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

Massachusetts Institute of Technology
Jeff Krupa
Dylan Rankin
Sang Eon Park

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

University of Toronto
Naif Tarafadar
Paul Chow

University of Washington
Scott Hauck
Shih-Chieh Hsu
Matthew Trahms
Dustin Werran

CERN
Vladimir Loncar
Jennifer Ngadiuba
Maurizio Pierini

UIC University of Illinois at Chicago
Zhenbin Wu

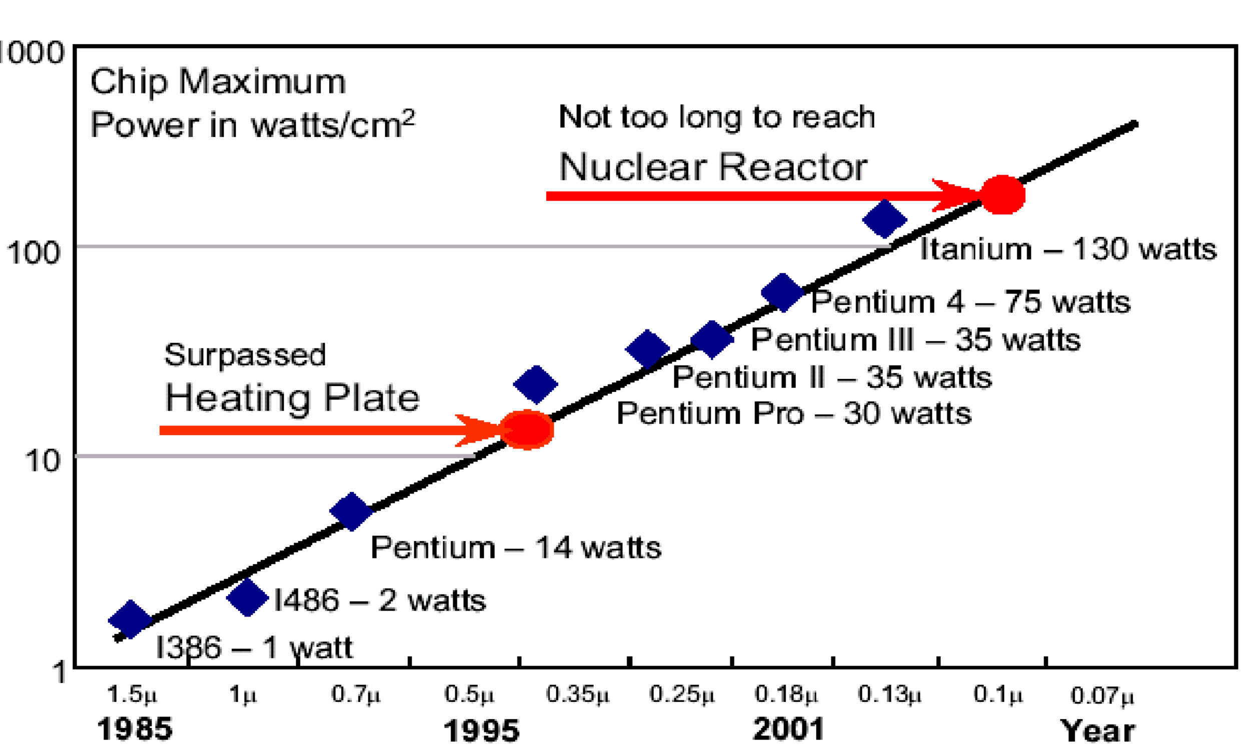
Columbia University in the City of New York
Giusppe Di Guglielmo

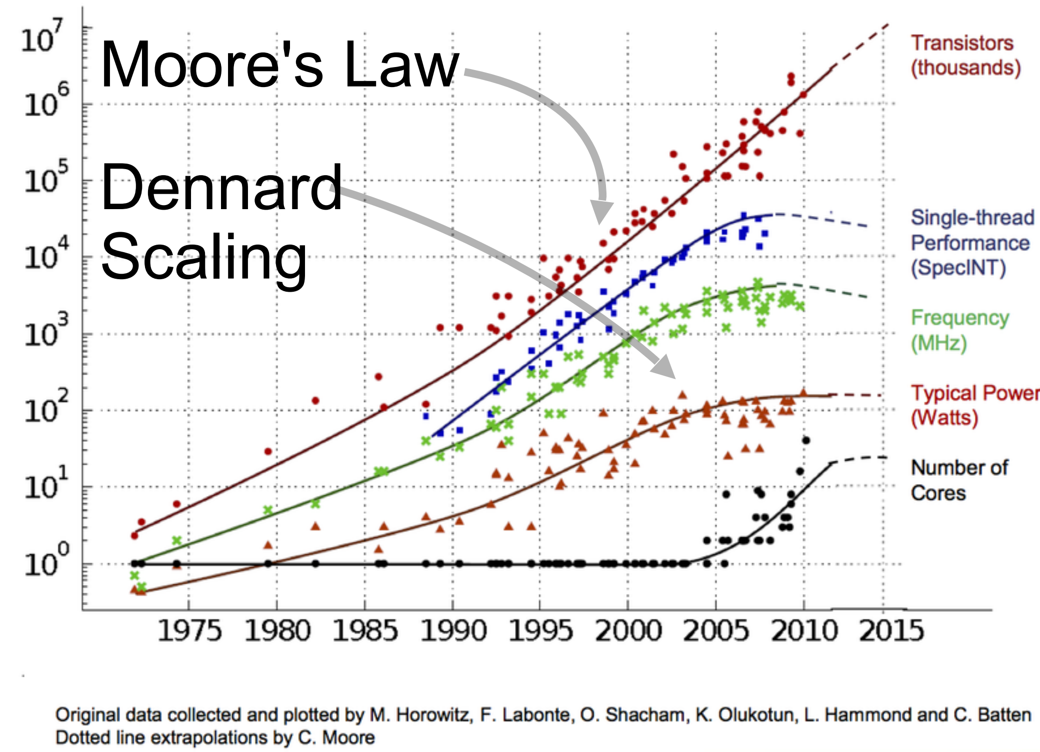
UCSD
Javier Duarte

Illinois
Mark Neubauer
Markus Crisziani

Beyond the Multicore Era

- Following the breakdown off Dennard scaling
  - Companies focused on multicore development
    - With many cores power limitations come up again



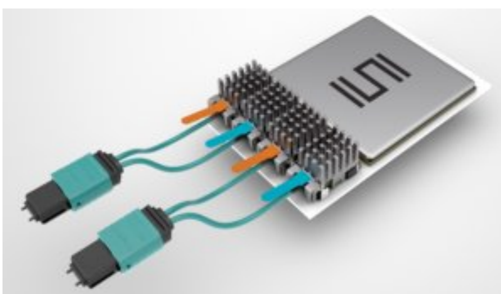
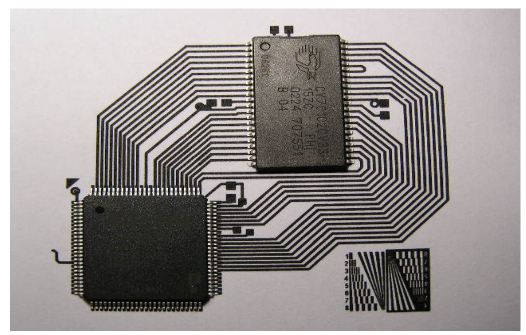
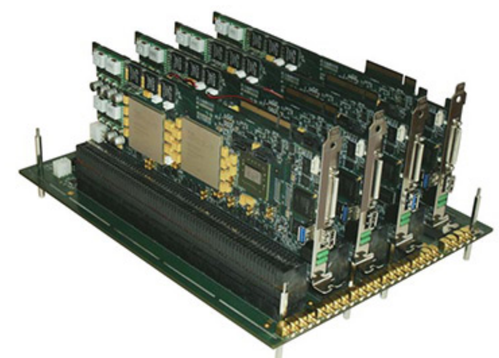
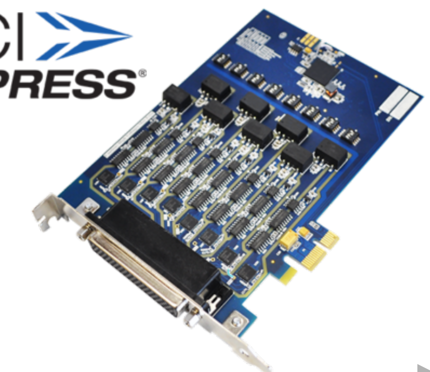


What becomes of the Post Multicore Era

Processors become specialized

https://www.cc.gatech.edu/~hadi/doc/paper/2011-isca-dark_silicon.pdf

Understanding Time Scales

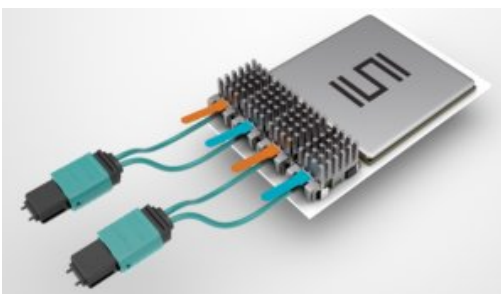
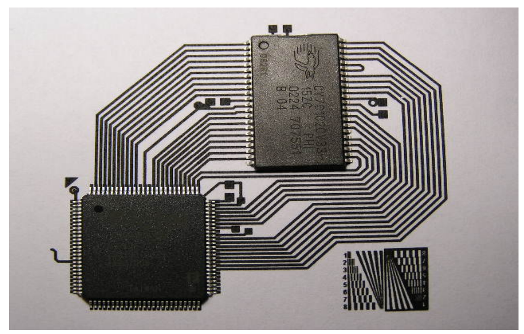
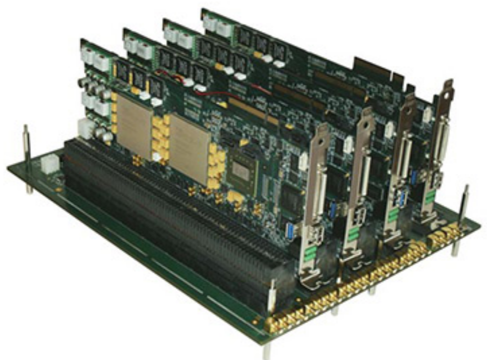
Optical to electrical to fpga pins	Chip(CPU) To FPGA On RAM	Board to Board on backplane	Stream through PCIexpress
			PCI EXPRESS 
<1µs	10-100µs	100µs-1ms	1ms-1s

Human interactions

LHC L1 Trigger

Understanding Time Scales

Optical to electrical to fpga pins	Chip(CPU) To FPGA On RAM	Board to Board on backplane	Stream through PCIexpress

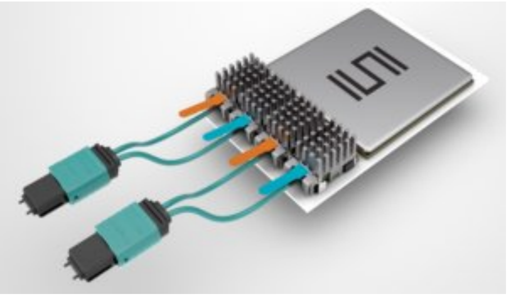
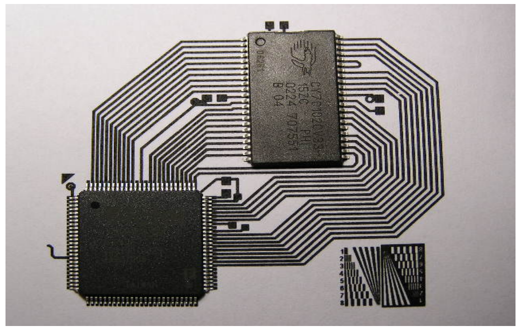
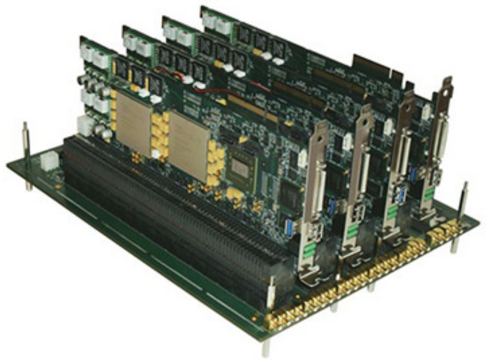

<1μs	10-100μs	100μs-1ms	1ms-1s

Human interactions

LHC L1 Trigger

LHC Physicists   Other Physicists   Industry

Understanding Time Scales

Optical to electrical to fpga pins	Chip(CPU) To FPGA On RAM	Board to Board on backplane	Stream through PCIexpress
			PCI EXPRESS 
<1μs	10-100μs	100μs-1ms	1ms-1s

All FPGA systems

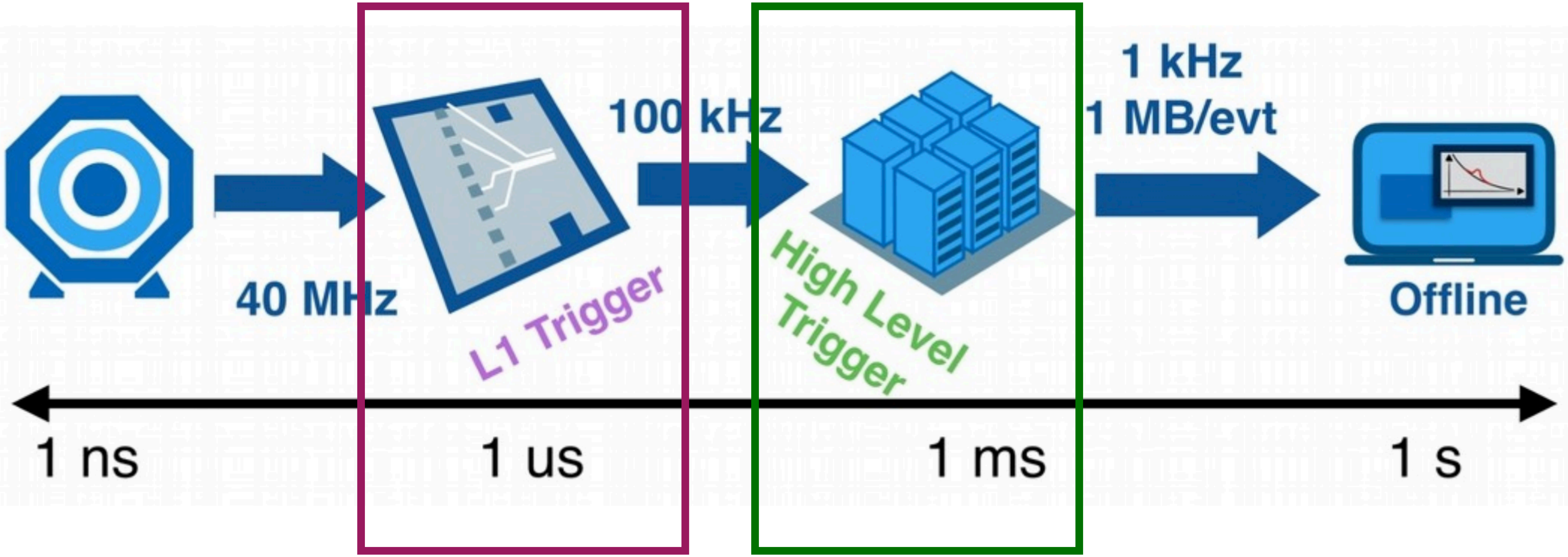
Hybrid GPU/CPU+FPGA systems

LHC Physicists

Other Physicists

Industry

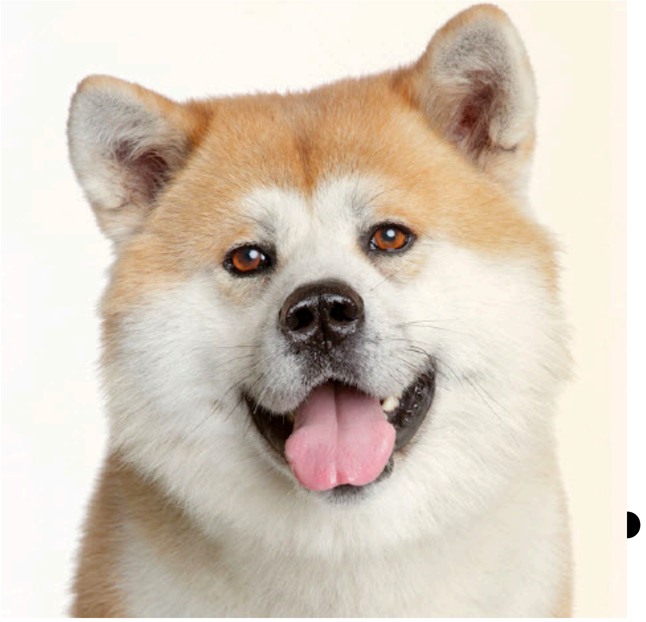
How do we process data?



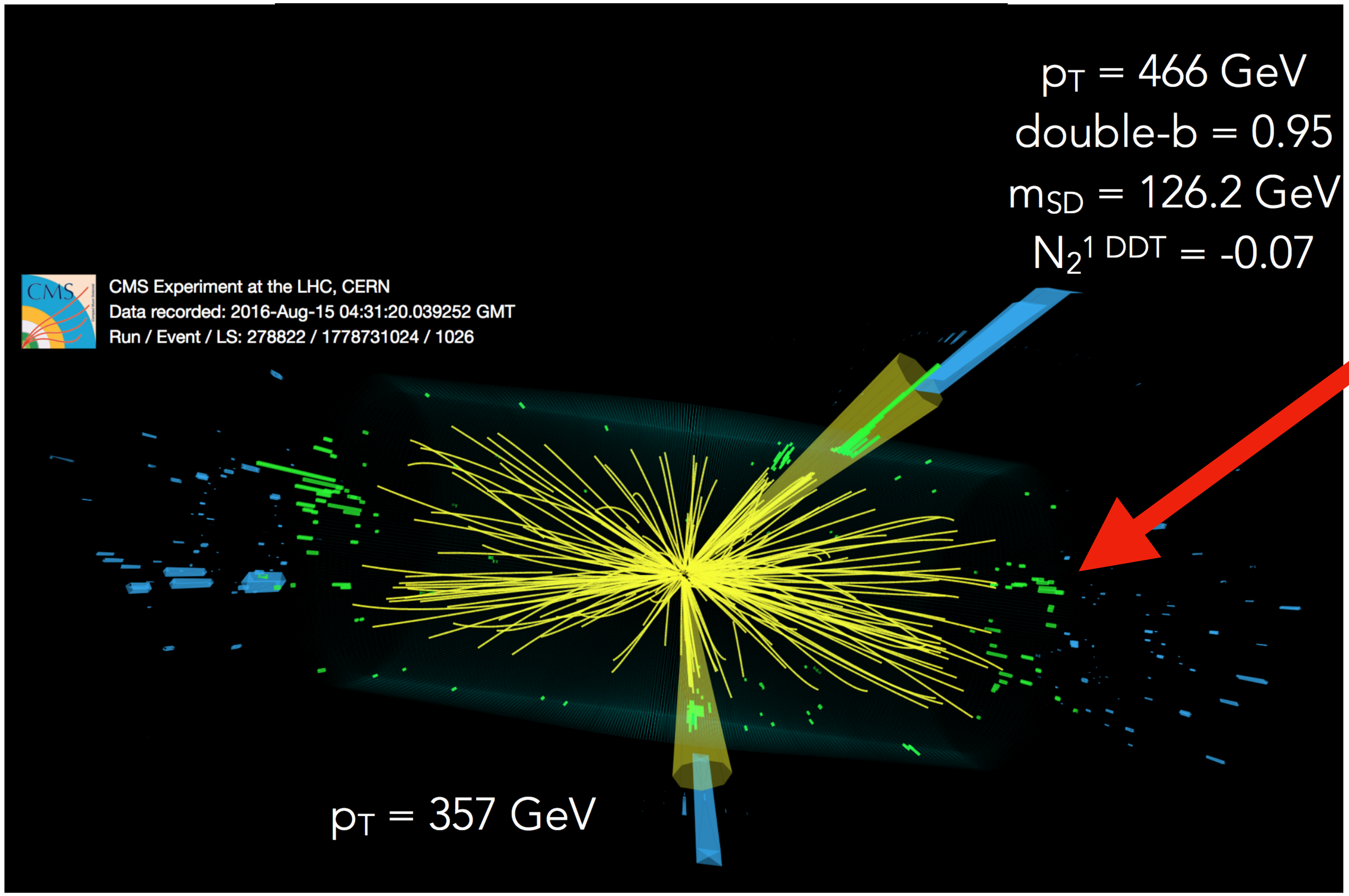
Conventional computing
Industry is building a lot of tools
Our input can drive innovation

Custom ASIC+FPGA System
ML needs to be done in <1μs
Requires a rethink of ML processing

Hidden gems?



- There is a plethora of physics that we throw out



Higgs boson right on the cusp of being thrown out

The dream

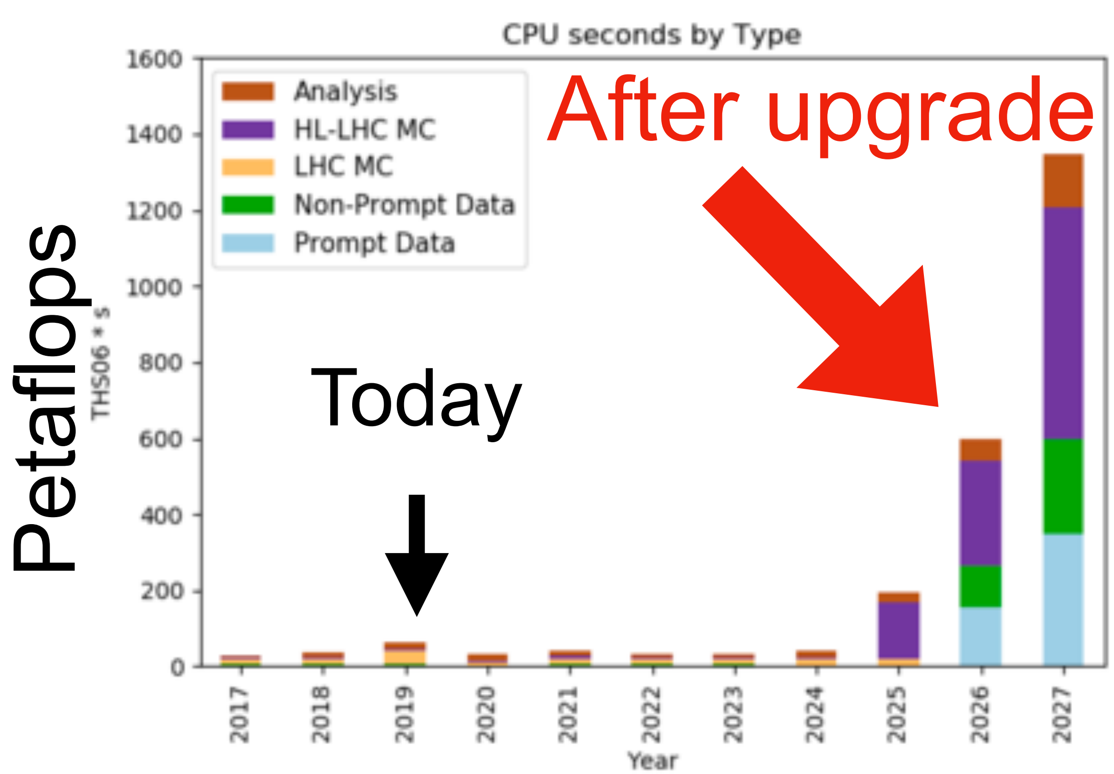
- At the moment:
  - We only get a full data of one in 100,000 collisions
  - There is interesting physics that we have to throw away

- 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

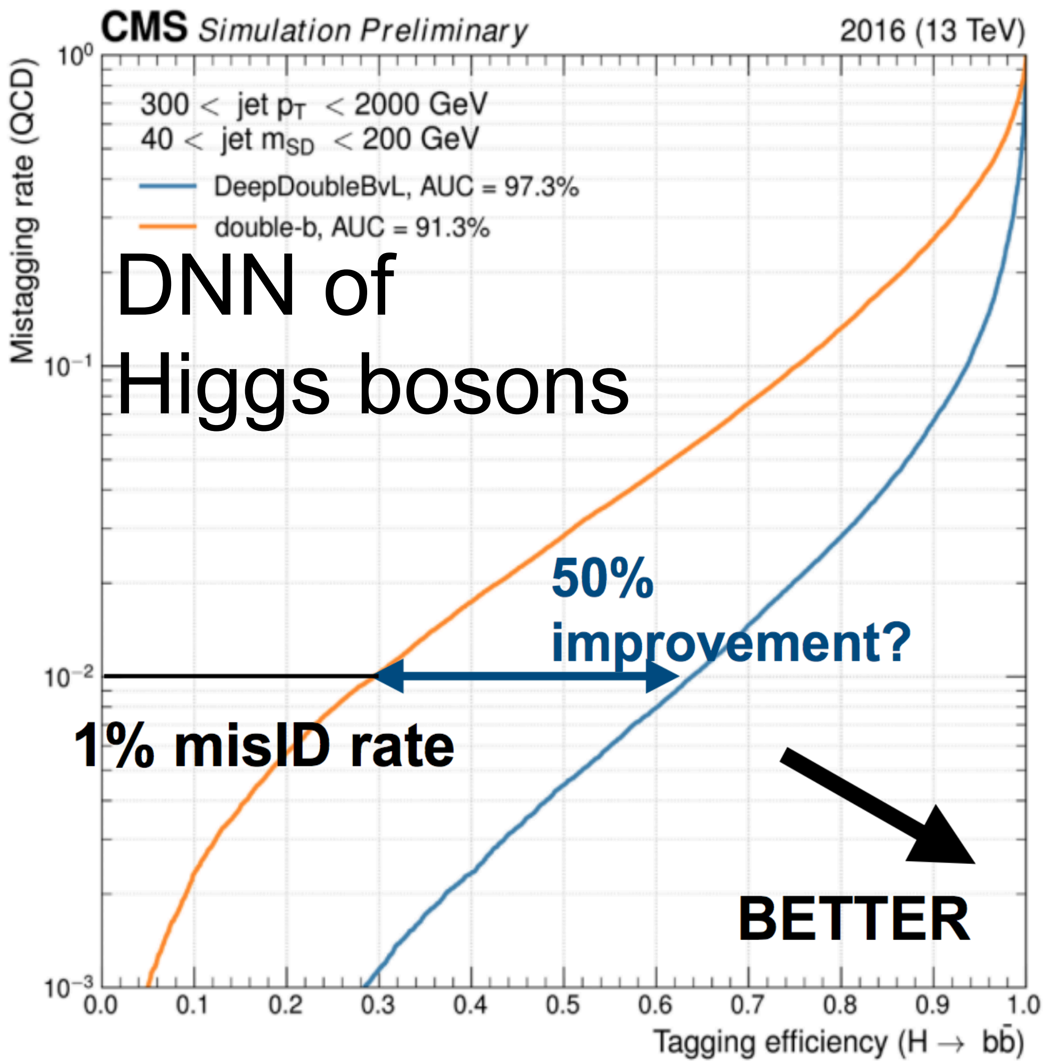
The Challenge

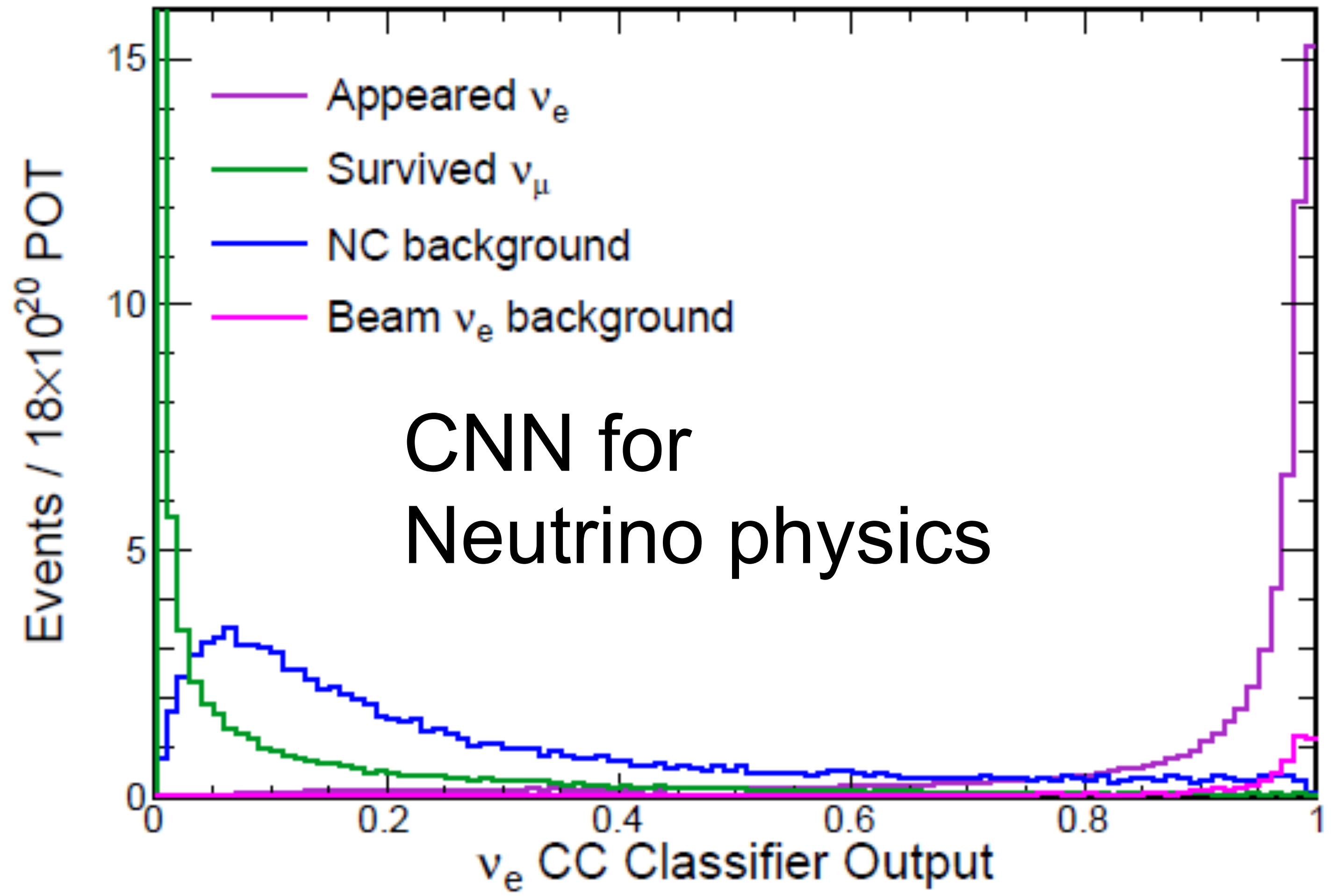
- We are upgrading the system
    - Our event size will be 10 times larger
    - And we have to take data at 5x the rate
        - Need this just to preserve our existing physics
- 10s of years of processing without modifying system

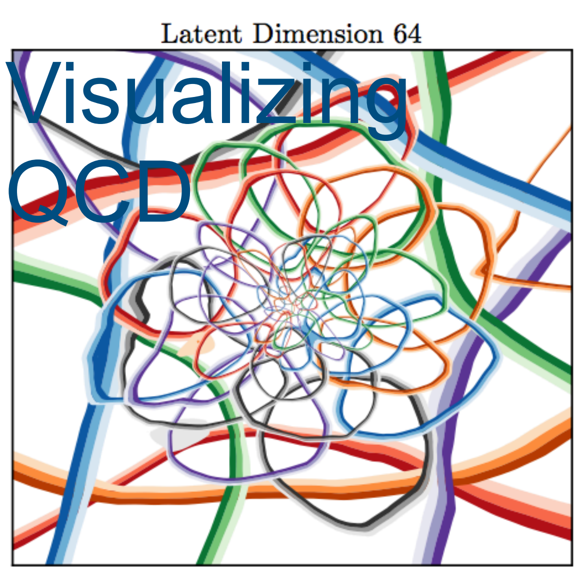
End of Dennard Scaling is about to hit us hard



