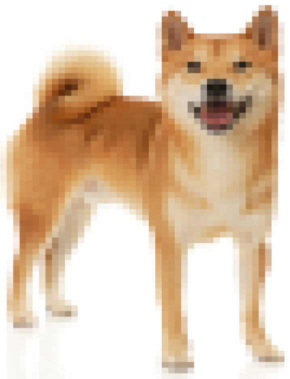


Services Takeaway

- Observe a ~10ms increase in latency when going to a service
  - Have observed large variations across network
  - Maintaining consistent network connection critical for running



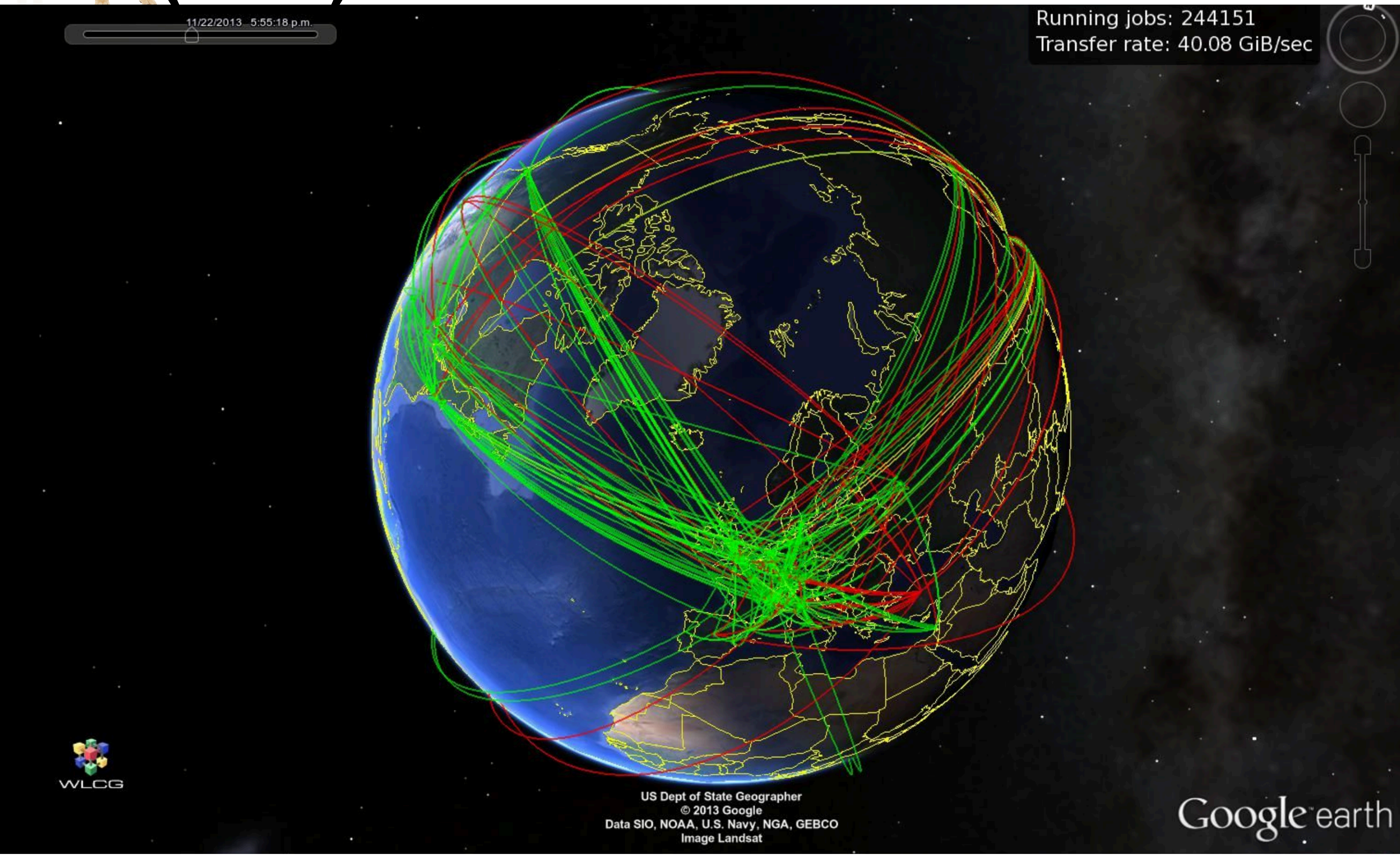




Throughput vs Latency

- Why are we limited to 500ms in latency?
  - 500ms at 100 kHz is 400 GB of data →not that much
  - With some redesign it is possible to increase this limit
    - Just need more disk as a buffer
- We still need to be able to process this data quick
  - That means we need to ensure throughput is high

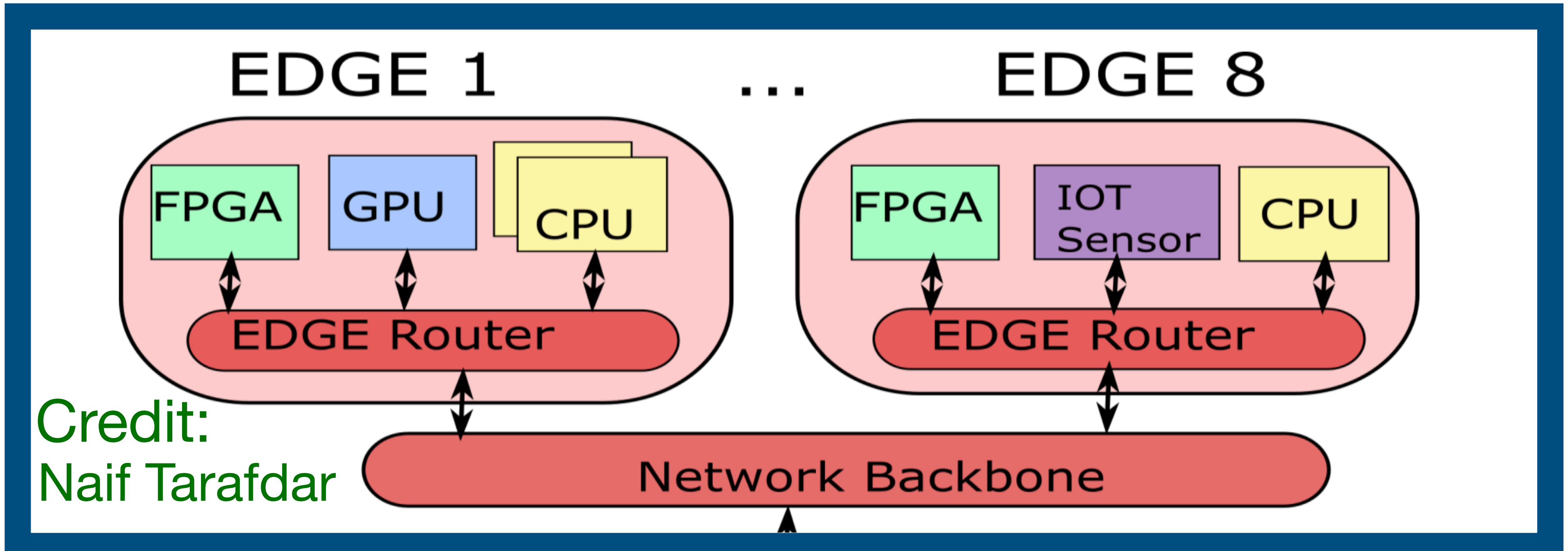
1 kHz (10s) LHC Computing Grid



1 kHz (10s)

Offline Reco

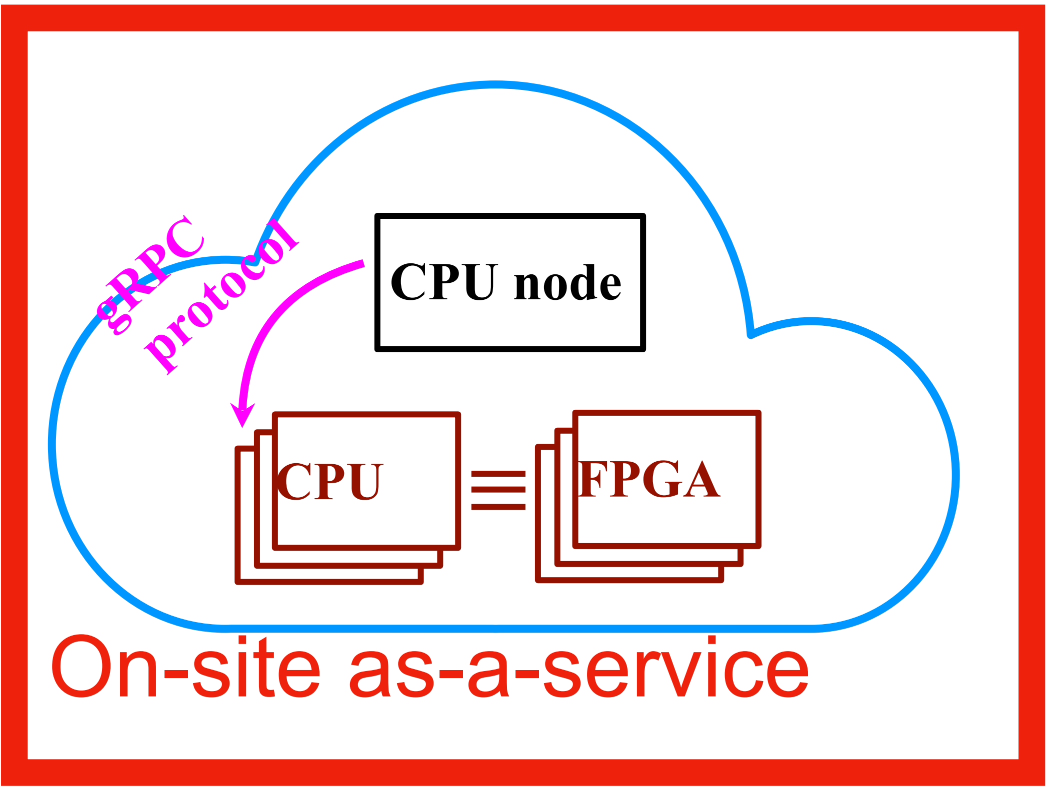
- At the final tier of reconstruction
  - Worldwide grid is roughly 0.75 Million cores 600 PB of data
  - Latency is not a critical limitation
    - Grid will have different technology all over (common protocol?)

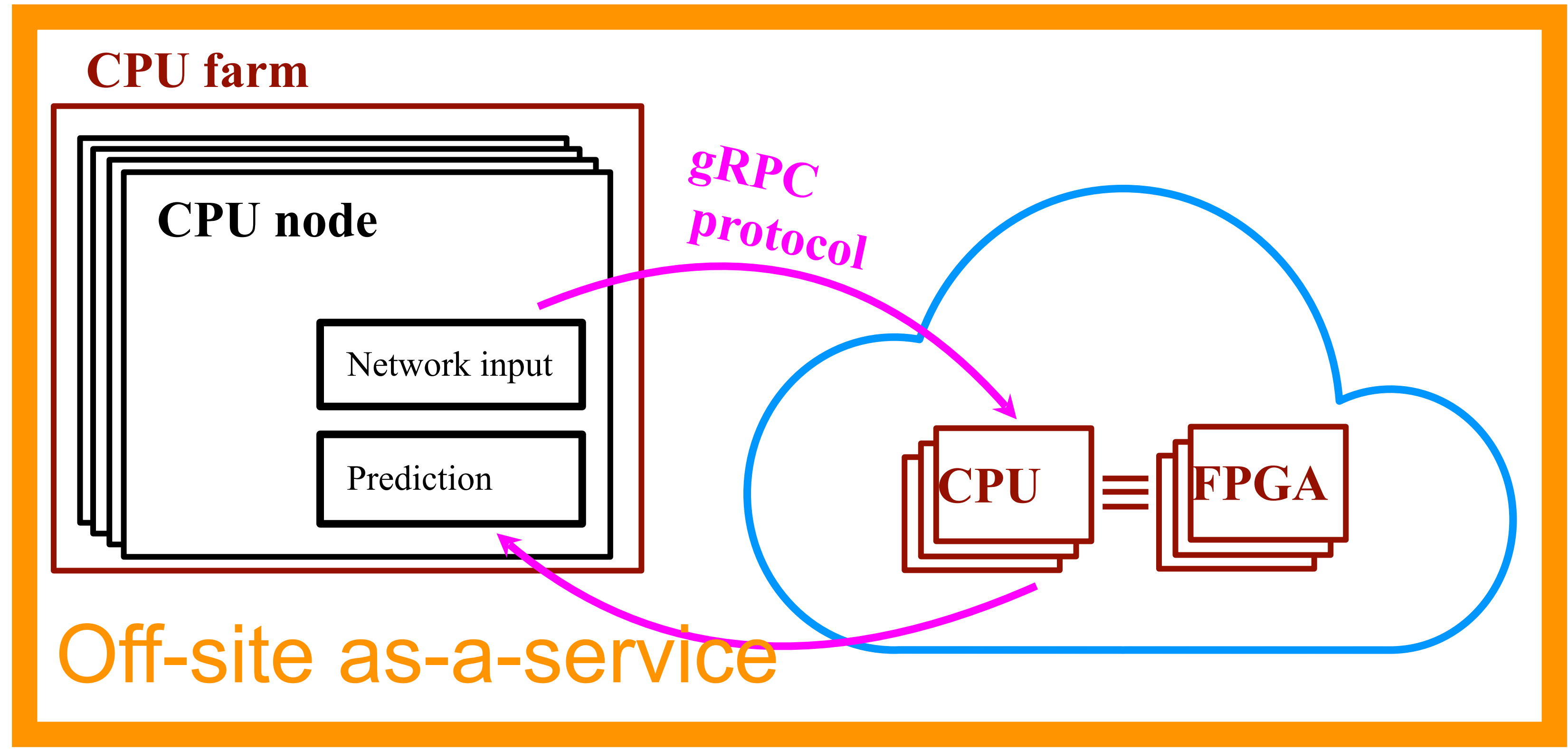


Service Options

Low latency Triggering (previous slides)

Larger latency but still large throughput (future slides)



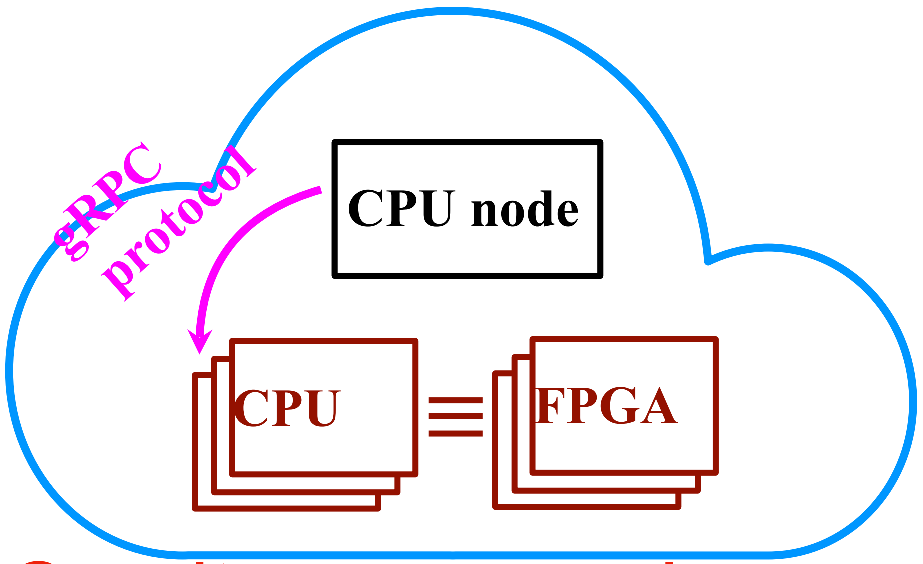


When latency not critical element : can go off-site to the cloud

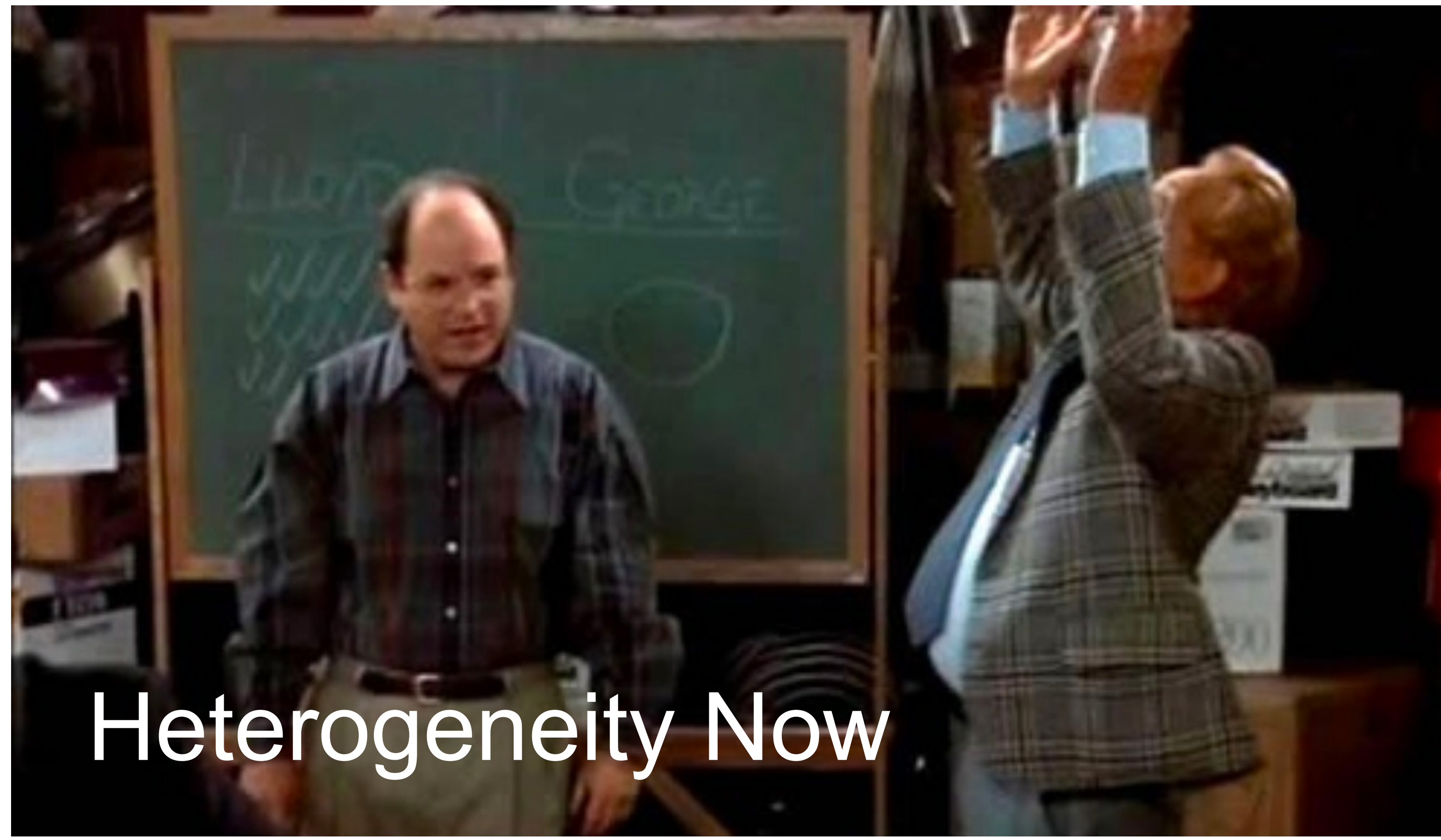
At the offline tier can switch to the cloud no→**Heterogeneity now**

Service Options

Low latency Triggering
(previous slides)