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

Deep Learning L1 Trigger

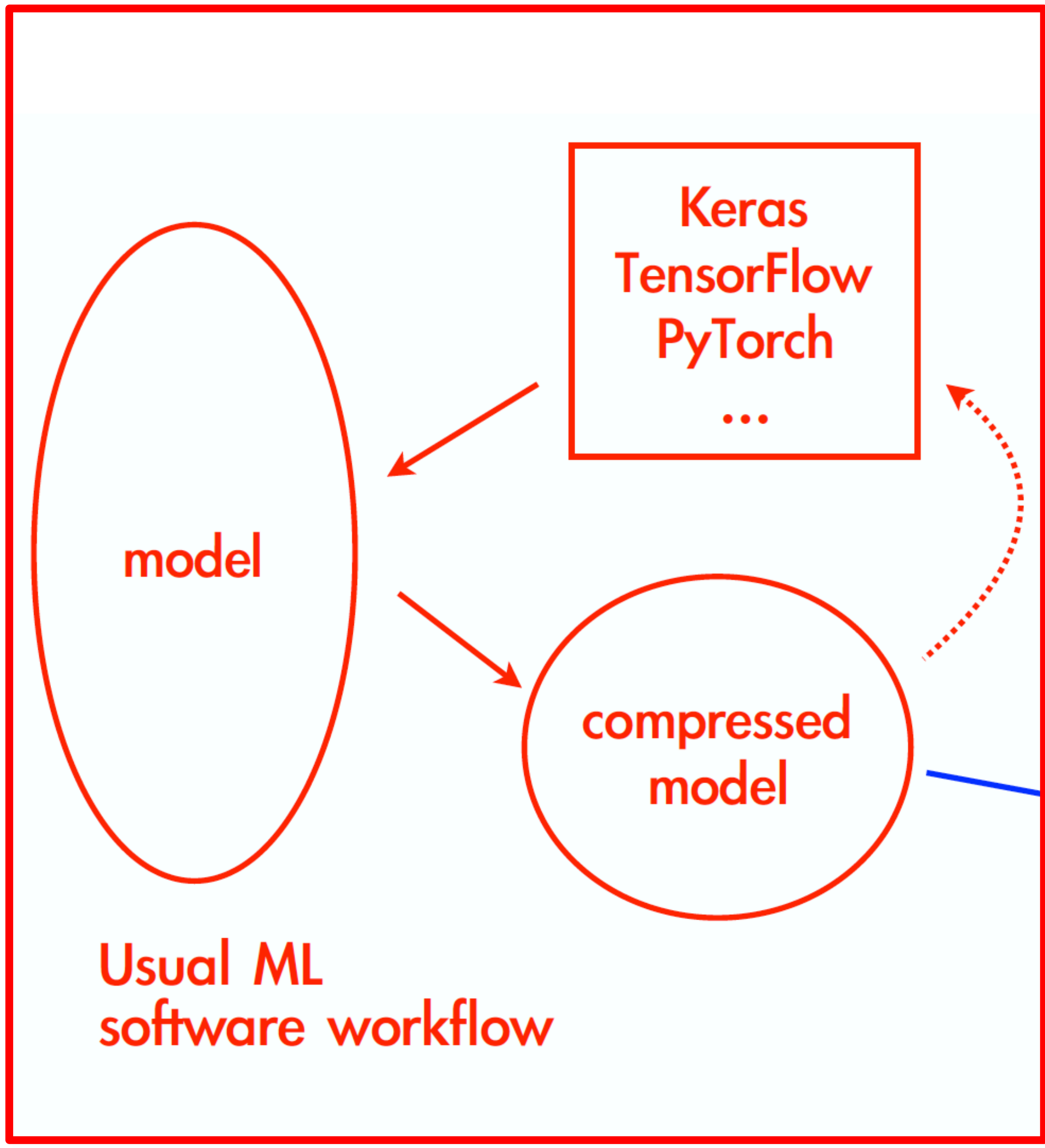
- Have at MOST 1μs to run an algorithm
  - We aim for algorithms that are in the 100ns range
- **Want to make the fastest possible algorithm**
- **Want to have the smallest initiation interval**
  - A collision is every 25ns (40 MHz)
  - We apply algorithms to multiple subsets of total event
    - That means we need applications in every X < 25ns

hls4ml

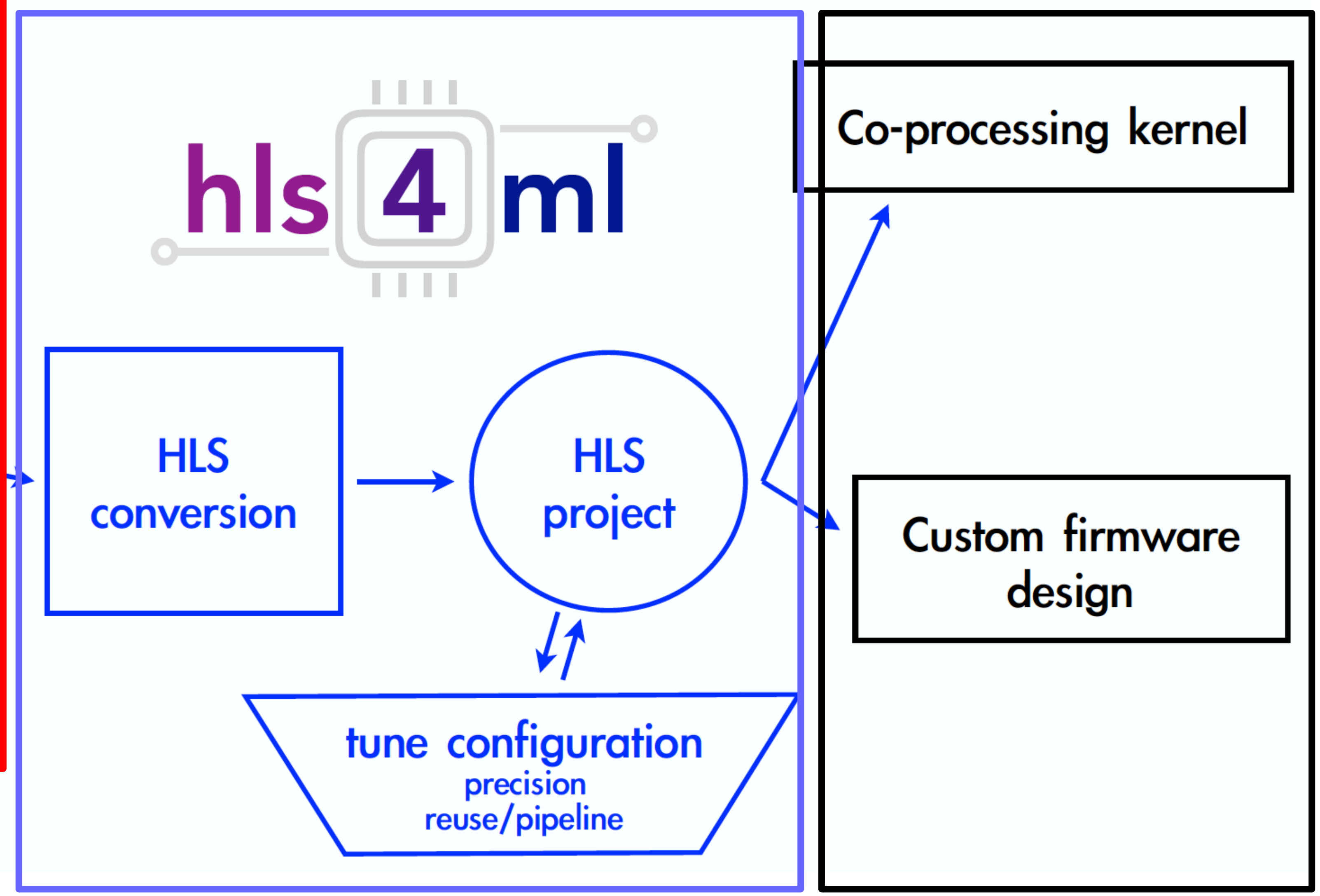
Summing Up the Data flow

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

Targeting Ultra low latency applications



Usual Training Step



HLS tuning

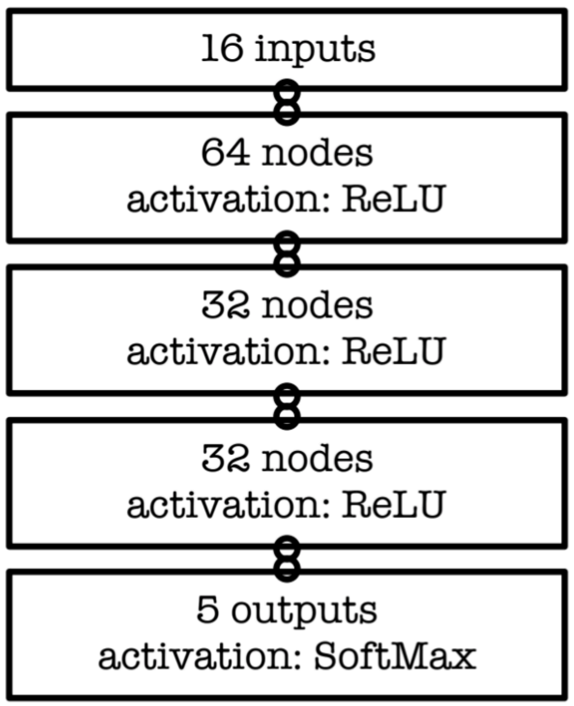
Final Product

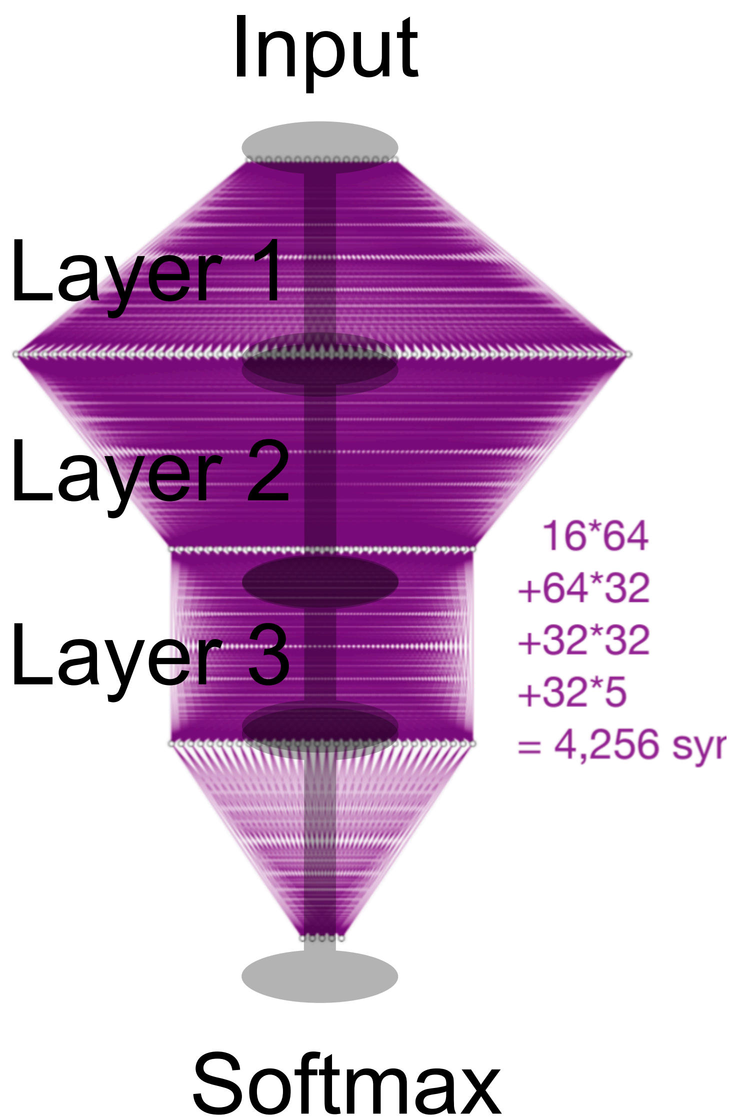
Support in HLS4ML for

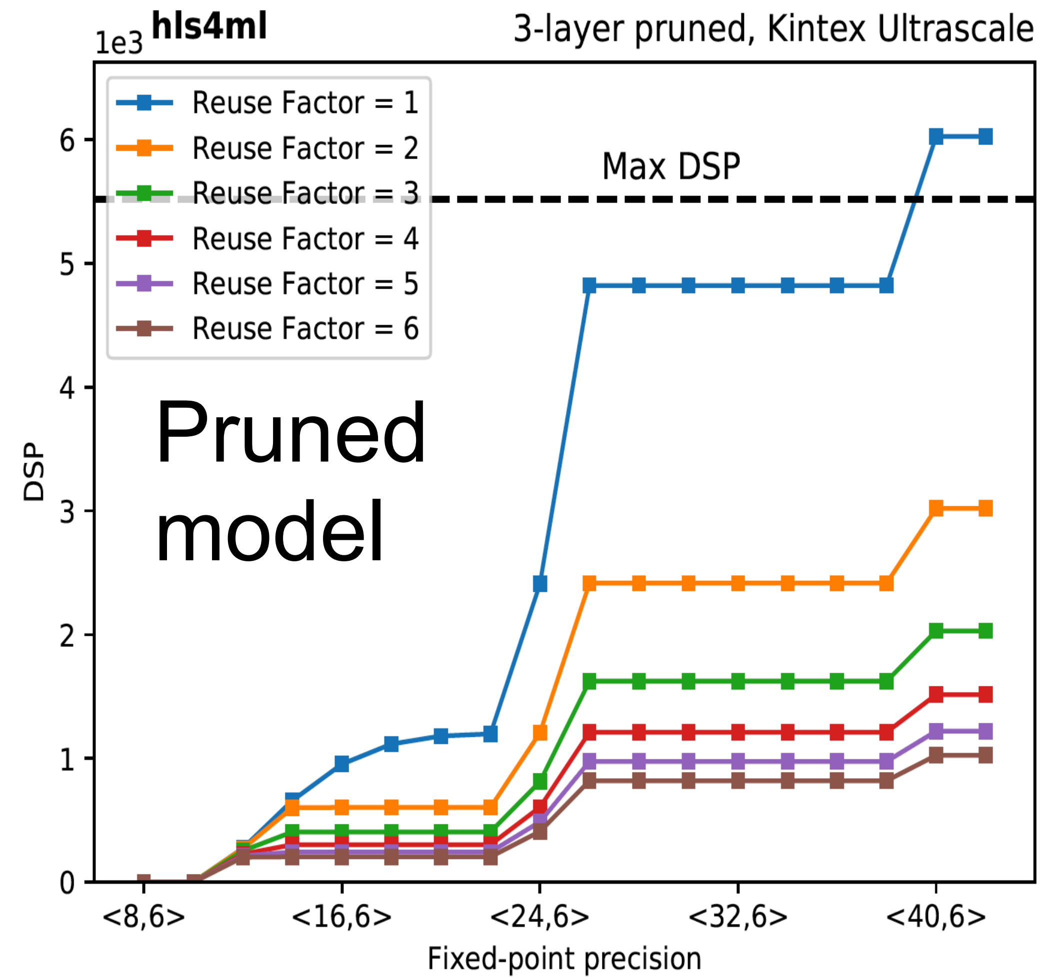
MLPs,CNNs,Binary/Tenary NNs,BDTs,Graph NNs,LSTM/GRUs

What can we run?

Case Study:
Particle Jet classifier







75ns latency new input every 5ns
fits in a a conventional FPGA (VU9P)

Design at the L1

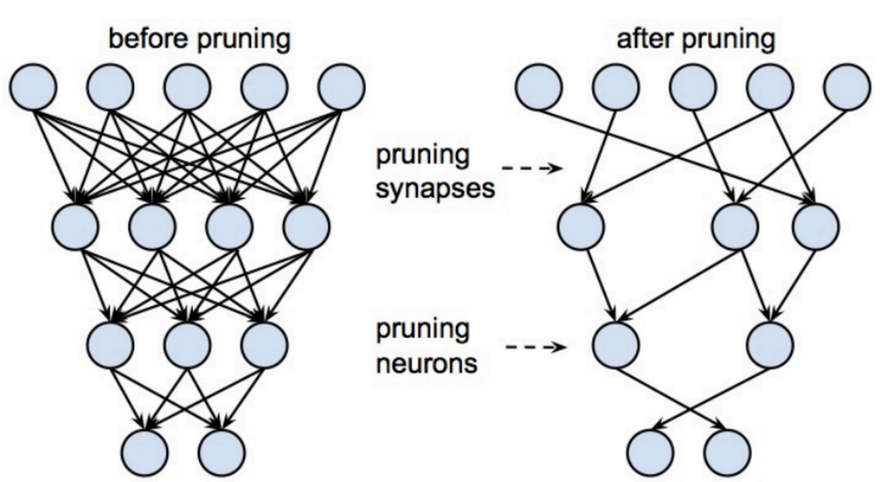
Focus on 3 ways to cut down resources

Is our algorithm overly complex?

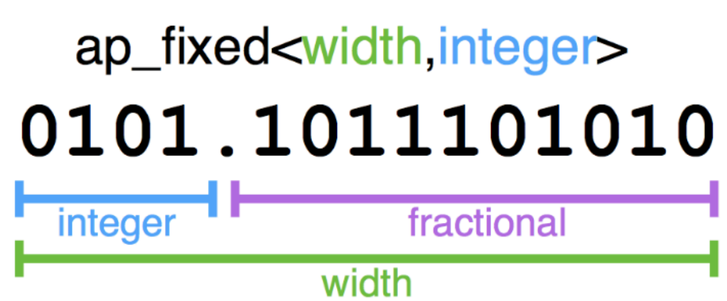
Are we too precise?

Does it really need to be this fast?

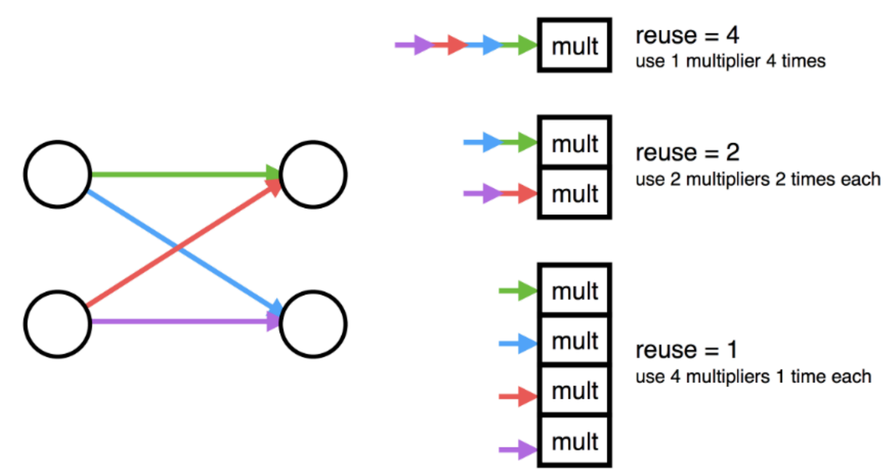
Algorithmic Compression



Quantization



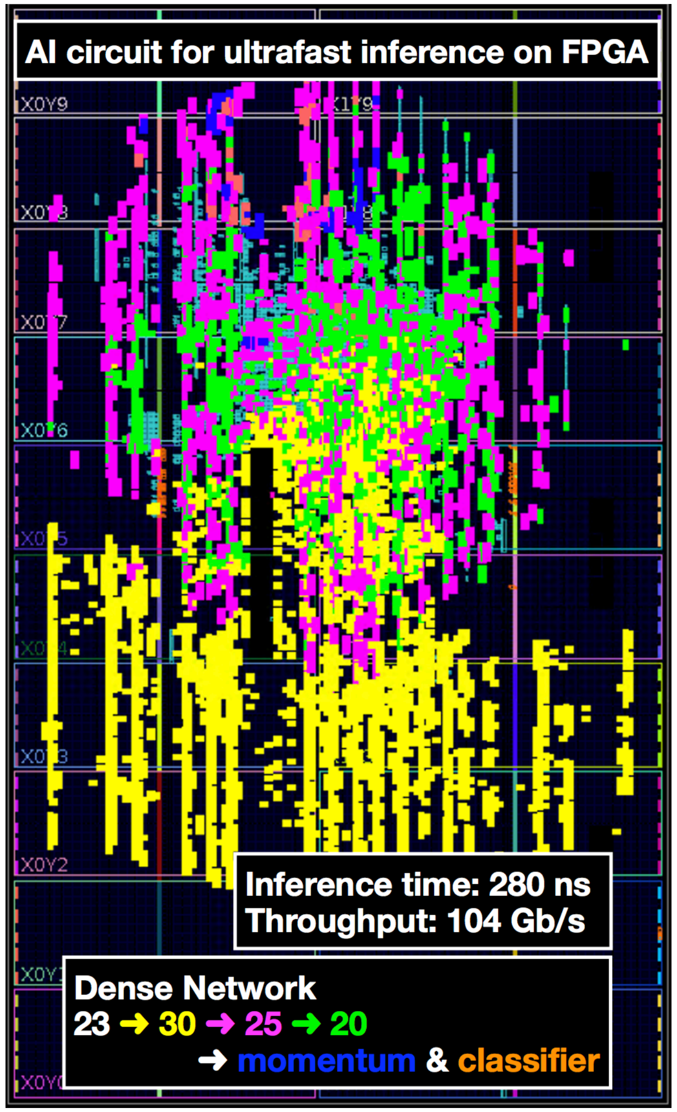
Reuse Factor

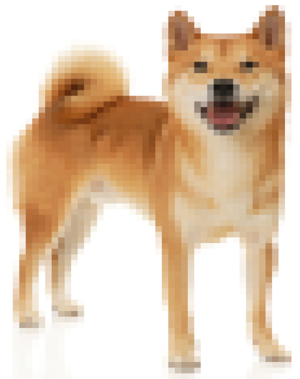


Current Status

- Tool: quickly adopted for a number of applications
  - Muon p_T reconstruction with an NN
  - Autoencoder at Level 1
  - NN based Tau lepton identification
  - Jet substructure at L1
  - .....
- Tutorials exsit using AWS f1 FPGA cluster

Targeting LHC Run 3

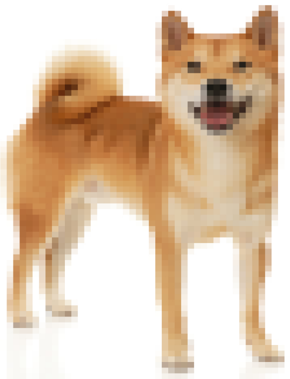


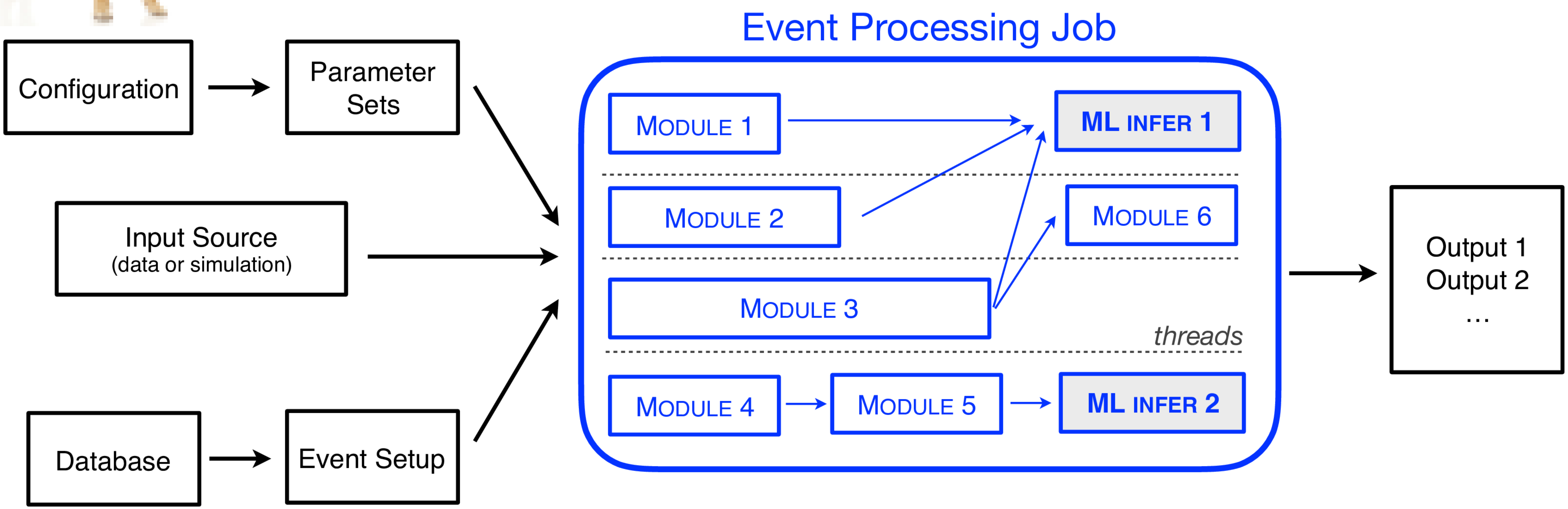


Deep Learning in HLT

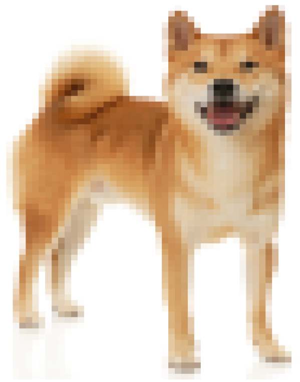
- Previous systems have been CPU only
  - New systems will likely be heterogeneous (FPGAs/GPUs…)
  - Some parts of reco can still benefit from use of CPU
- With timescales at the level of milliseconds
  - Utilize industry tools to use heterogeneous systems
  - Can consider GPUs for parallelized processing
- ML on the millisecond timescale used by many industries
  - ML inference engines (ML-as-a-service) MLaas

Reco Strategy





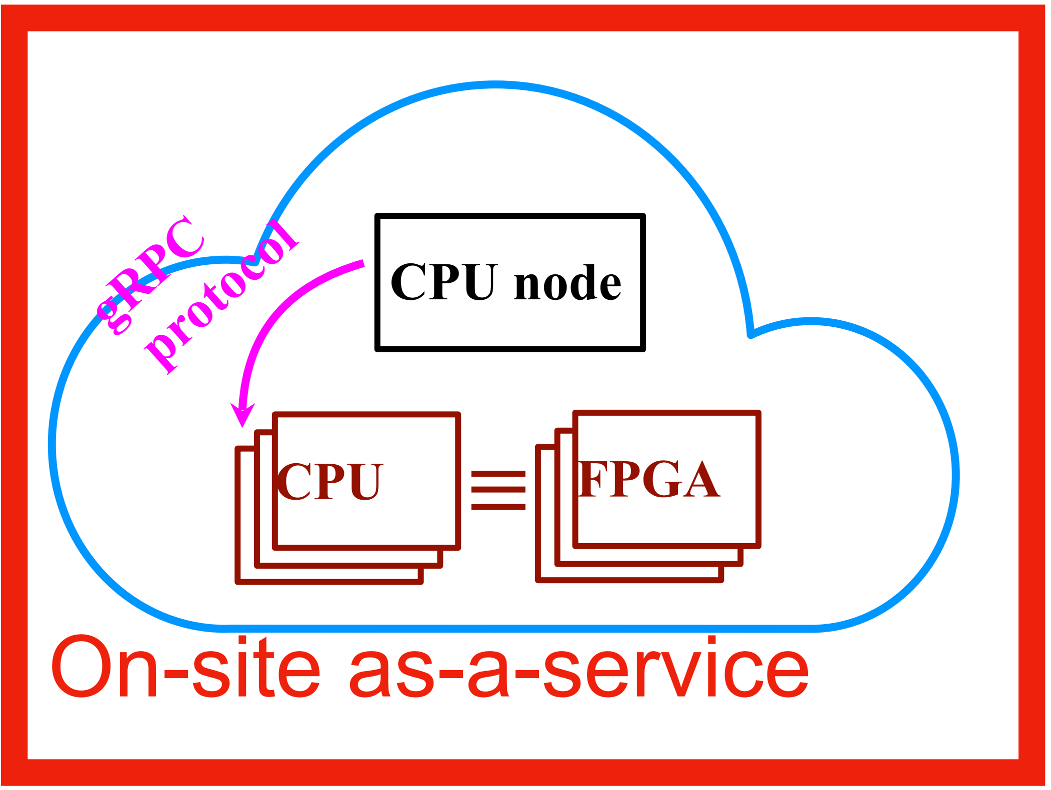
- Complicated scheme of modules
  - While some parts are parallelizeable
  - Collision level analysis built in by construction **(Batch 1)**

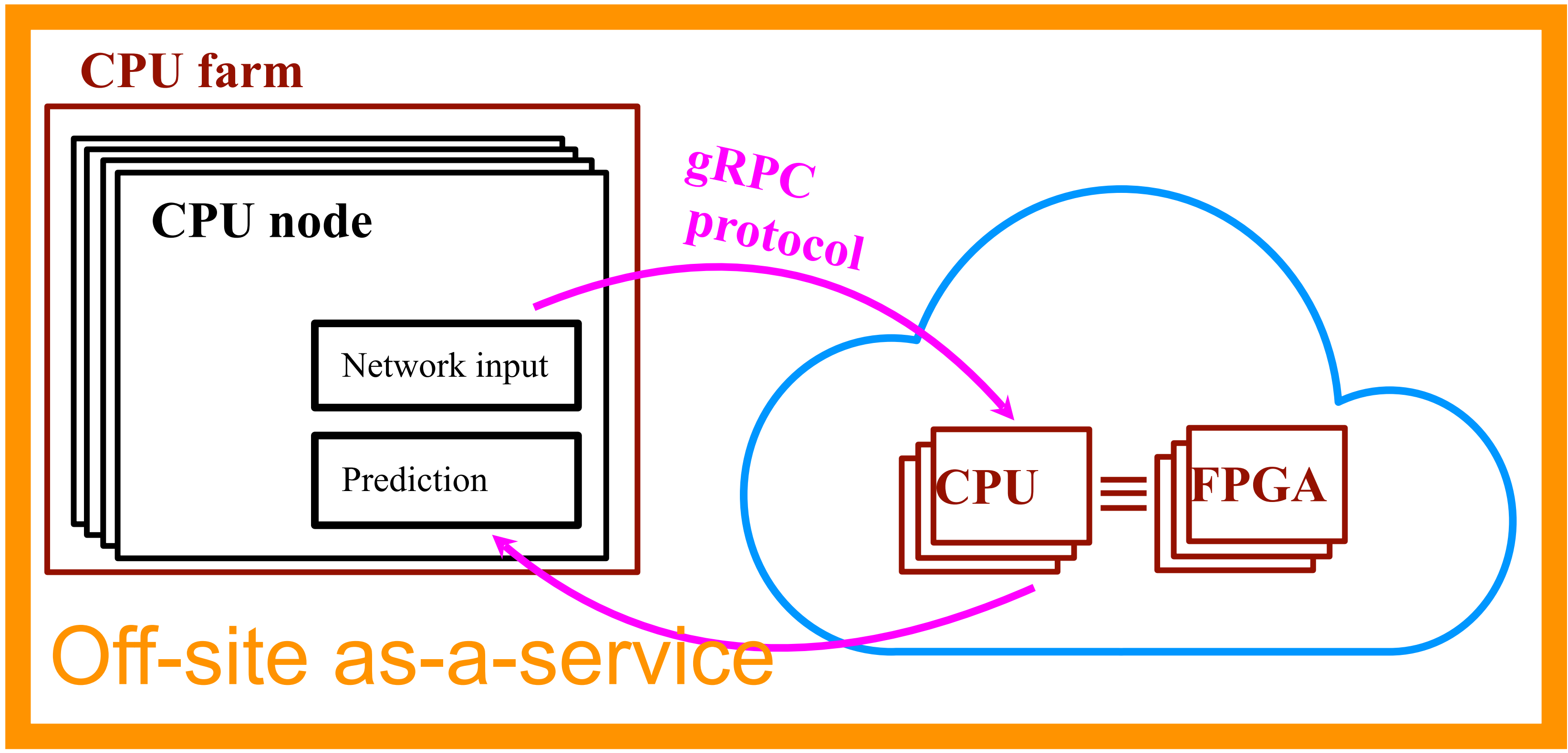


Service Options

Low latency Triggering

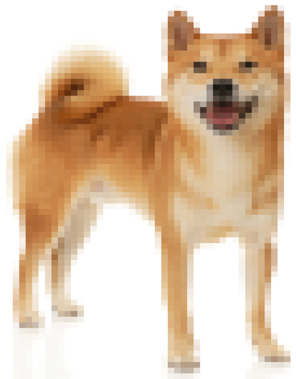
Larger latency but still large throughput (future slides)





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

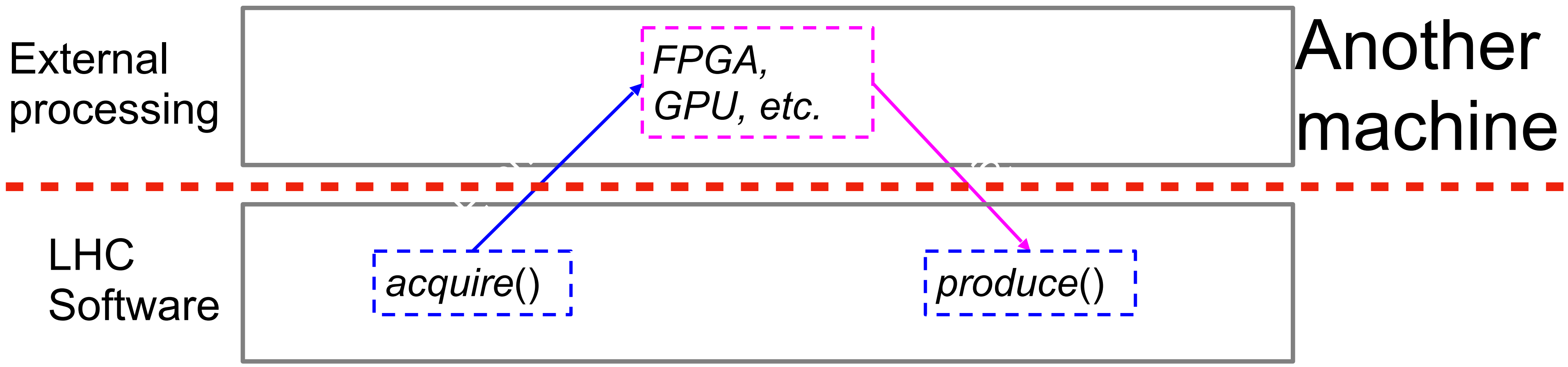
For timescales in <100ms, this is not an option



Offloading to Hardware

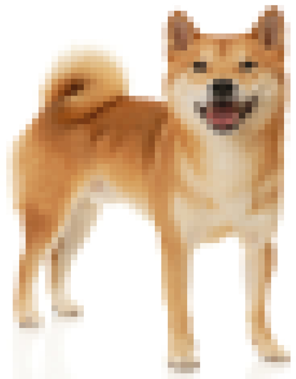
- To run these algorithms within our software

Asynchronous task based processing



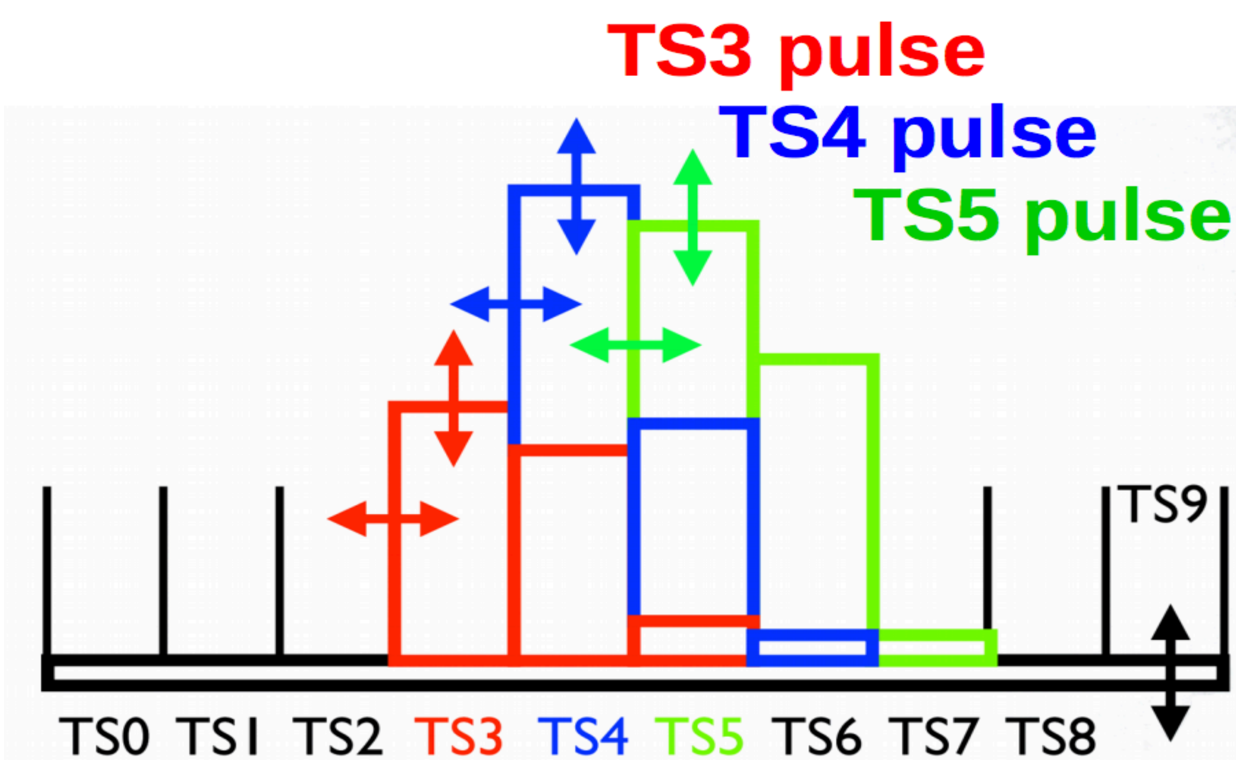
Non-blocking: schedule other tasks while waiting