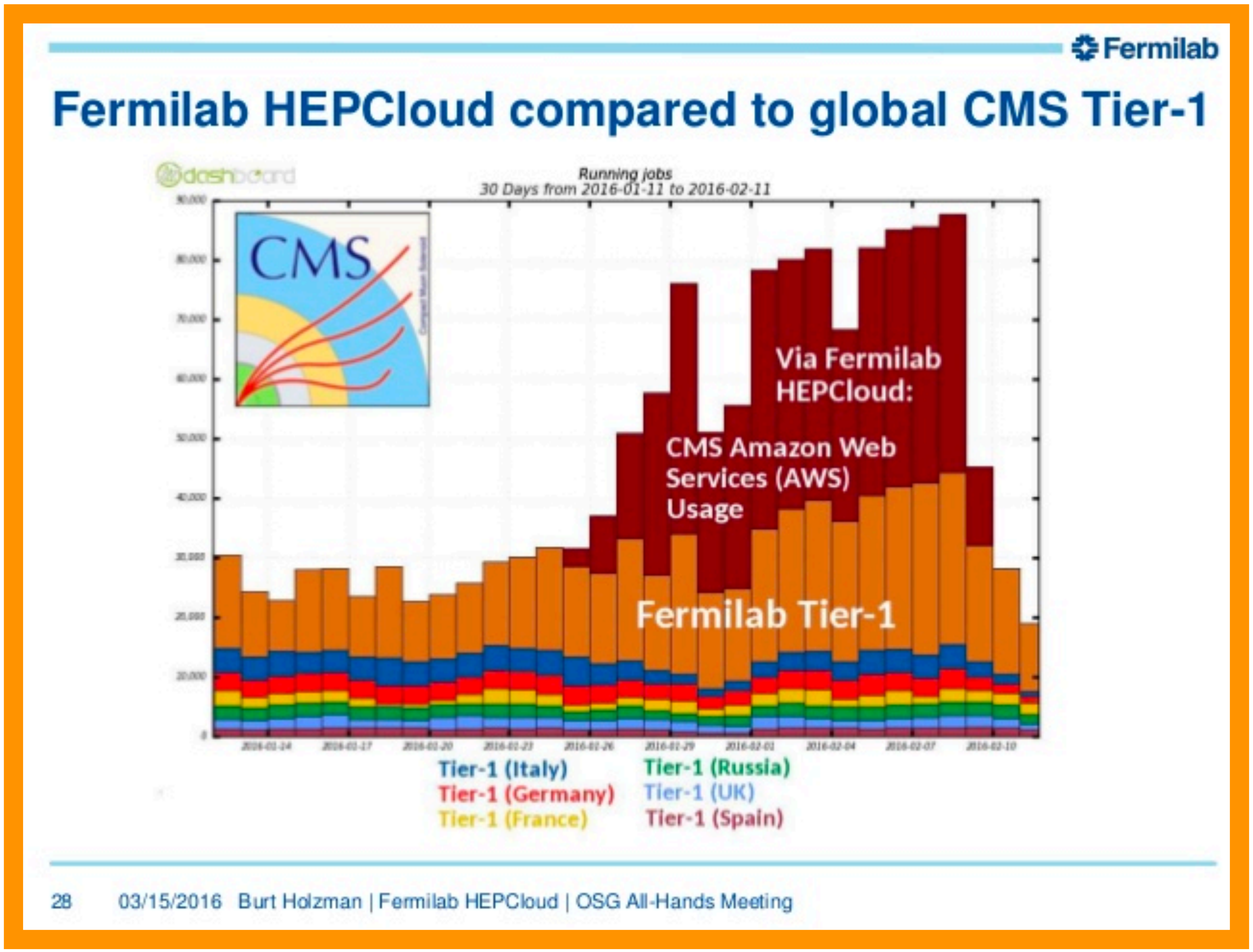
On-site as-a-service



When latency not critical element : can go off-site to the cloud

At the offline tier can switch to the cloud no→**Heterogeneity now**

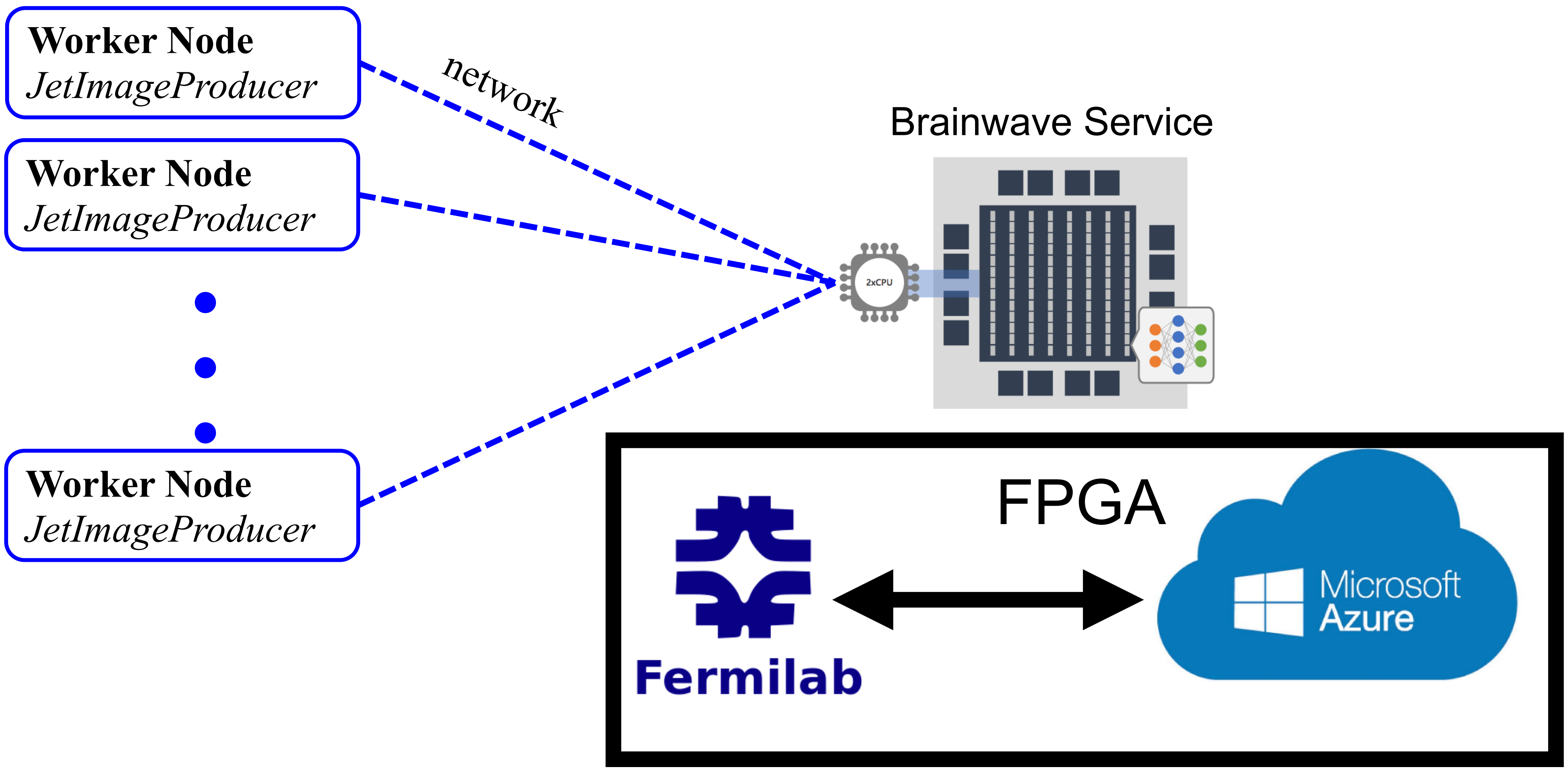
Service in Cloud



We have already done this with CPUs in the cloud

Throughput

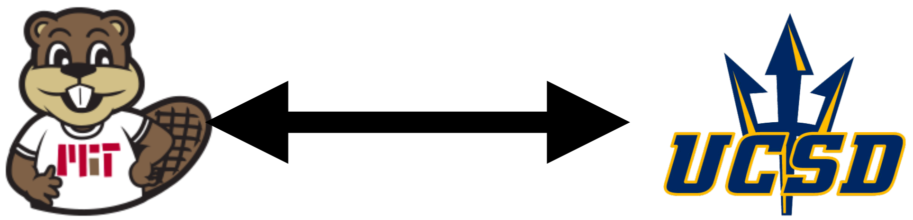
- Despite the longer latency we can have one node serve many



- With this setup how many nodes until system has to throttle down
- Bottlenecks can come from network, not just service

Benchmark #1

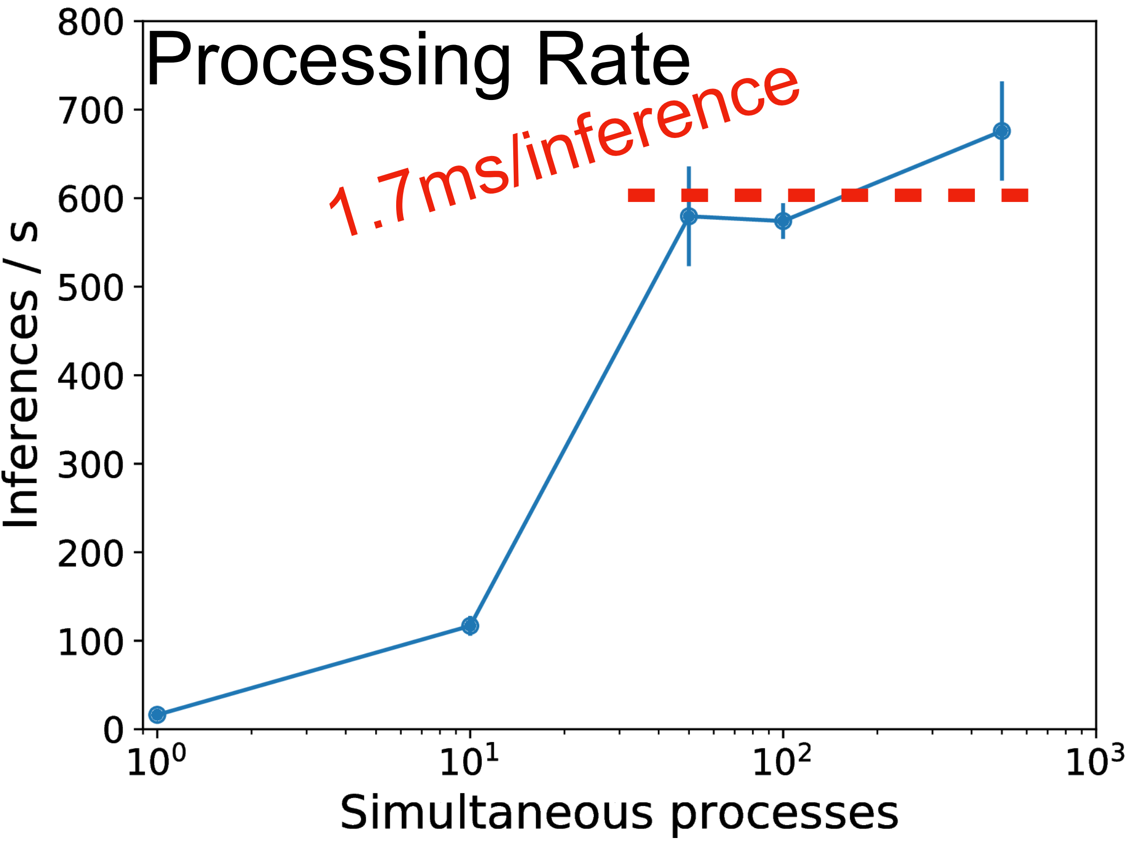
- Throughput is driven by the actual minimum latency of algo
  - For FPGA algo latency is 0.08ms→working to get there
- Cloud have to deal with additional slow down from networking

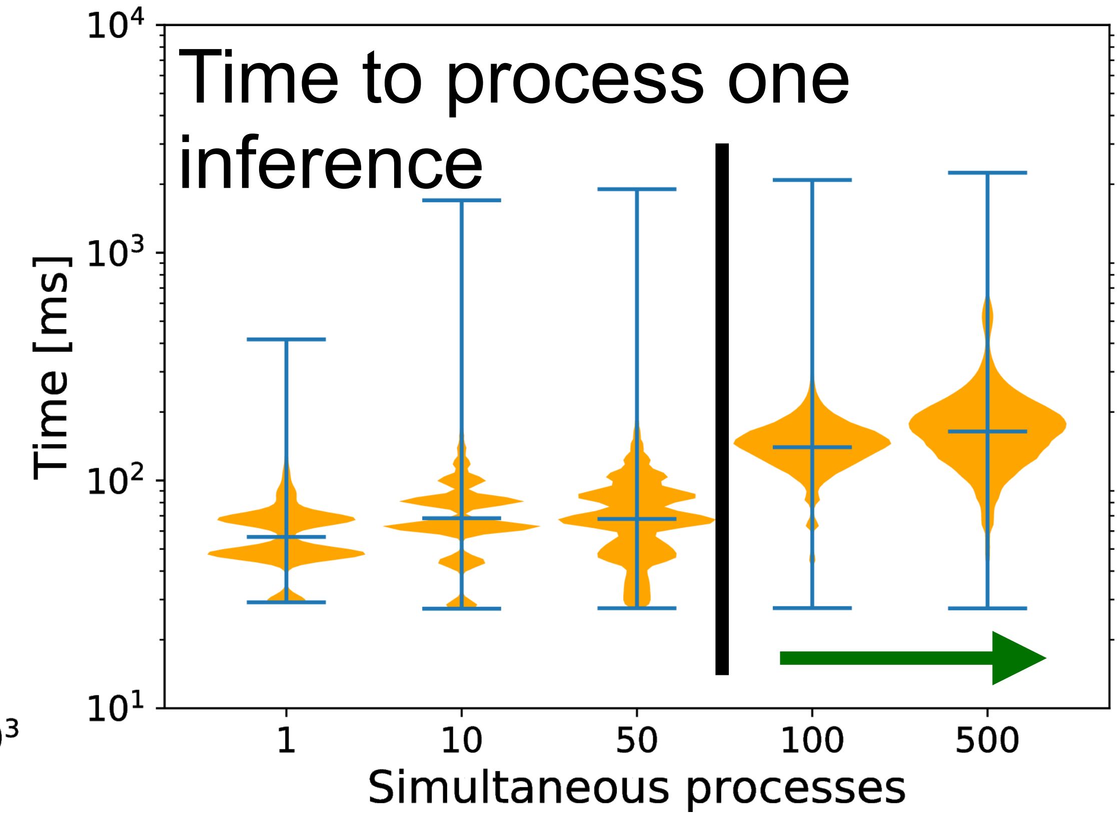


Algo	Per Event	+On-site aaS	+Cloud aaS	Ping	On/Cloud put
Old	50ms	N/A	N/A	N/A	N/A
NN CPU	15ms	N/A	N/A	N/A	N/A
NN GPU(1080 Ti)	3ms (prelim)	16ms	90ms	75ms	1ms/30ms*
NN FPGA	2ms	TBD(<16ms)	TBD	TBD	>0.1ms

*Cloud throughput on GPU still to be scrutinized

Benchmark #2





Can Serve 50-100 nodes with 1 FPGA and no loss

Algo	Per Event	+On-site aaS	+Cloud aaS	Ping	On/Cloud* put
CPU	1.75s	N/A	N/A	N/A	N/A
GPU Batch 1	7ms	23ms	97ms	75ms	5ms/20ms*
GPU Batch 32	3ms	240ms	975ms	75ms	8ms/20ms*
FPGA	1.7ms	15ms	60ms	25ms	1.7 ms

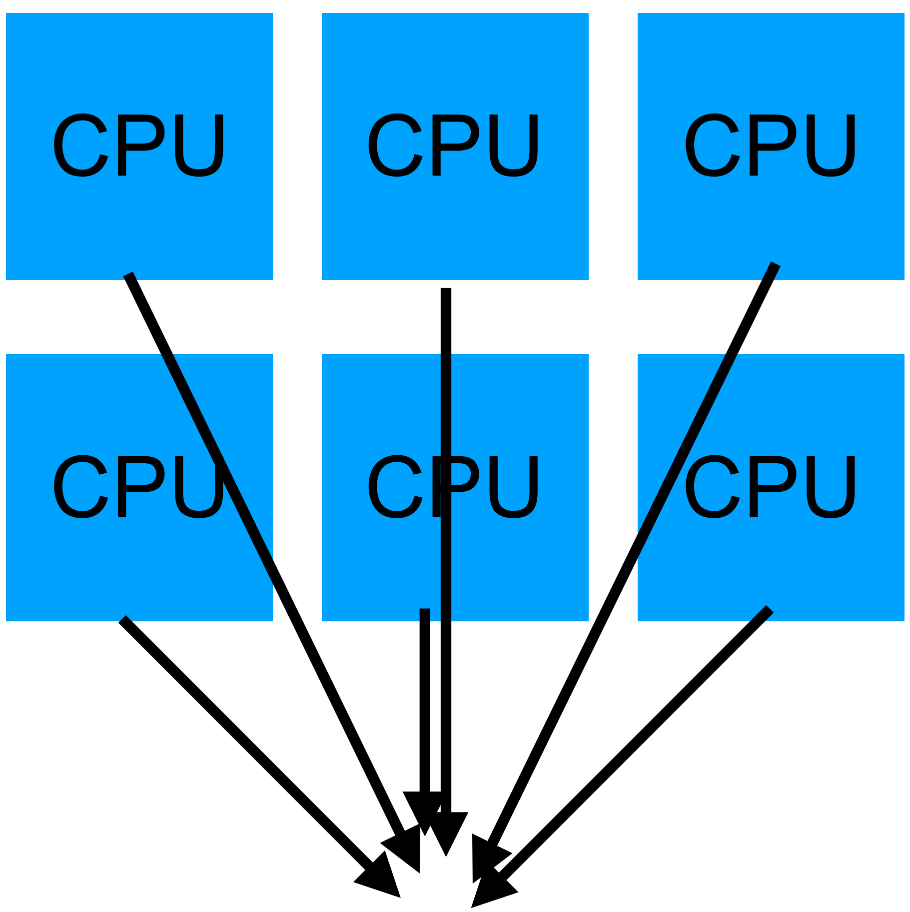
*Cloud throughput on GPU still to be scrutinized

Takeaways

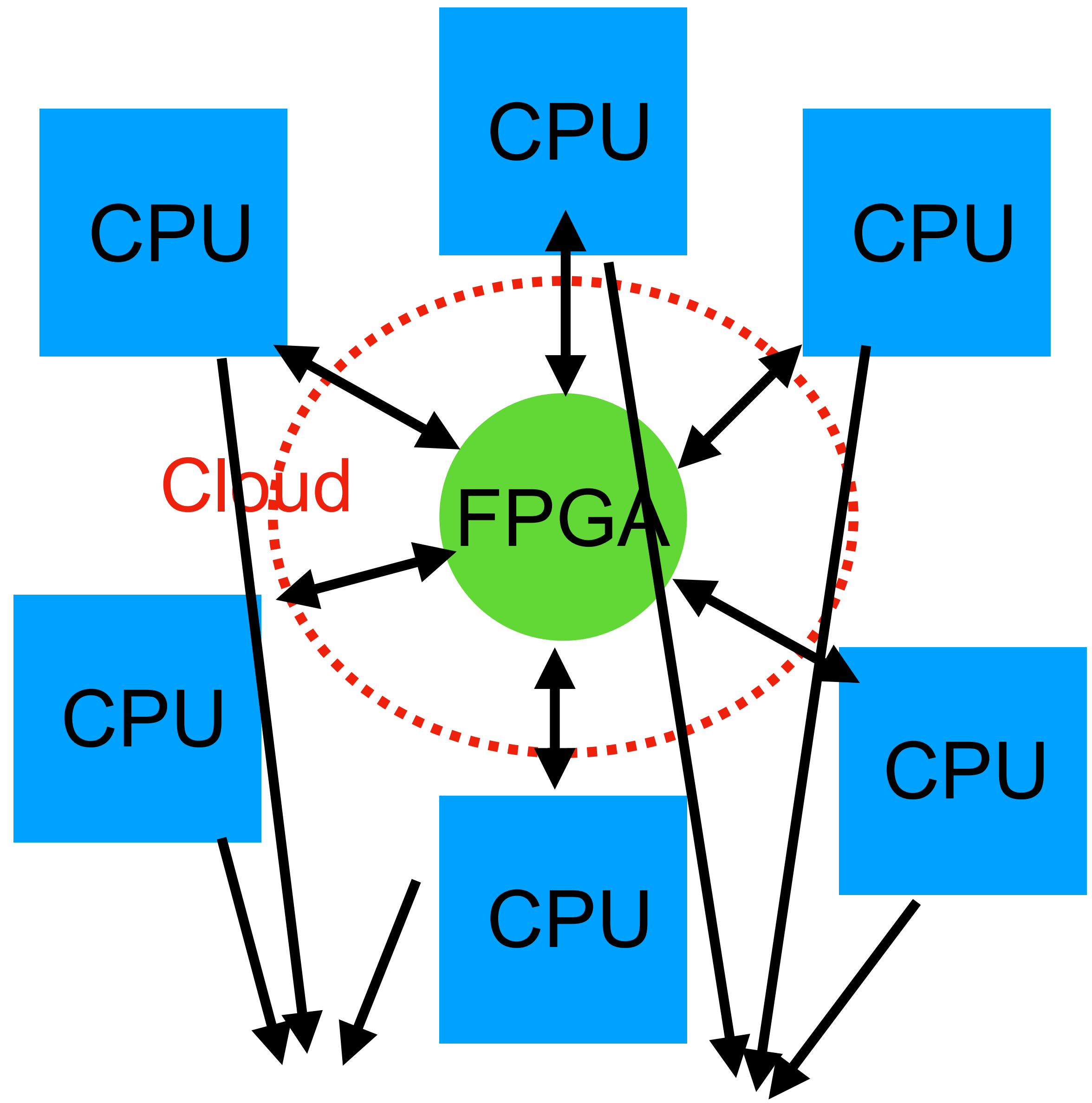
- When large speedups are present in overall throughput
  - Where as-a-service starts to really shine
  - Can think about one service for many machines
  - Will take a latency hit in our system from this
    - This is something we can deal with
- Our next step is bringing the studies to scale
  - Can we serve many thousands of processes at once?

What have we learned?

- With large speedups we can redesign our system



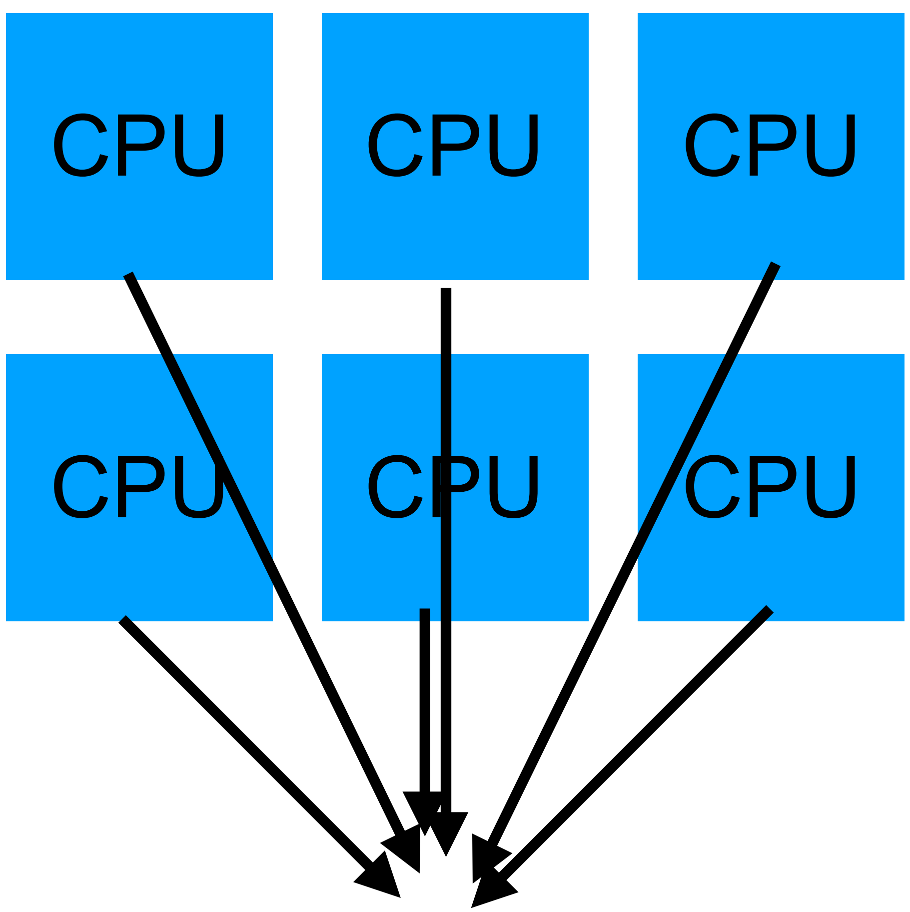
Process event by event



Process (event by event)?
outsource to aaS

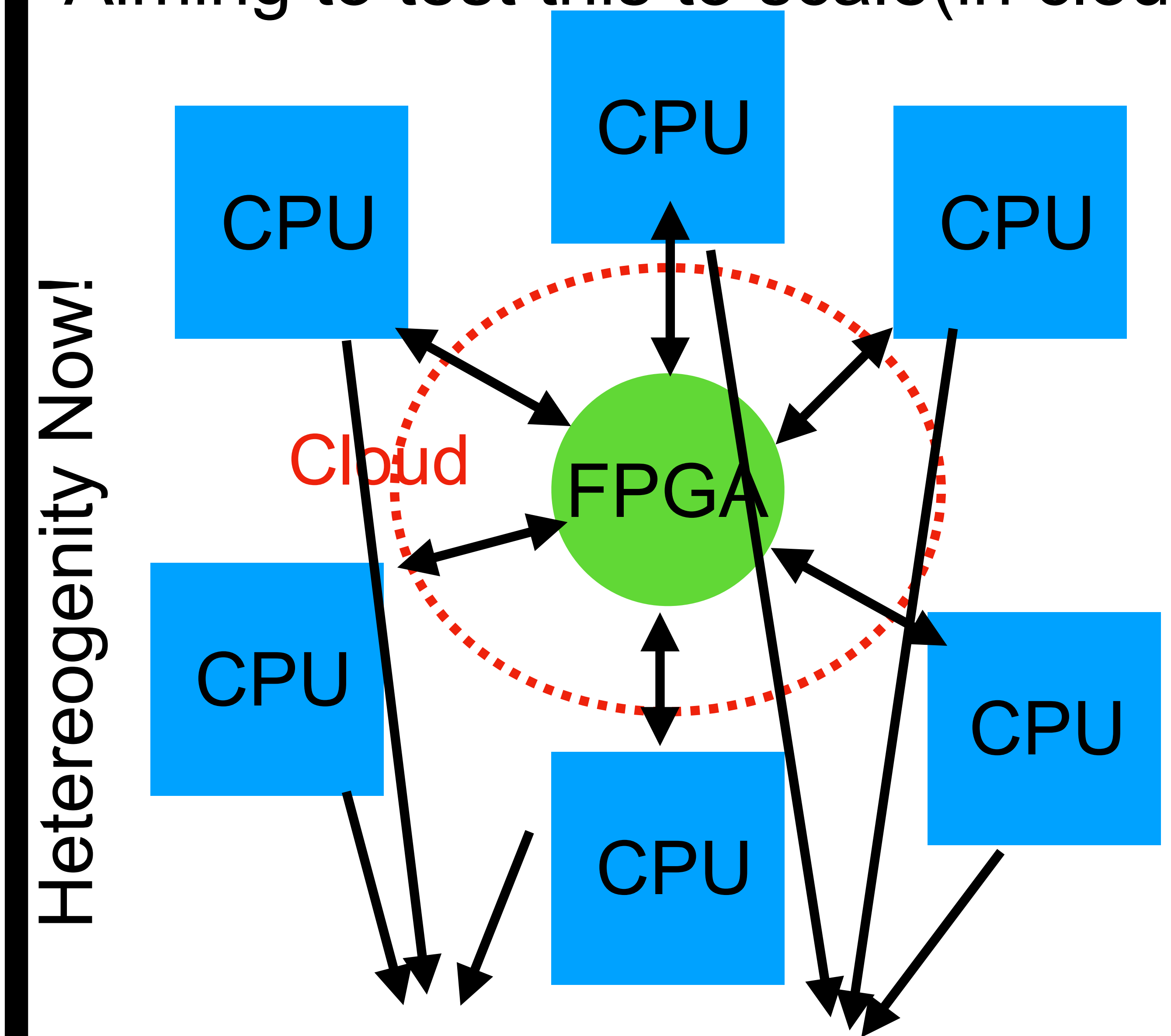
What have we learned?

- With large speedups we can redesign our system



Process event by event

Aiming to test this to scale(in cloud)



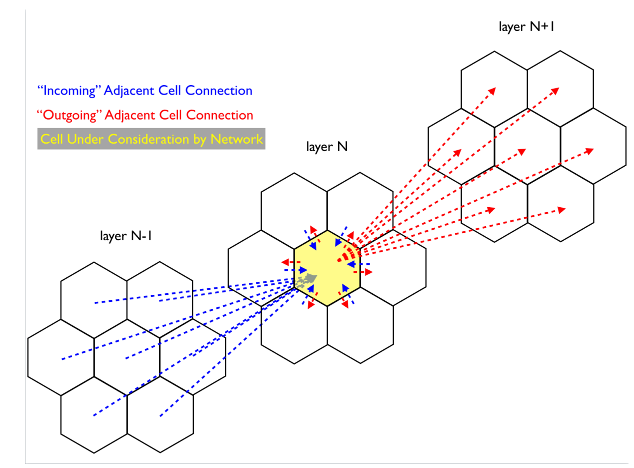
Process (event by event)?
outsource to aaS

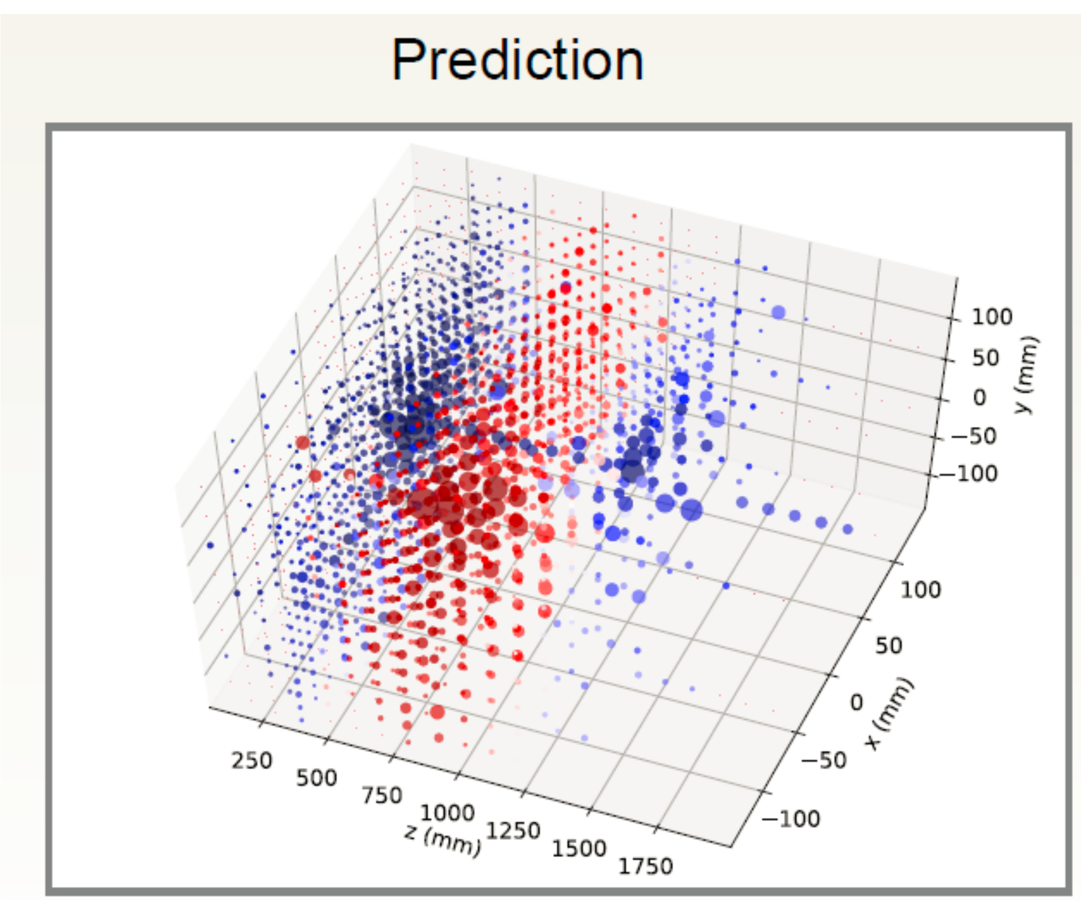
Going Beyond

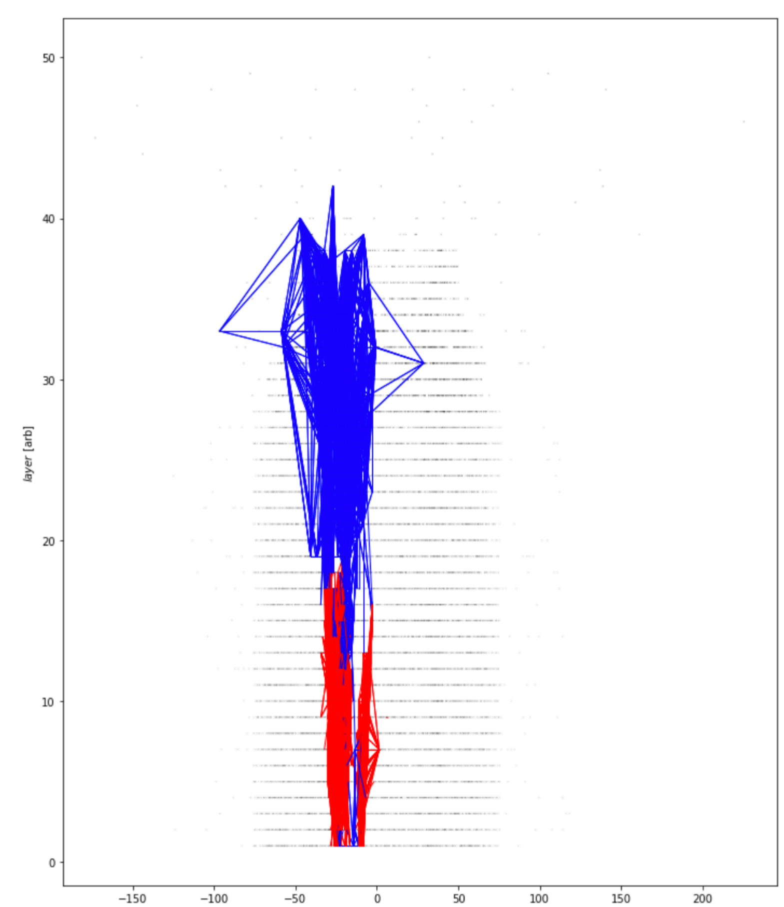
- To be really effective aim for flexibility in NN design
    - Have many different NN architectures to solve many different probs
    - Adapting to industry(Resnet50/Bert/…) not a good option
- Multi-FPGA/…. support
    - Adapting to FPGAs/… will want to avoid CPU altogether
    - Can take advantage of inherent speedups and networking on FPGA
- Throughput adaptations in our computing model
    - Latency limits not critical: can consider alternative computing models

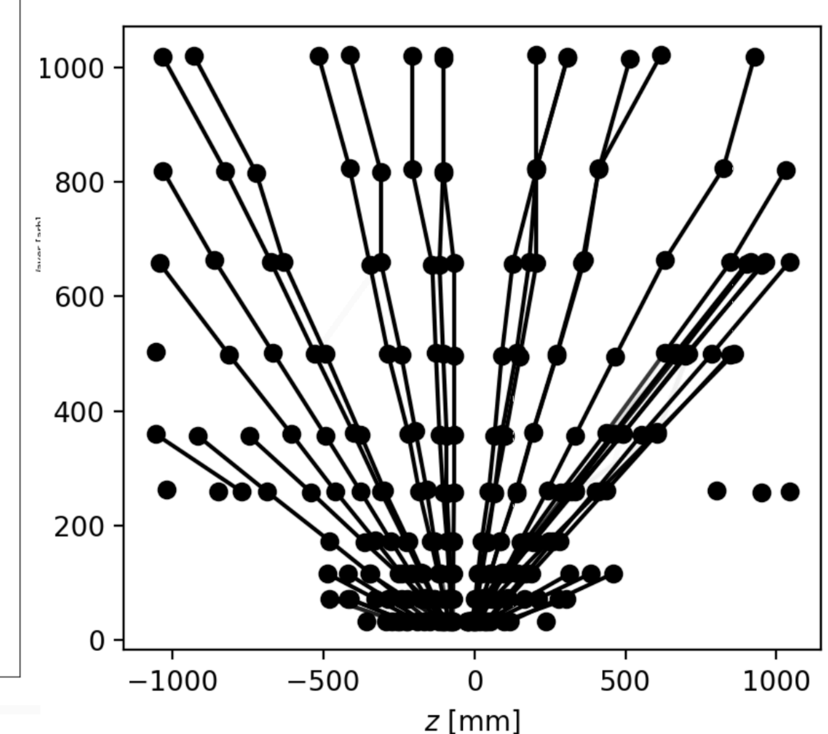
What about ML?

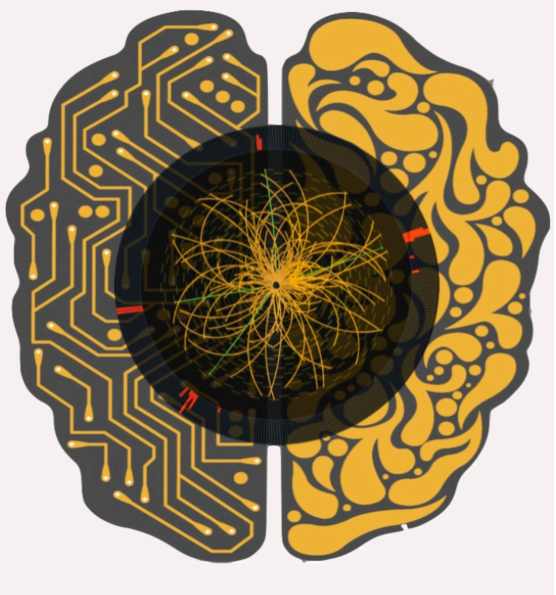
- Rapid adoption to improve reconstruction quality
- Effective for newer detectors with large numbers of channels
- Large dedicated effort within HEP comunity











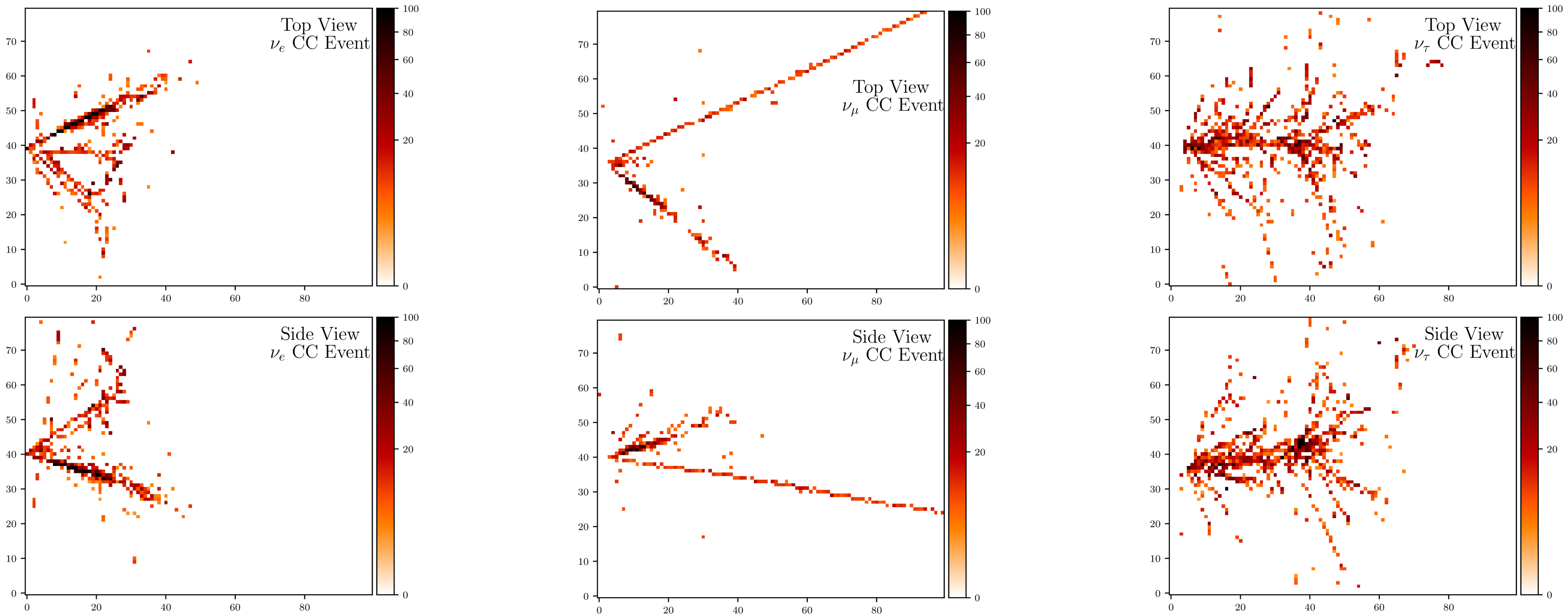
Beyond the LHC

https://fastmachinelearning.org/

- This talk has focused on data reconstruction at the LHC
- Are quickly identifying other cases with the same issues
- Have extended our collaboration to incorporate everybody
    - Inaugural workshop can be found here https://indico.cern.ch/event/822126/
    - You too can join our Fast Machine Learning effort

Lets consider a few examples

Neutrino Event Reconstruction



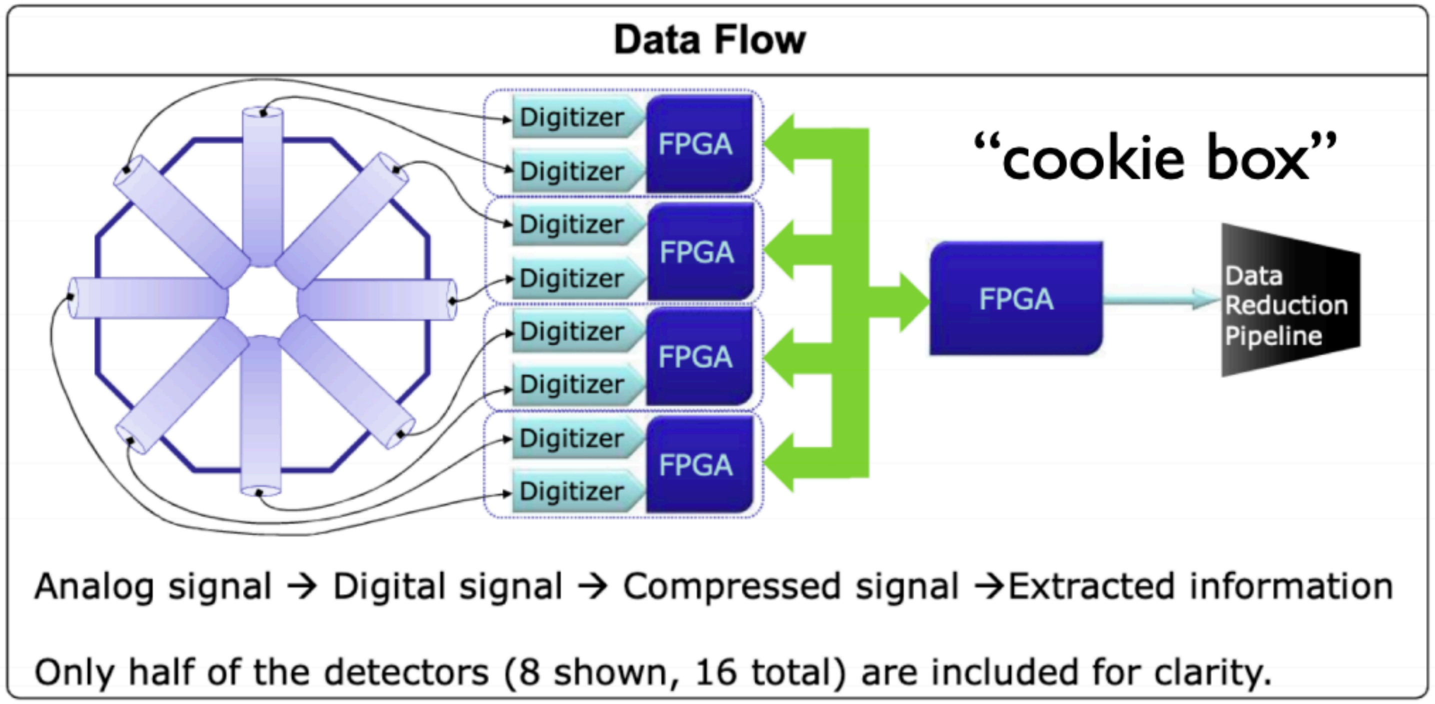
Reconstruction can be performed with a CNN (Resnet-like)

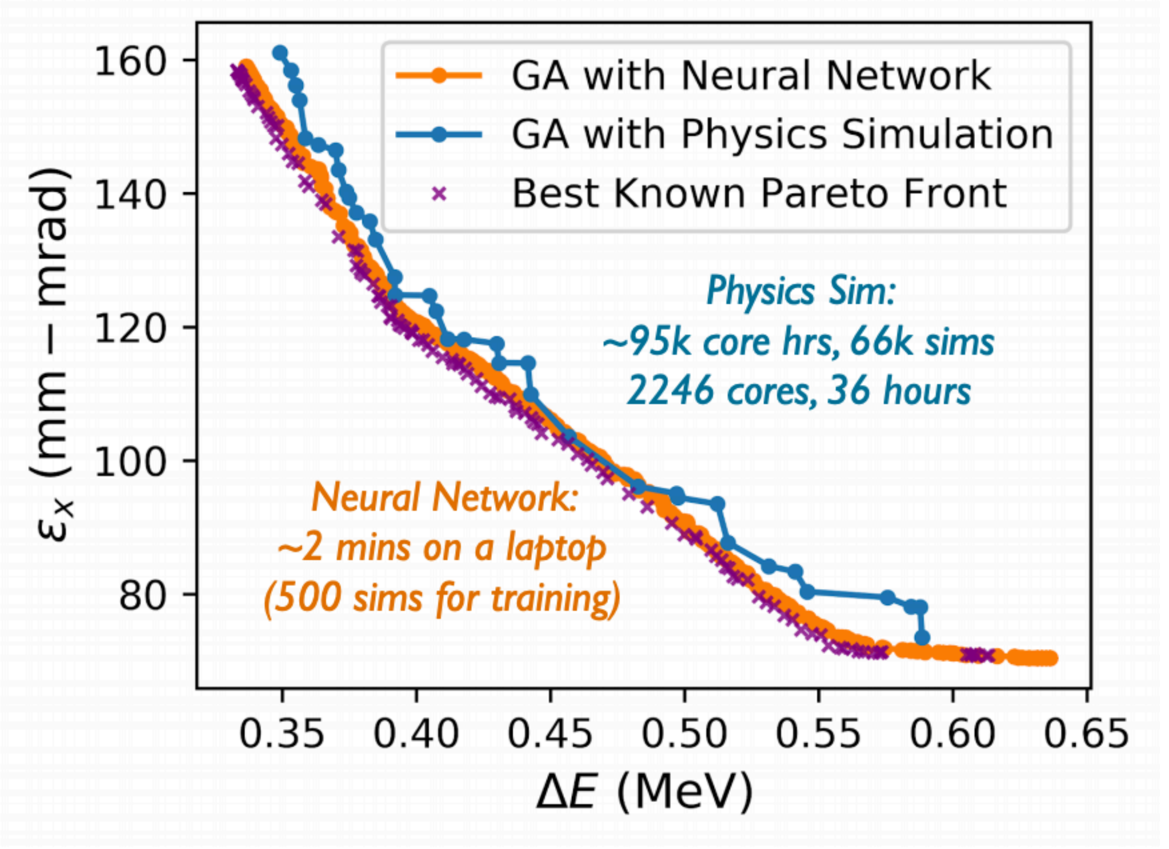
Future detectors will have to deal with 40 Tb/s of data

They will aim for per-event latency < 2ms to find Supernovae

Particle Accelerators

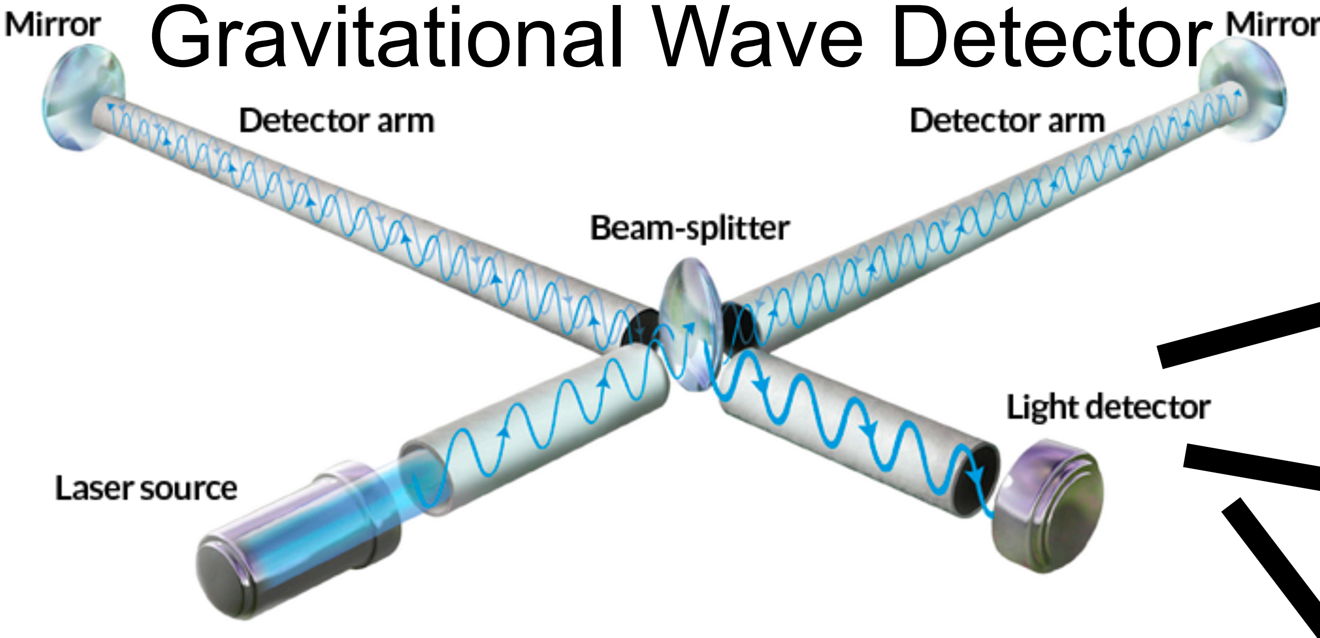
- Demands for high speed control of accelerator systems
- Large data rates to monitor and control beam dynamics
- Have had continual success with ML solutions for modeling