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



Examples

Hcal Reco



Energy reconstruction of Hadronic showers

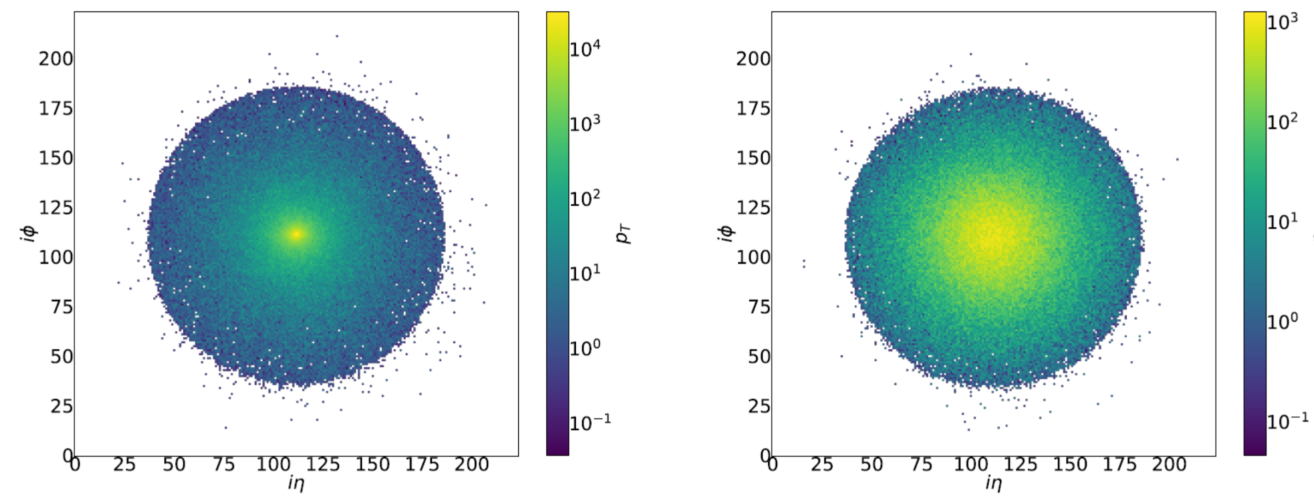
Simple energy regression

16000 times per collision

Batch N per particles

Image Top tagger

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



Top quark identification

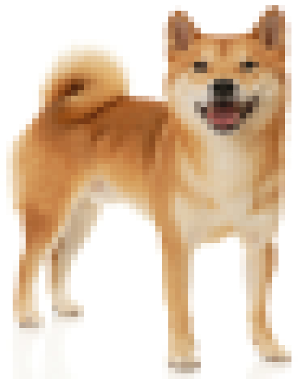
Here we use Resnet50 as benchmark

Complicated identification

Many inputs

1-2 times per collision

Batch 1 per Event



Performance

Hcal Reco

Algo	Per Event
Old CPU	50ms
NN CPU	15ms
NN GPU(1080 Ti)	3ms
NN FPGA	2ms
+Off Machine	+10ms
+Offsite	+40ms
throughput+offsite	3ms

GPU : Use tensorflow+tensorrt
FPGA: HLS4ML+SDAccel

Image Top tagger

Algo	Per Event
CPU	1.75s
GPU Batch 1	7ms
GPU Batch 32	2ms
FPGA	1.7ms
+Off Machine	+10ms
+Offsite	+40ms
throughput+offsite	1.7ms

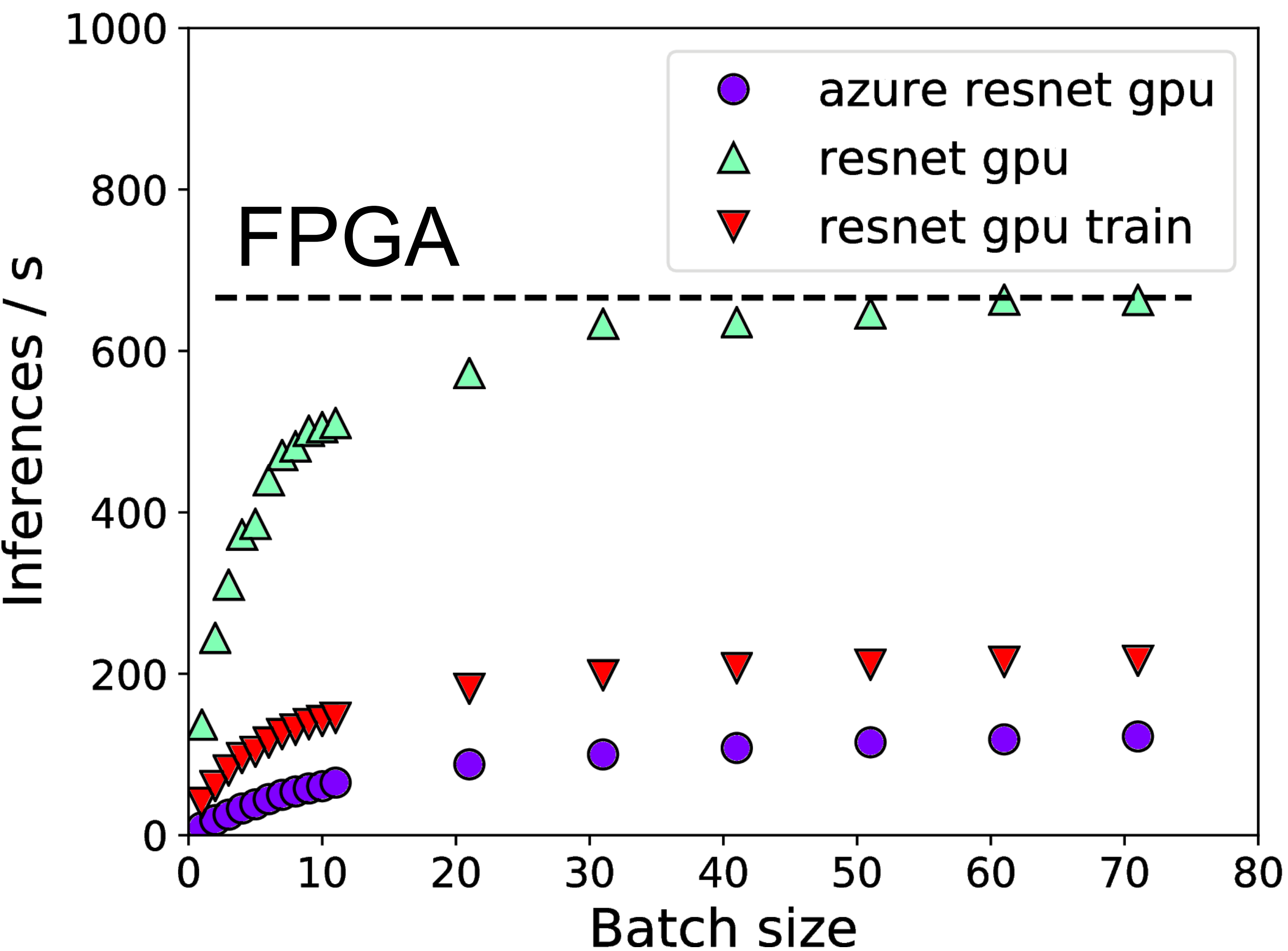
GPU : Use tensorflow+tensorrt
FPGA: Microsoft Azure

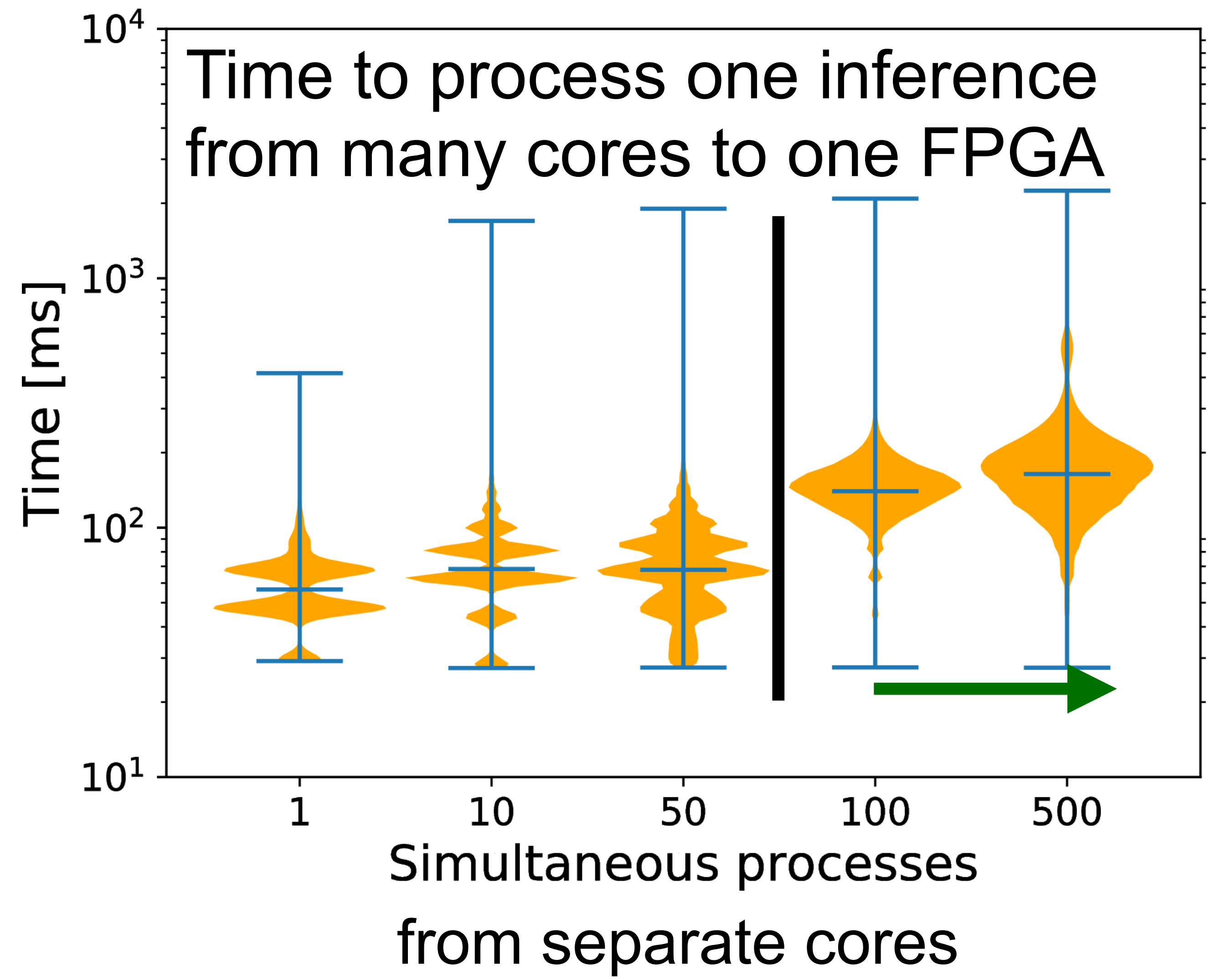
No GPU nor FPGA code needed to implement these optimally!

Batch 16000 per particles

Batch 1 per Event

Processing Technology



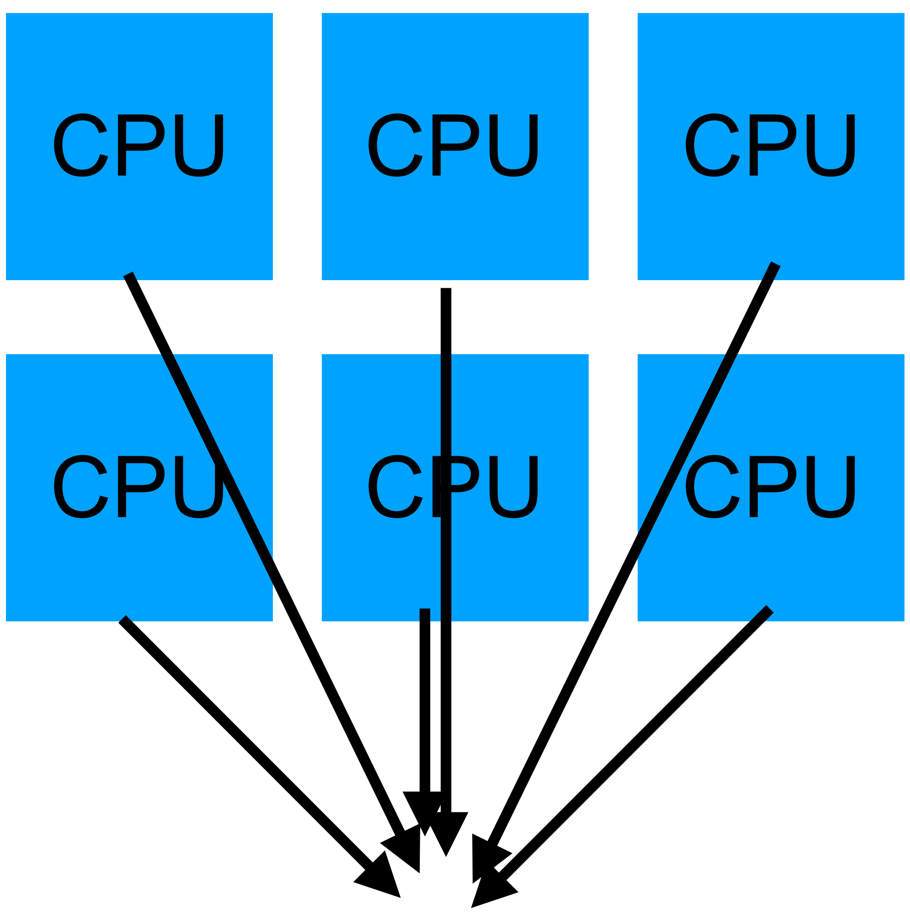


With a GPU we can get to FPGA level of throughput, but long latency

Speedups on a single FPGA can serve many different CPU cores

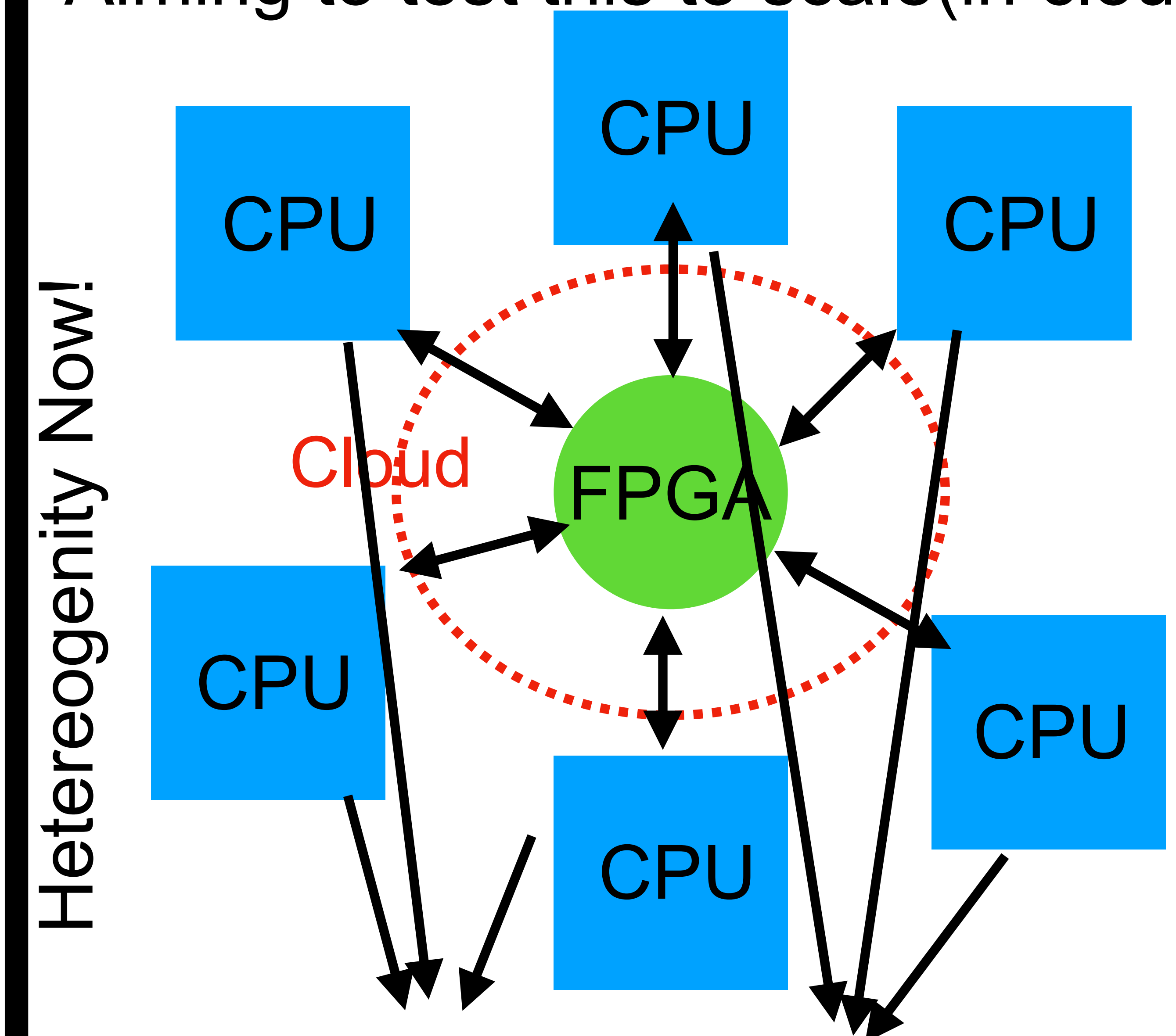
What have we learned?

- With large speedups we can redesign our system



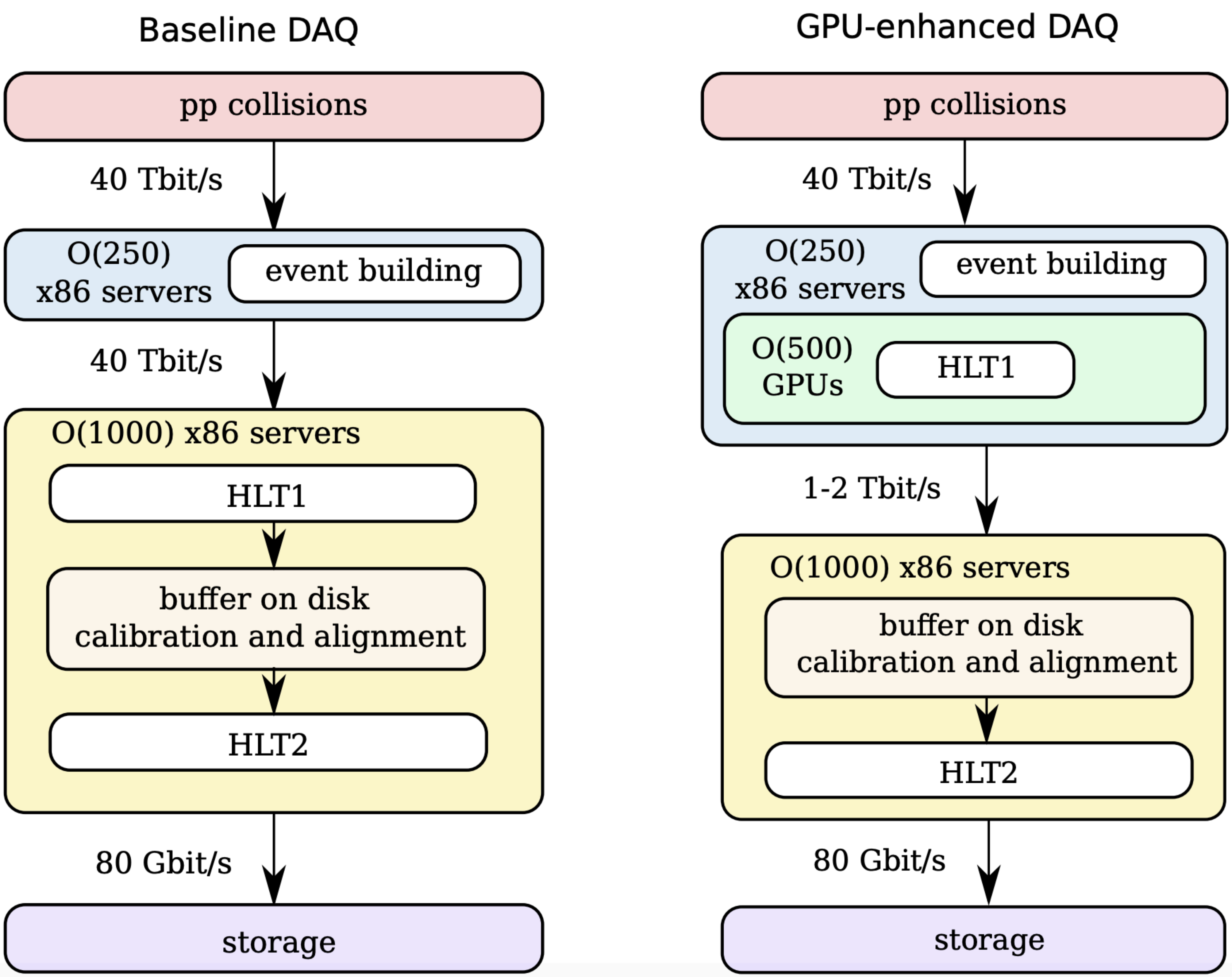
Process event by event

Aiming to test this to scale(in cloud)



Process (event by event)?
outsource to aaS

Alternative GPU model



Full Reconstruction algorithm ported to GPU

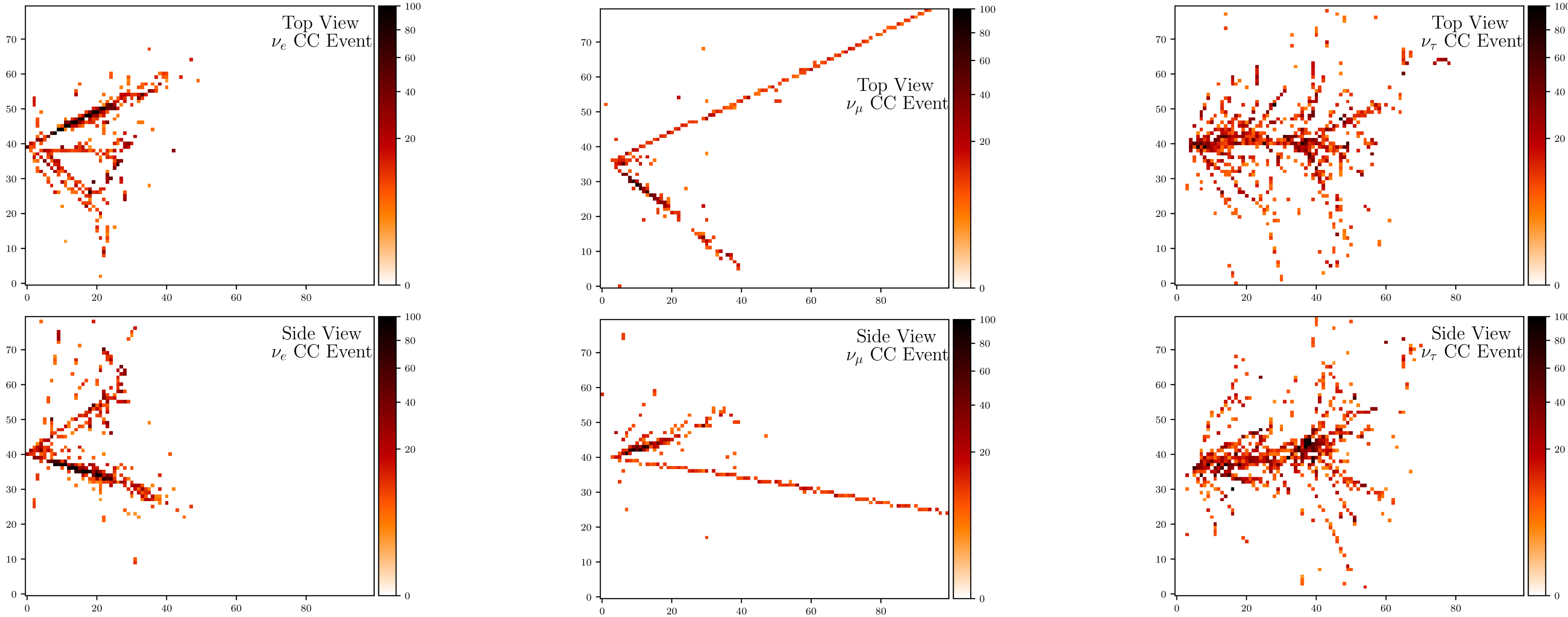
Beyond the LHC

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

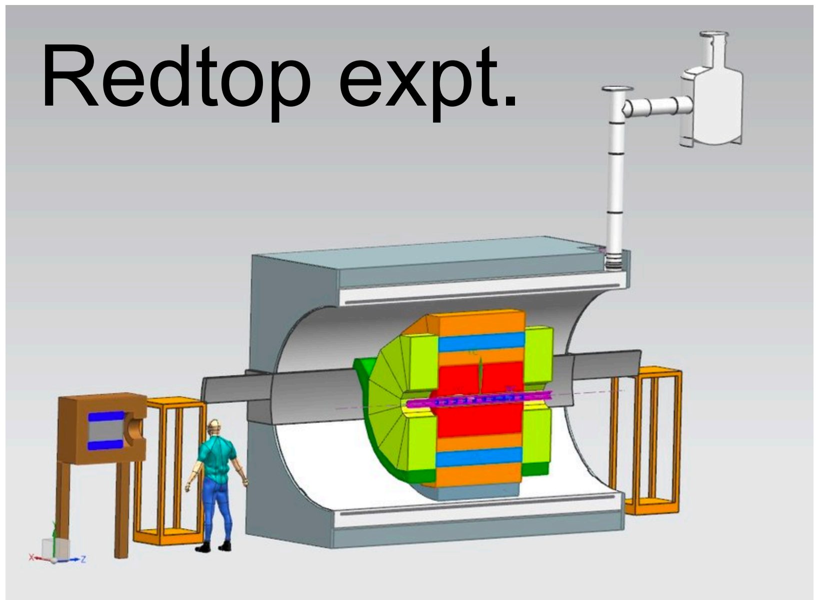


Reconstruction can be performed with a CNN (Resnet-like)

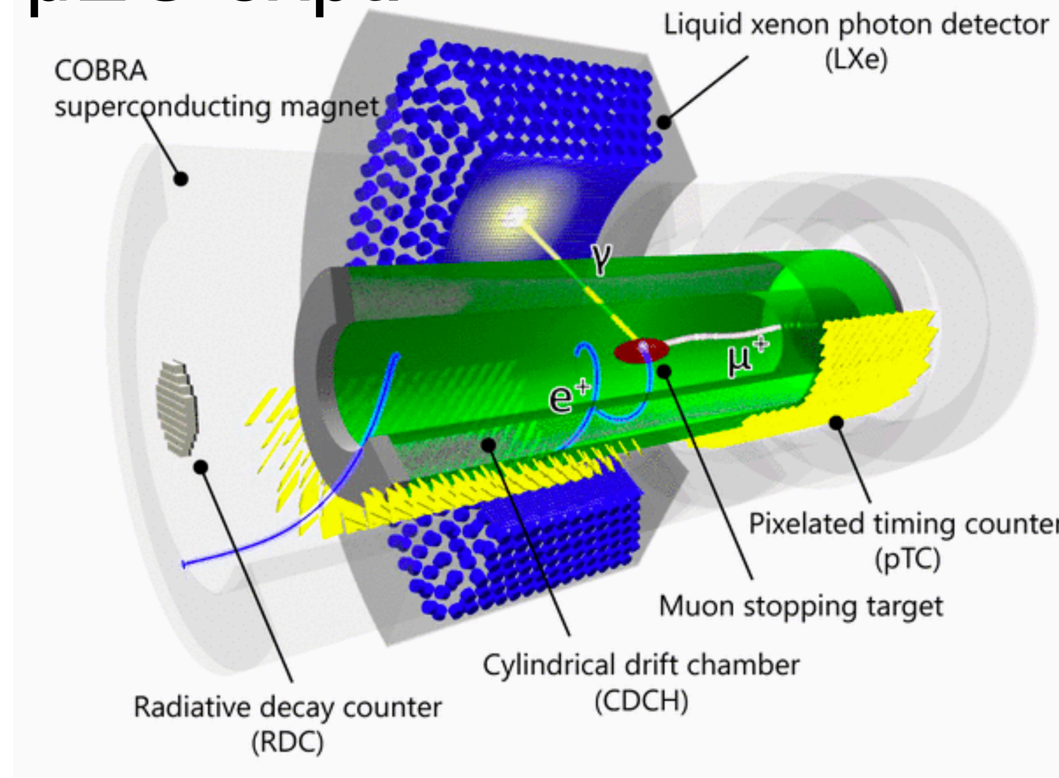
Future detectors will have to deal with 40 Tb/s of data

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

Beam Dump Exps.



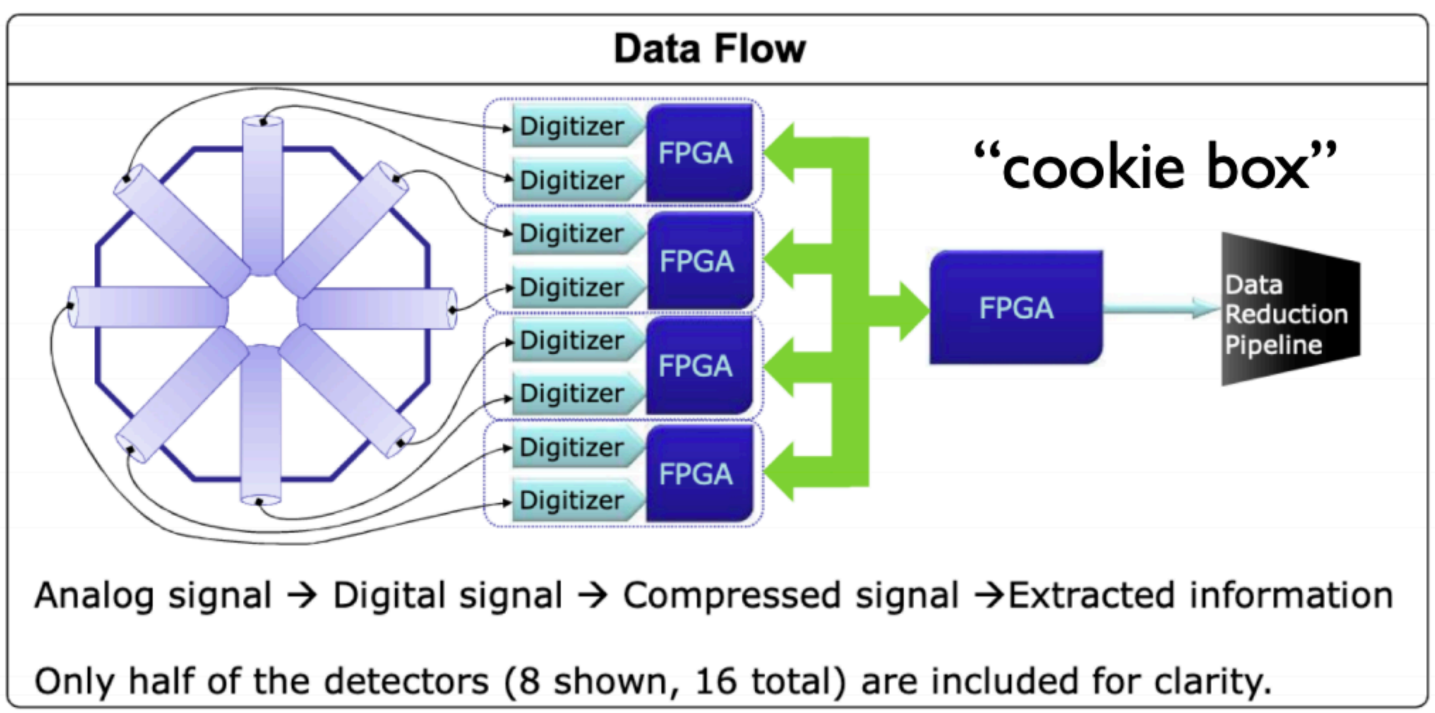
μEG expt.