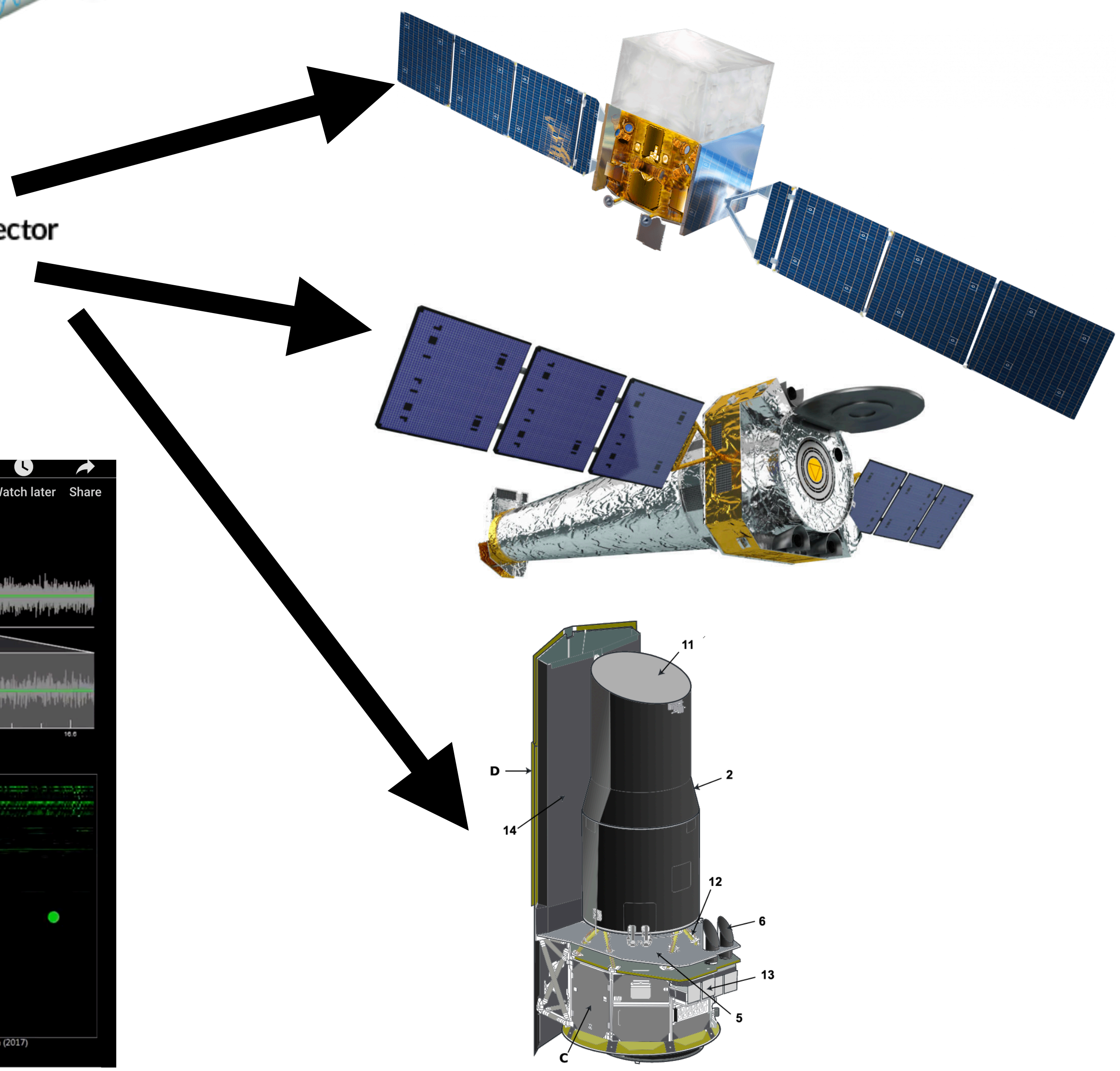


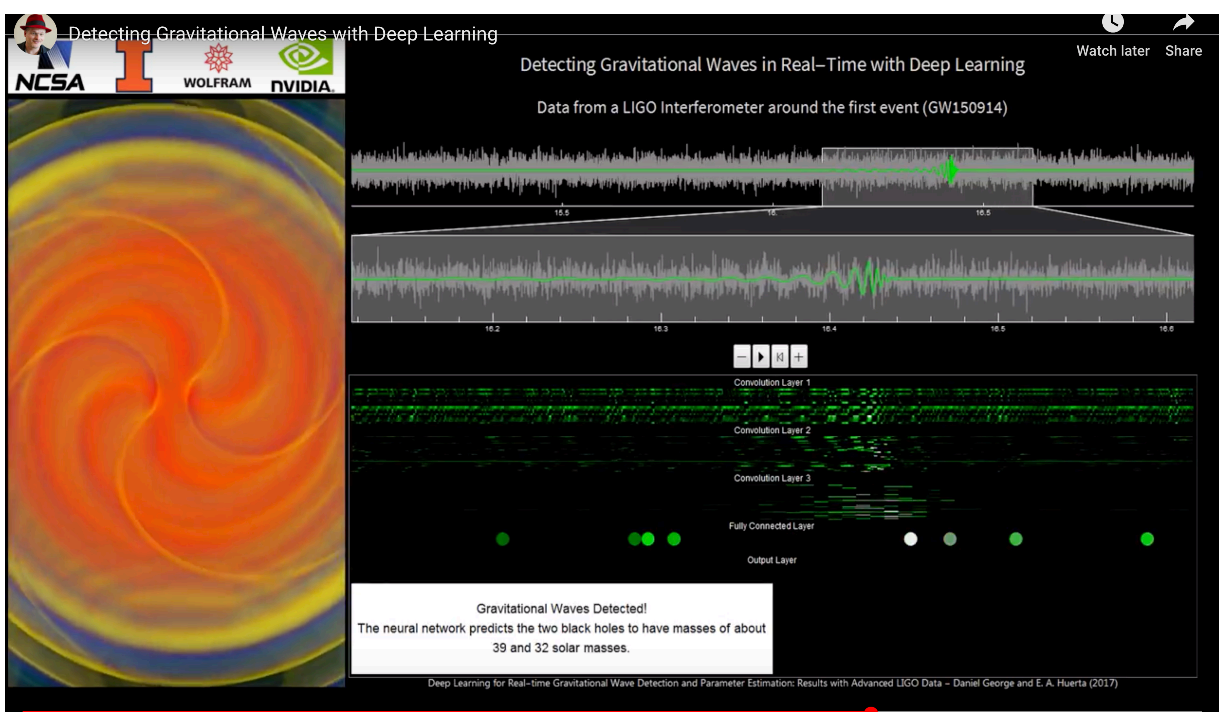
Gravitational Wave Detection

Gravitational Wave Detector



Telescopes





Fast identification of gravitational waveforms to signal satellite and other telescopes for astronomical phenomenon multi-messenger astronomy

Astrophysics

Lens type

Survey	Galaxy	Quasar	Supernovae
Today (all)	**1000**	**<50**	**2**
DES	**2,000**	**120**	**5**
LSST	**120,000**	**8,000**	**120**
Euclid	**170,000**	**-**	**-**

Nord+2016; Collett+2015; Gavazzi+2008; Oguri+Marshall, 2010

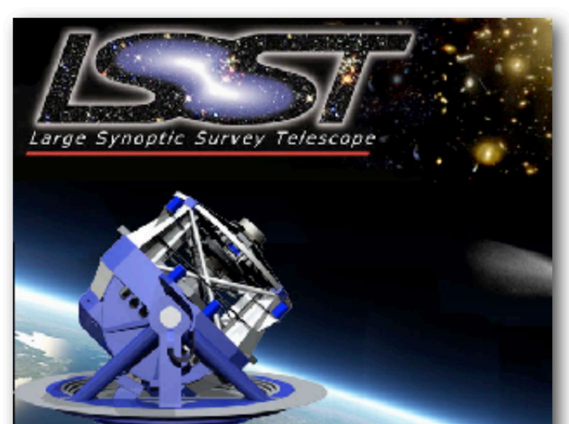
LSST will produce over **10 million** transient alerts per night.

With LSST in 2022 Astrophysics datasets reach petabyte data scales with large and complicated feature analysis

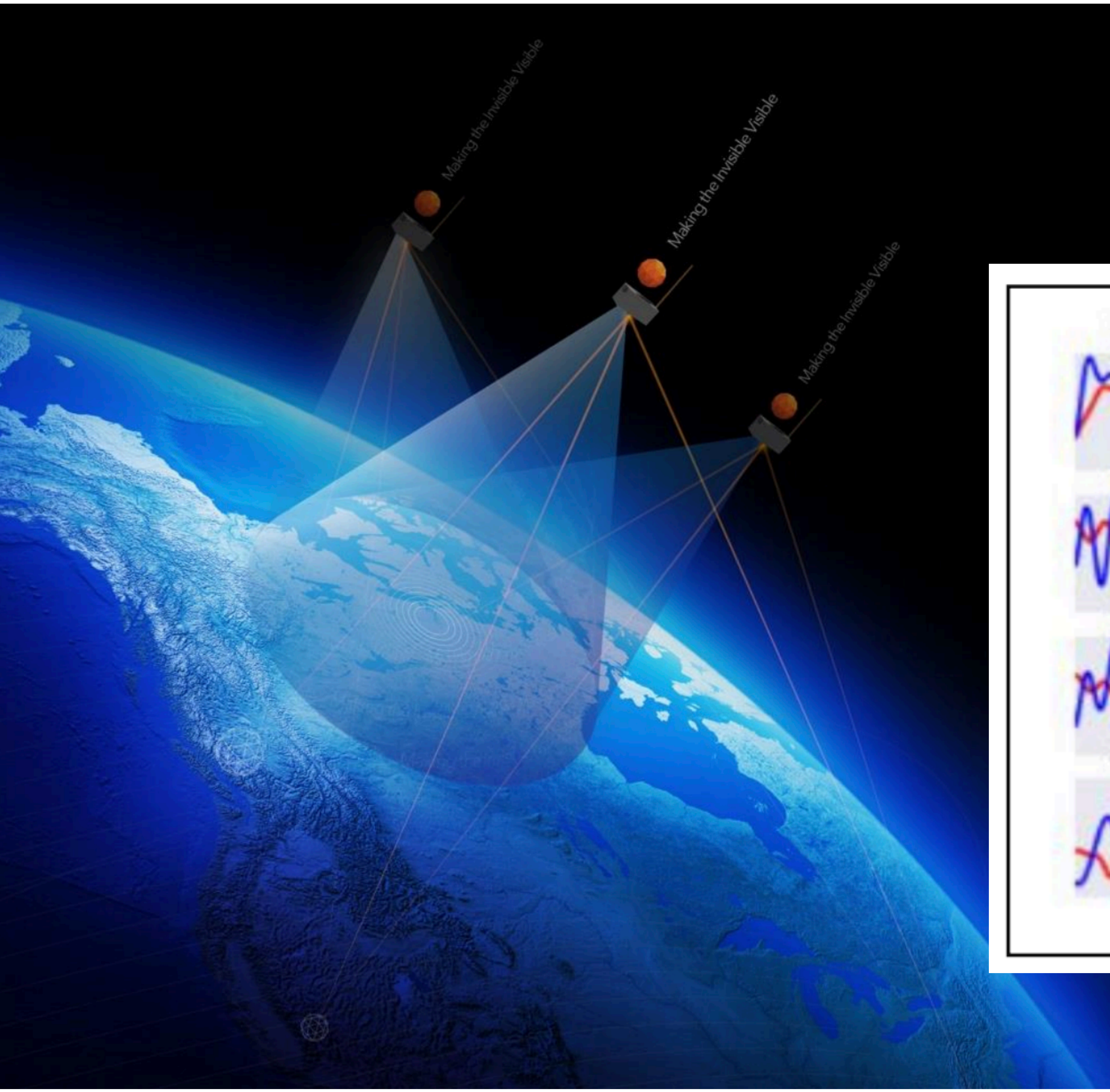
SDSS I-II
2000-08
2.5-meter mirror
O(10^8) Galaxies
10k sq. deg.
0.2 TB/Night

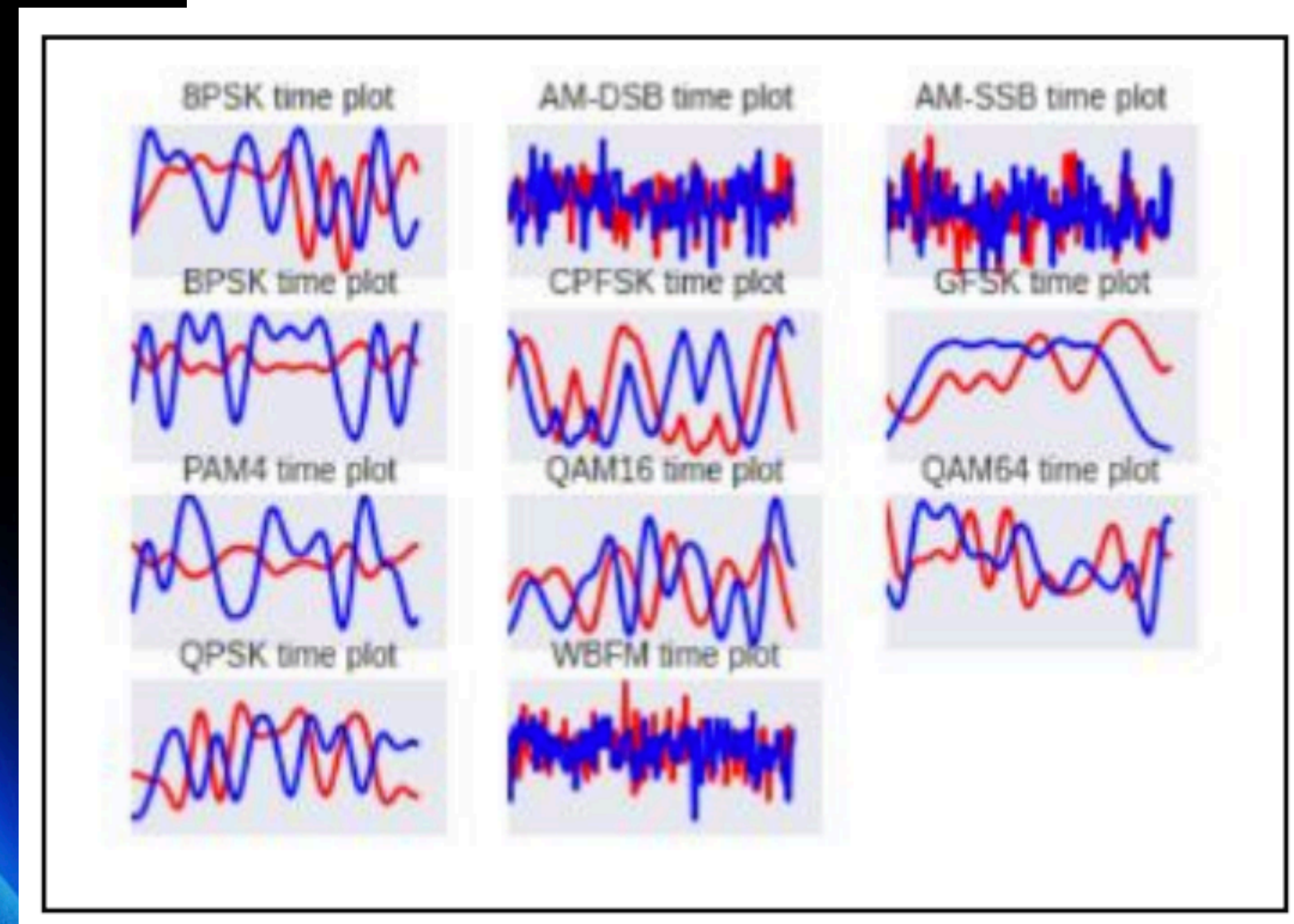
DES
2013-18
4-meter
O(10^8) Galaxies
5k sq. deg.
1 Tb/Night

LSST
2022-32
8.4 -meter
O(10^{10}) Galaxies
20k sq. deg.
20 Tb/Night



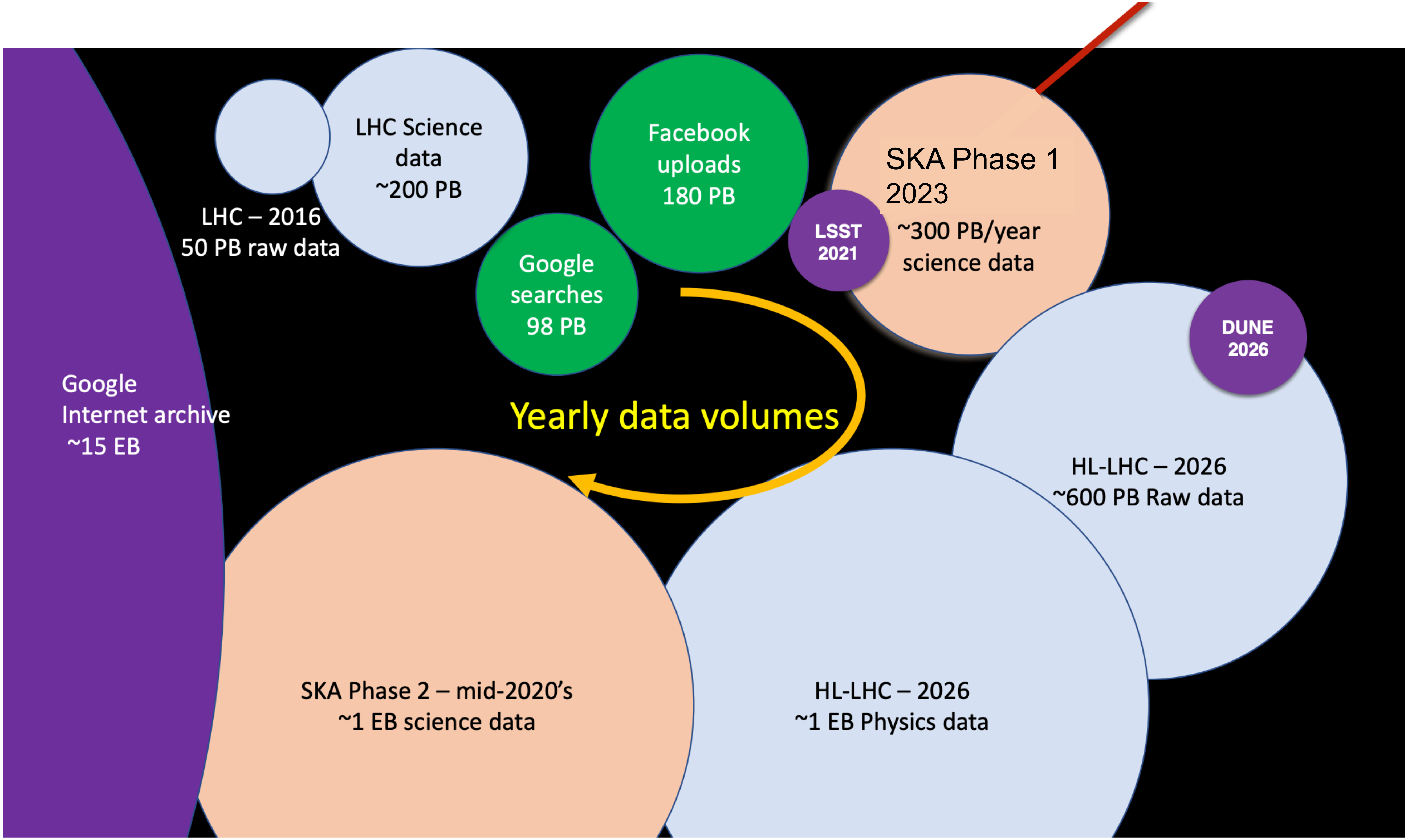
Identification of transients require real time processing of all data


Making the Invisible Visible

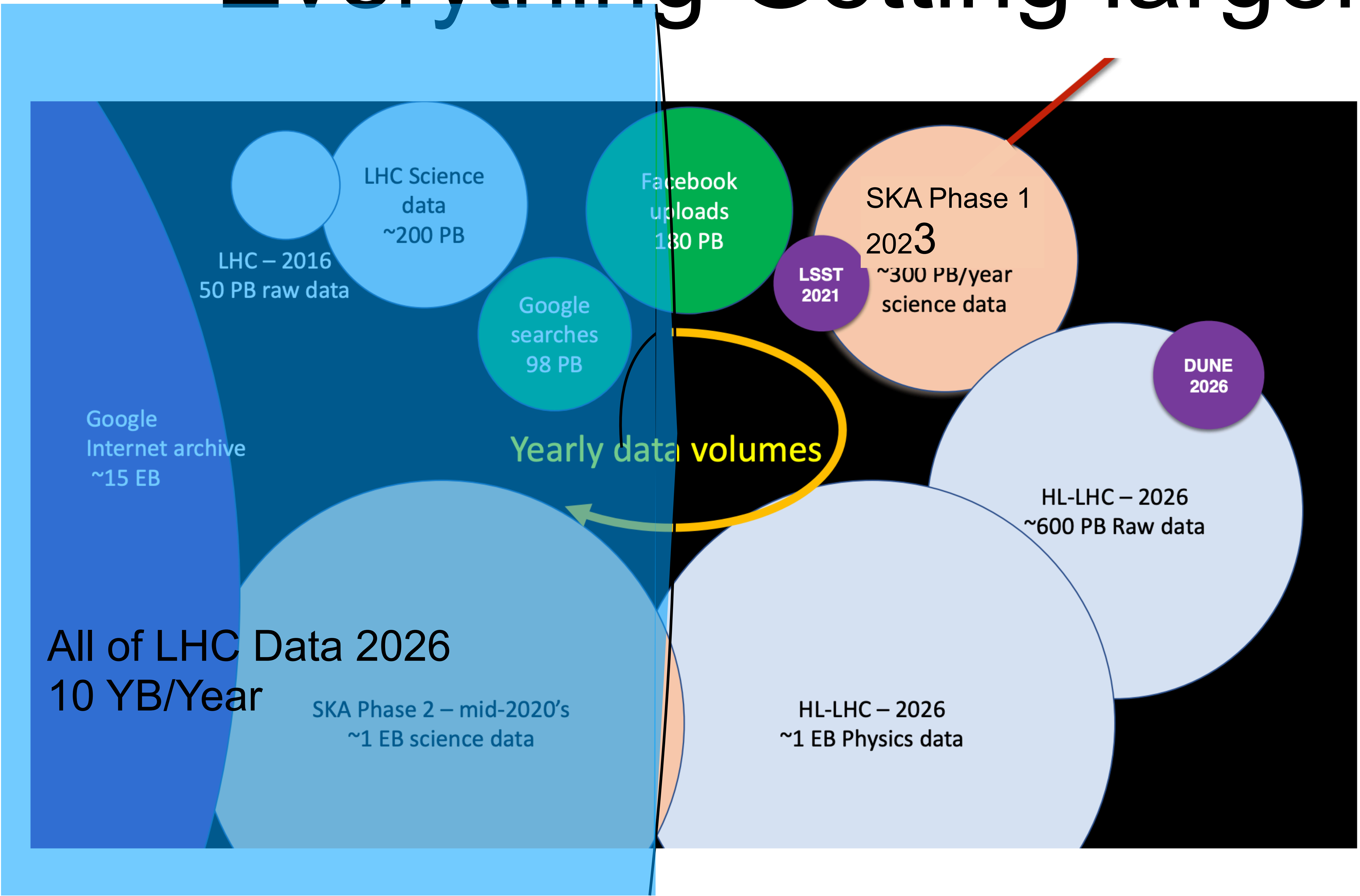

8PSK time plot
AM-DSB time plot
AM-SSB time plot
BPSK time plot
CPFSK time plot
GFSK time plot
PAM4 time plot
QAM16 time plot
QAM64 time plot
QPSK time plot
WBFM time plot

Many More

Everything Getting larger



Everything Getting larger



Conclusions

- Large scale campaign underway to adopt deep learning everywhere
- Scale of data processing in physics is getting larger
    - With large datasets come huge scientific potential
    - Processing of large data is a real challenge
- Have demonstrated ML+ Heterogeneous computing works
    - Parallelization of NNs and eff of FPGAs give large speedups
    - Exploring cloud and edge based service solutions

Conclusions

Getting closer to analyzing all of our data

In science has the potential to open new doors

Thanks!

INTERNET2®

NSF

DEPARTMENT OF ENERGY · UNITED STATES OF AMERICA

XILINX
ALL PROGRAMMABLE™

Microsoft

MIT
Quest for
Intelligence

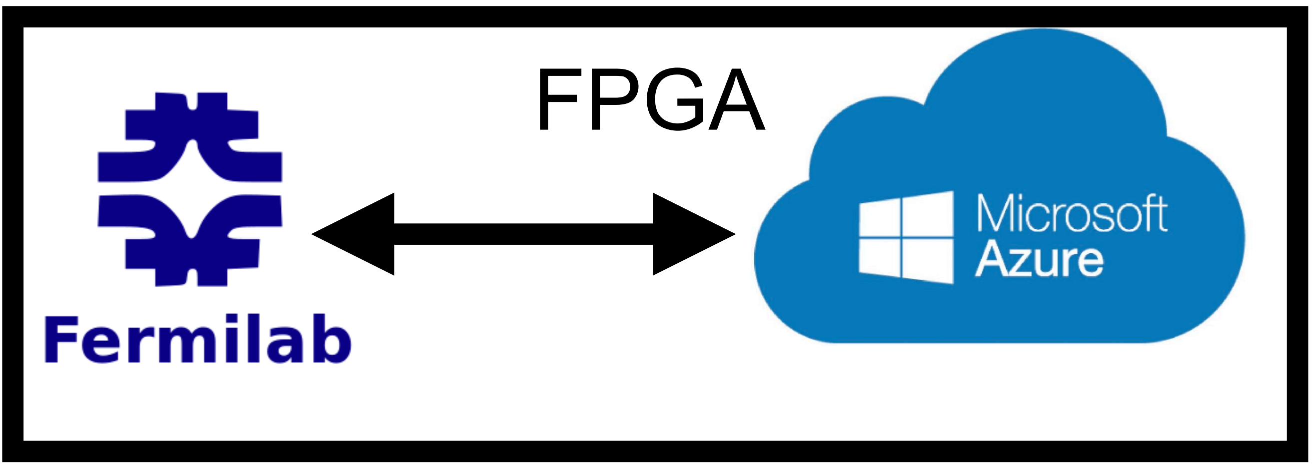
Benchmark #1

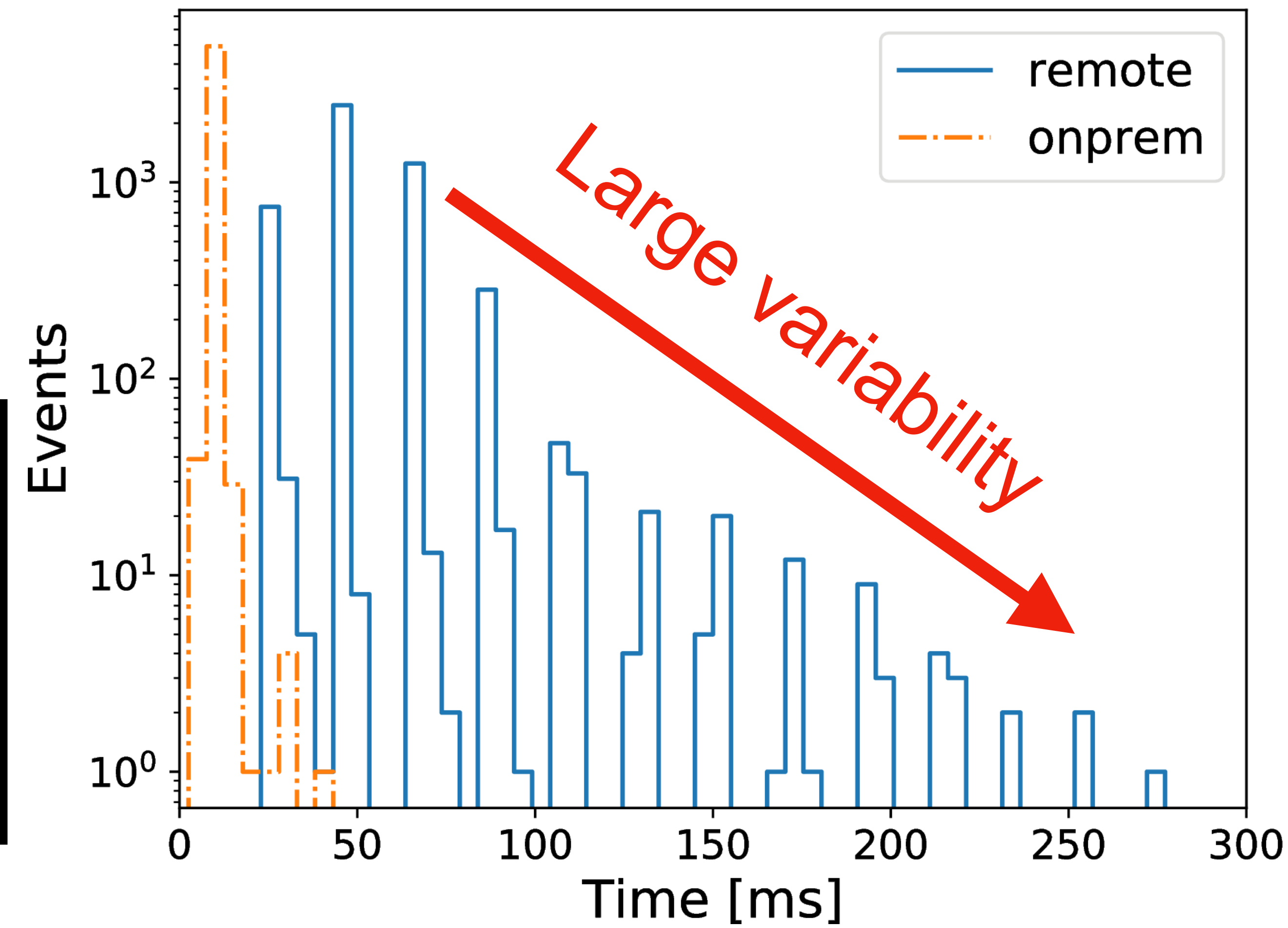
- Send our 16k inference from MIT to GPU at UCSD
  - Ping time is 75ms (speed of light google map distance is 32ms)
  - To UCSD and back takes ping time + 16ms
- Still working on test with FPGA (soon)

Algo	Per Event	+On-site aaS	+Cloud aaS	Ping
Old	50ms	N/A	N/A	N/A
NN CPU	15ms	N/A	N/A	N/A
NN GPU(1080 Ti)	3ms (prelim)	16ms	90ms	75ms
NN FPGA	2ms	TBD(<16ms)	TBD	TBD

Benchmark #2

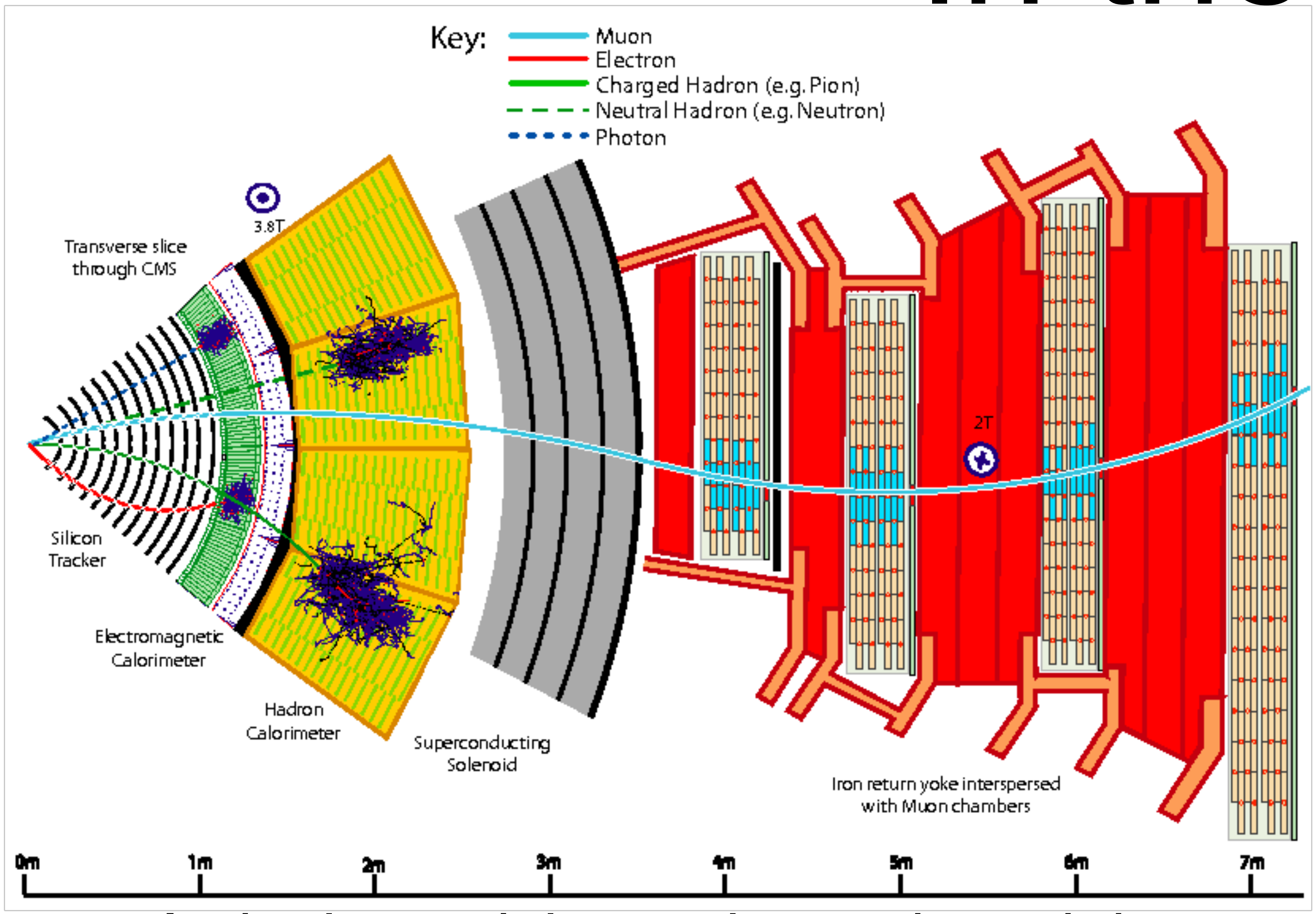
UCSD to MIT for GPU
FNAL to Azure for FPGA





Algo	Per Event	+On-site aaS	+Cloud aaS	Ping
CPU	1.75s	N/A	N/A	N/A
GPU Batch 1	7ms	23ms	97ms	75ms
GPU Batch 32	2ms	240ms	975ms	75ms
FPGA	1.7ms	15ms	60ms	25ms

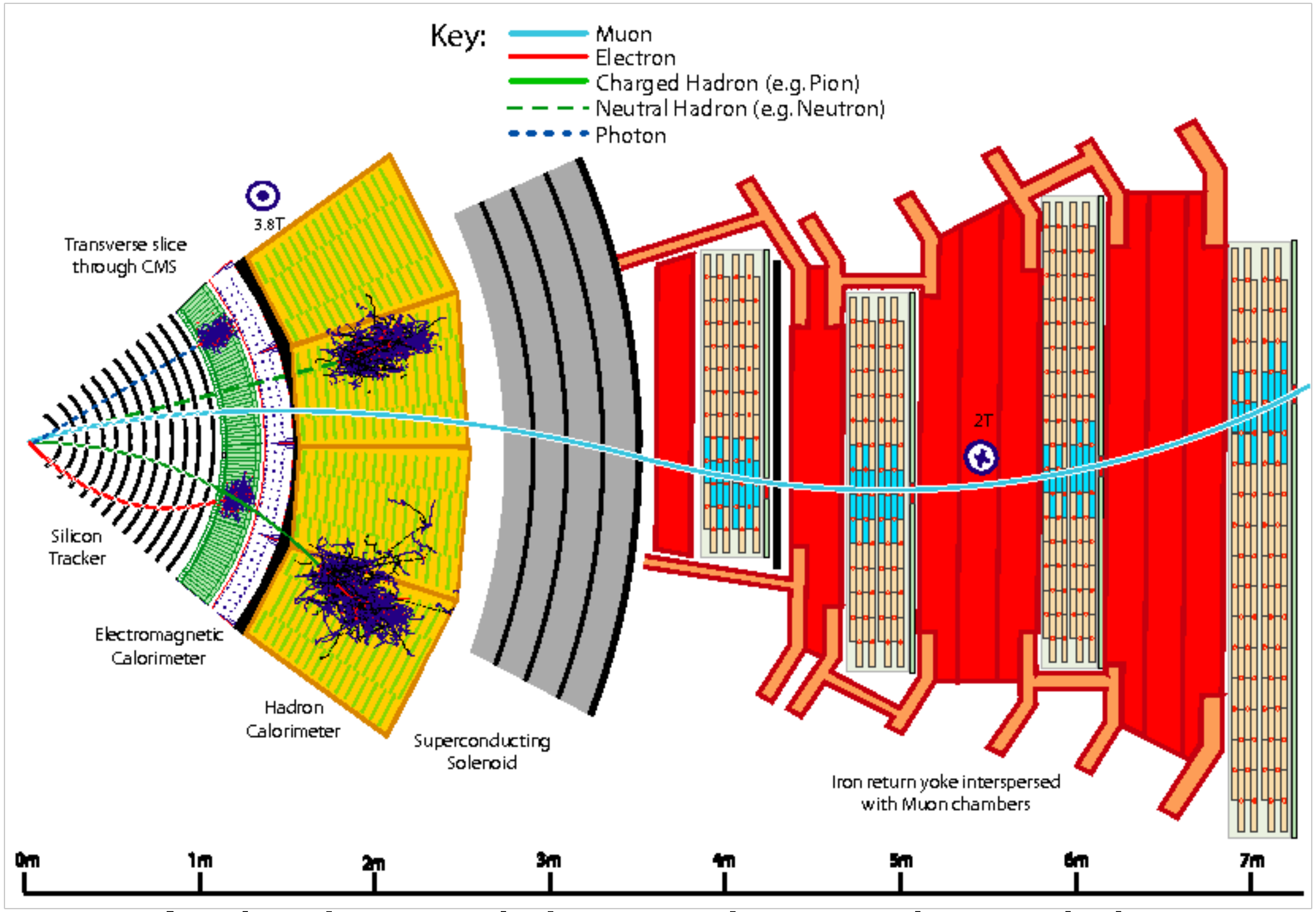
In the detector



All reconstruction is separated on an event by event level

- A single particle can leave deposit in many detectors
  - Each detector deposit a complex and different topology
  - Reconstruction of particles/detectors can be parallelized

Reconstruction of Objects



Batch 1 Per Event

All reconstruction is separated on an event by event level

- A single particle can leave deposit in many detectors
  - Each detector deposit a complex and different topology

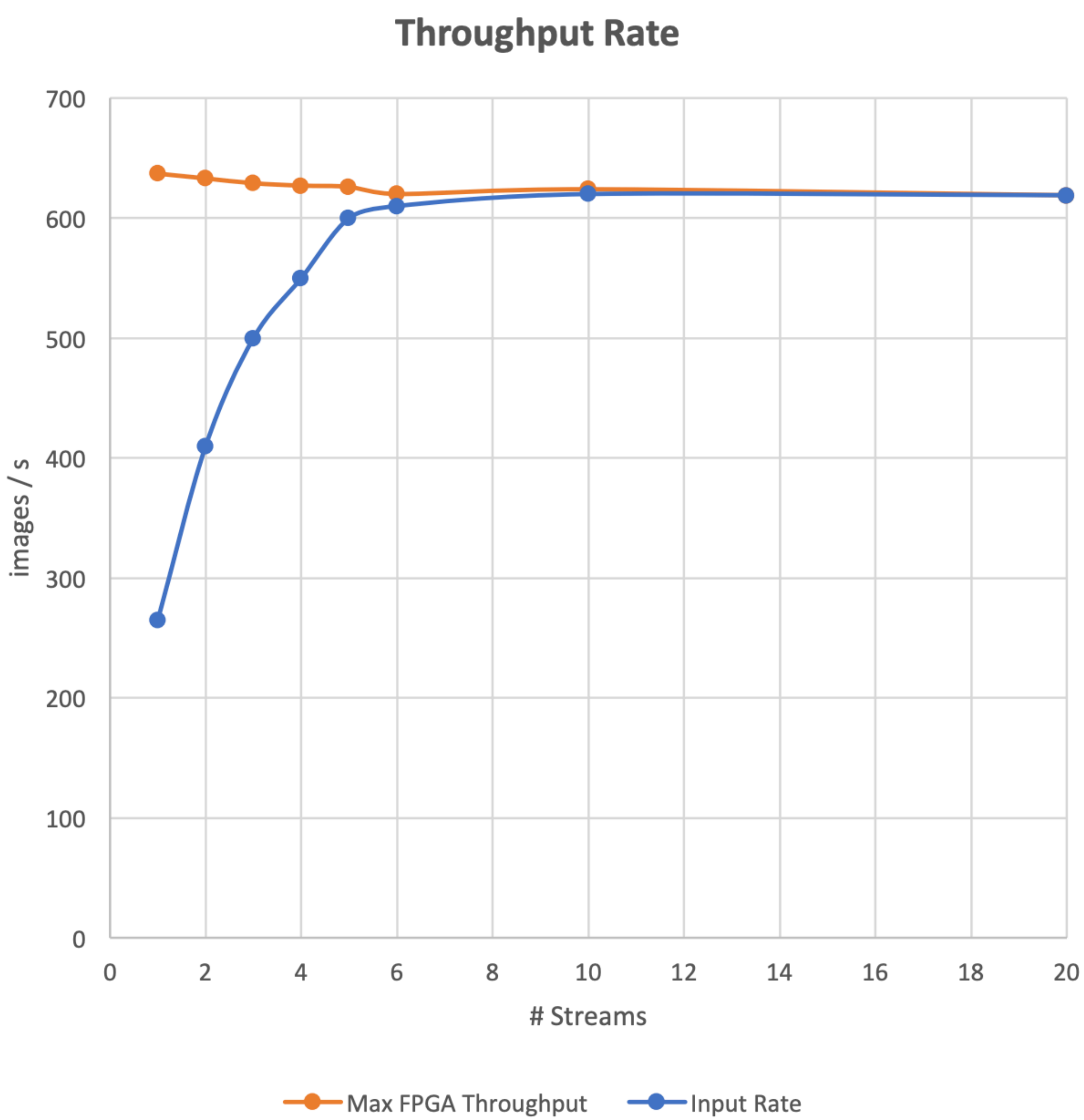
Batch N Per Particle

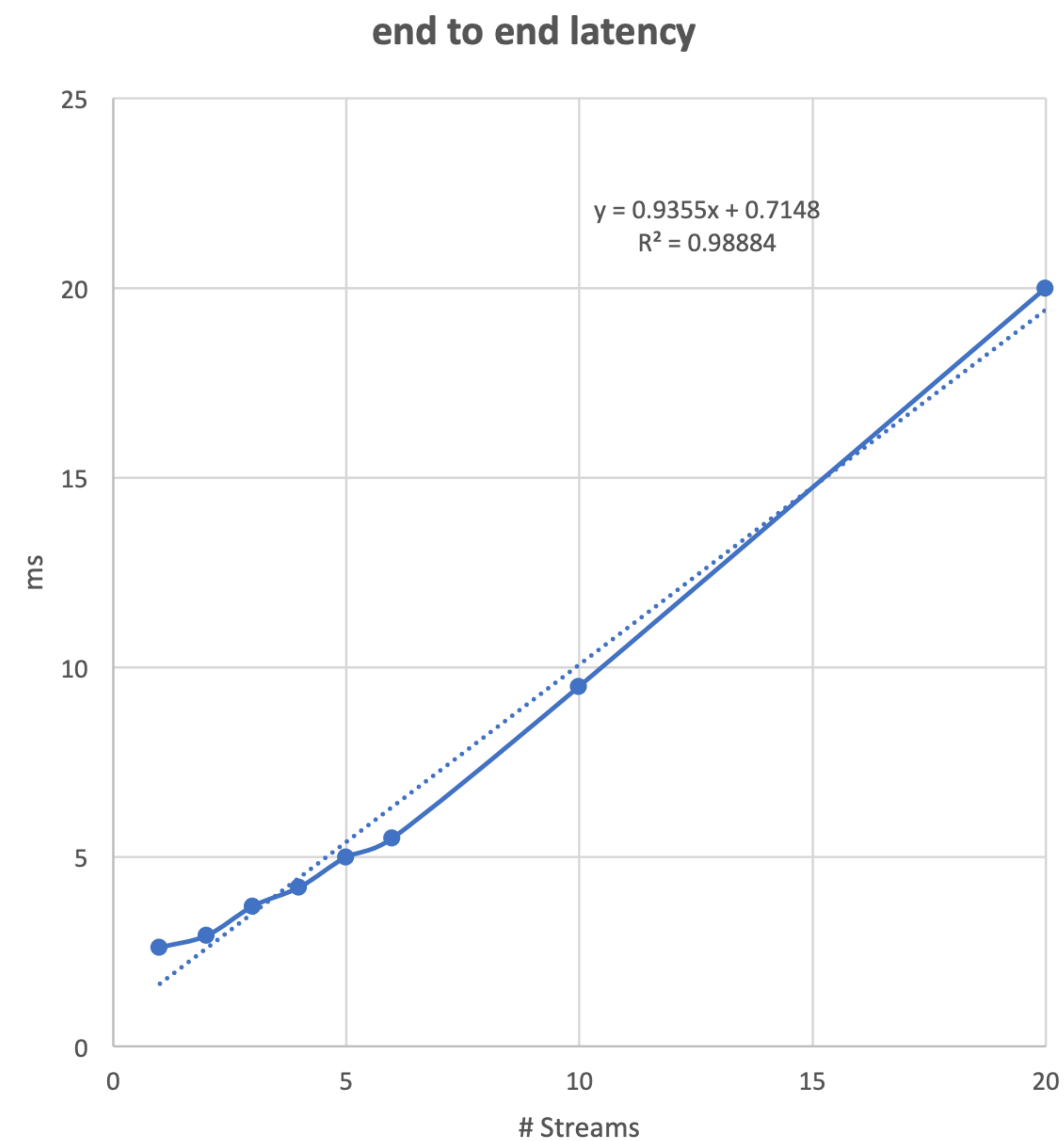
- Reconstruction of particles/detectors can be parallelized