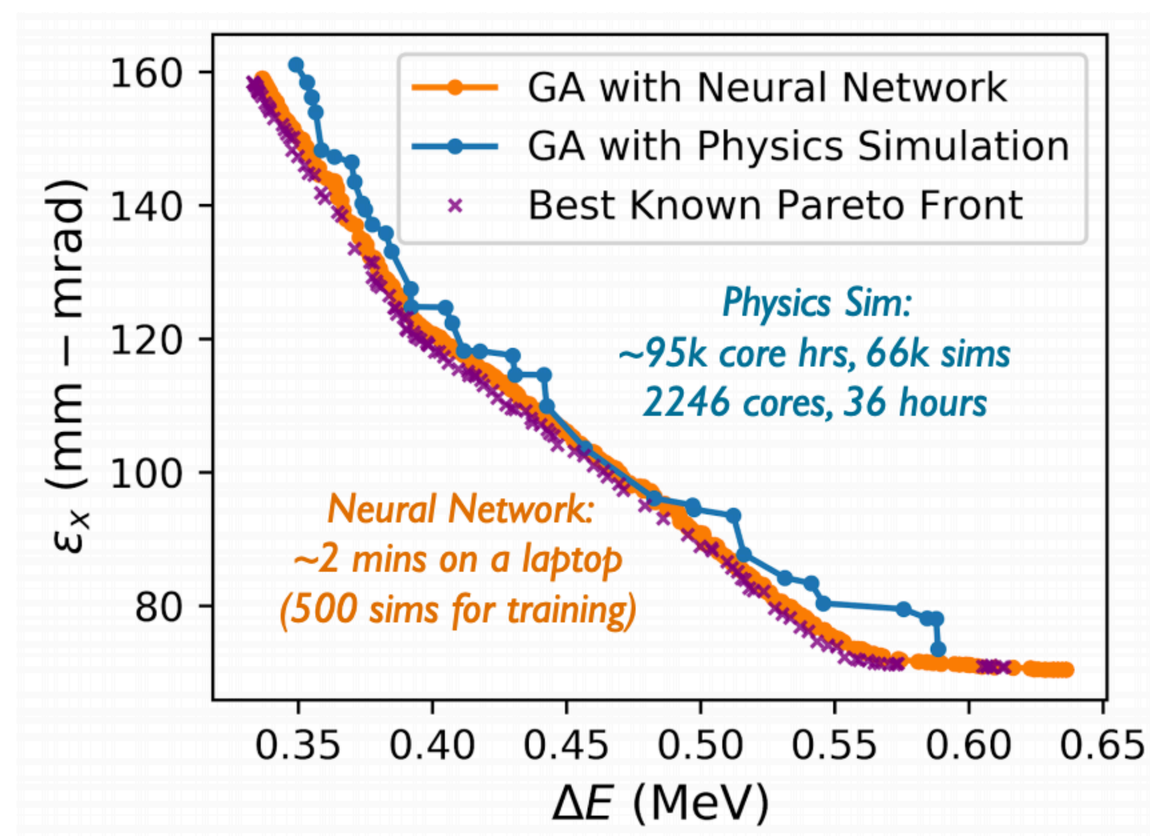


- Very large data rate expts benefit from real time processing
  - A wide variety of beam dump experiments would benefit
  - ML is a great way to preprocess and compress data
  - Allows for fast high rate processing

Particle Accelerators

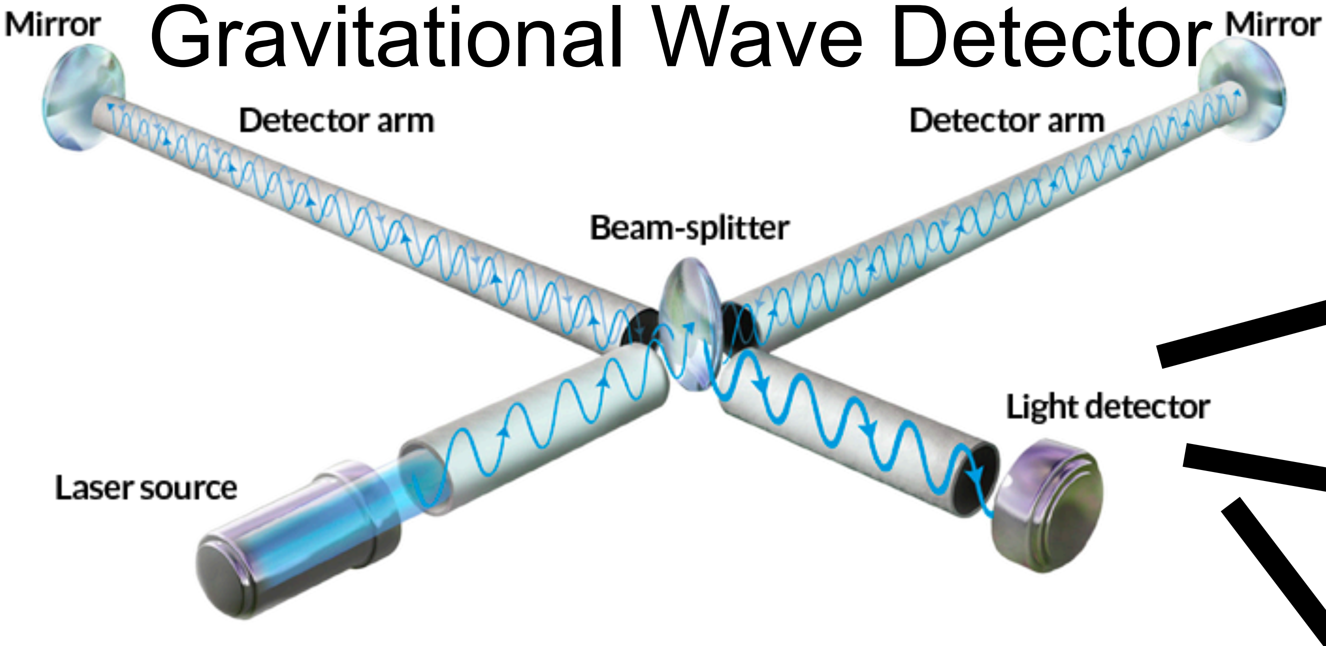
- Demands for high speed control of accelerator systems
- Large data rates to monitor and control beam dynamics
- Have had continual success with ML solutions for modeling



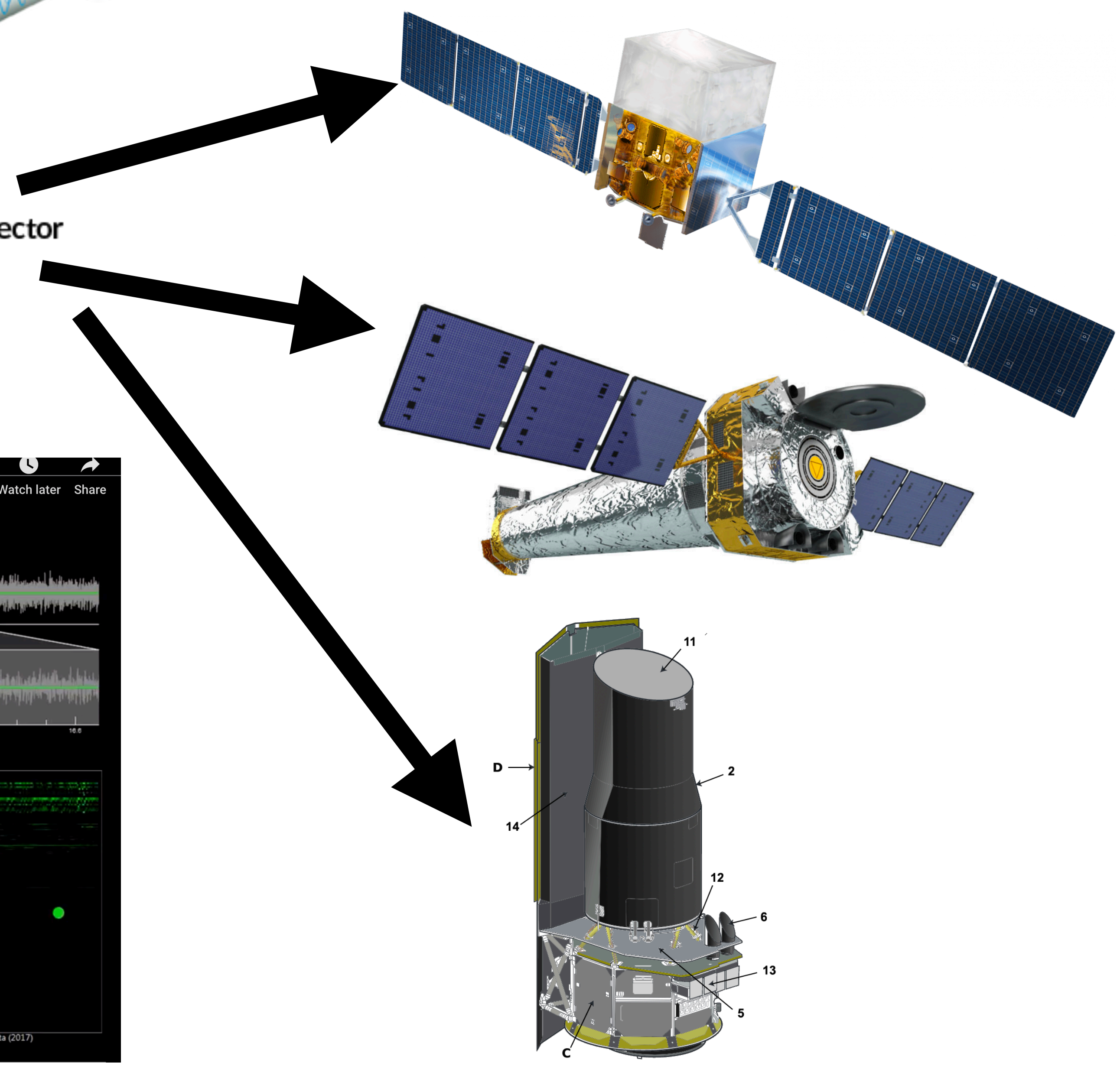


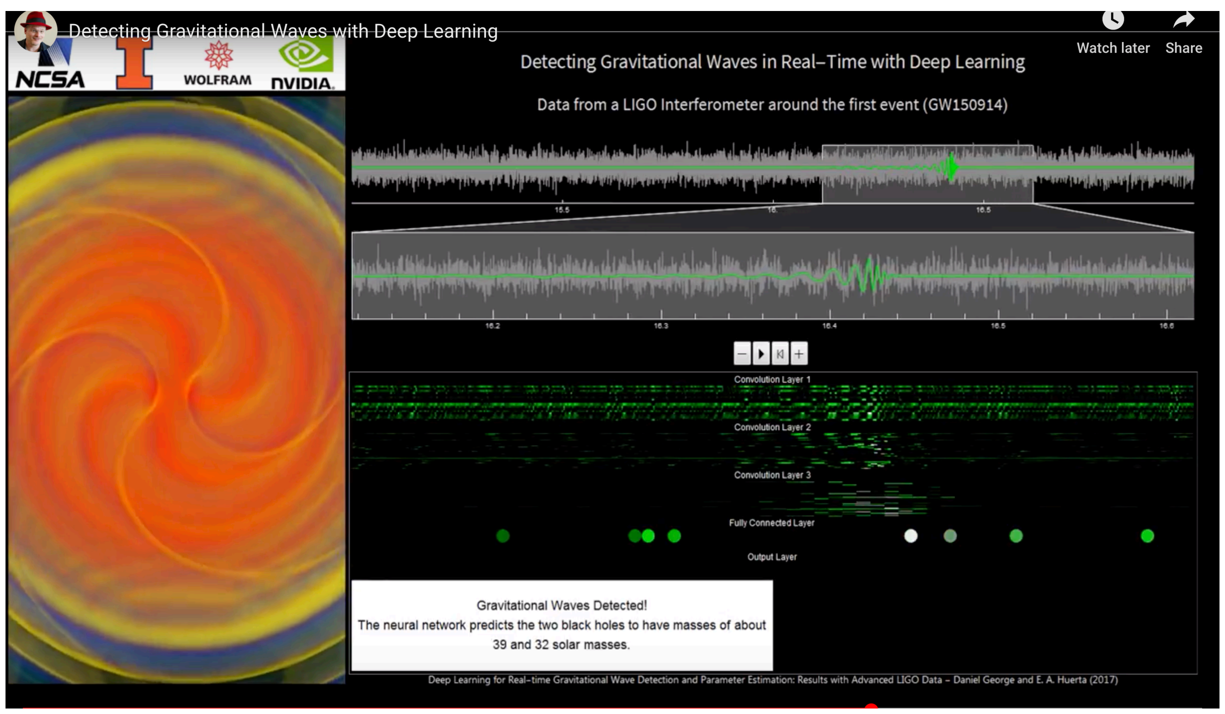
Gravitational Wave Detection

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

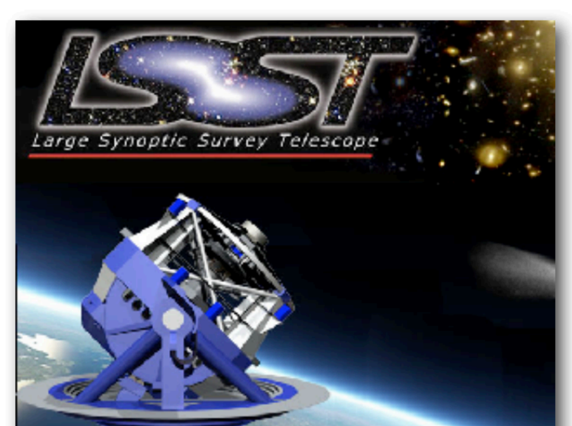
LSST will produce over **10 million** transient alerts per night.

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

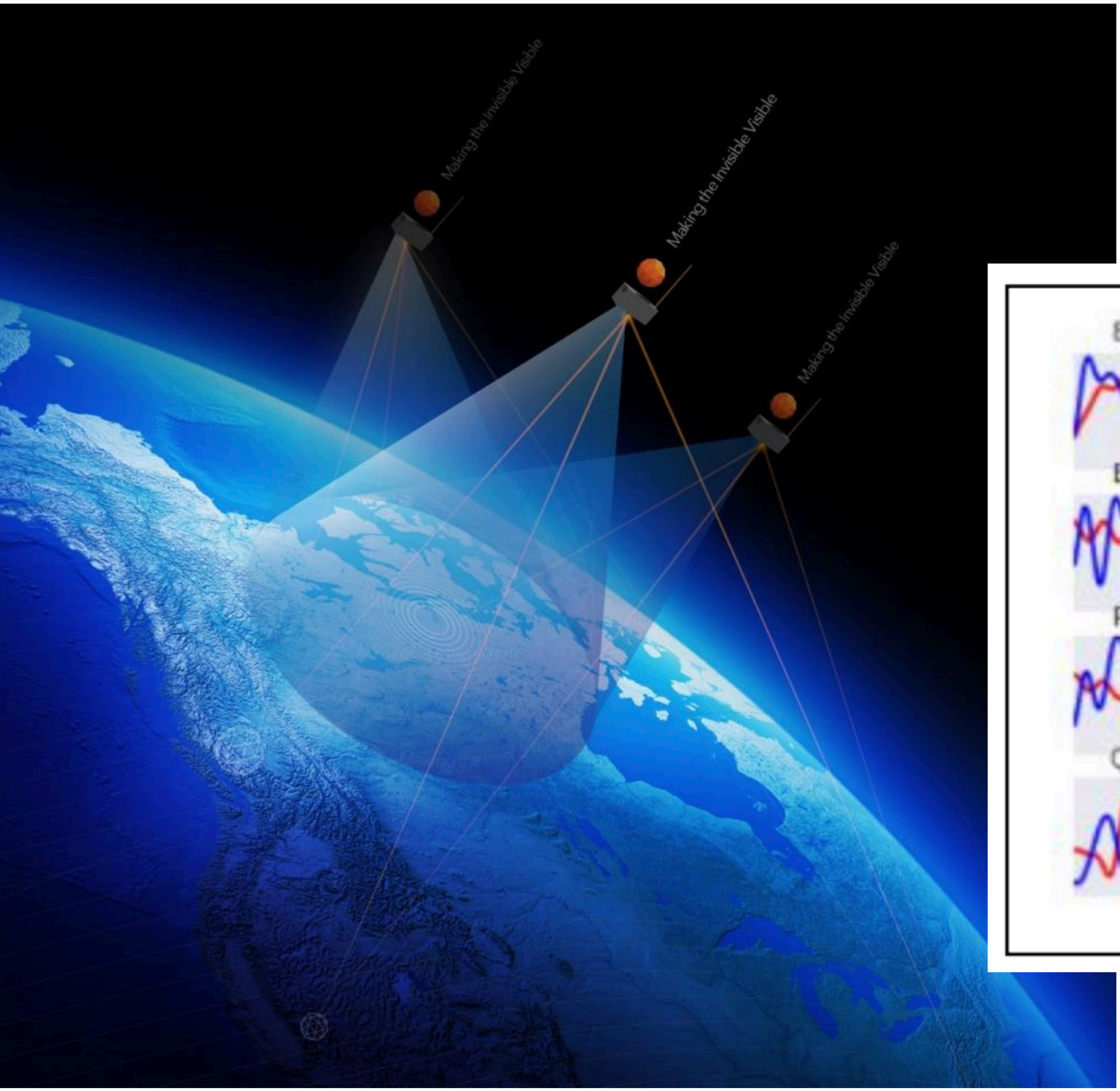
SDSS I-II
2000-08
2.5-meter mirror
O(10^8) Galaxies
10k sq. deg.
0.2 TB/Night

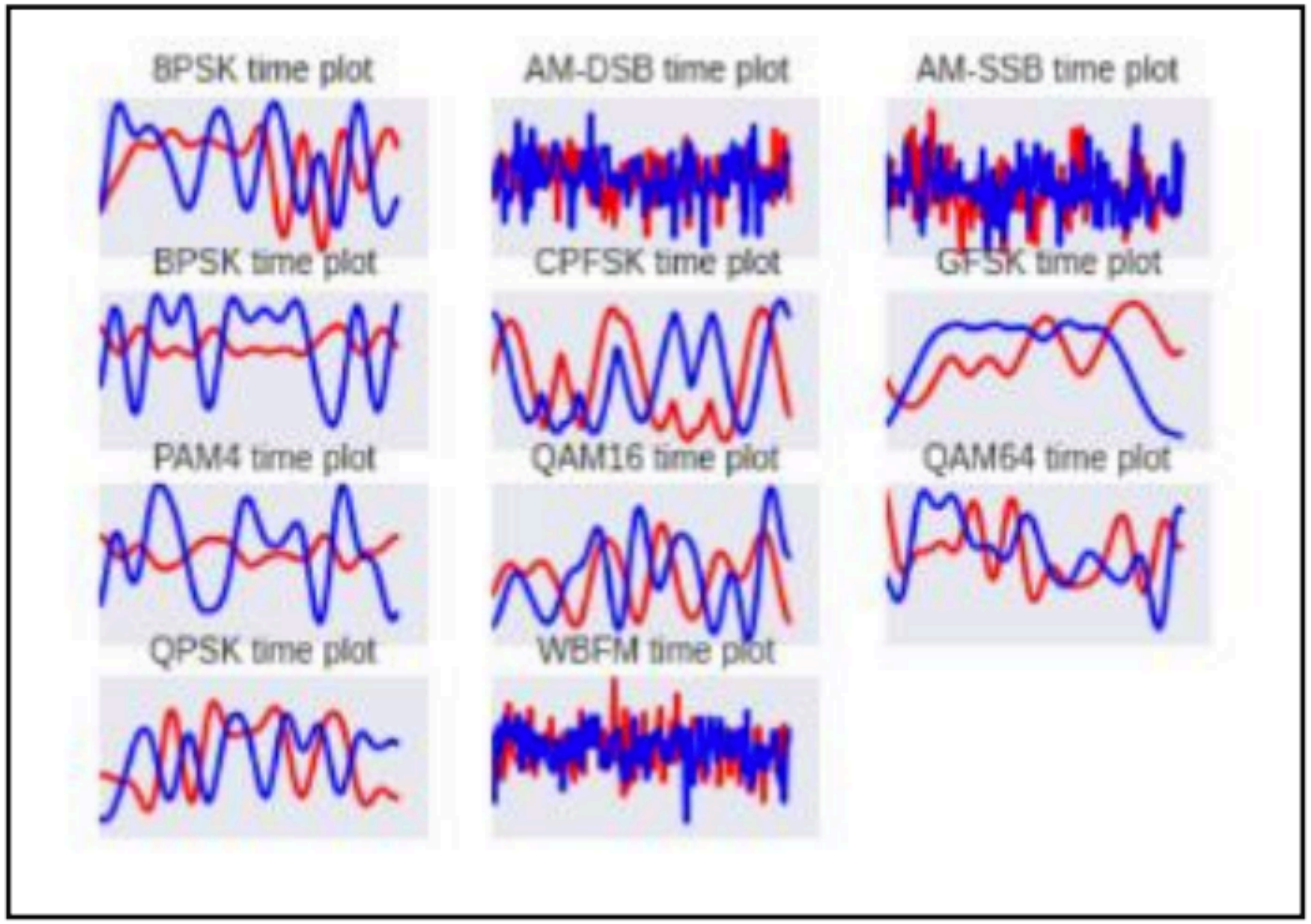
DES
2013-18
4-meter
O(10^8) Galaxies
5k sq. deg.
1 Tb/Night

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



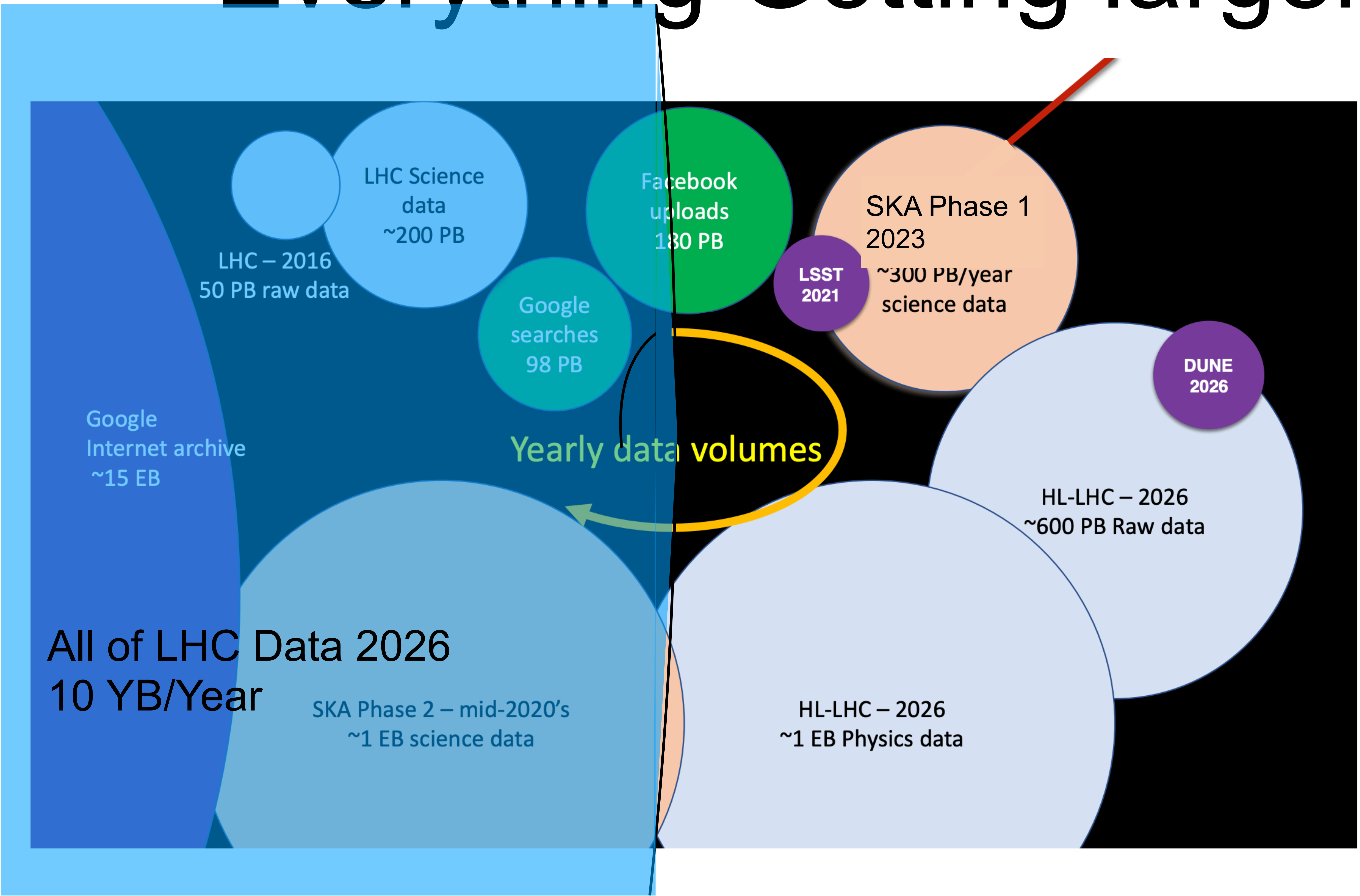
Identification of transients require real time processing of all data


Making the Invisible Visible
Making the Invisible Visible
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

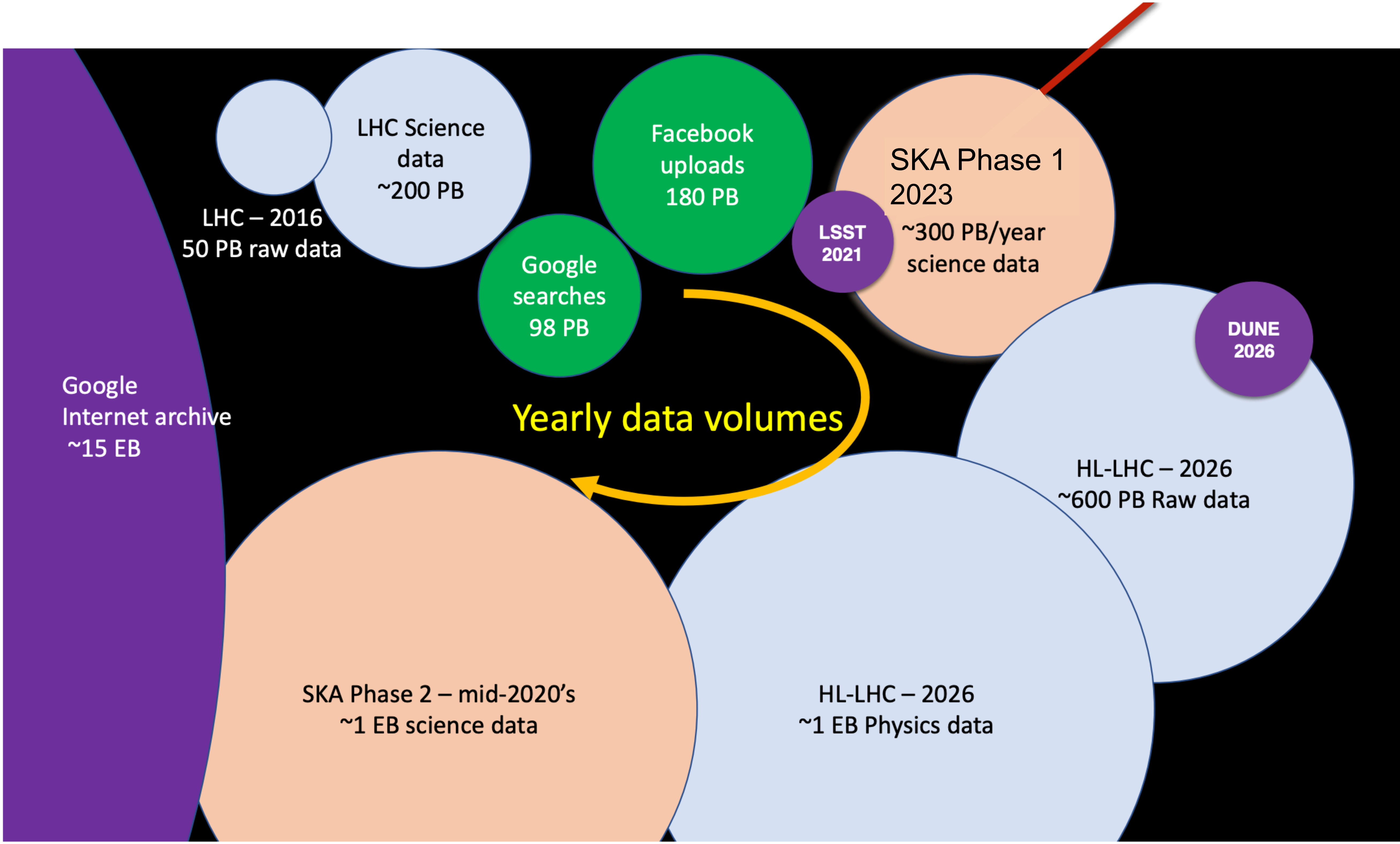


Conclusions

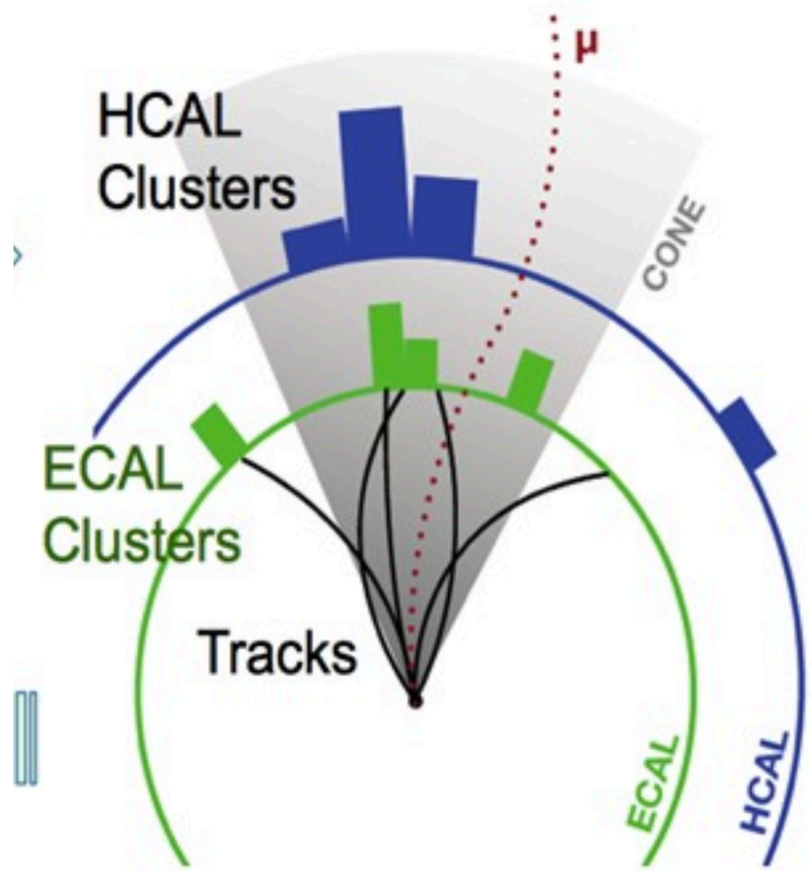
- Large scale campaign underway to adopt deep learning everywhere
- Scale of data processing in physics is getting larger
    - With large datasets come huge scientific potential
- Have demonstrated ML+ Heterogeneous computing works
    - Parallelization of NNs and eff of FPGAs give large speedups
        - Can be used both for very low latency systems
        - GPUs are a viable alternative for longer latency

Thanks!

Everything Getting larger

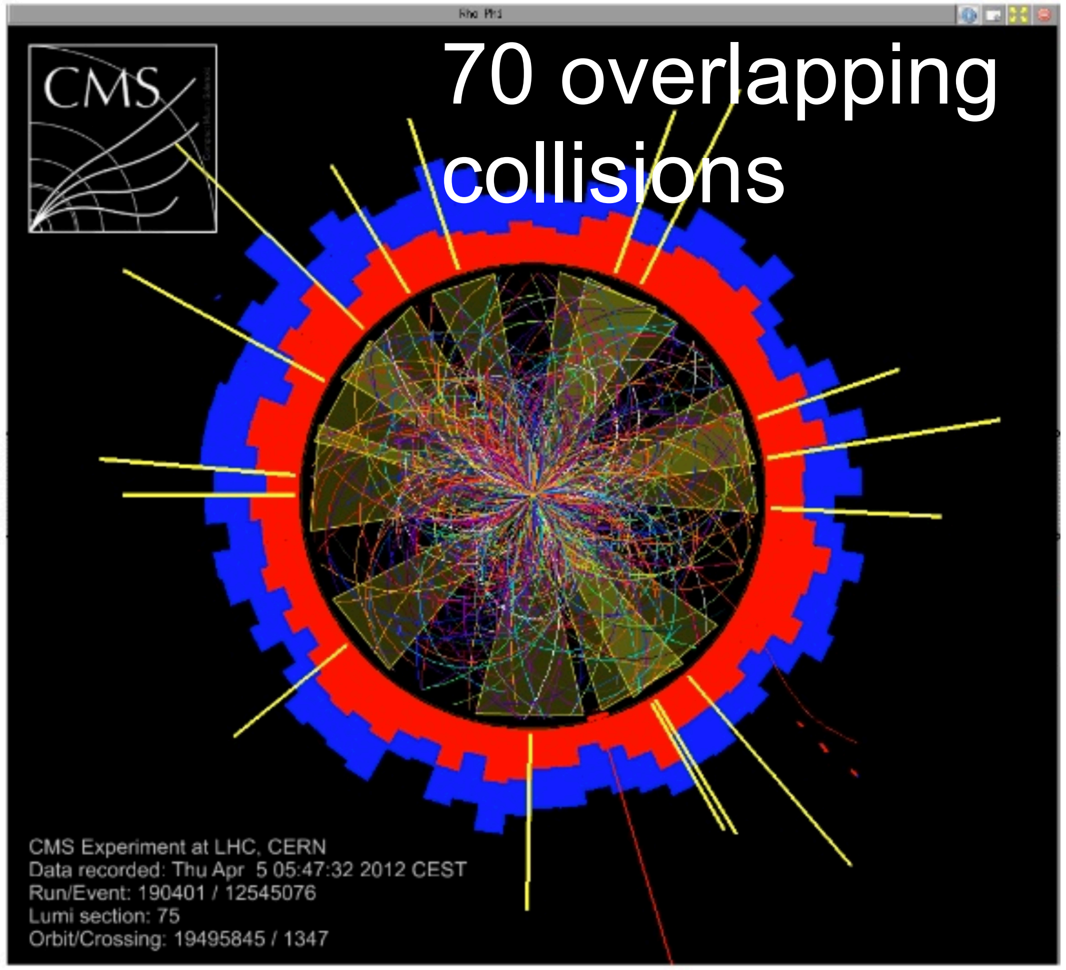


Reconstruction Challenge



LHC reconstruction involves combining many different detectors in to particles

Batch N per particles



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

In the future will be > 200 collisions

Batch 1 per Event

Data Flow in CMS

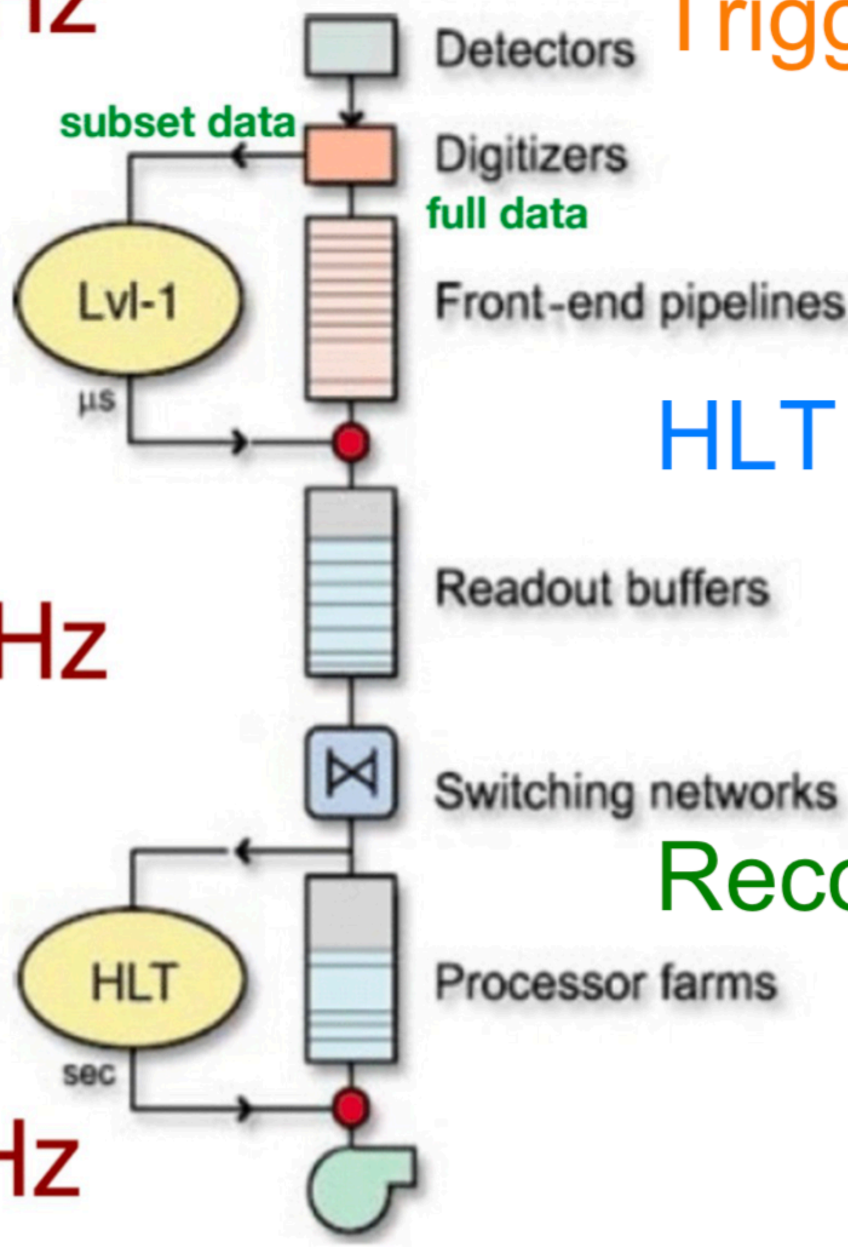
40 MHz

Data reduction

100 kHz

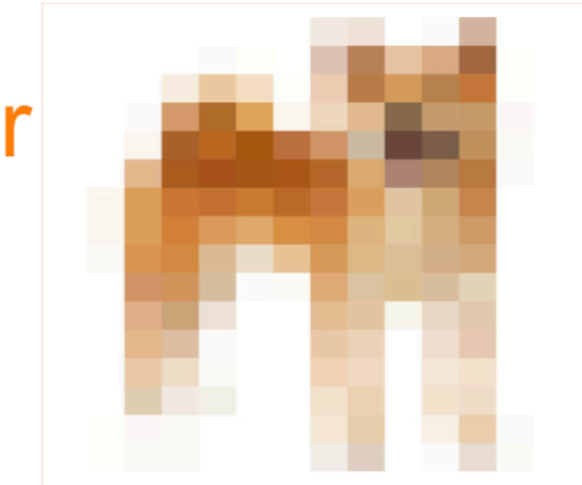
Data reduction

500 Hz



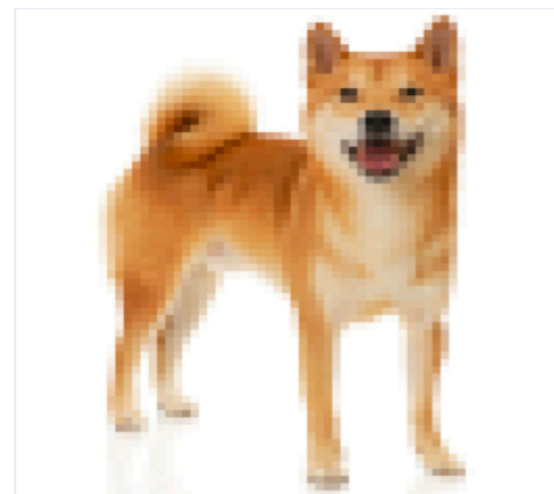
"Offline Computing"
Grid, O(10) Pb

L1 Trigger



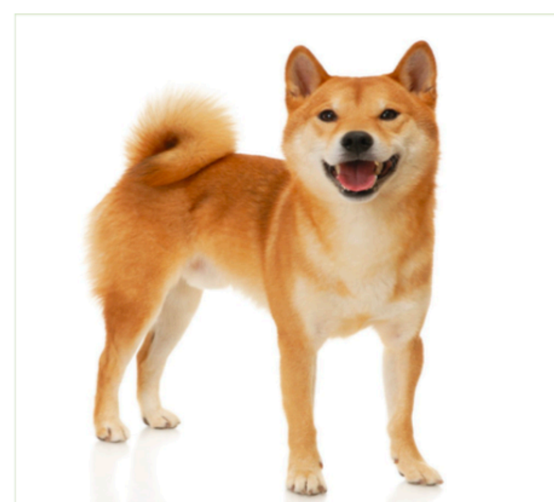
High speed
Low granularity
readout (10µs)