Xilinx ML Suite

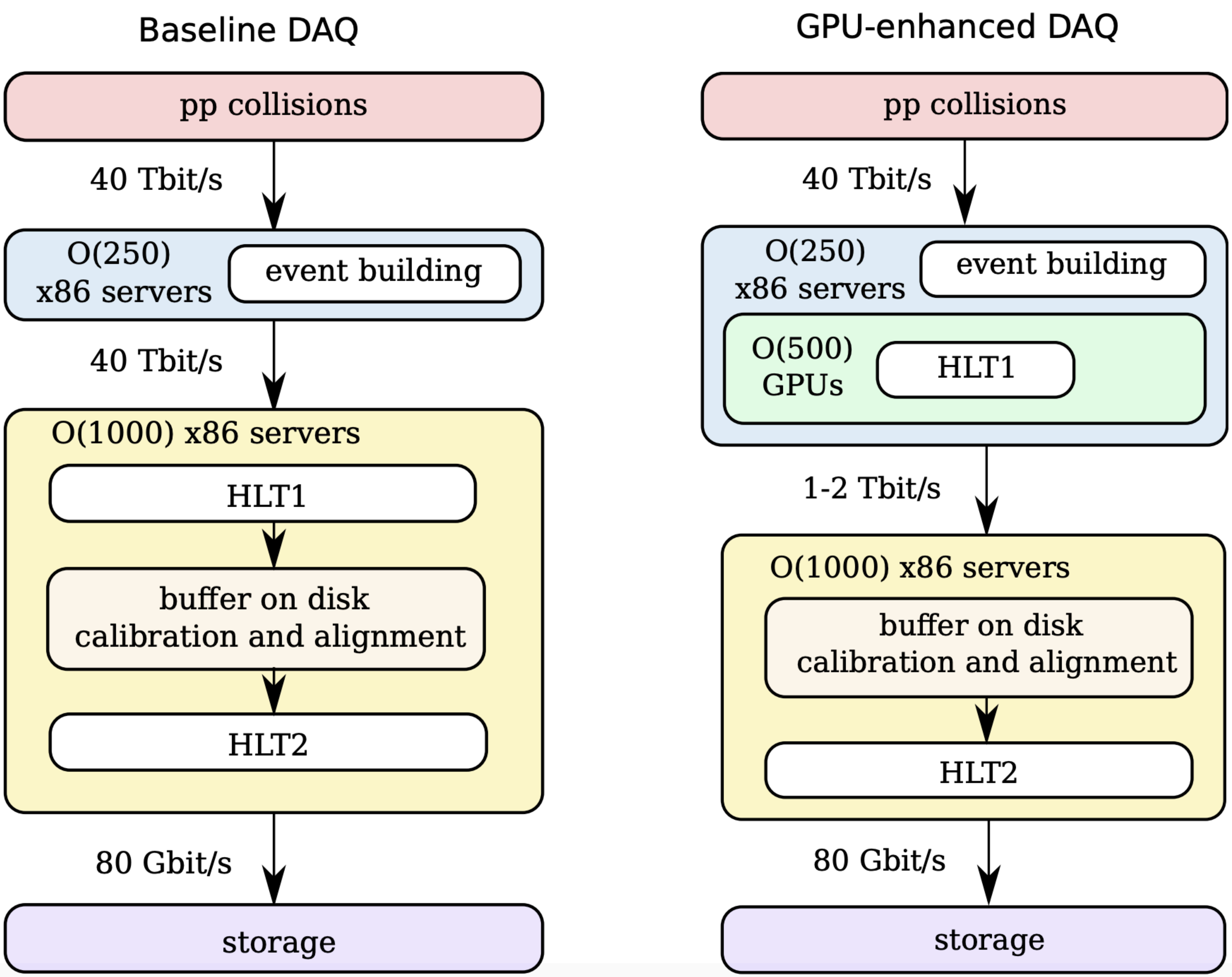
- Consider Googlenet example

```
requests avg latency: 16.855598 ms
time     avg latency: 2.07637 ms
```



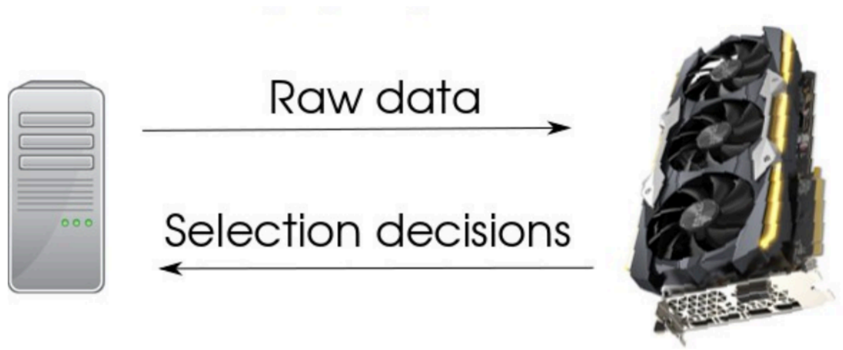


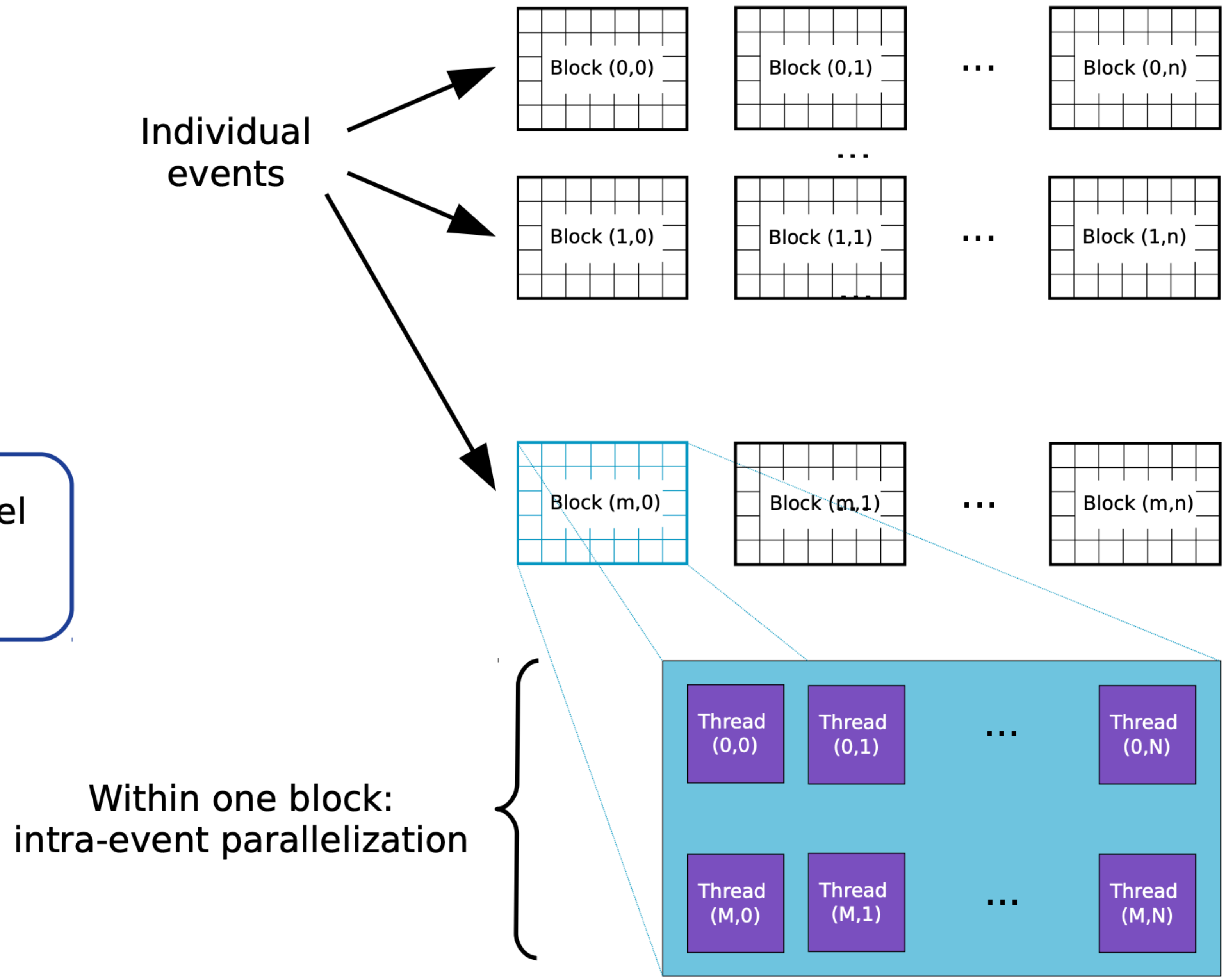
Alternative GPU Model



Full Reconstruction algorithm ported to GPU

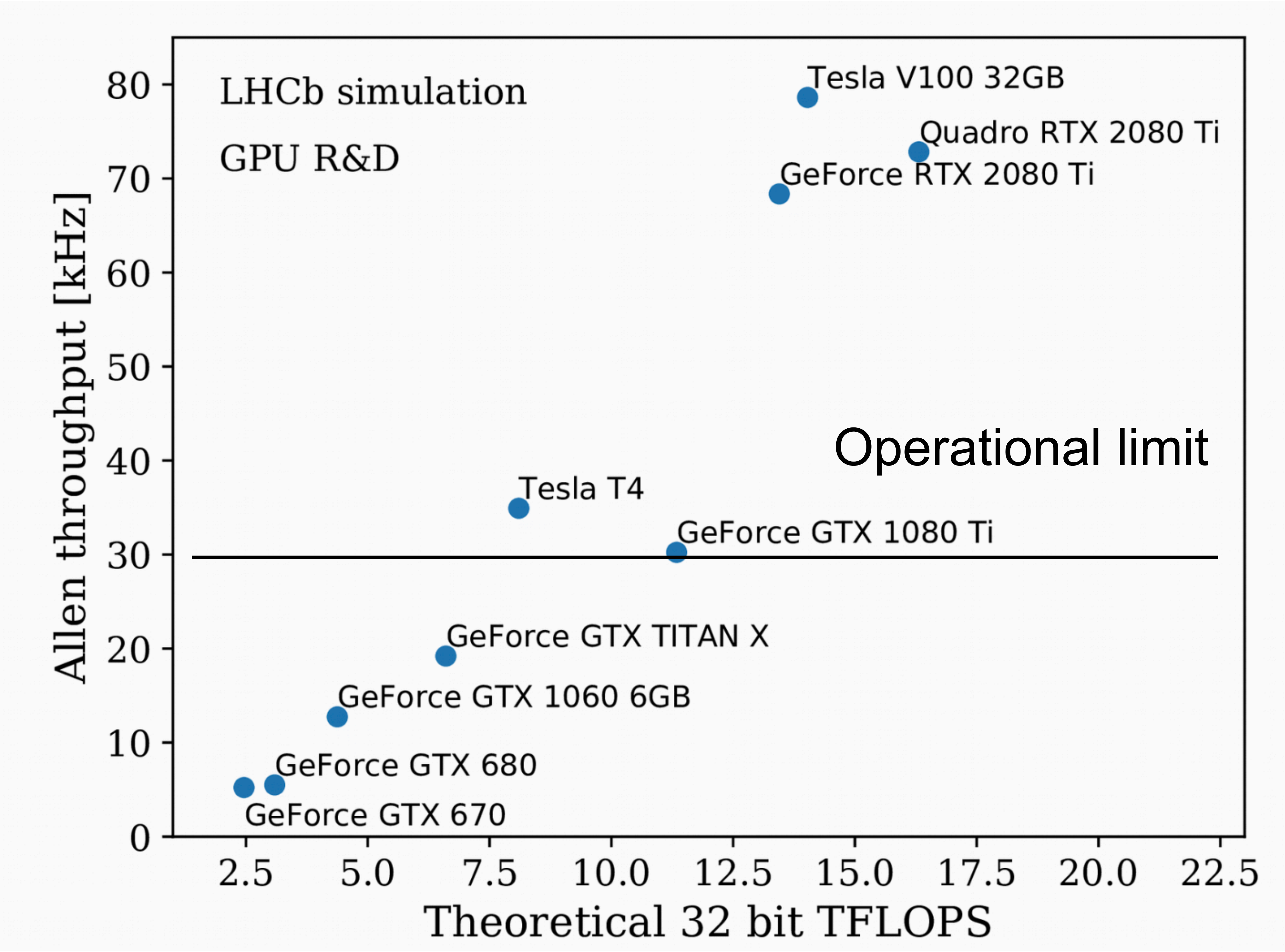
Alternative GPU Model





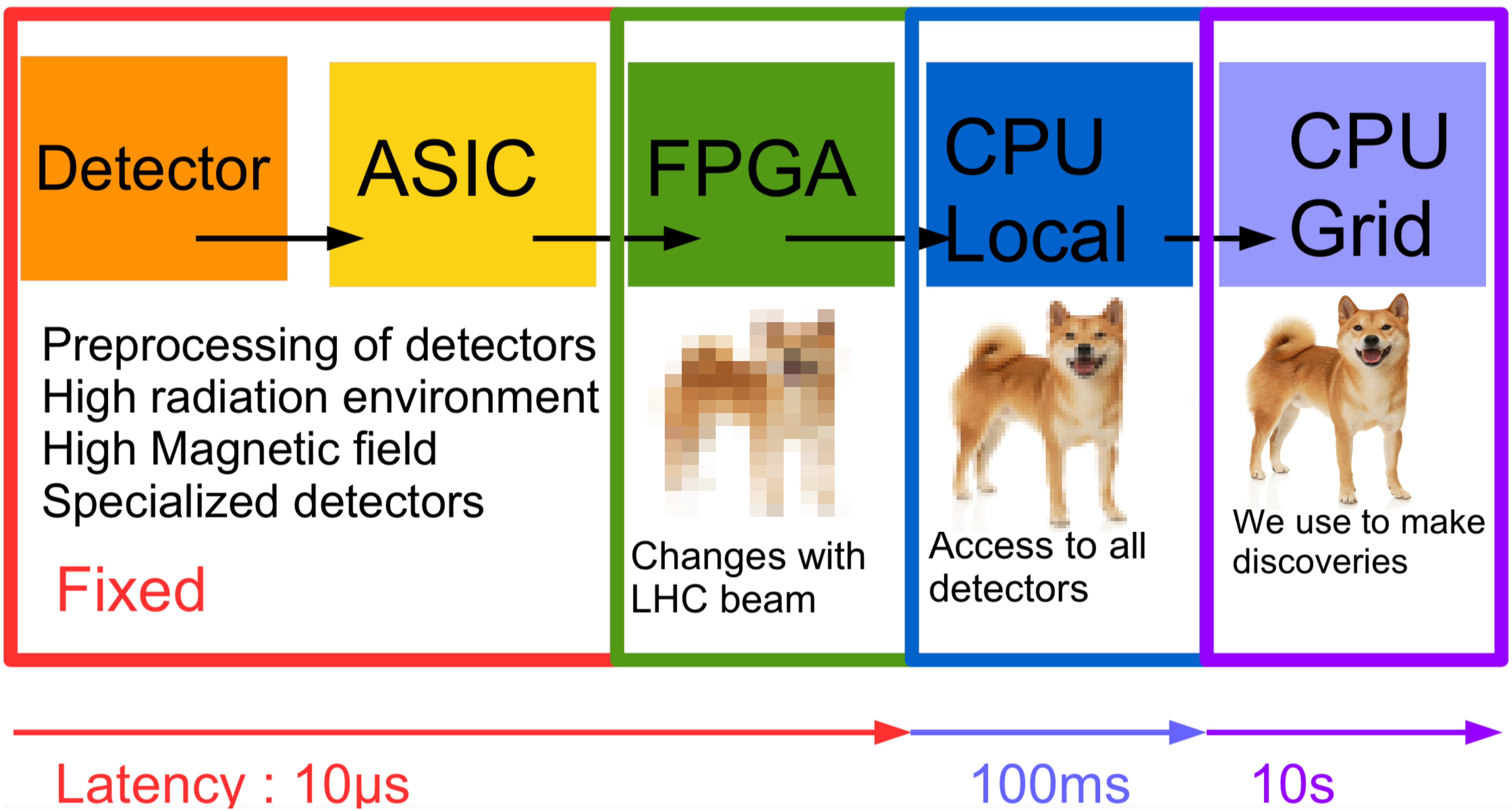
- Process thousands of events in parallel
- Single precision only

Alternative GPU Model

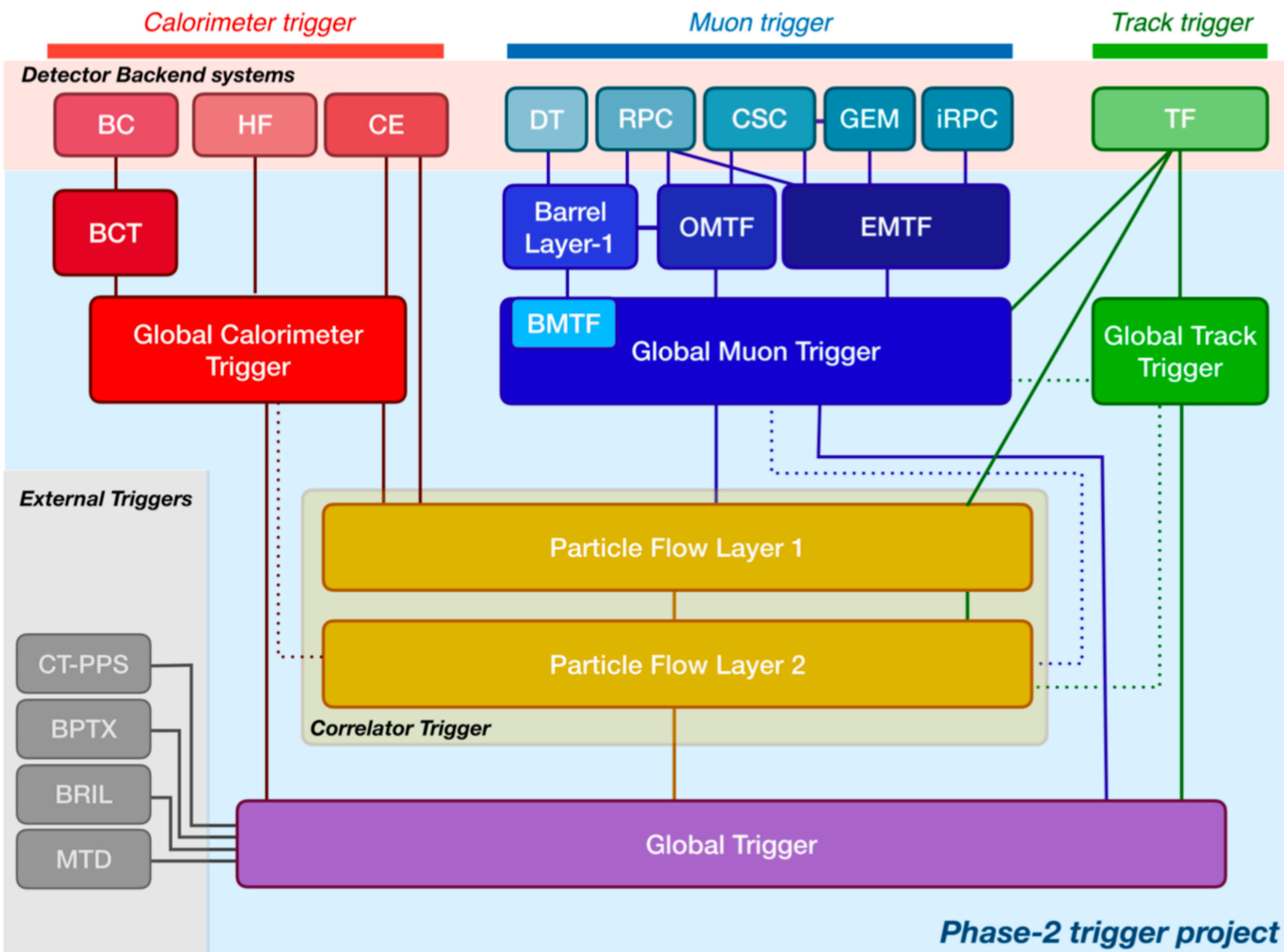


Another View of Same

Collision rate is 40 MHz
A new collision every 25ns



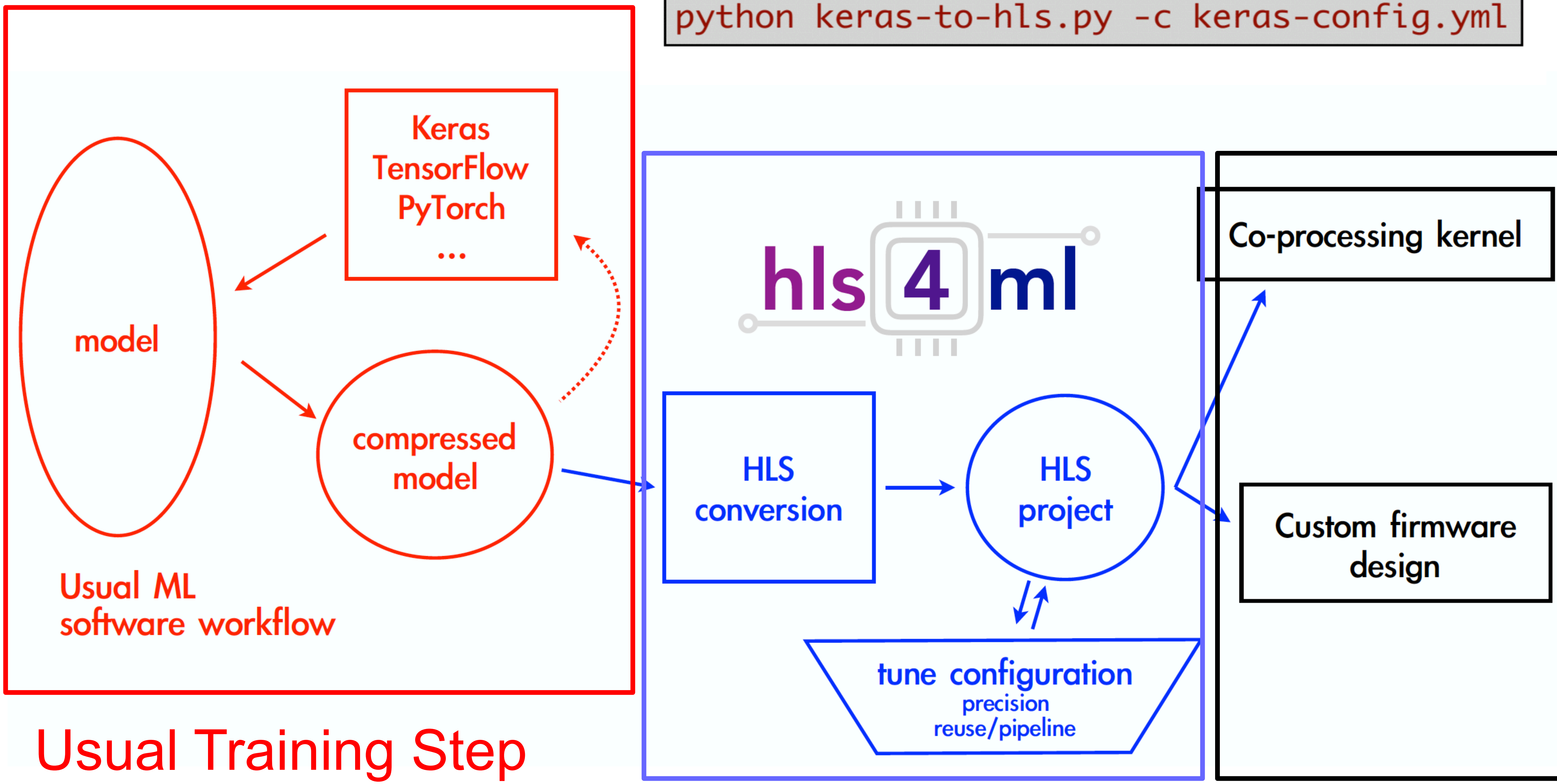
40 MHz (10μs) Systems



Each Block represents O(30) FPGAs w/50 Tb/s bandwidth 1μs latency

Summing Up the Data flow

```
python keras-to-hls.py -c keras-config.yml
```



Usual Training Step

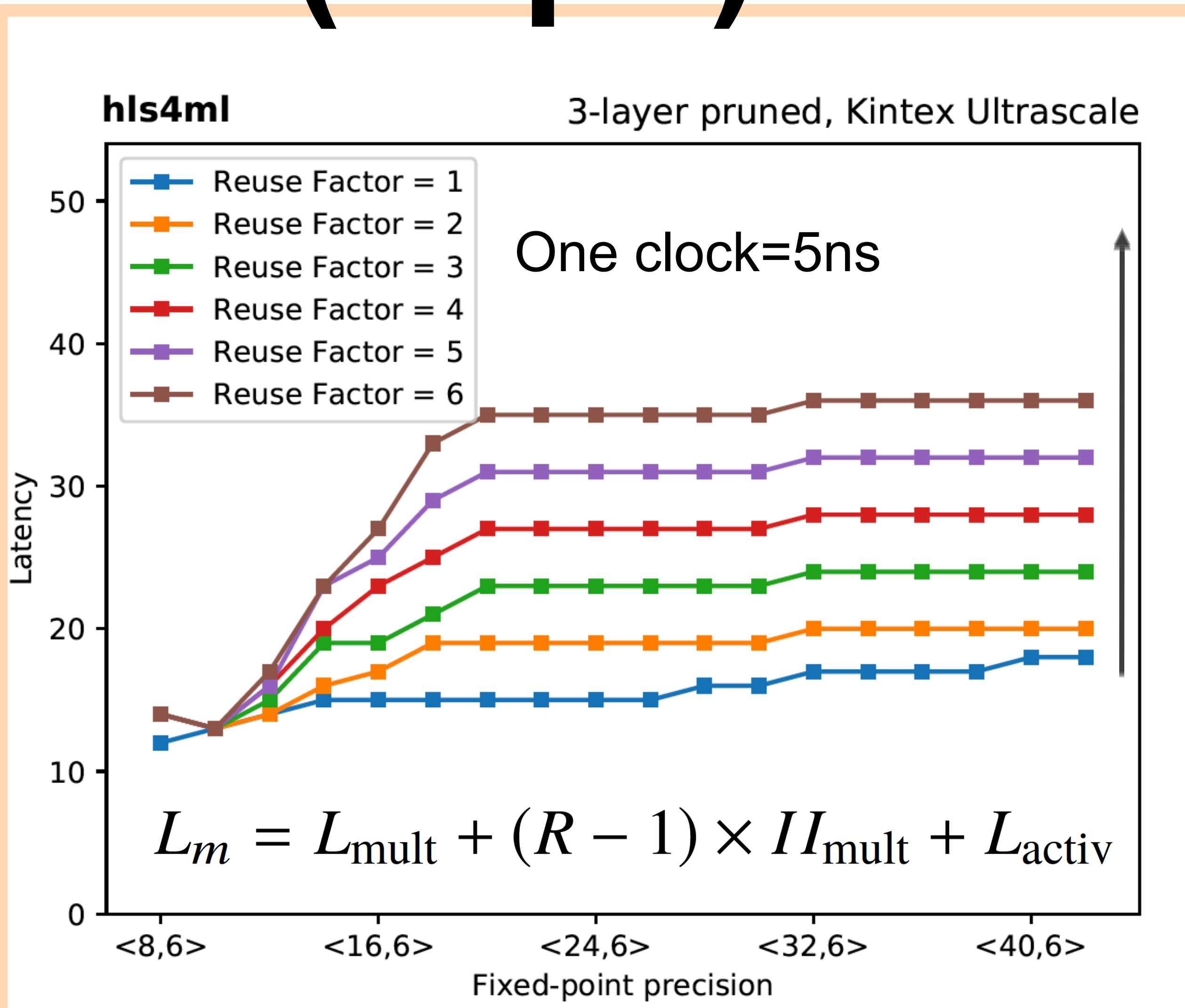
HLS tuning

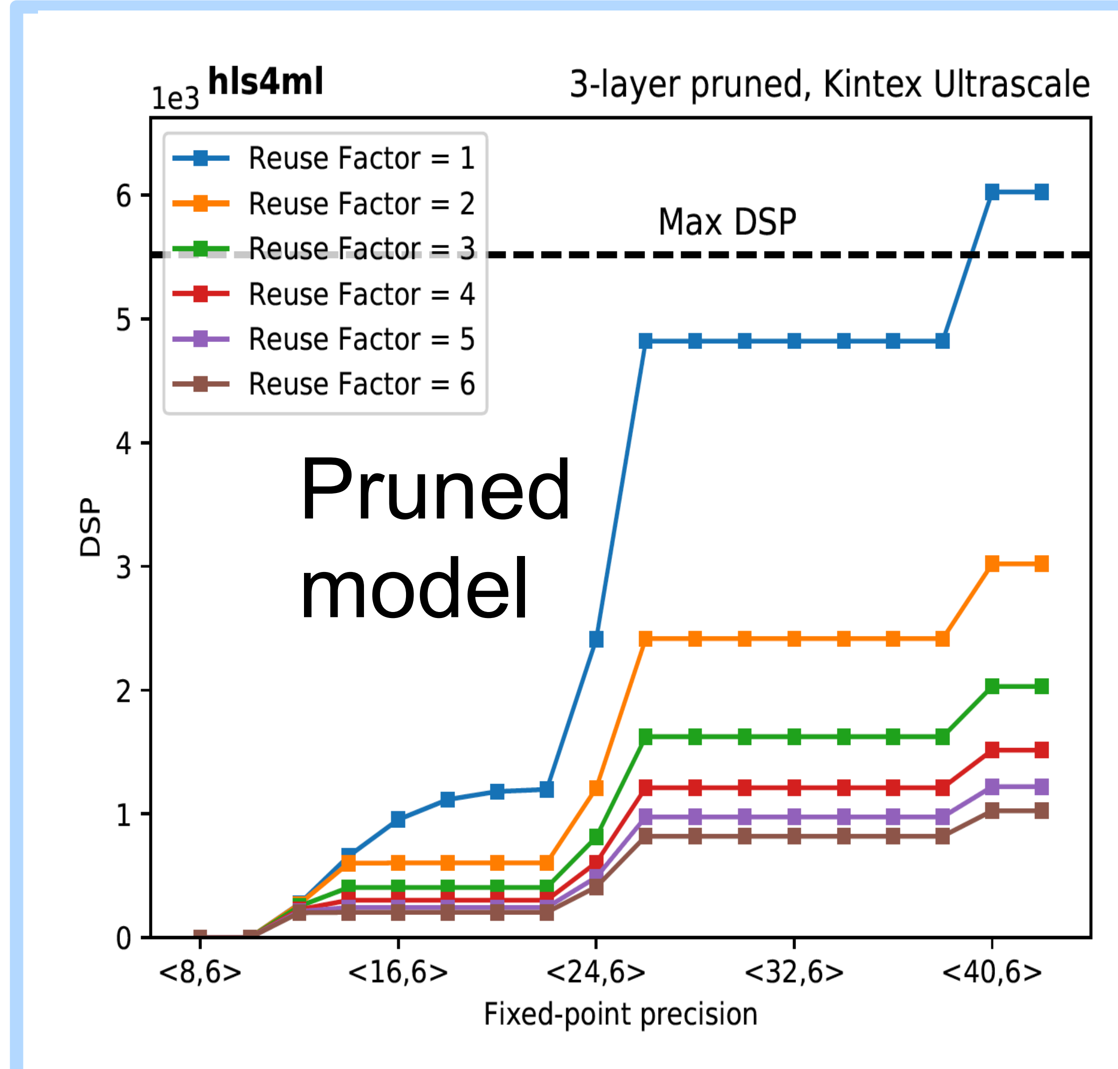
Final Product

Targeting Ultra low latency applications

40 MHz (10µs)

Example Performance





3-Layer NN 75ns latency
with an II of 1

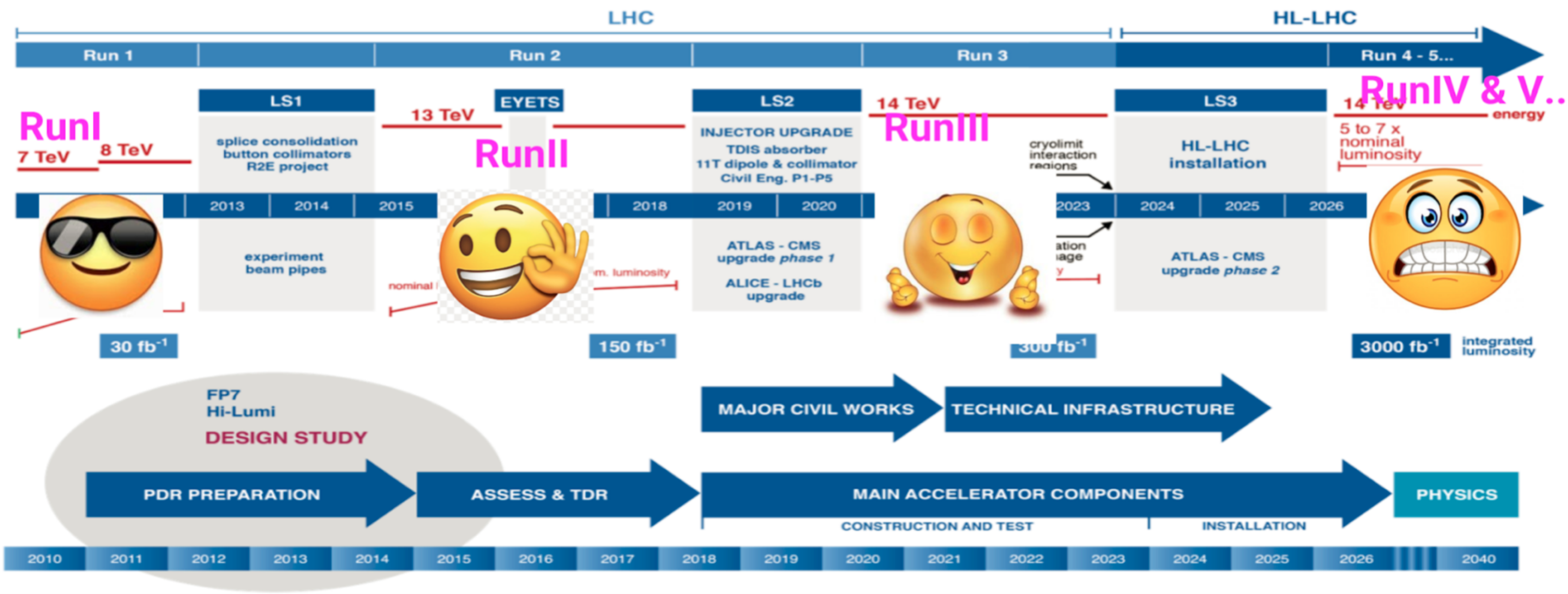
Latency (in clocks) gets worse
With reuse factor
Consistent with sharing resources

Tuneable reuse of DSPs
and BRAM to get latency
and II in ns timeslaes

What is a collision?

- LHC collides 60 protons at the same time
  - Eventually will become 200 protons at the same time
  - Collisions occur at 40 MHz
  - Expect roughly 1000(2000) particles per collision now(future)
    - Particles can leave deposits in many detectors
  - Aim to reconstruct aggregate properties of these collisions
- LHC Detector is roughly 100 Million channels
  - After zero suppression we have 8MB per collision

A More detailed View



MS Databox Edge

- **Data Box Edge:** A Microsoft *hardware-as-a-service solution* with an FPGA inside, installed at FNAL

```
iot_service = \
  IotWebservice.deploy_from_image(
    ws,
    iot_service_name,
    Image(ws, image_name),
    deploy_config,
    iothub_compute
    )
```

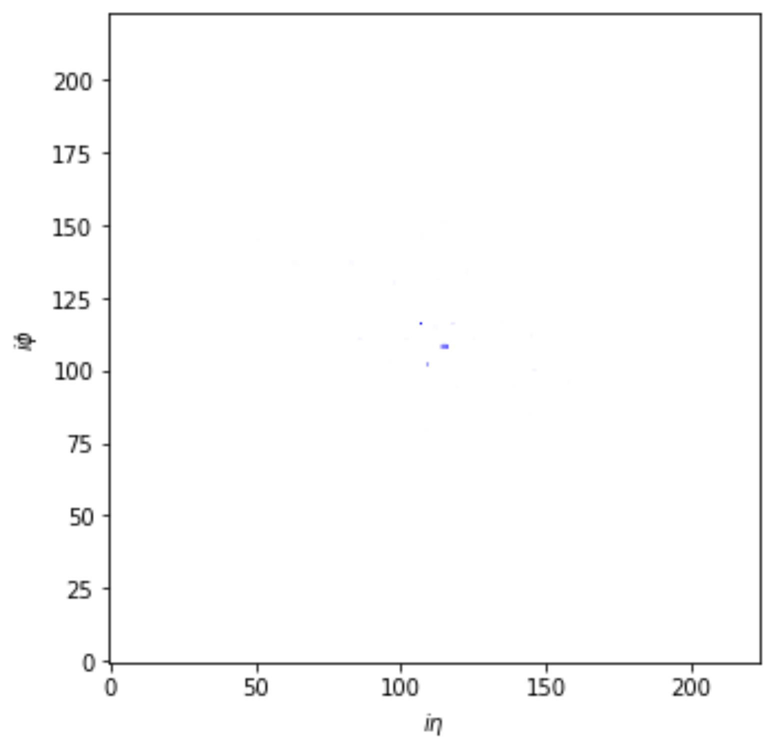
- **Deploy pre-trained NNs** using a CLI or a python SDK

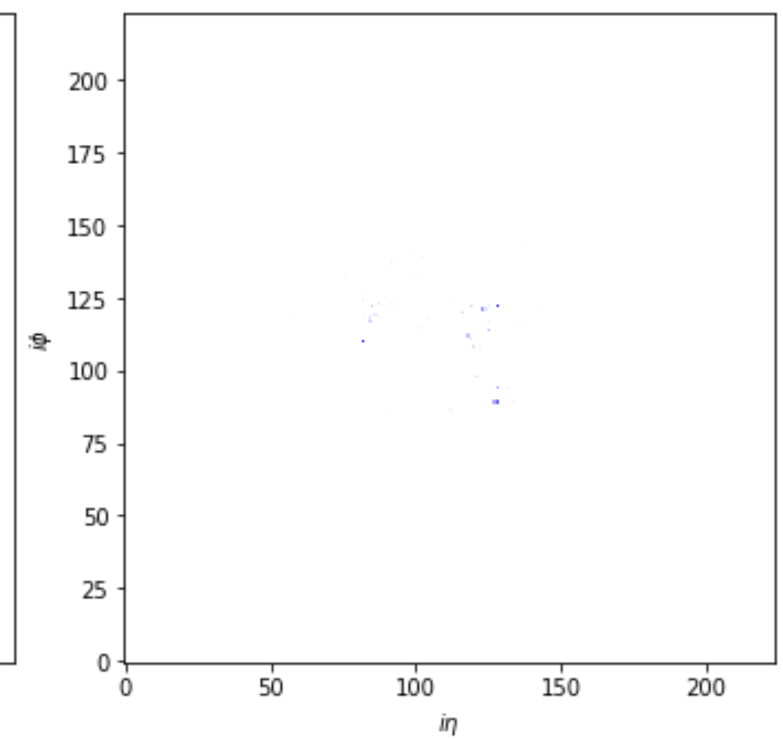
- **Inference from a client** by sending data over gRPC

```
client = PredictionClient(
    address = address.fnal.gov, port = 50051,
    use_ssl = False,
    service_name = module_name
    )
result = client.score_numpy_arrays(
    input_map = {'Placeholder:0' : np_array}
    )
```

Jet Tagger Example

- Distinguish between top quarks and QCD using **224x224** **single-color images**
    - Images: collected energy in the η/ϕ plane (detector coordinates)





Previous inference results

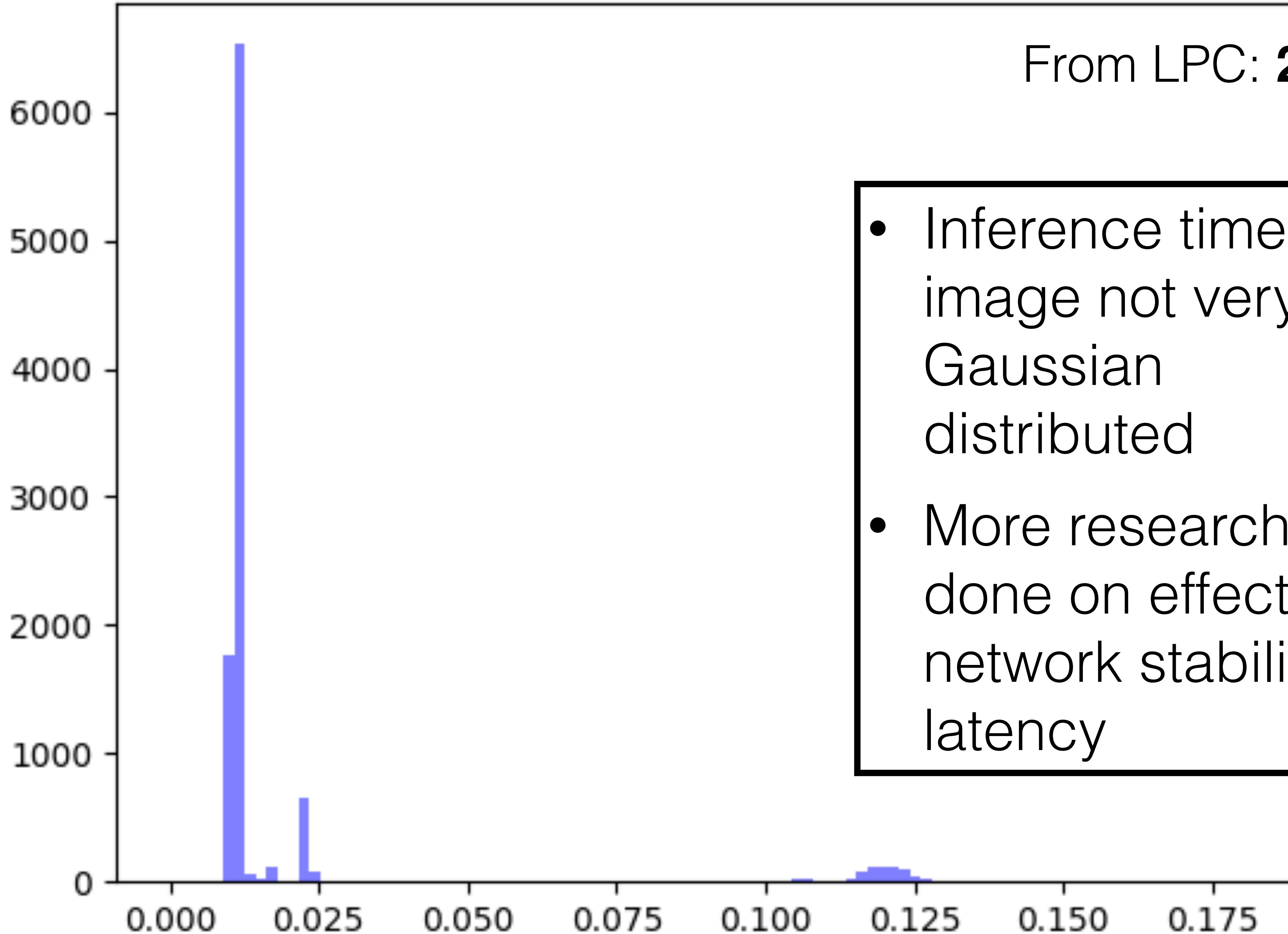
- On a single CPU: **~500 ms**
- On Azure Kubernetes Cloud Service: **~60-80 ms** (depending on distance)
- Deployed at Azure Data Center in Viriginia (2018): **~10 ms**

Using Data Box Edge

- Docker container directly on DBE: **14 ms** ±*25*
- From LPC: **20 ms** ±*30*
- From laptop at FNAL: **68 ms** ±*27*
- From LXPLUS @ CERN: **168 ms** ±*62*

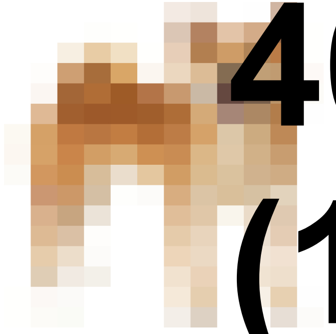
Timing

From LPC: **20 ms ±3**



time (s)

- Inference times per image not very Gaussian distributed
- More research to be done on effects of network stability/latency



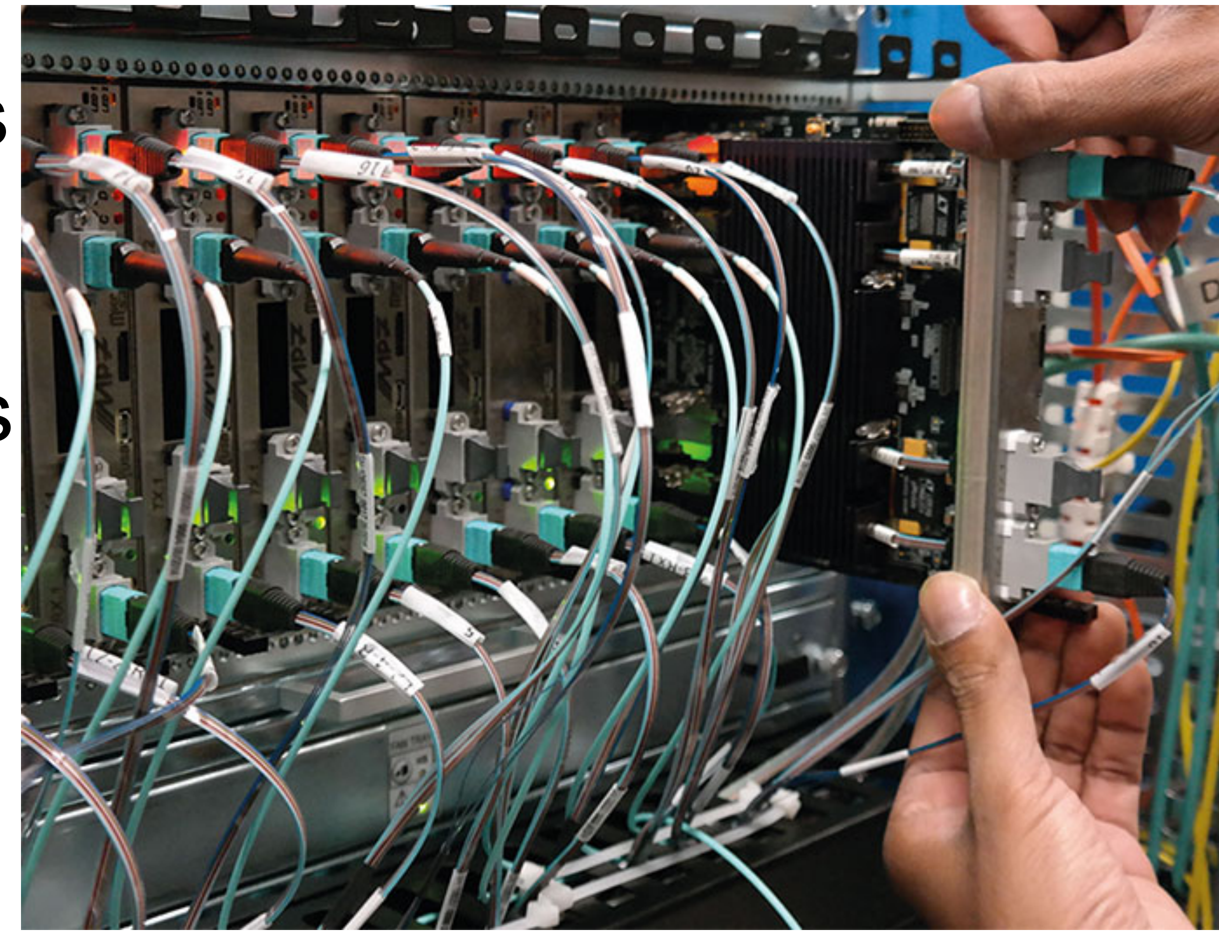
40 MHz (10μs)