HLT



Intermediate
speed (100ms)
better readout

Reco

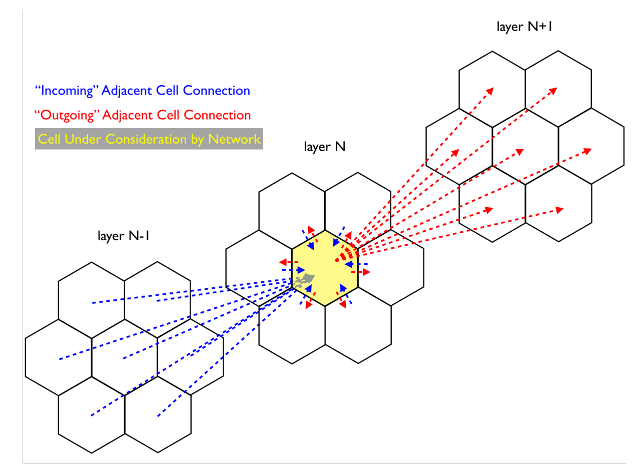


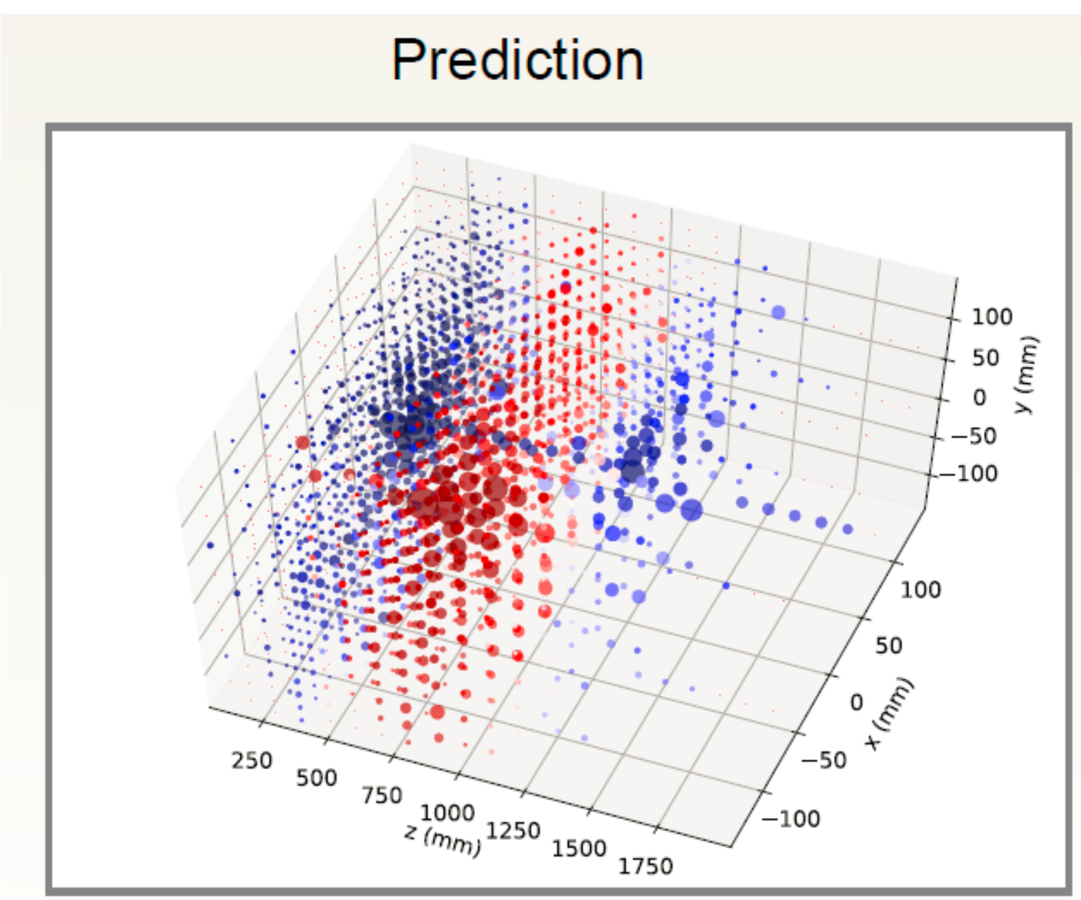
Full data readout
(10s)

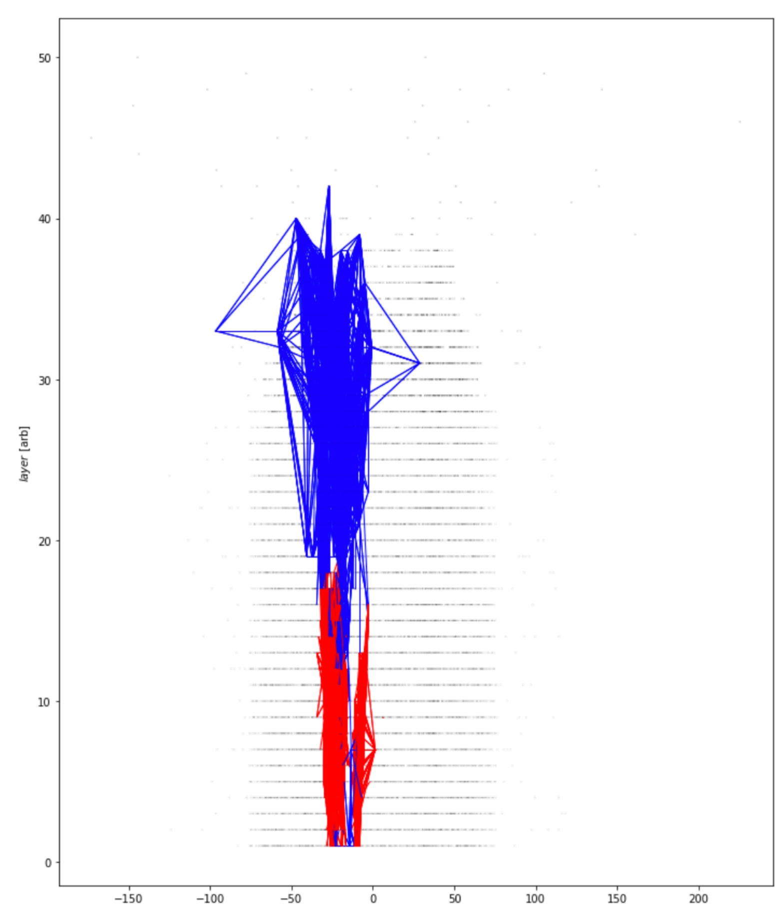
Despite the large rate reduction we still store many Petabytes of data

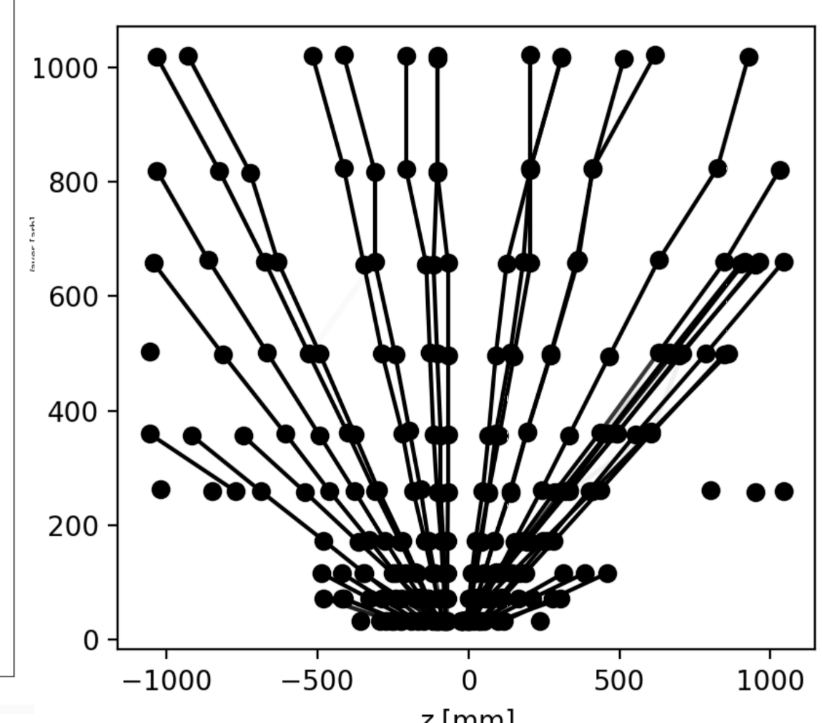
Current trends in HEP

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



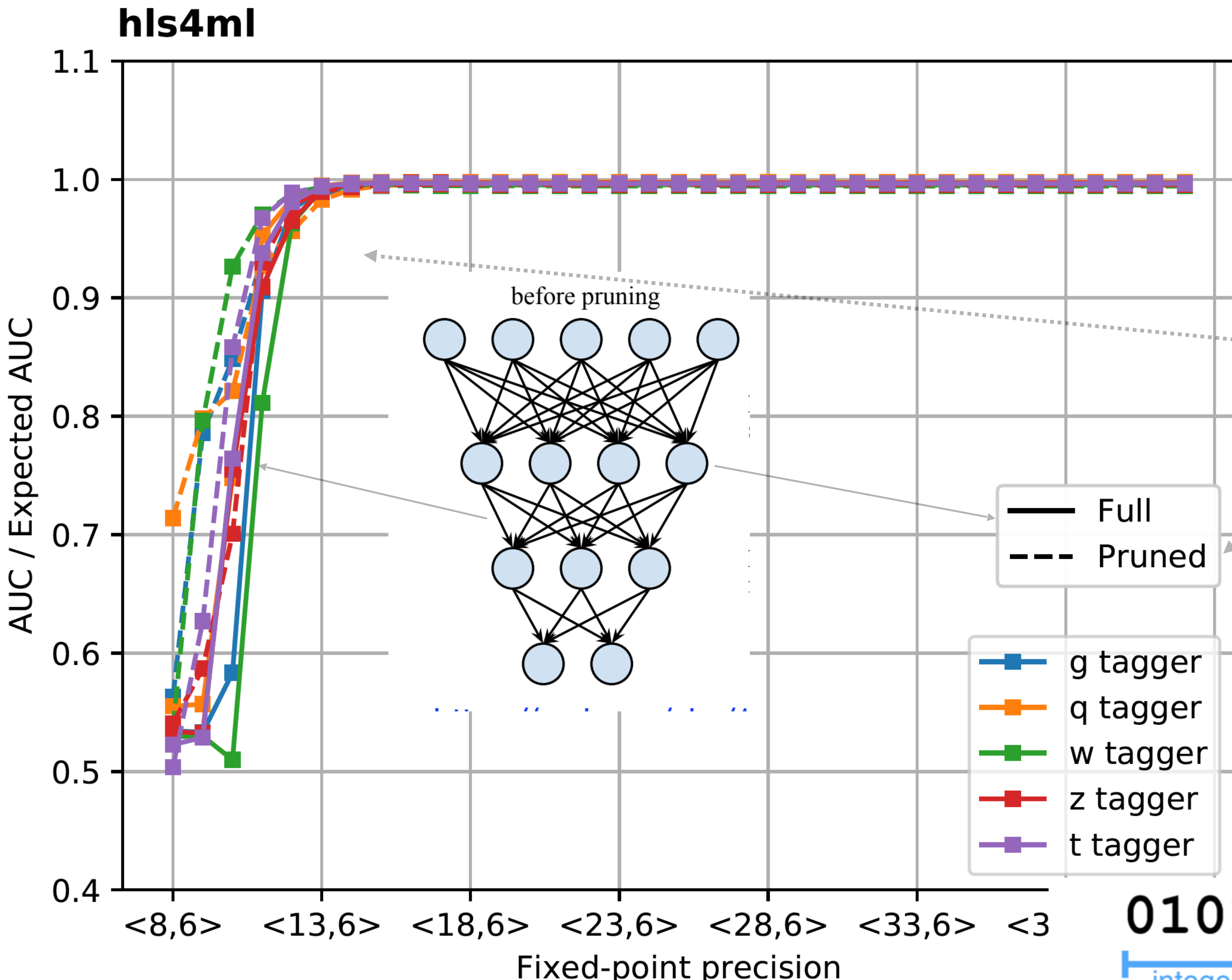


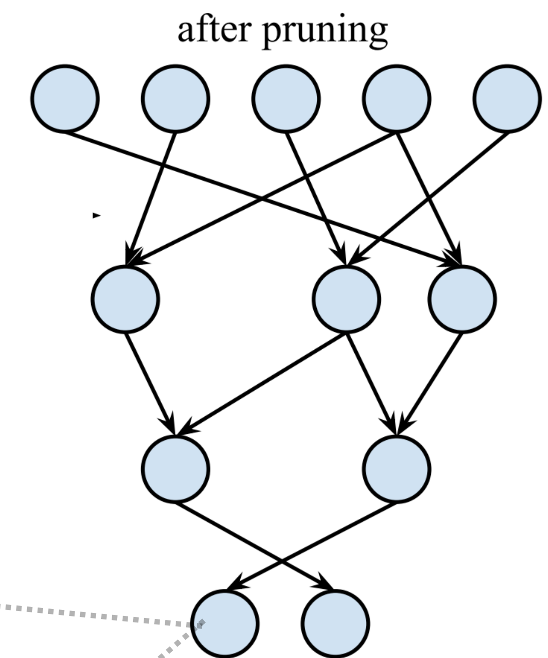


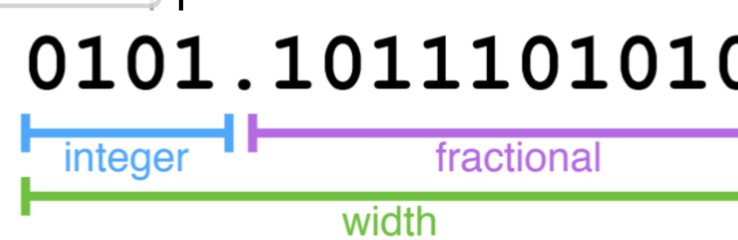


Quantization

- FPGAs support arbitrary precision

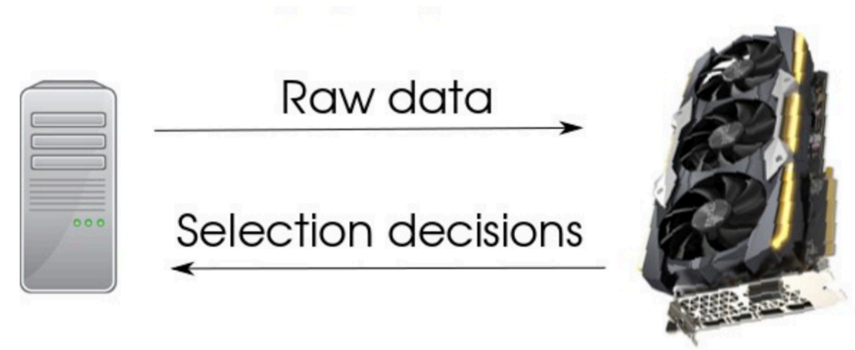


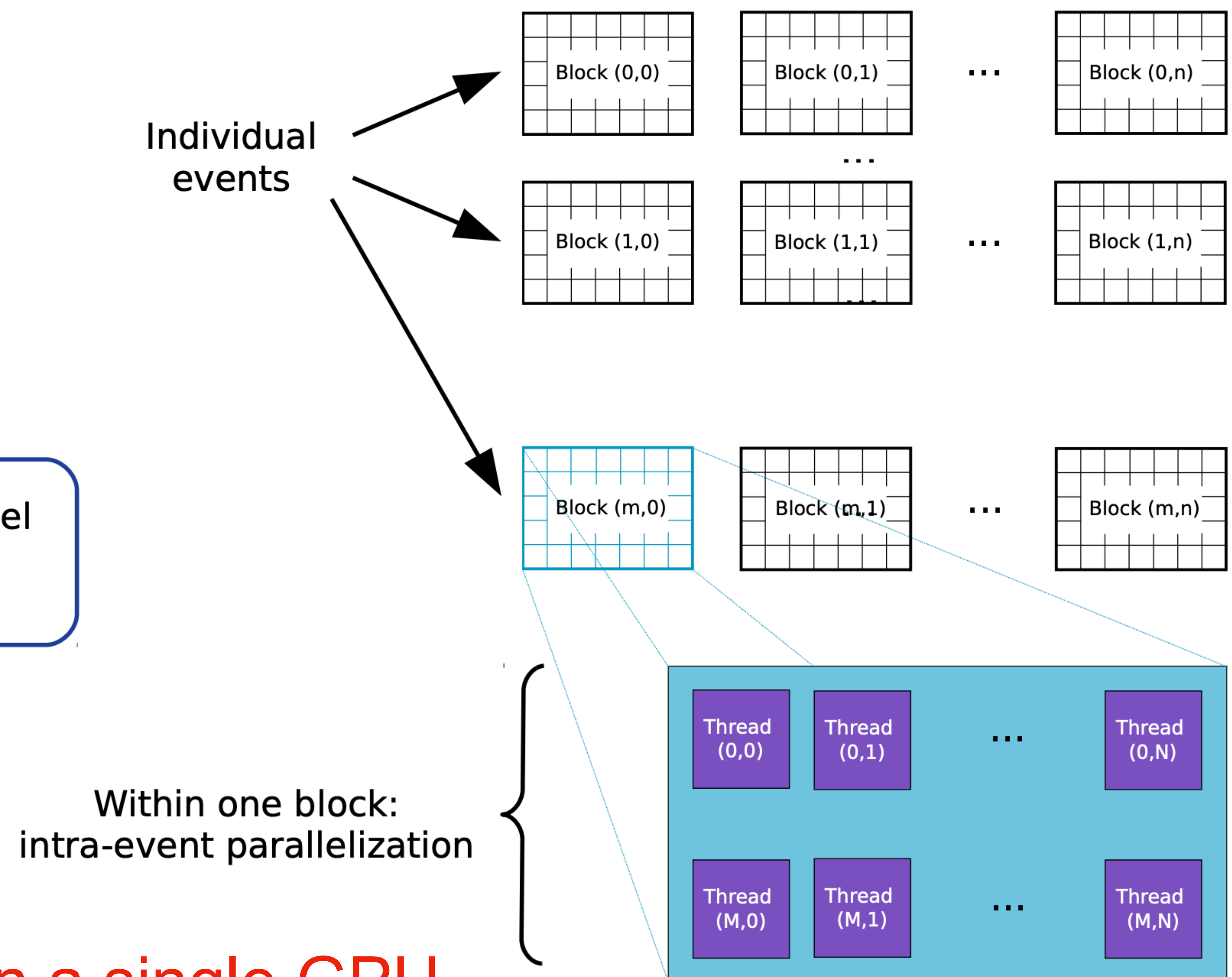




<Total bit width, integer bits above decimal>

Alternative GPU model





- Process thousands of events in parallel
- Single precision only

Batch over many events on a single GPU
Long latency
Small event size(compared to CMS/ATLAS)

Thanks!

INTERNET2

NSF

DEPARTMENT OF ENERGY
UNITED STATES OF AMERICA

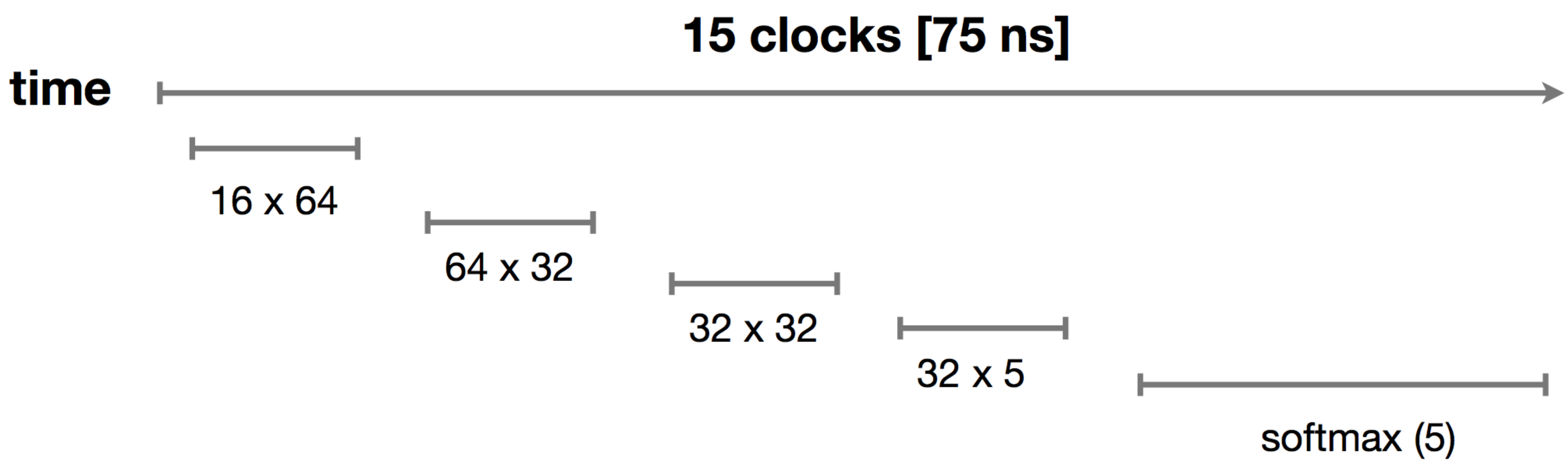
XILINX
ALL PROGRAMMABLE™

Microsoft

MIT
Quest for
Intelligence

Overall Performance

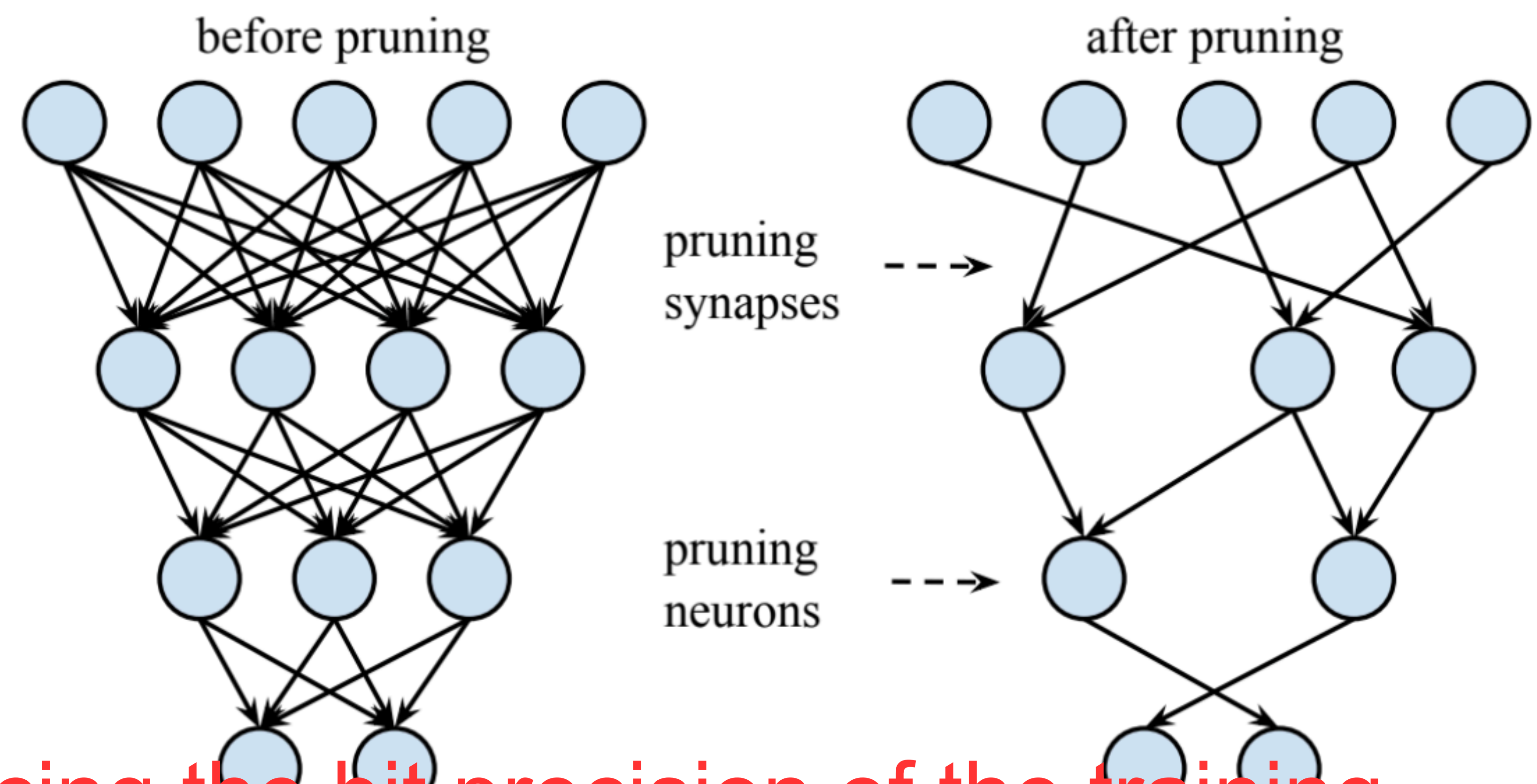
- Set the Reuse to 1 (Max speed) Precision to 18,8
  - Benchmarked this on a Virtex 7 (xc7vx690tffg1927-2)



Reuse = 1	BRAM	DSP	FF	LUT
Total	13	1116	47k	35k
% Usage	~0%	20%	3%	5%

Algorithm Compression

- Compression is a critical aspect to reduce ML
  - Allows us to put much larger networks on an FPGA
- Two key elements which *only* FPGAs can do
  - Pruning of the weights (removes multiplications)



  - Reducing the bit precision of the training

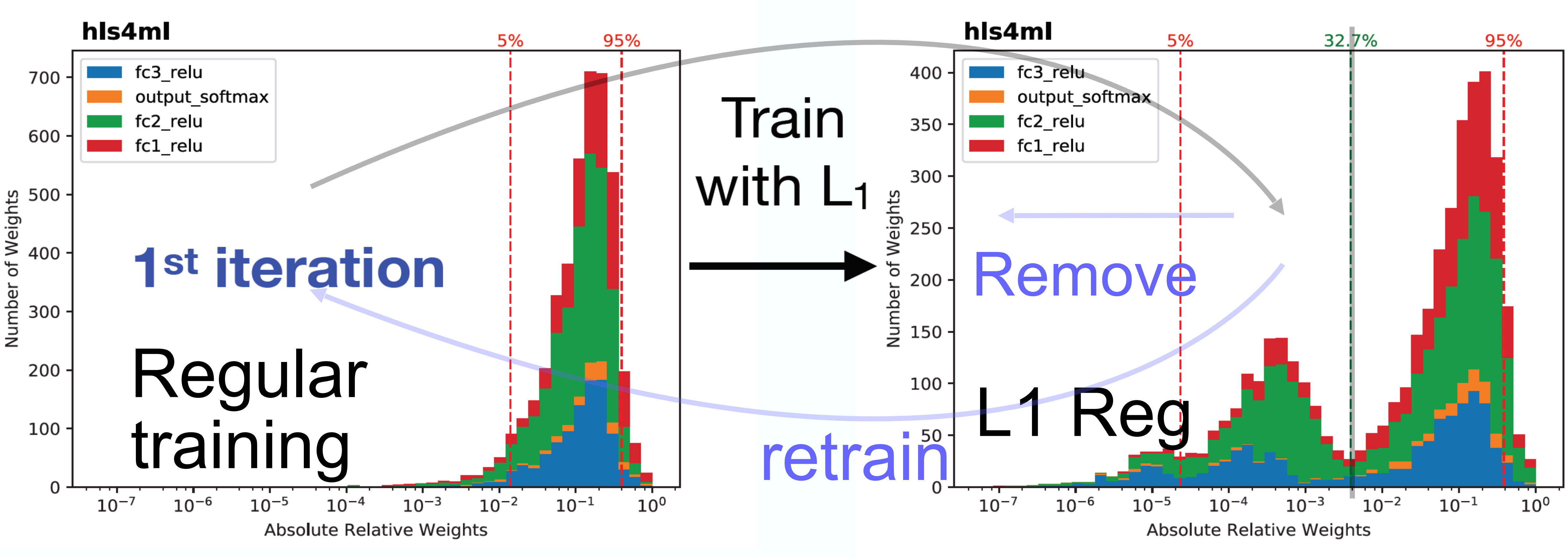
Network Compression

Usual cross entropy

$$L_i = \boxed{-\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)} + \boxed{\lambda_1 \,|\, w \,|}$$

L1 Regularization

- Add a regularization term to the loss function
  - Helps to force weights to smaller values



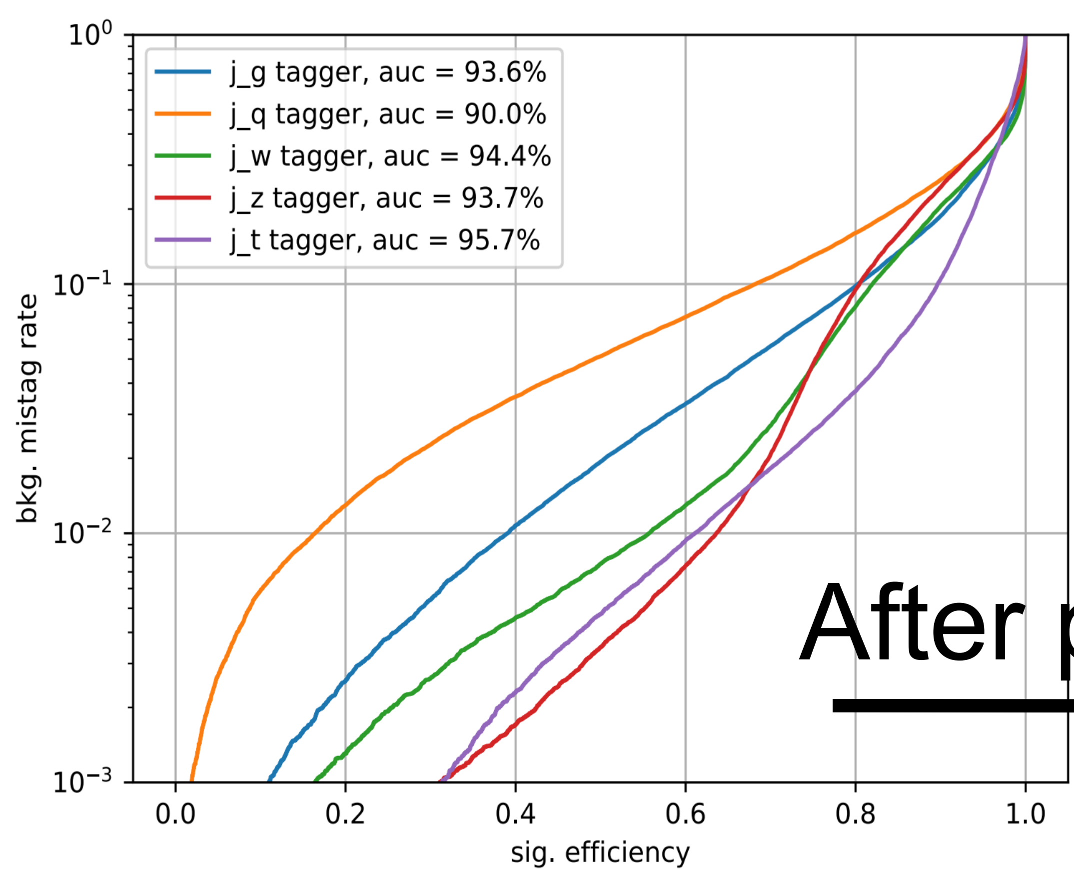
Weight Pruning

Usual cross entropy

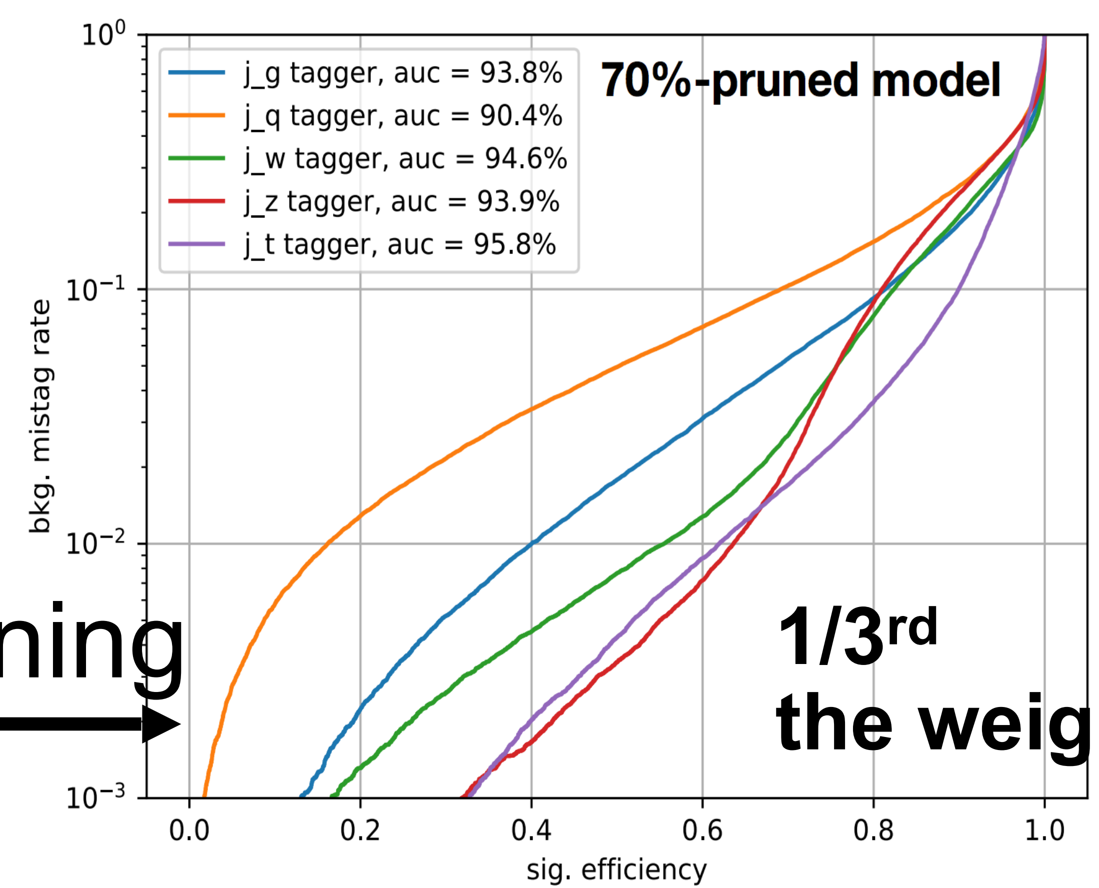
$$L_i = \boxed{-\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)} + \boxed{\lambda_1 \mid w \mid}$$

L1 Regularization

- Add a regularization term to the loss function
  - Helps to force weights to smaller values