L1 Trigger

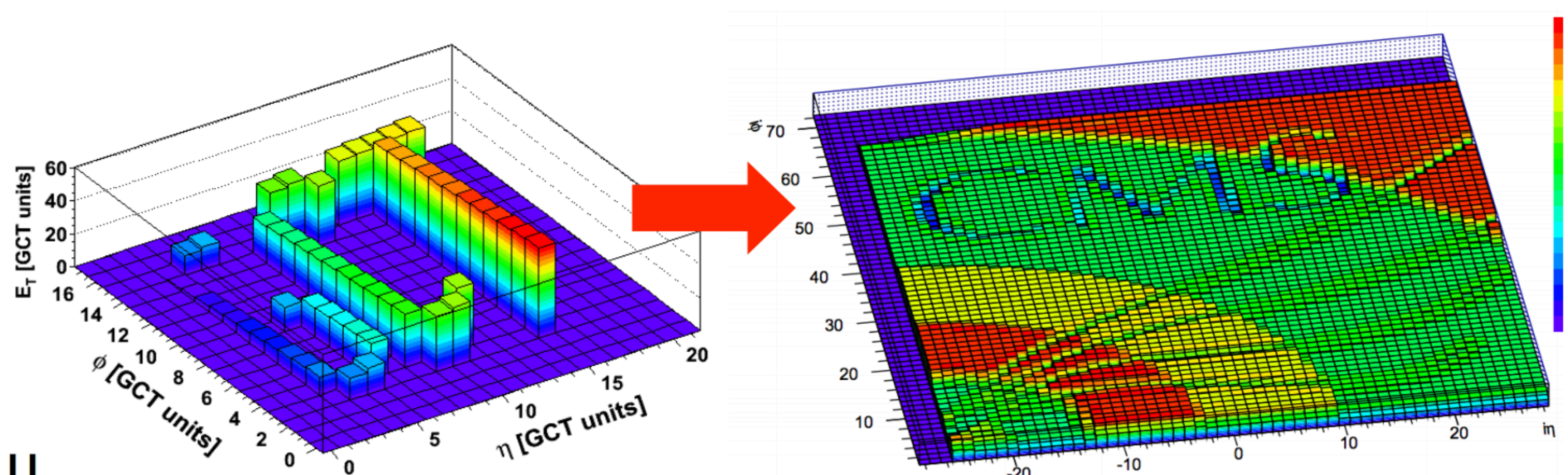
Have to take a new event every 25ns

Interconnected FPGAs
 direct optical links between the chips
48-112 Links per chip
Links run at 10-25 Gbps

Full system is O(1000) FPGAs



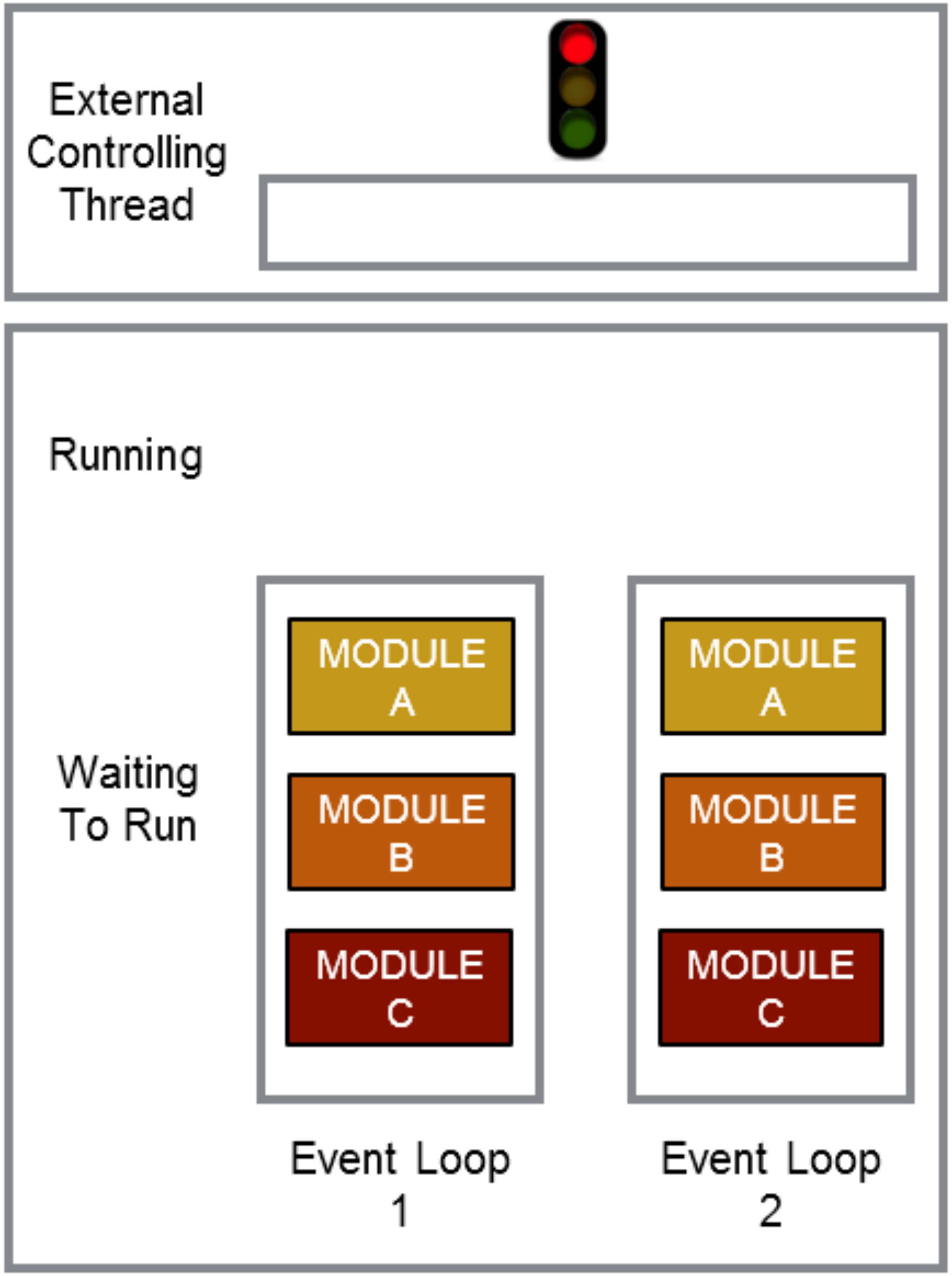
As FPGAs get larger so has the resolution of our detector



External Work in CMSSW (1)

Setup:

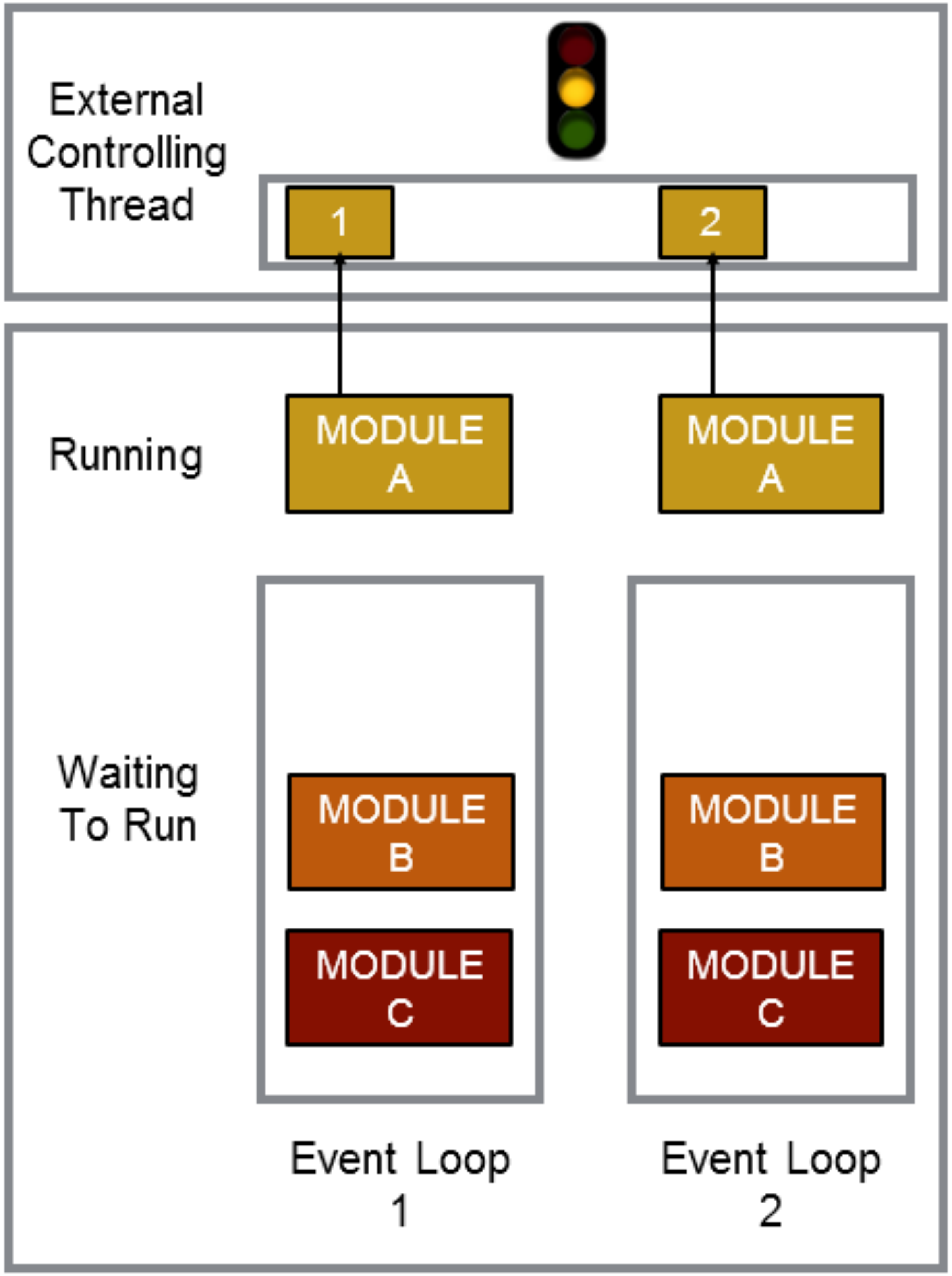
- TBB controls running modules
- Concurrent processing of multiple events
- Separate helper thread to control external
- Can wait until enough work is buffered before running external process



External Work in CMSSW (2)

Acquire:

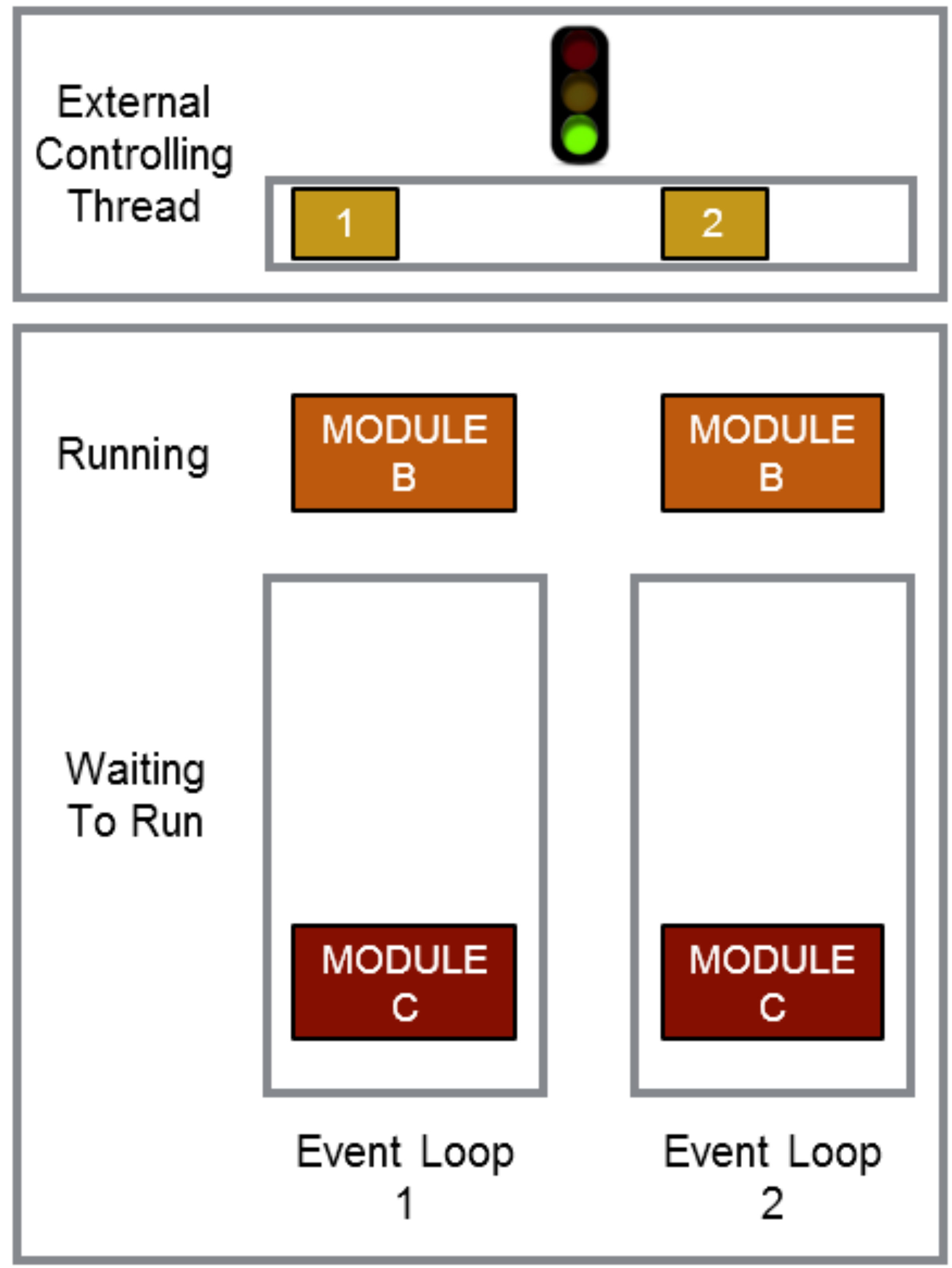
- Module *acquire*() method called
- Pulls data from event
- Copies data to buffer
- Buffer includes callback to start next phase of module running



External Work in CMSSW (3)

Work starts:

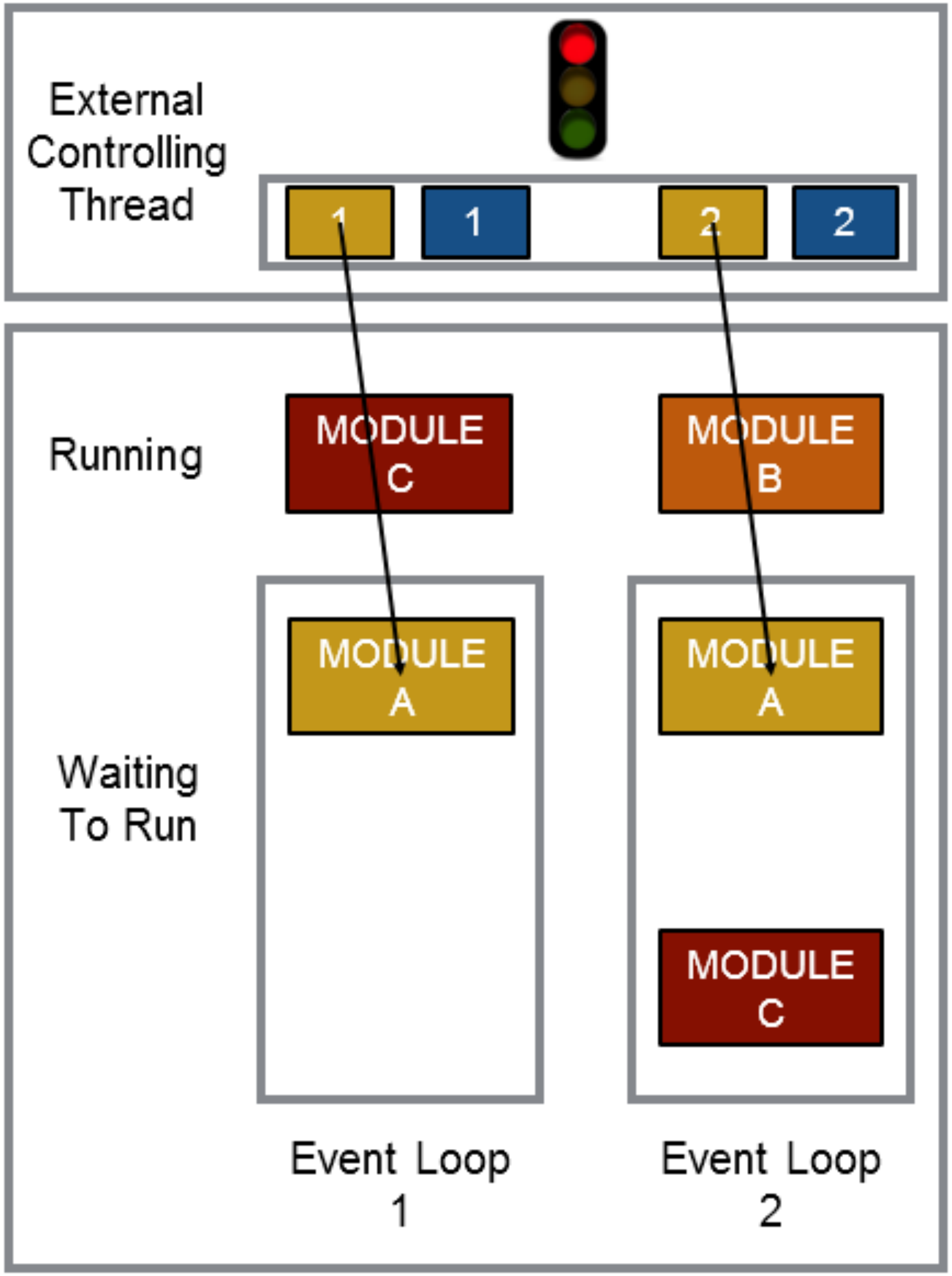
- External process runs
- Data pulled from buffer
- Next waiting modules can run (concurrently)



External Work in CMSSW (4)

Work finishes:

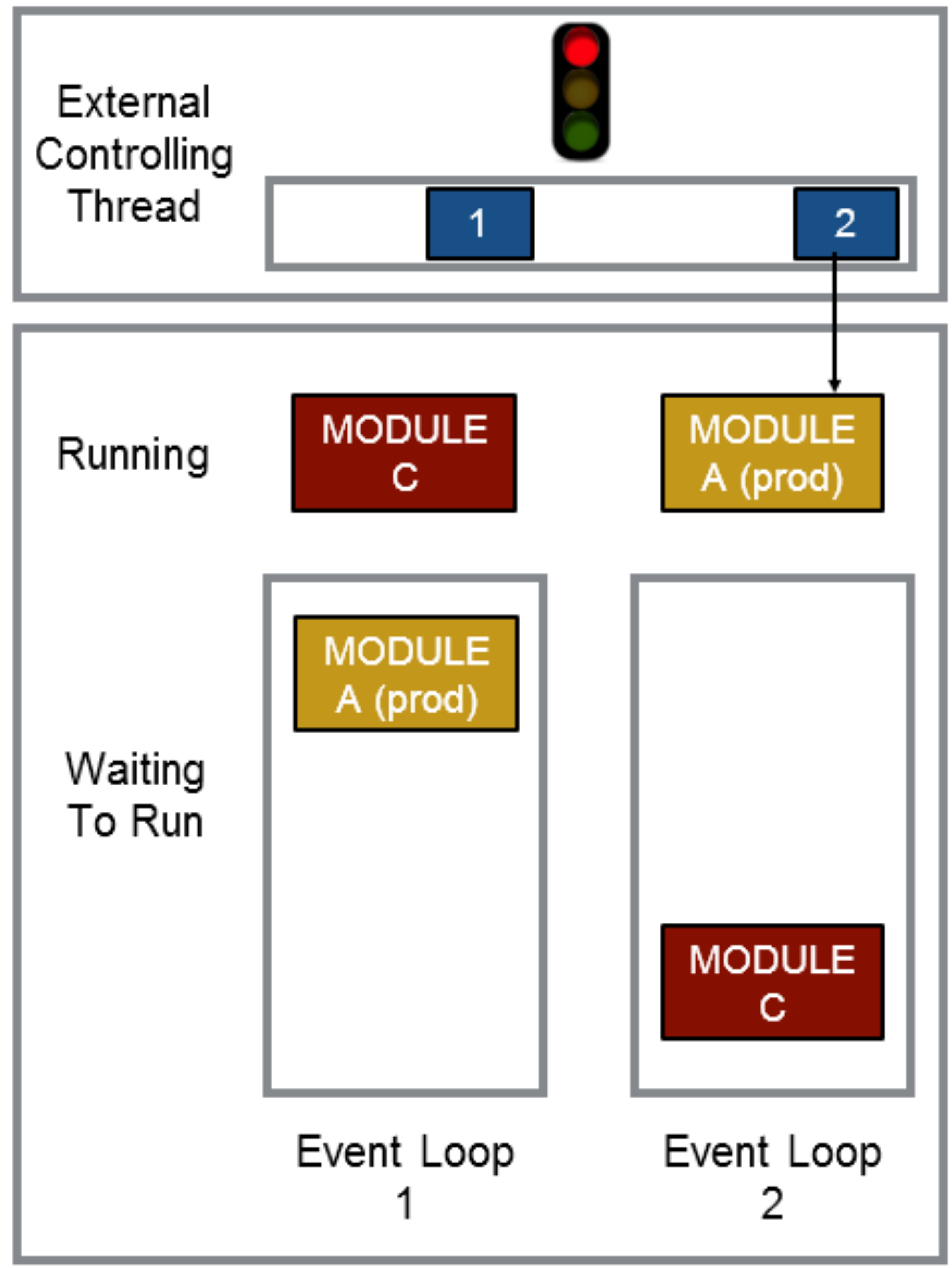
- Results copied to buffer
- Callback puts module back into queue



External Work in CMSSW (5)

Produce:

- Module *produce*() method is called
- Pulls results from buffer
- Data used to create objects to put into event



Sonic and Friends