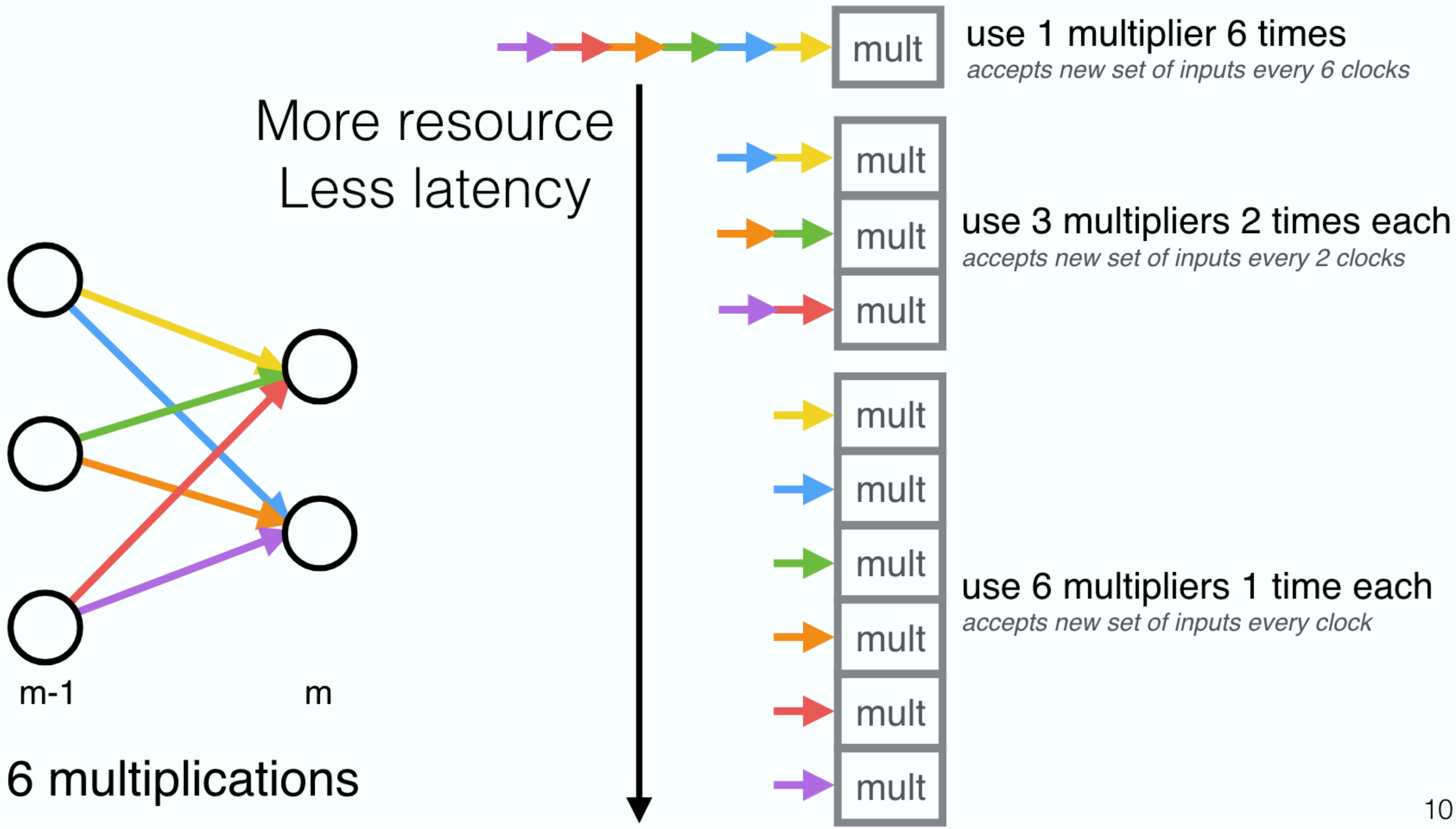
After pruning



1/3rd the weights

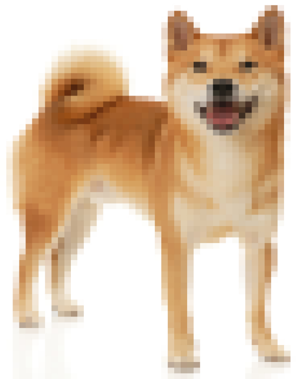
Reuse Factor

- By gauging pipeline we can adjust resource usage

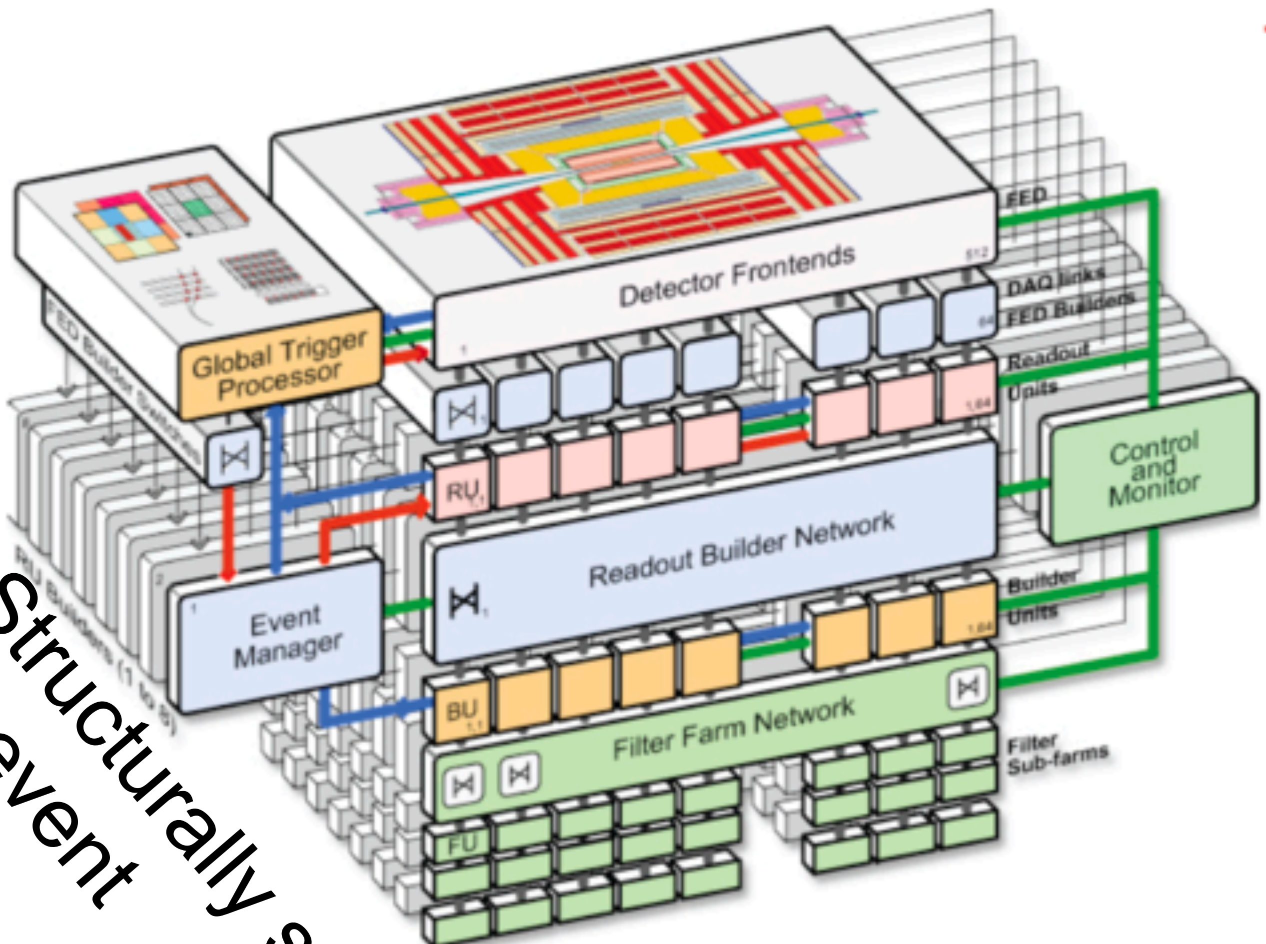


Takeaways

- 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

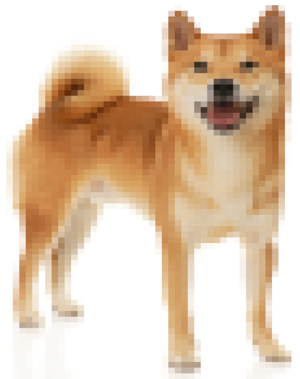


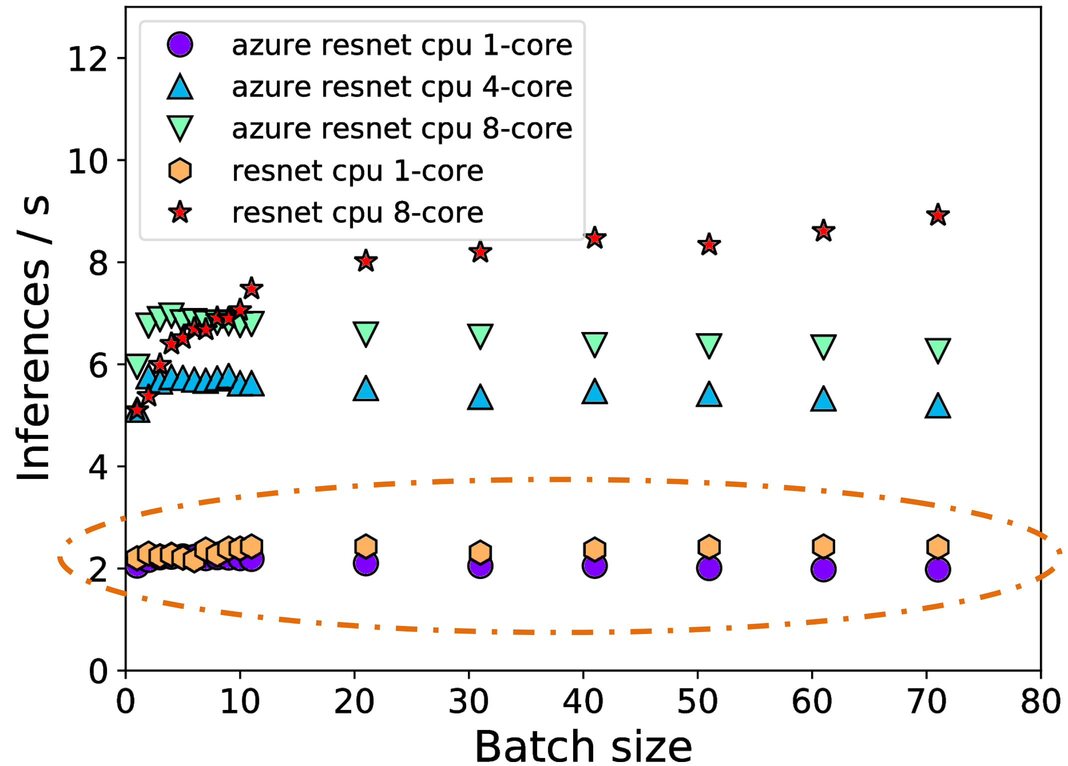
Structurally split by event

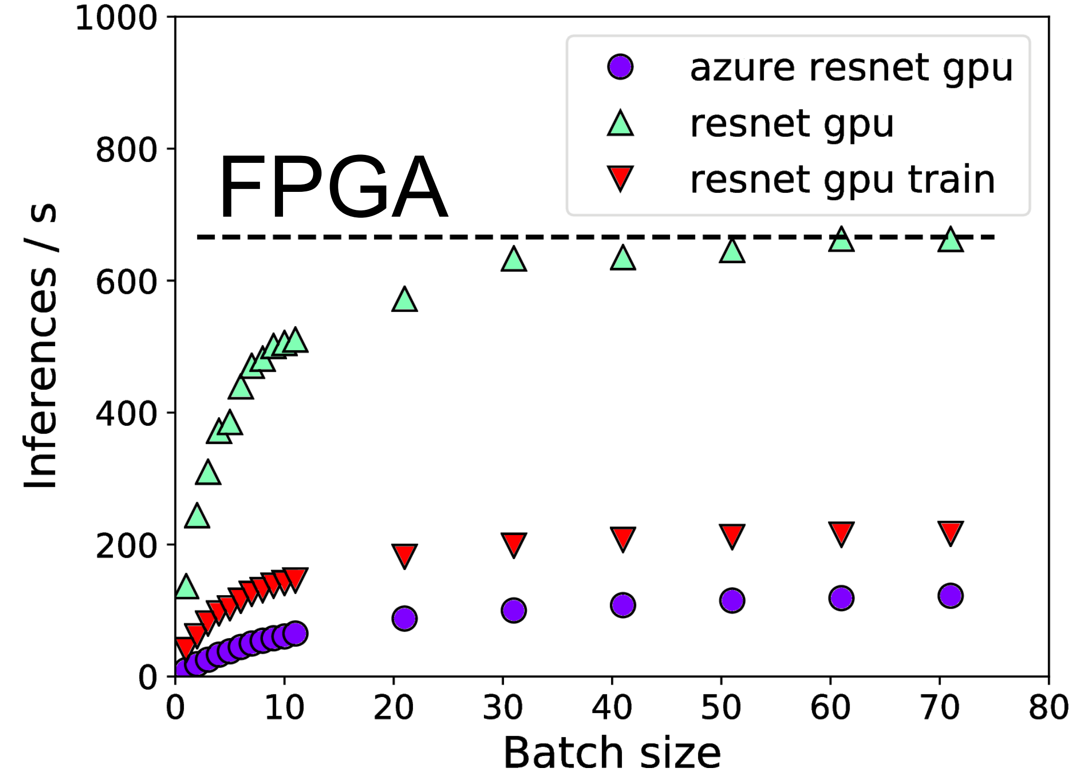
- 100 kHz of collisions in
- 1kHz of collisions out
- <500ms to analyze collision
- Currently
  - A local computing cluster
  - System is all CPUs

- Experiments are considering GPU/CPU system for 2022

How Fast is It?

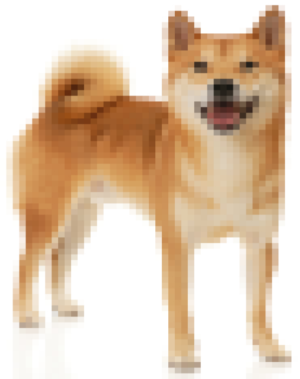






Algo	Per Event
CPU	1.75s
GPU Batch 1	7ms
GPU Batch 32	2ms
FPGA	1.7ms

With an FPGA can get 1.7ms inference time at batch 1
With a GPU can get 2ms/img time at batch 70



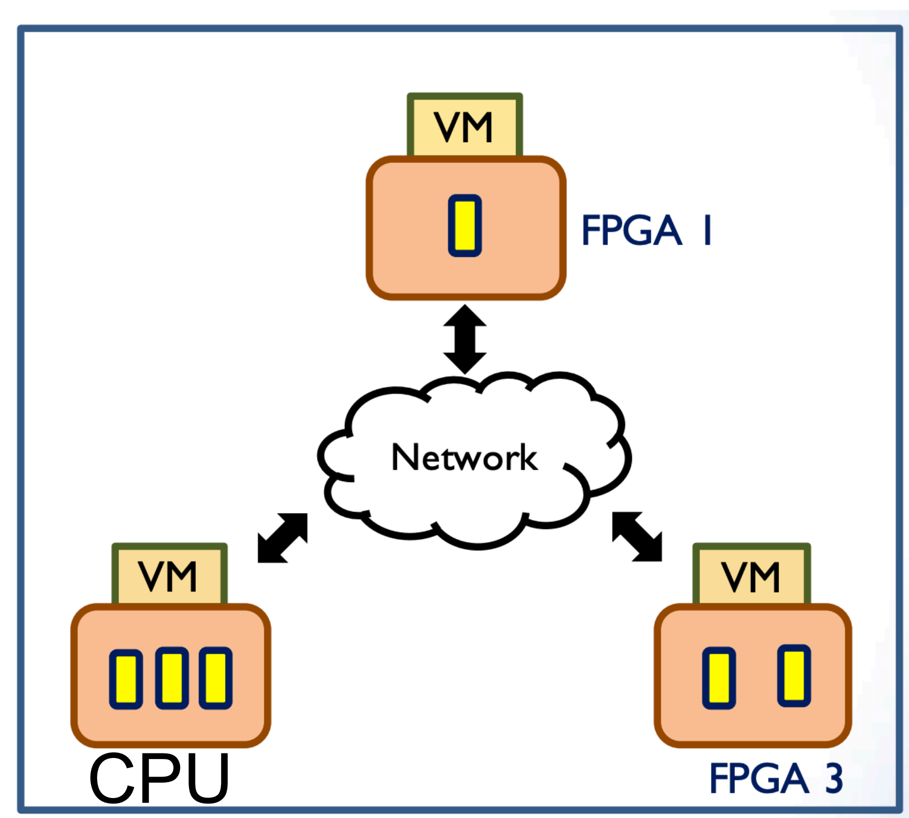
Benchmark #1

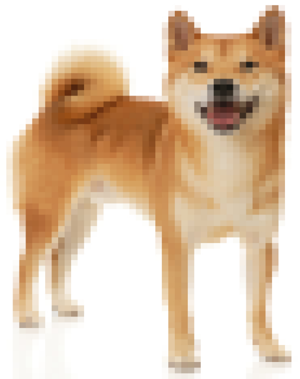
- GPU as a service
  - Using tensor-rt-server
    - Industry standard
  - Latency : 16ms

- 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

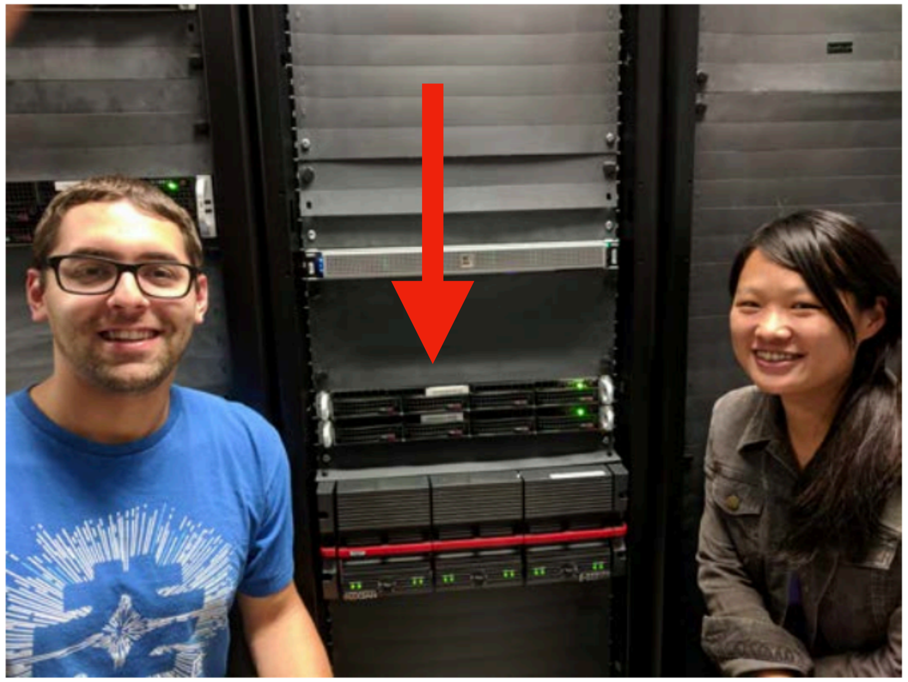
Batch 1 latency: 15ms

Microsoft Databox Edge

From local CPU
to FPGA system at FNAL

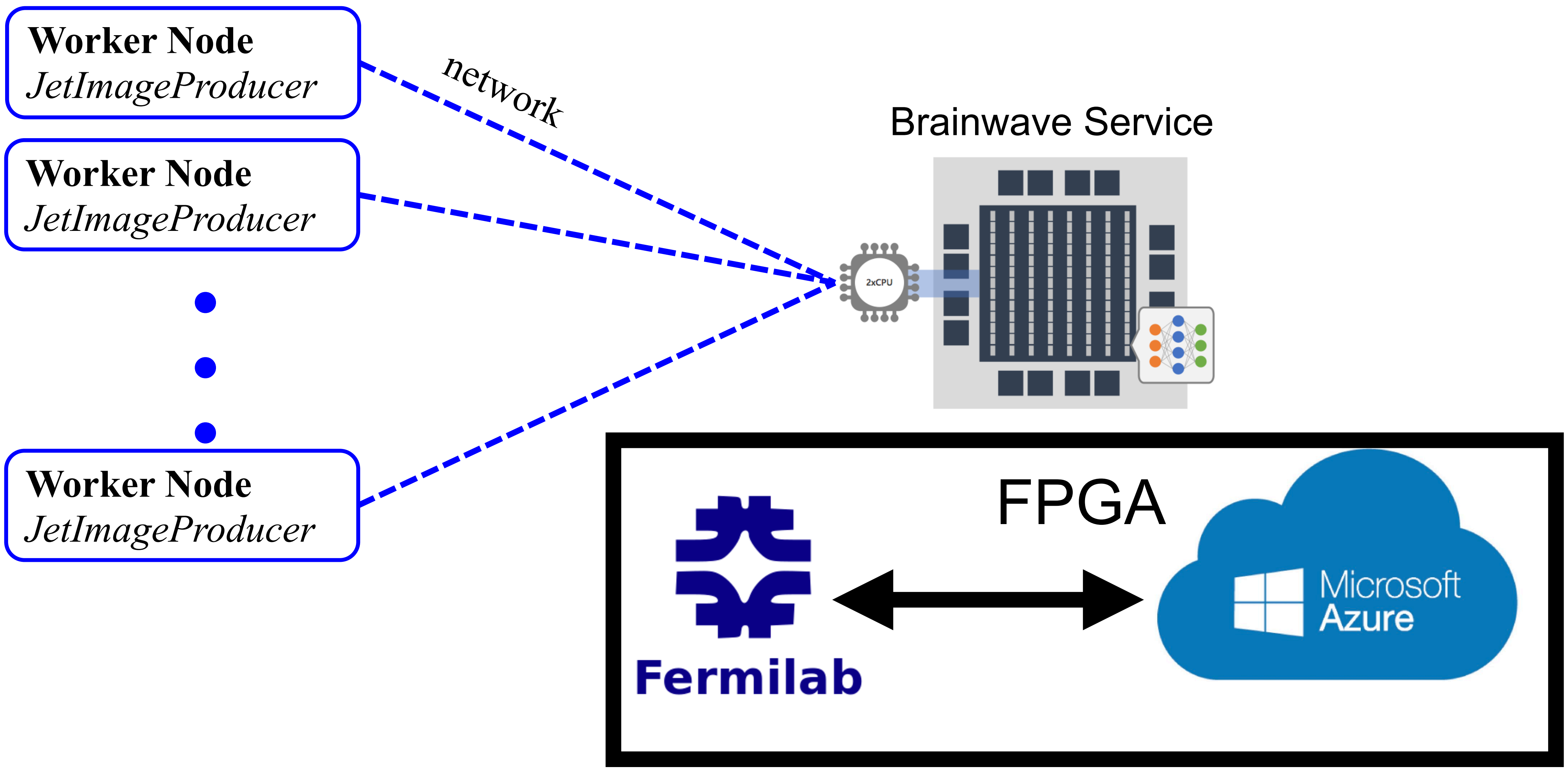
Batch 1 latency: 20ms

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



Throughput

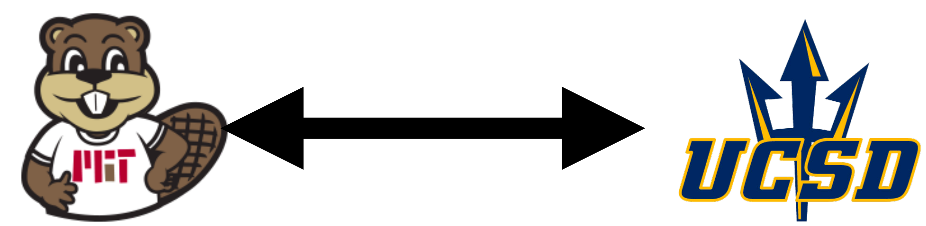
- Despite the longer latency we can have one node serve many



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

Benchmark #1

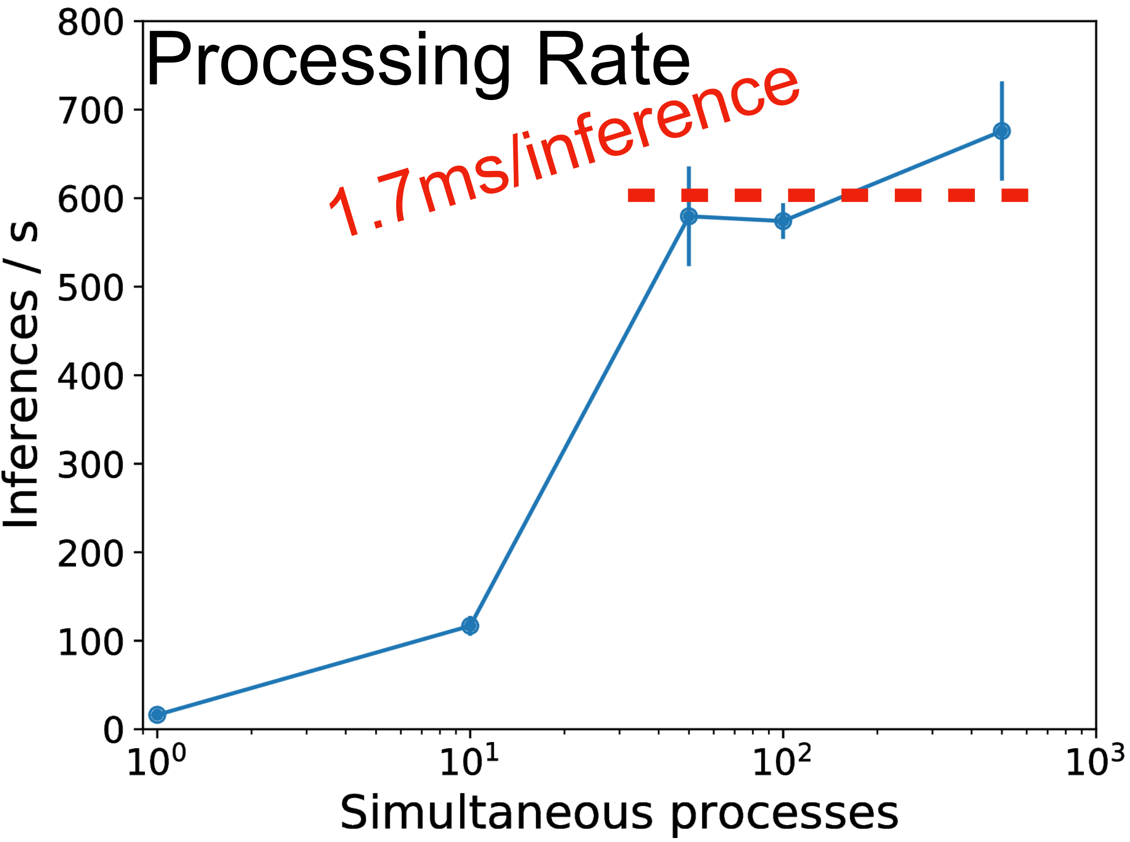
- Throughput is driven by the actual minimum latency of algo
    - For FPGA algo latency is 0.08ms→working to get there
- Cloud have to deal with additional slow down from networking

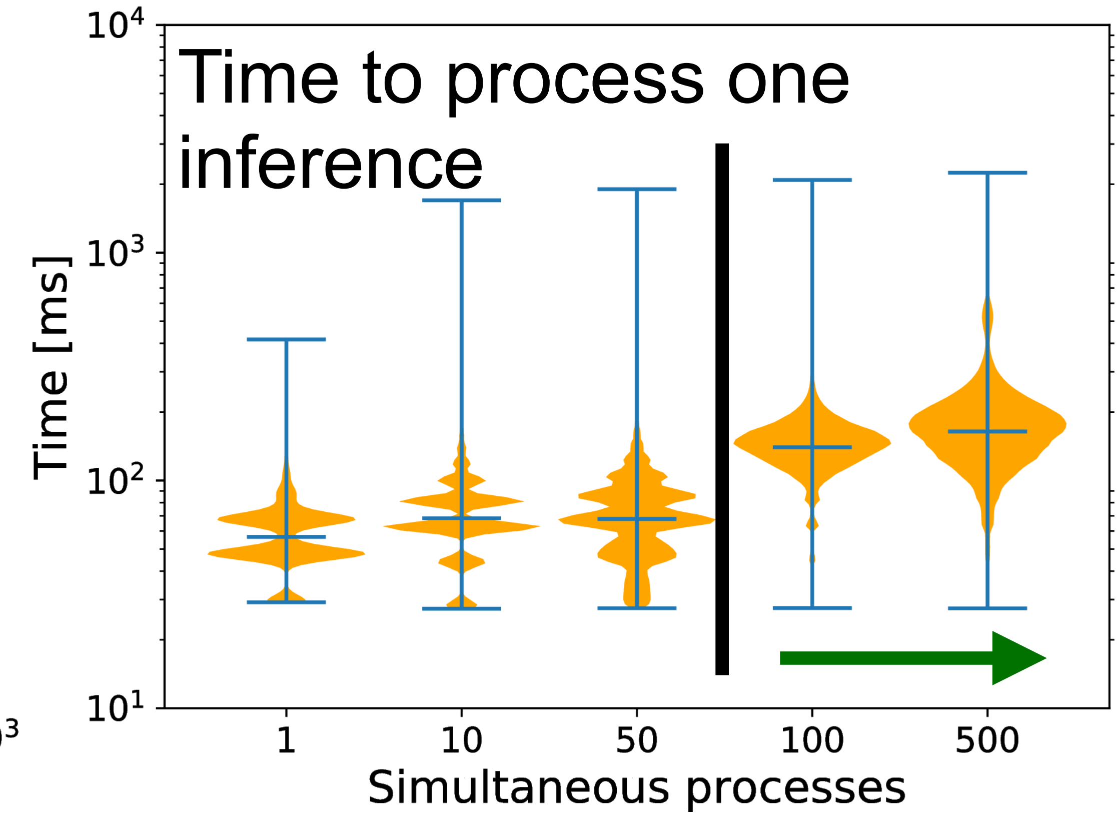


Algo	Per Event	+On-site aaS	+Cloud aaS	Ping	On/Cloud put
Old	50ms	N/A	N/A	N/A	N/A
NN CPU	15ms	N/A	N/A	N/A	N/A
NN GPU(1080 Ti)	3ms (prelim)	16ms	90ms	75ms	1ms/30ms*
NN FPGA	2ms	TBD(<16ms)	TBD	TBD	>0.1ms

*Cloud throughput on GPU still to be scrutinized

Benchmark #2

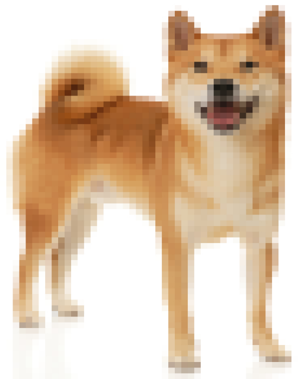




Can Serve 50-100 nodes with 1 FPGA and no loss

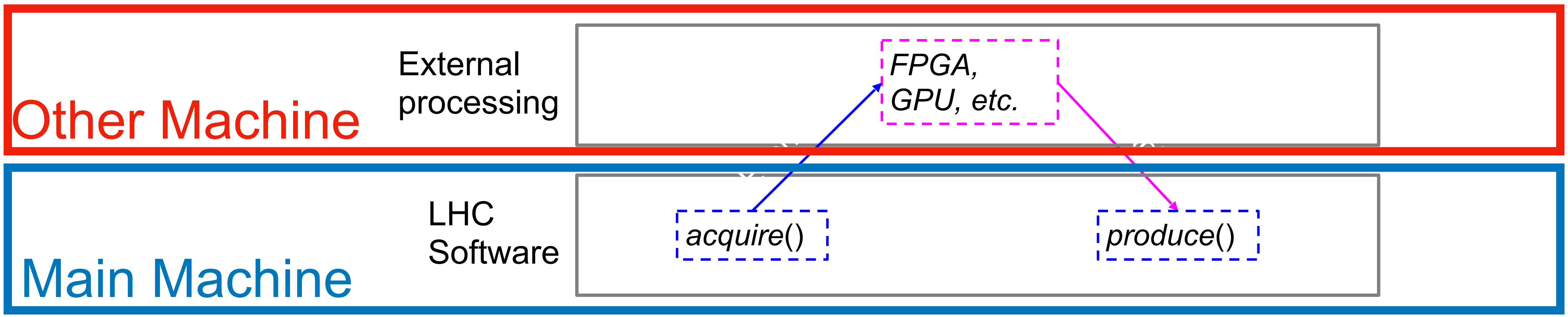
Algo	Per Event	+On-site aaS	+Cloud aaS	Ping	On/Cloud* put
CPU	1.75s	N/A	N/A	N/A	N/A
GPU Batch 1	7ms	23ms	97ms	75ms	5ms/20ms*
GPU Batch 32	3ms	240ms	975ms	75ms	8ms/20ms*
FPGA	1.7ms	15ms	60ms	25ms	1.7 ms

*Cloud throughput on GPU still to be scrutinized

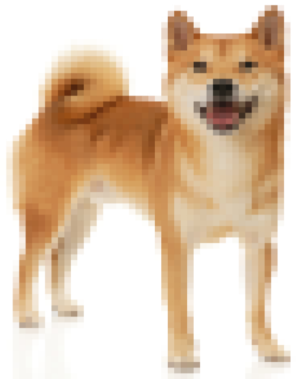


Idea #2: Services

- To run these algorithms within our software

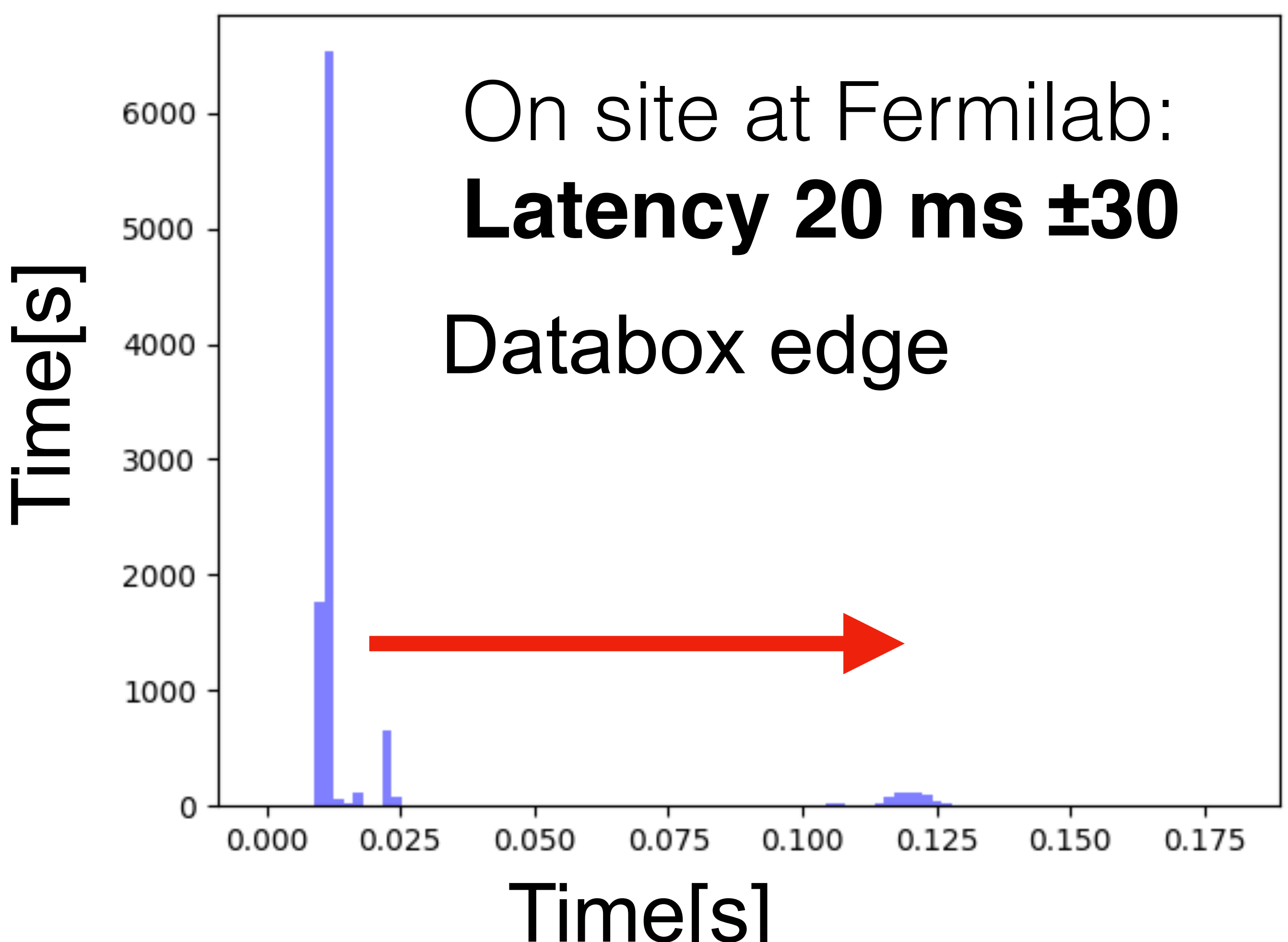


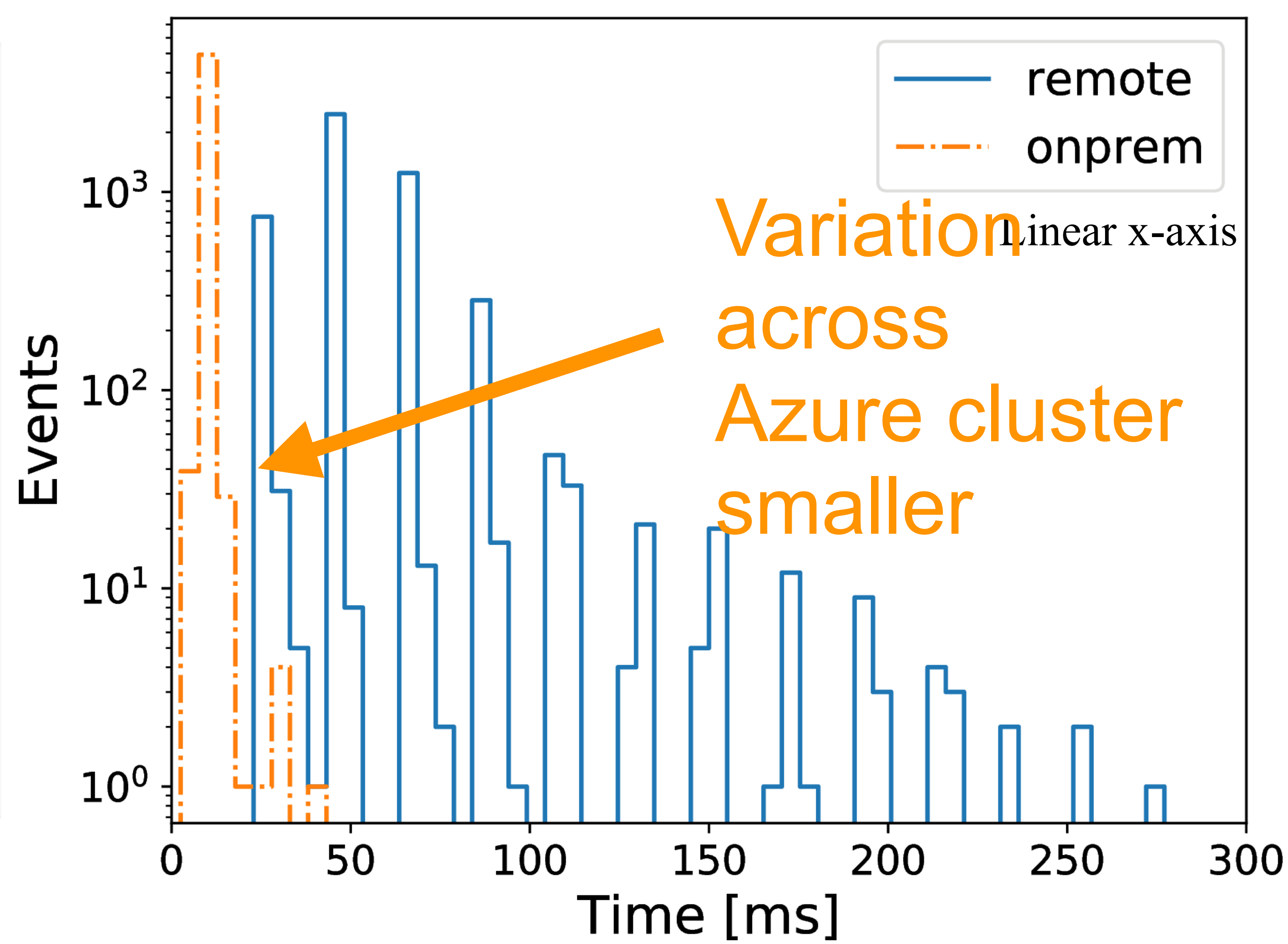
- SONIC : Services for Optimized Network Inference on Coprocessors
- Strategy
  - Use the same benchmarks as before
  - Now wrap these with gRPC protocol between different machines

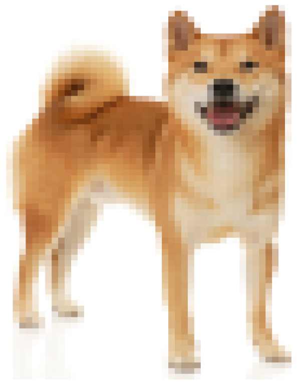


Services Takeaway

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





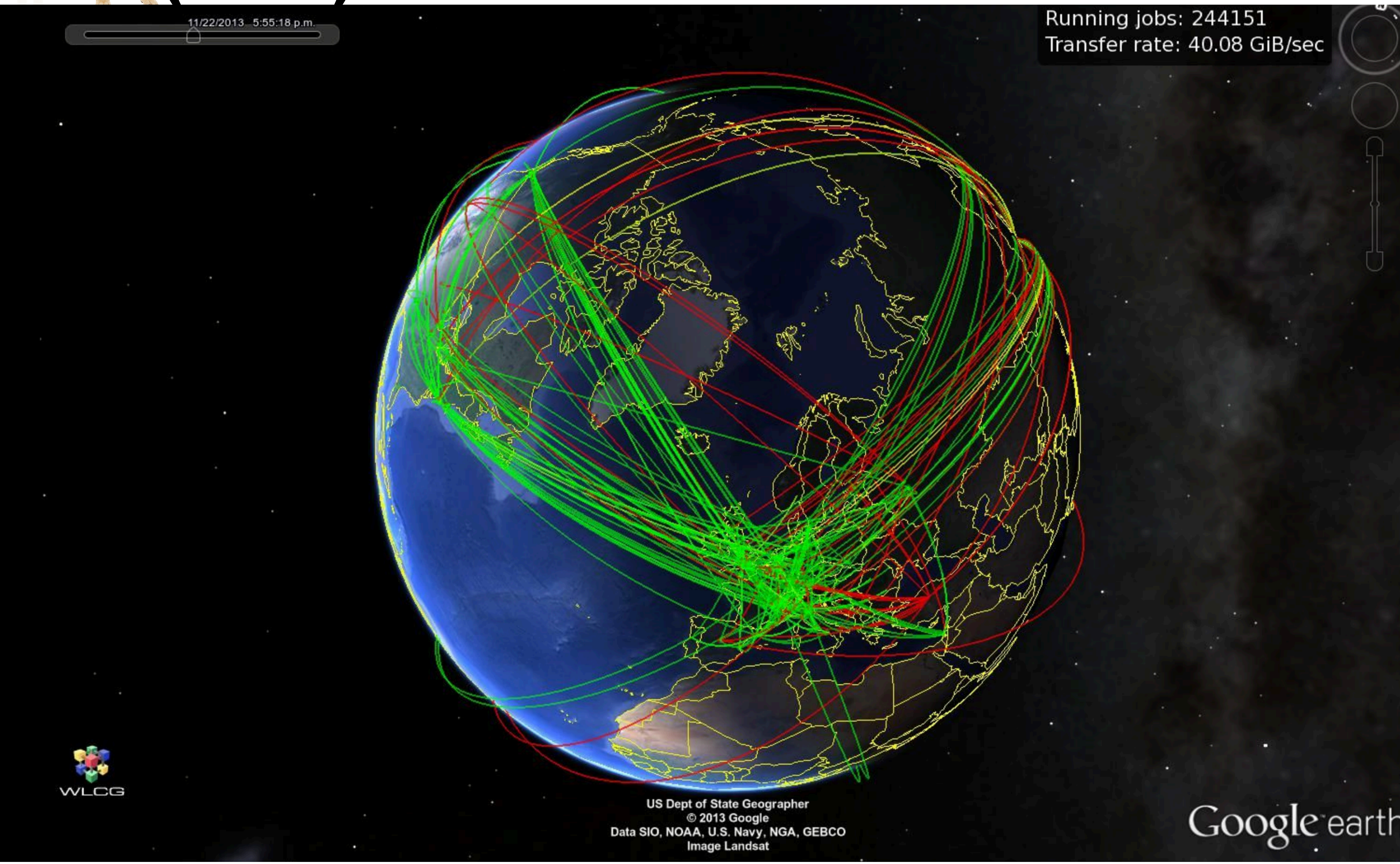


Throughput vs Latency

- 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)

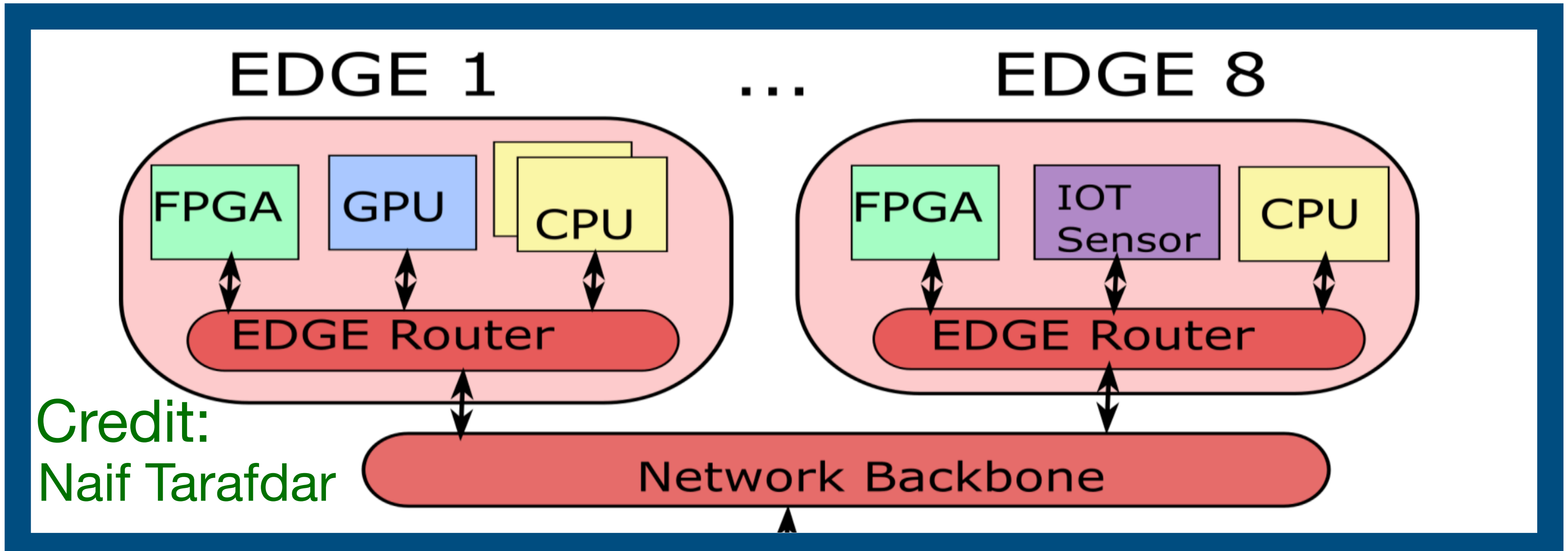
LHC Computing Grid



1 kHz (10s)

Offline Reco

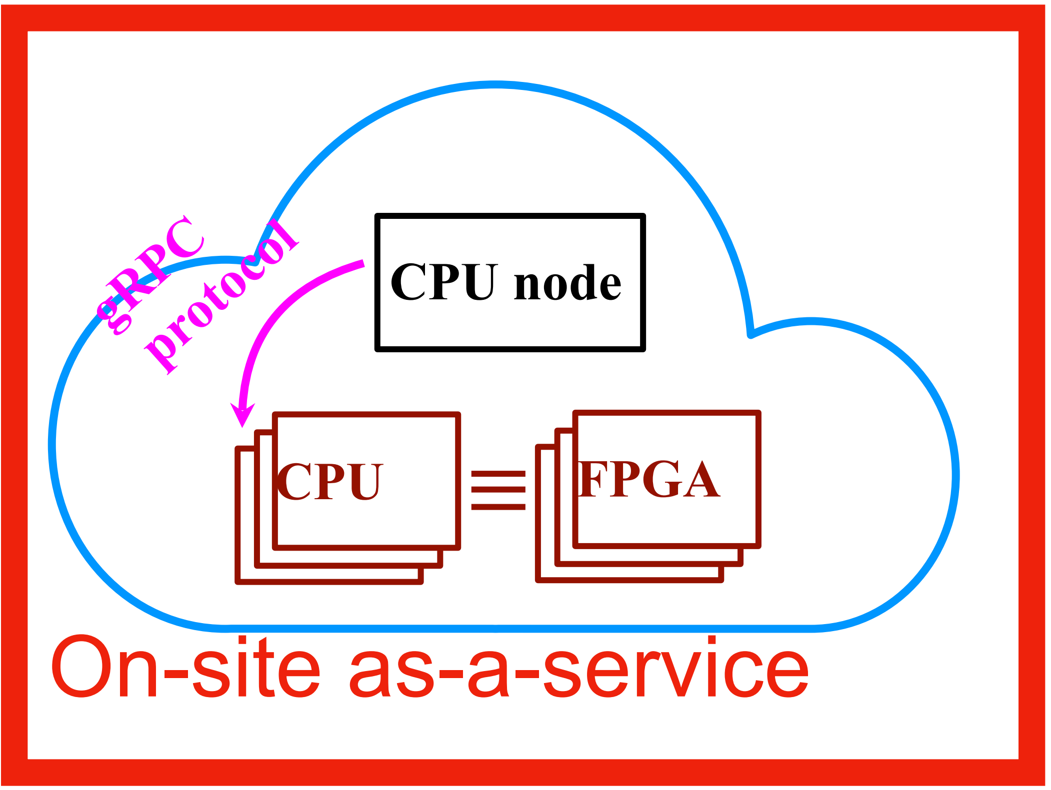
- At the final tier of reconstruction
  - Worldwide grid is roughly 0.75 Million cores 600 PB of data
  - Latency is not a critical limitation
    - Grid will have different technology all over (common protocol?)

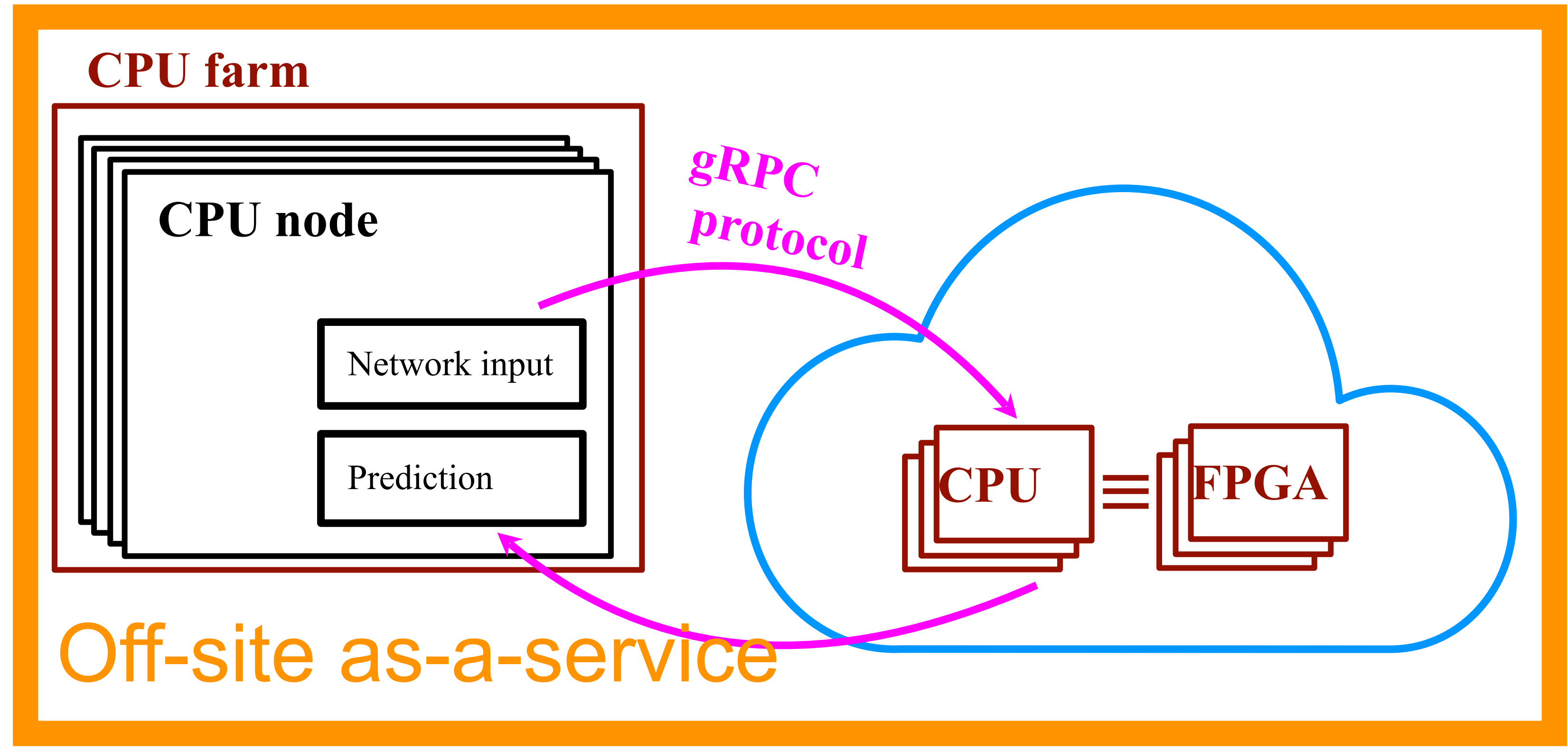


Service Options

Low latency Triggering (previous slides)

Larger latency but still large throughput (future slides)



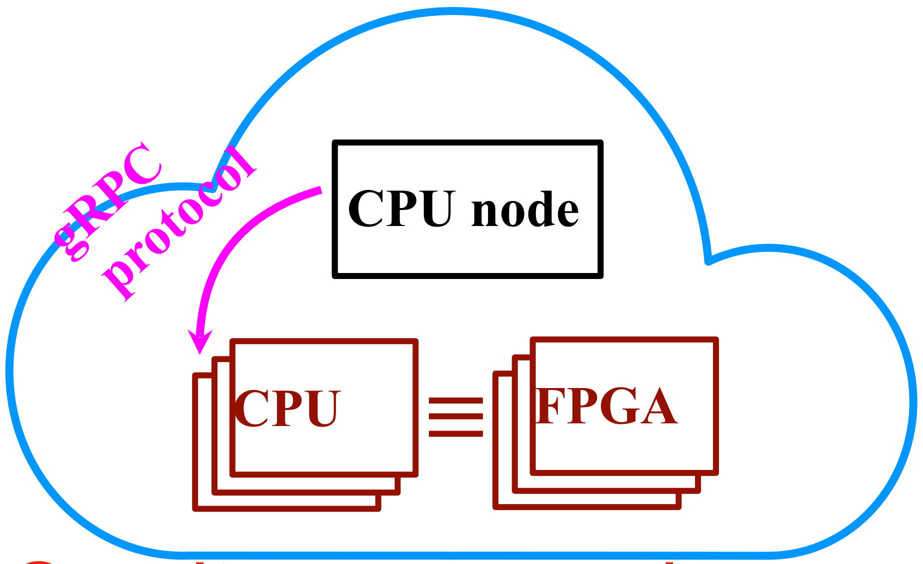


When latency not critical element : can go off-site to the cloud

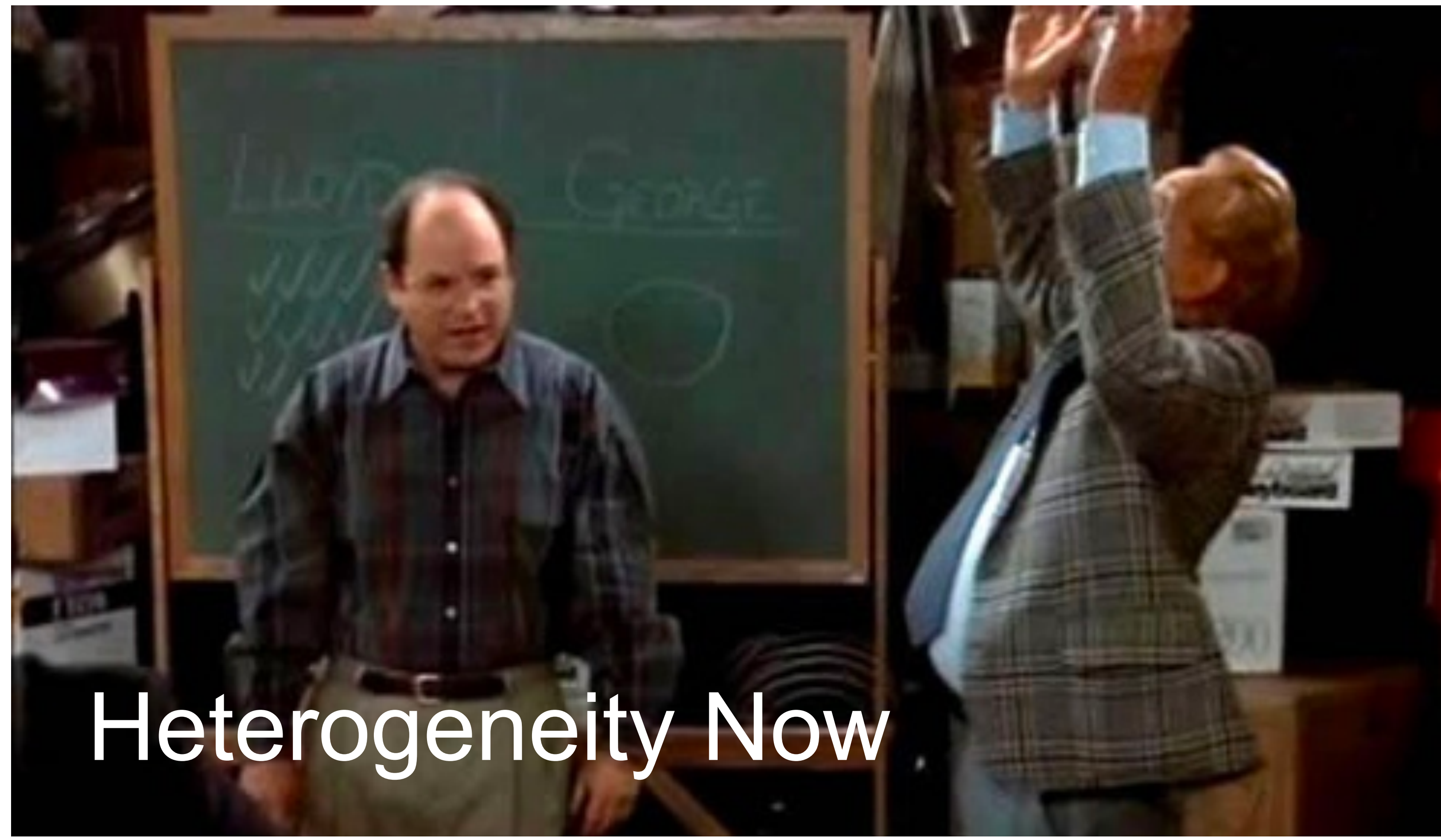
At the offline tier can switch to the cloud no→**Heterogeneity now**

Service Options

Low latency Triggering
(previous slides)

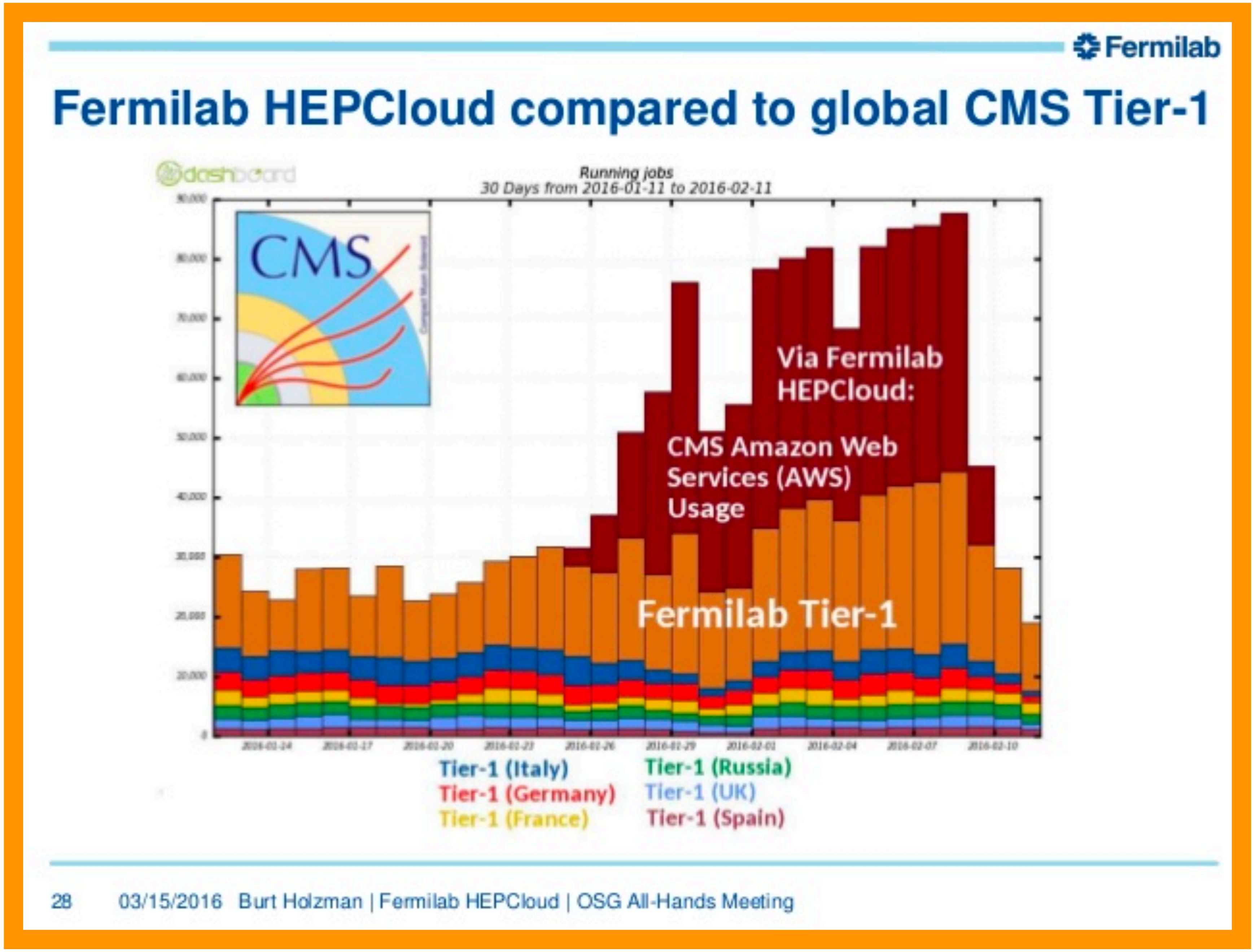


On-site as-a-service



When latency not critical element : can go off-site to the cloud
At the offline tier can switch to the cloud no→**Heterogeneity now**

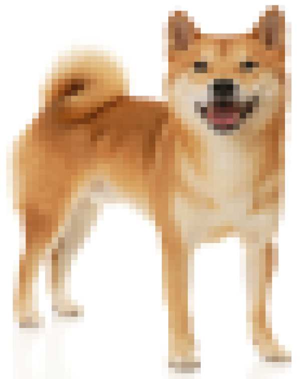
Service in Cloud



We have already done this with CPUs in the cloud

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



Idea #1

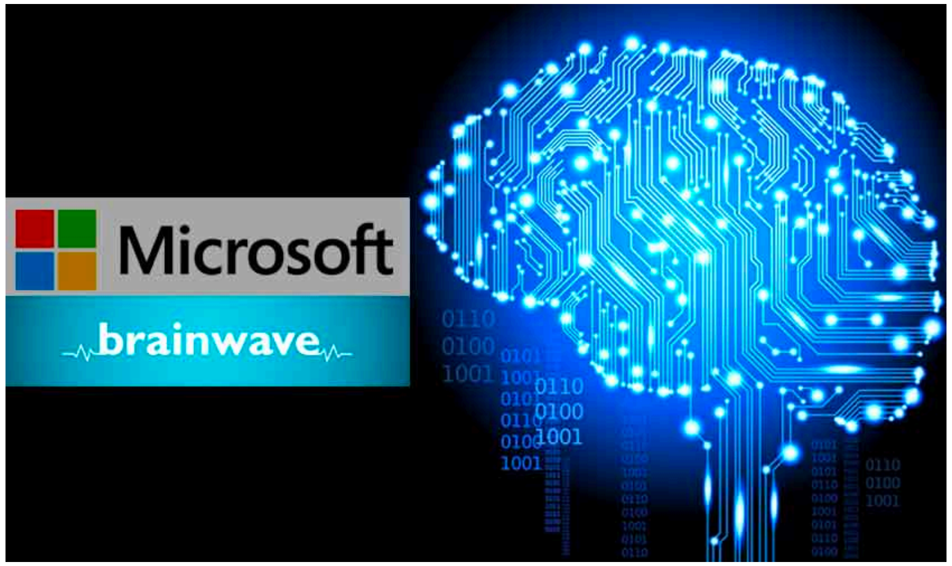
Benchmark #1

XILINX
ALL PROGRAMMABLE™

hls4ml

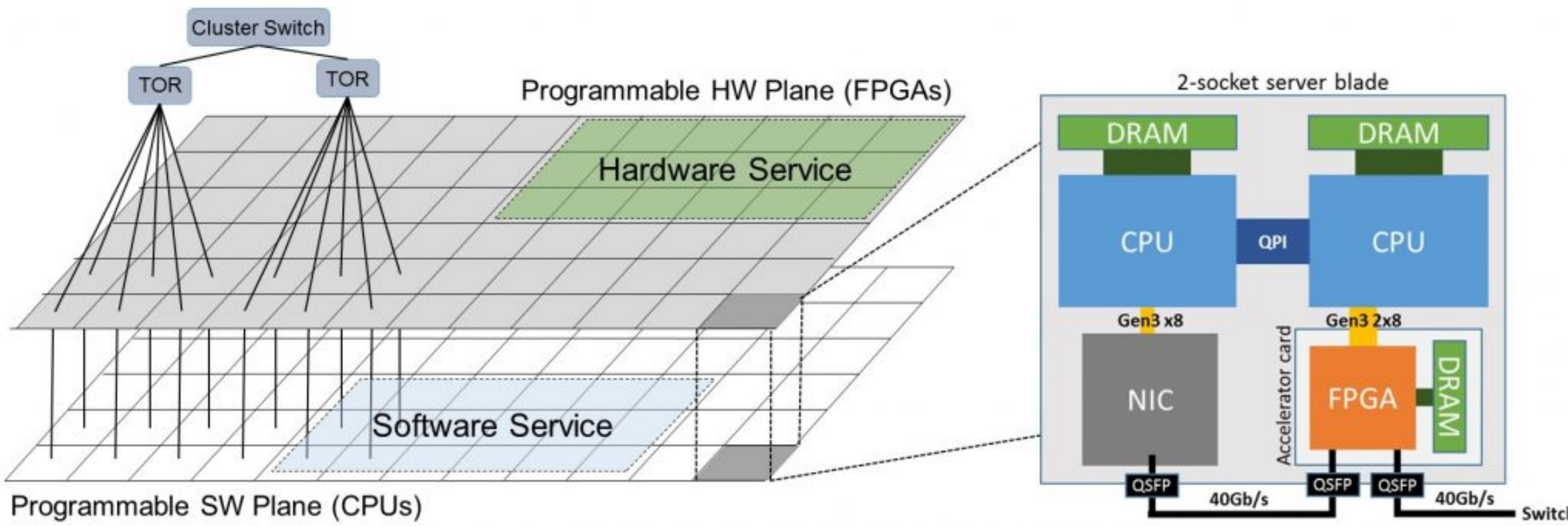
Benchmark #2

intel
STRATIX® 10
inside™



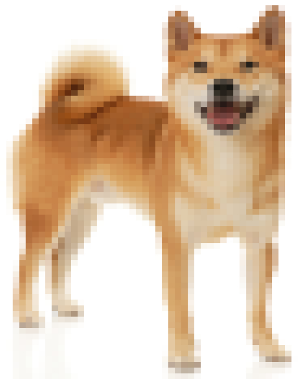
*Also investigating Xilinx ML suite(see backup) + Intel Open Vino

Microsoft Brainwave



Brainwave supports:
- ResNet50
- ResNet152
- DenseNet121
- VGGNet16

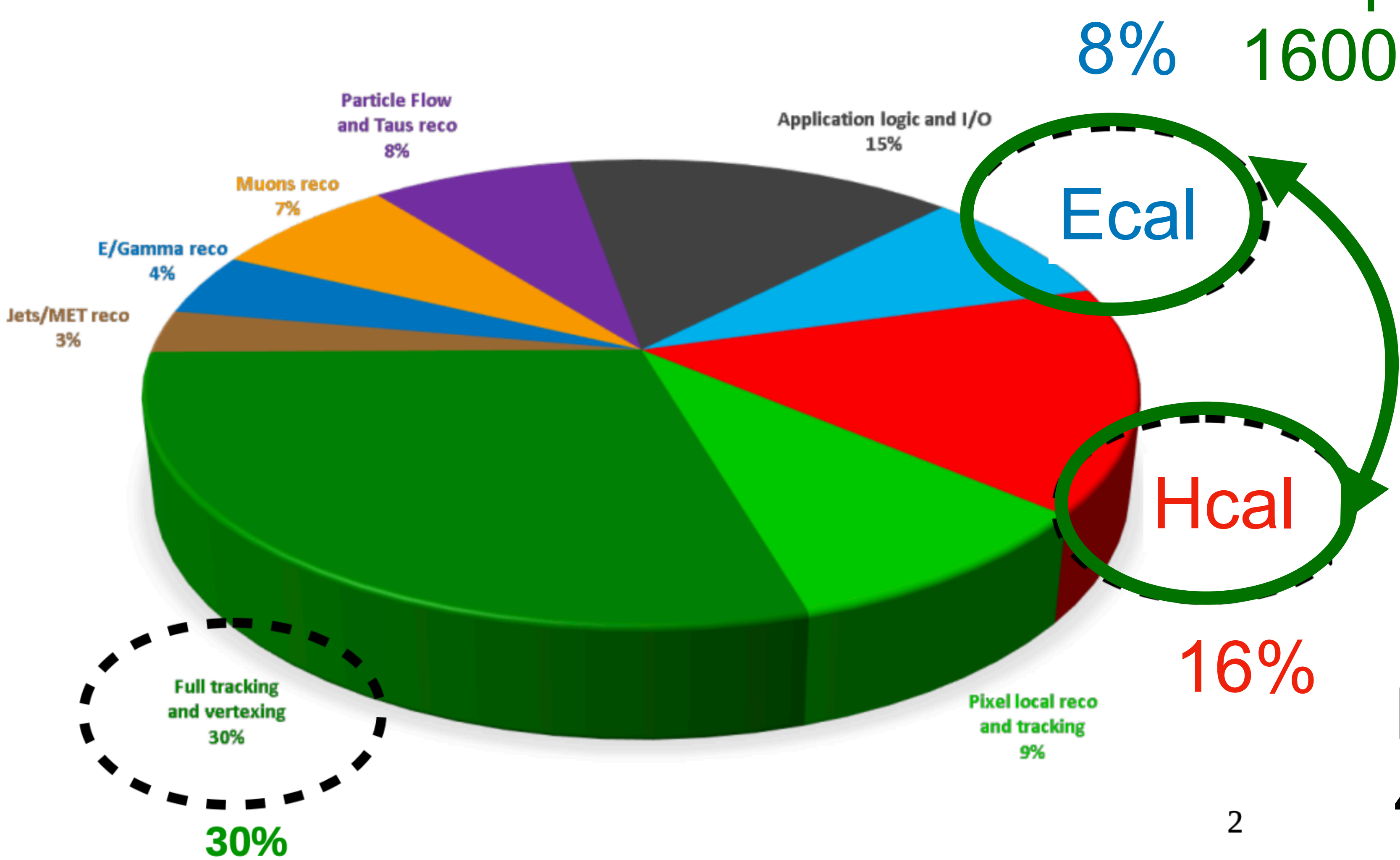
- Full FPGA interconnected fabric setup-as-a-service
  - Capable of running many different NN architectures
  - Relying on the NPU framework for ML compilation
  - (Very) optimized use of ML on the FPGA



Benchmark #1

Time budget per algorithm

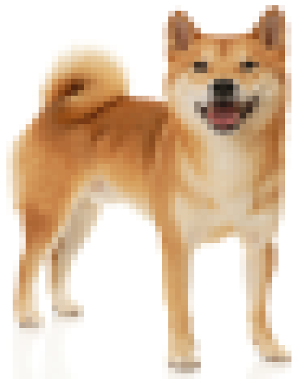
Energy reconstruction of
Hadronic showers
Simple energy regression
16000 times per collision



Roughly 25% of our computing budget

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

Already developed algo w/good performance

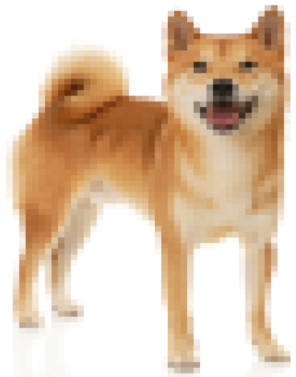


How Fast is it?

- 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**

Many different Top Tagging attempts

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

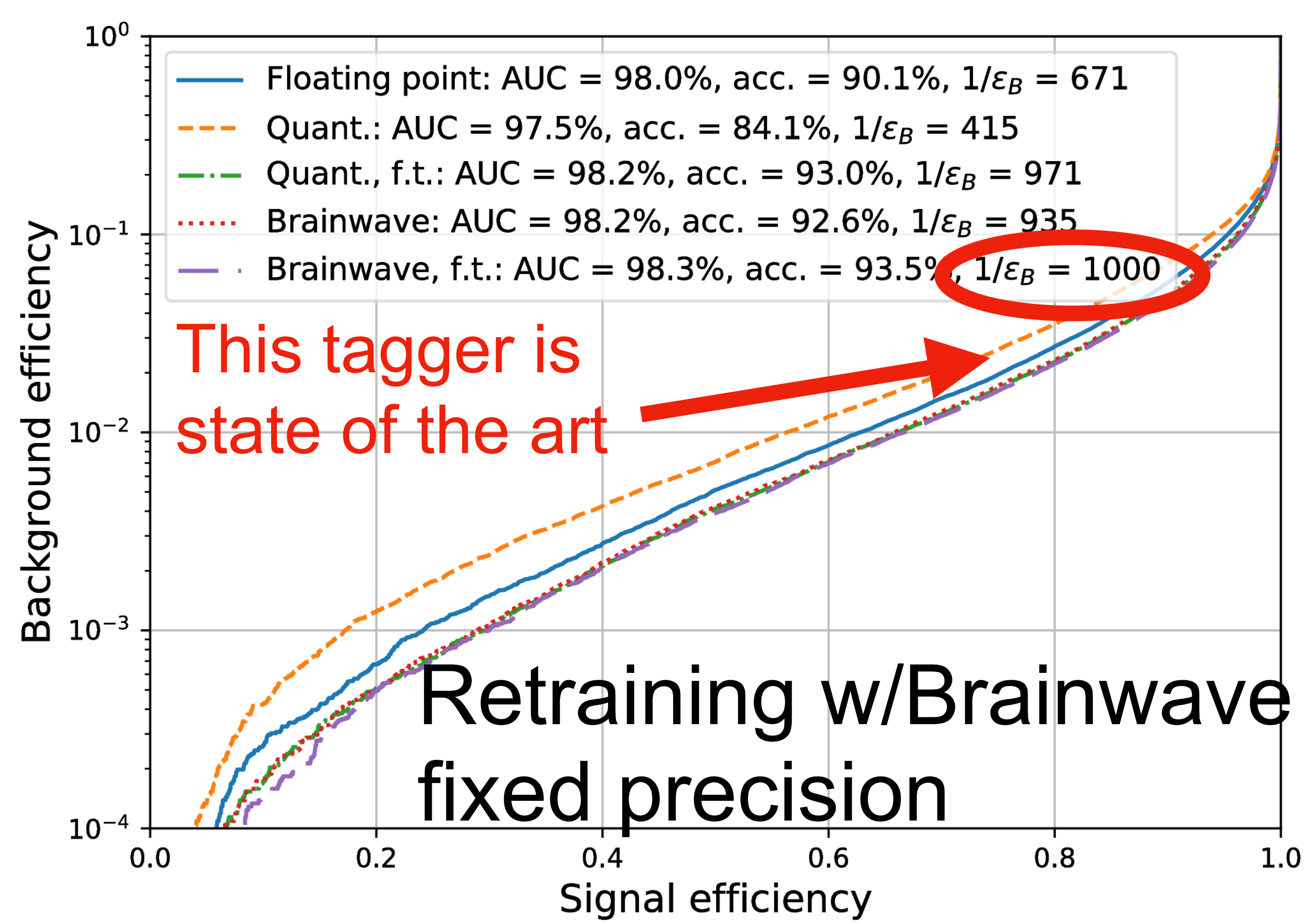
Approach	AUC	Acc.	1/eB (@ eS=0.3)	Contact	Comments
LoLa	0.980	0.928	680	GK / SImon Leiss	Preliminary number, based on LoLa
LBN	0.981	0.931	863	Marcel Rieger	Preliminary number
CNN	0.981	0.93	780	David Shih	Model from *Pulling Out All the Tops with Computer Vision and Deep Learning (1803.00107)*
P-CNN (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

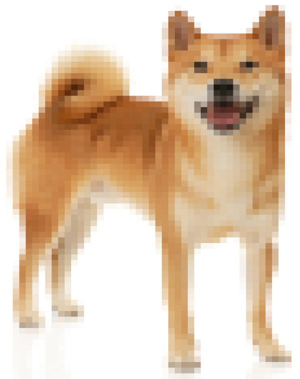
Worlds Best Tagger:

AUC=98.4% acc.=93.7% $1/\varepsilon_B = 1160$

Our Tagger:

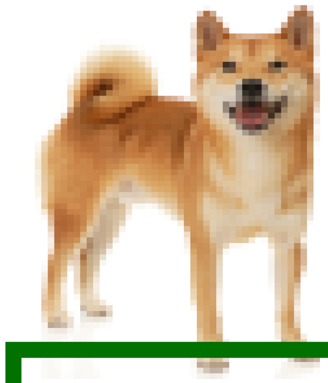
AUC=98.3% acc.=93.5% $1/\varepsilon_B = 1000$





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



Improving Performance

Idea #1

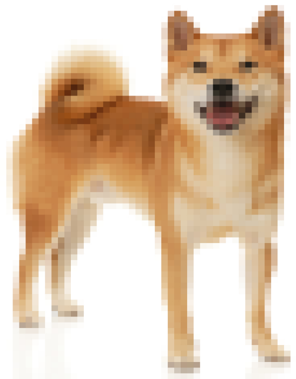
- Buy a GPU/FPGA card for each node
  - Pro: Can be done now   Con: Massive code rewrite

Idea #2

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

Idea #3

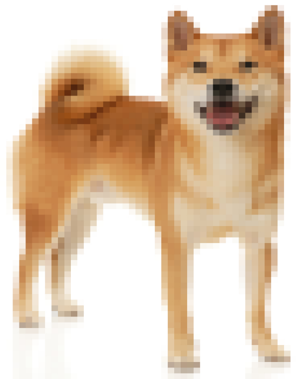
- Port what we can to ML and rely on existing/new tools
  - Pro: We like ML   Con: Redisgn algorithms can be hard



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)

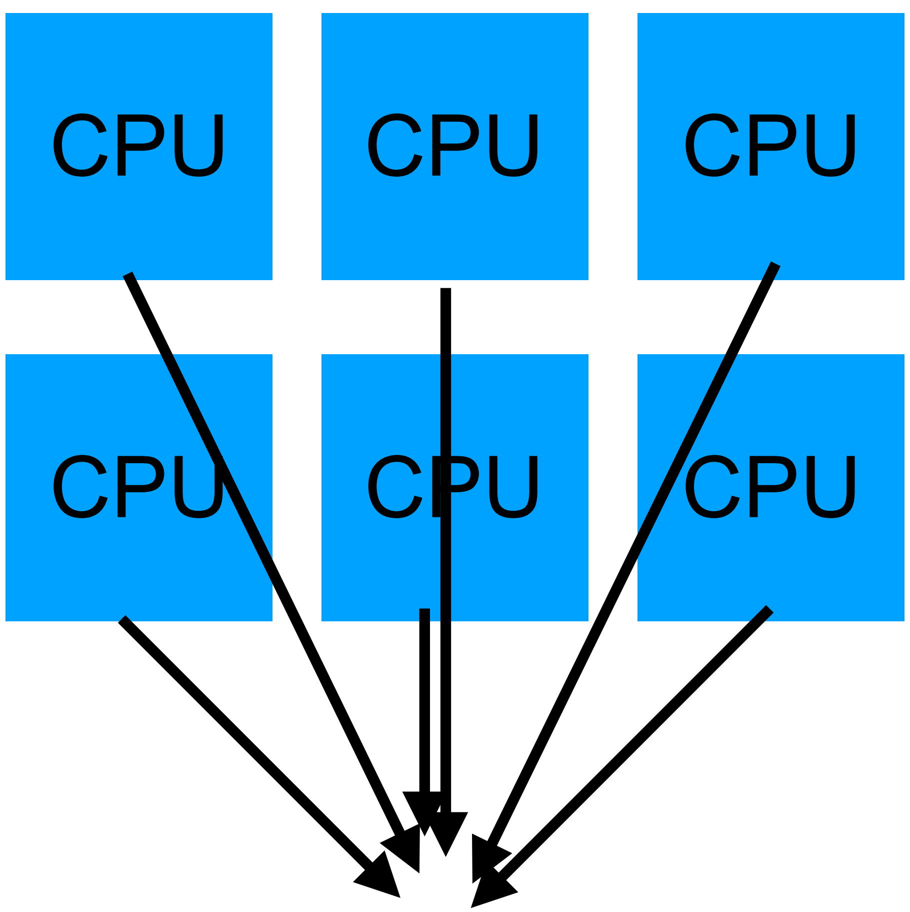
ML is highly parallelizeable→Big speed ups

Takeaways

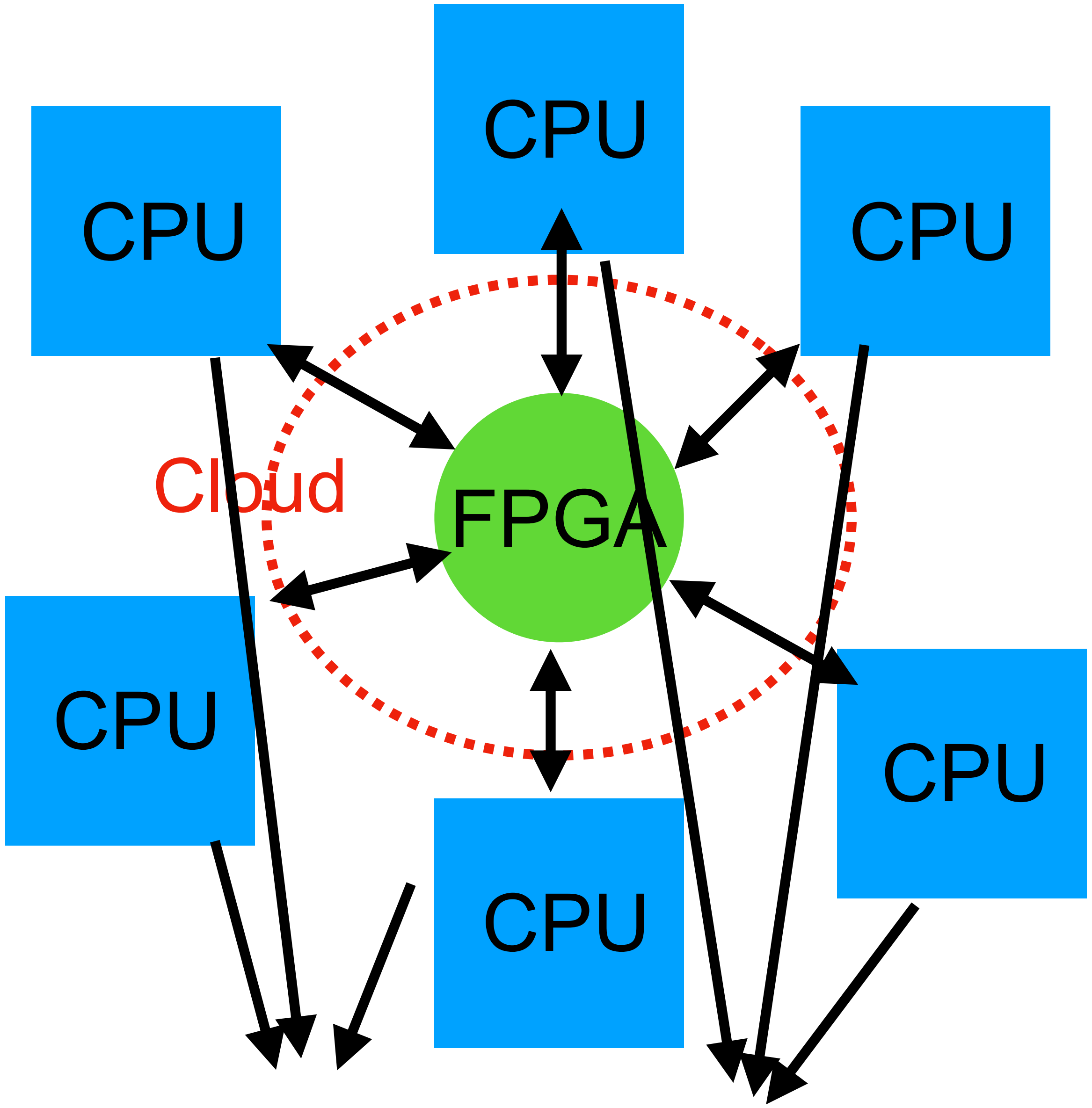
- When large speedups are present in overall throughput
  - Where as-a-service starts to really shine
  - Can think about one service for many machines
  - Will take a latency hit in our system from this
    - This is something we can deal with
- Our next step is bringing the studies to scale
  - Can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

What have we learned?

- With large speedups we can redesign our system



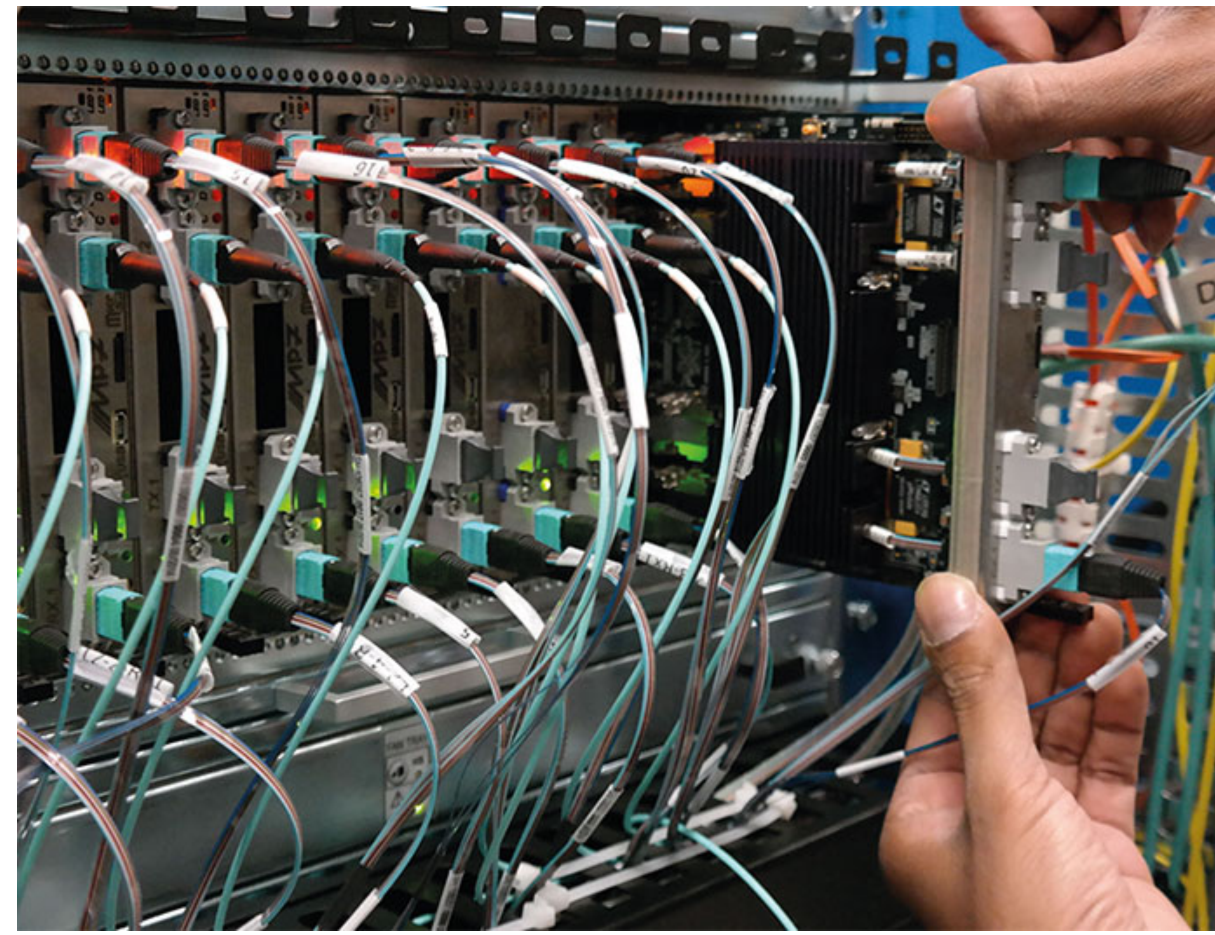
Process event by event



Process (event by event)?
outsource to aaS

40 MHz (10μs)

L1 Trigger

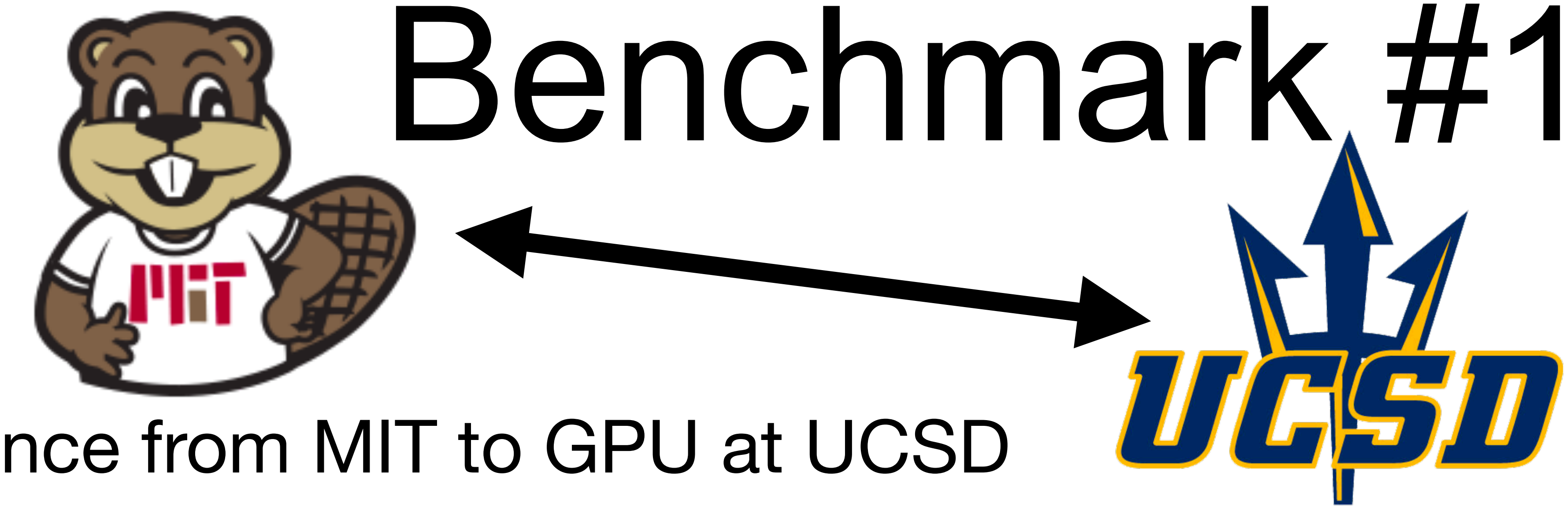


A new event every 25ns

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

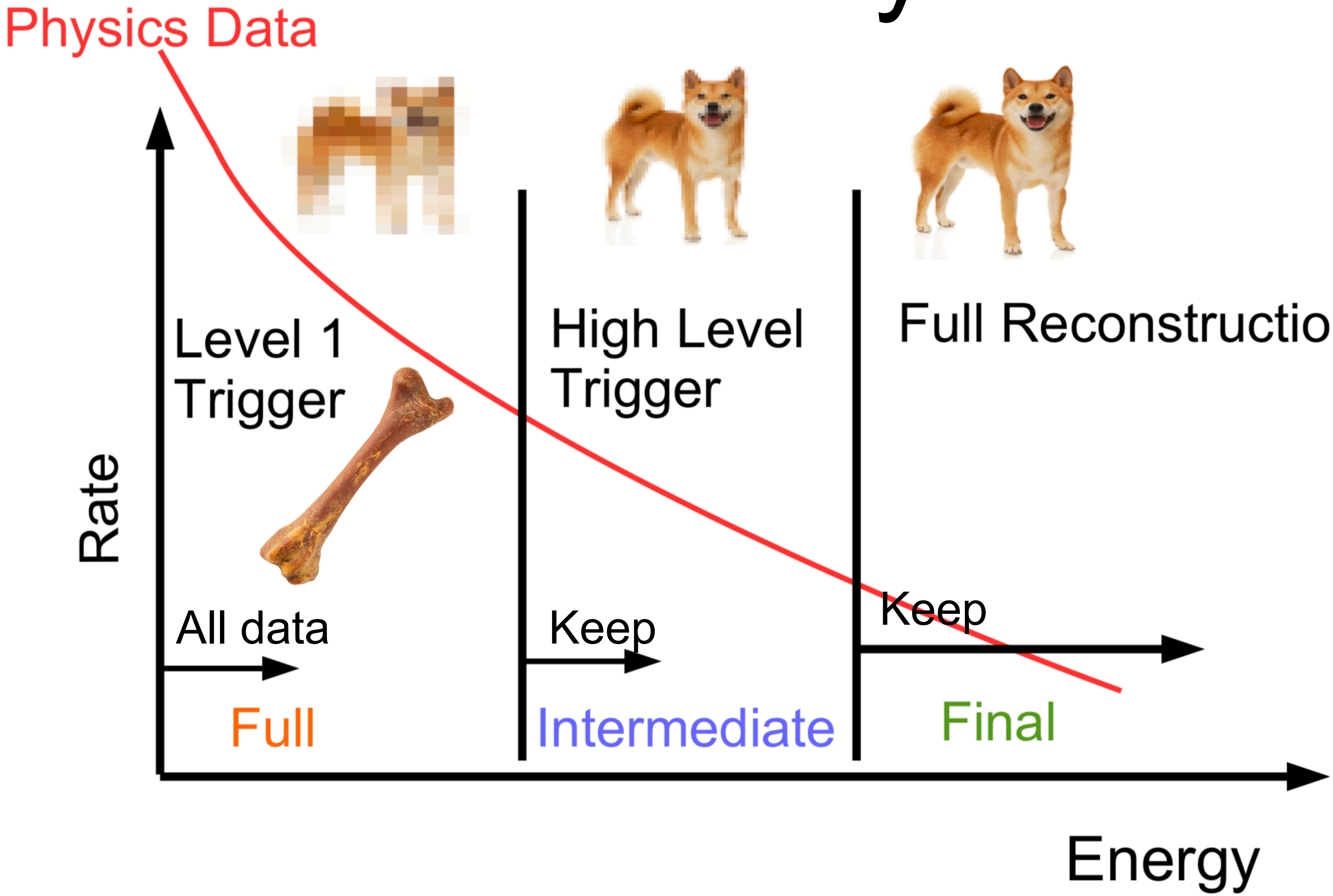
Benchmark #1



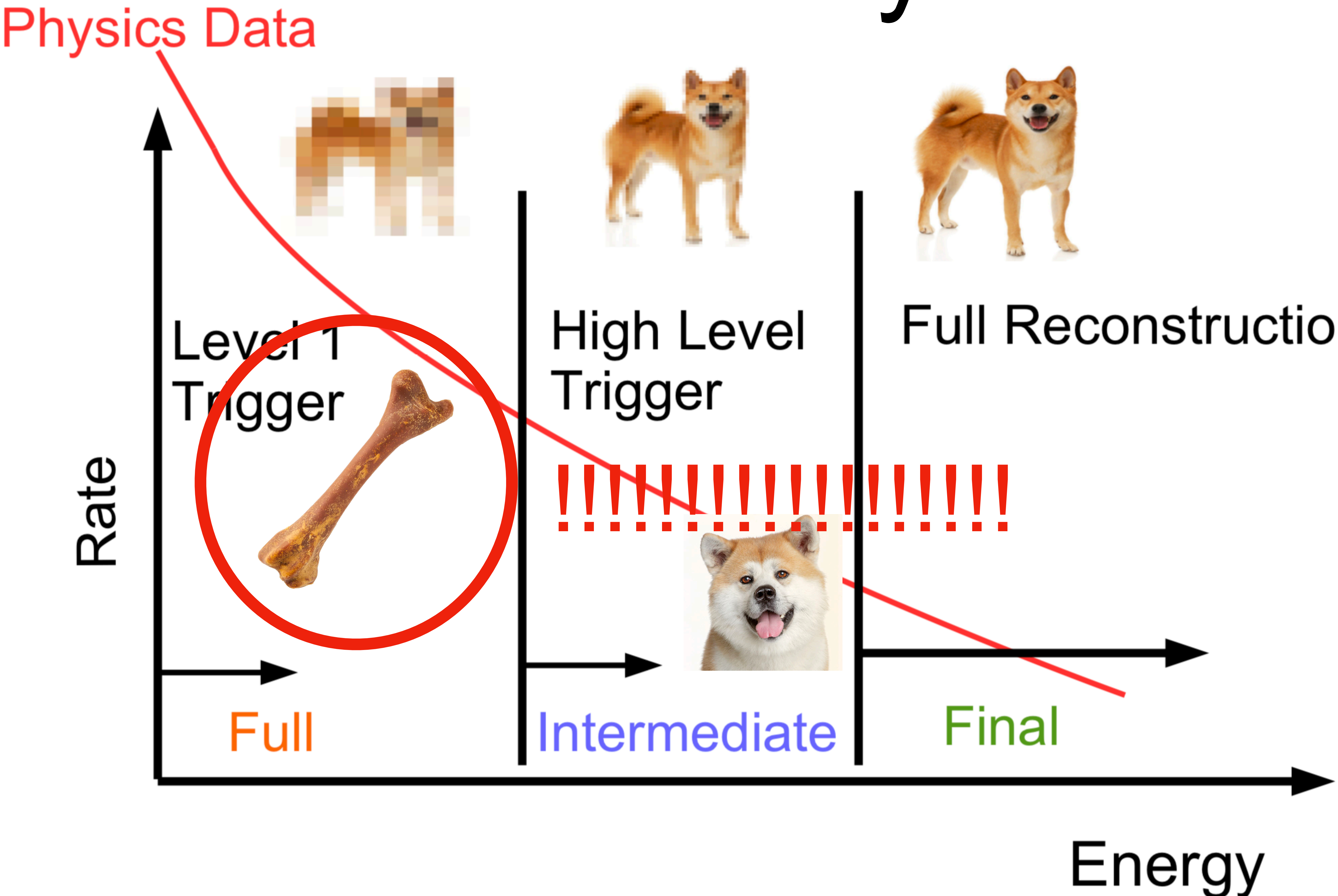
- Send our 16k inference from MIT to GPU at UCSD
  - Ping time is 75ms (speed of light google map distance is 32ms)
  - To UCSD and back takes ping time + 16ms
- Still working on test with FPGA (soon)

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

The Physicist View



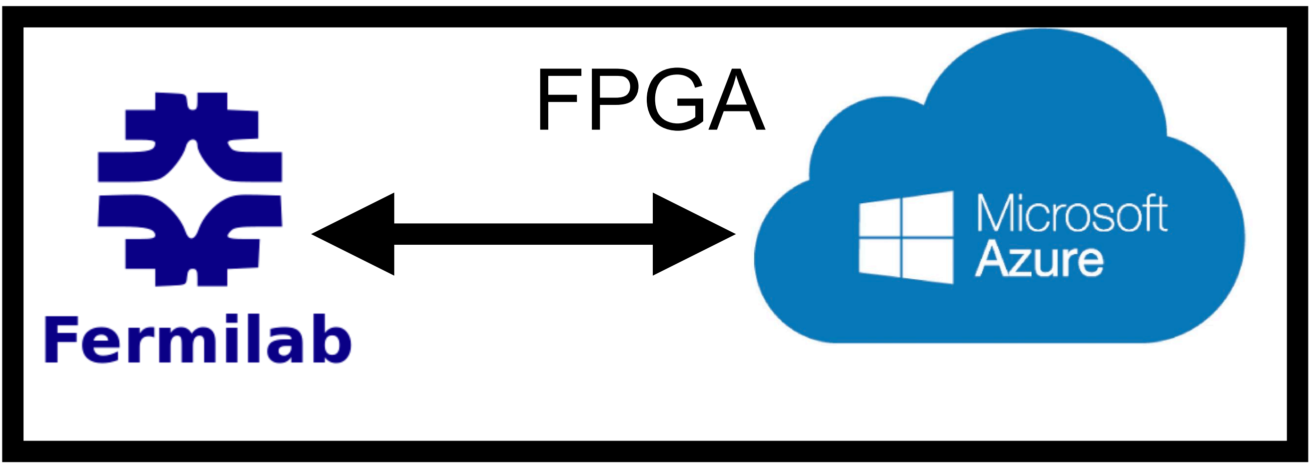
The Physicist View

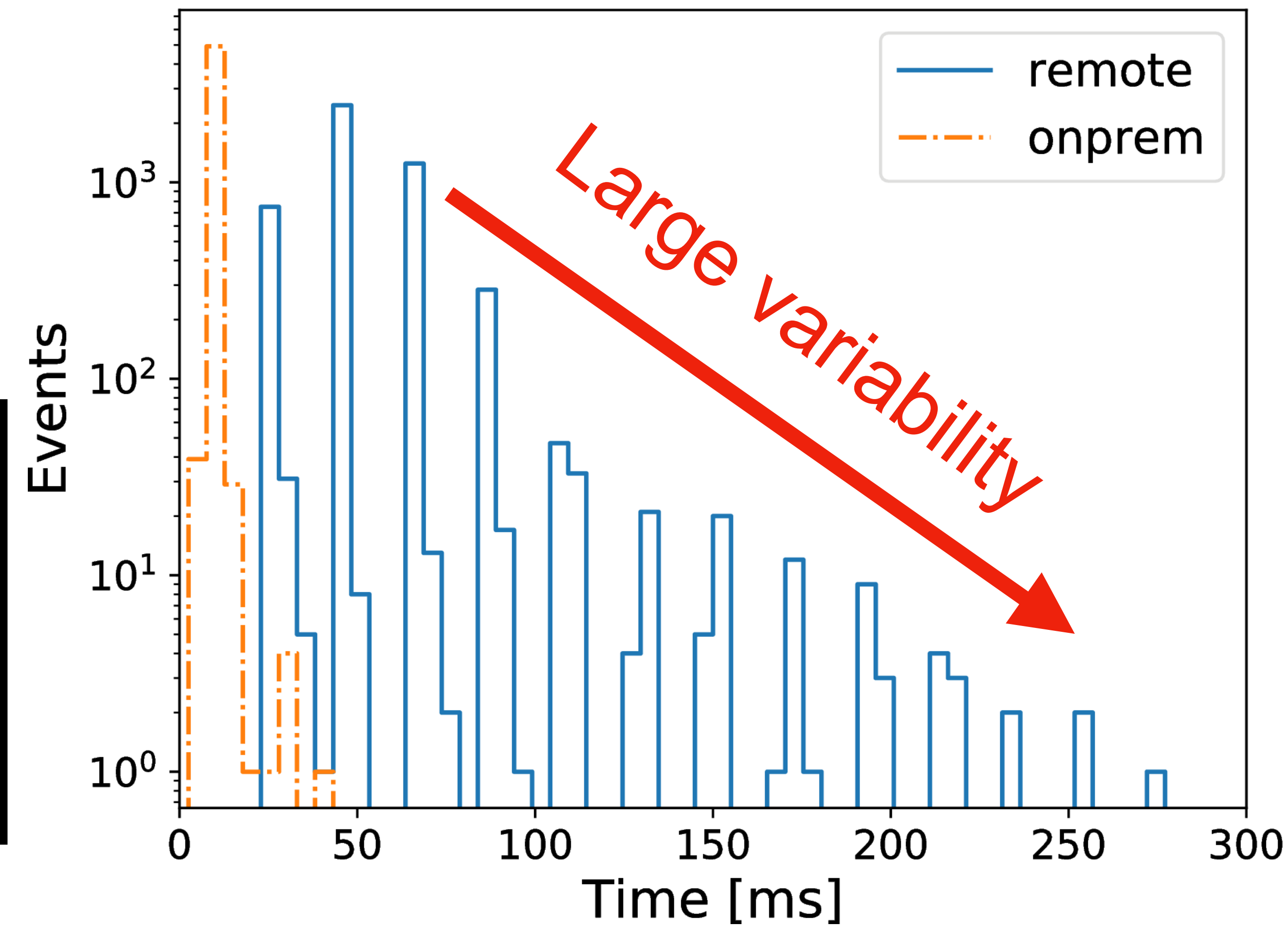


ML at all Tiers will help to recover missing physics

Benchmark #2

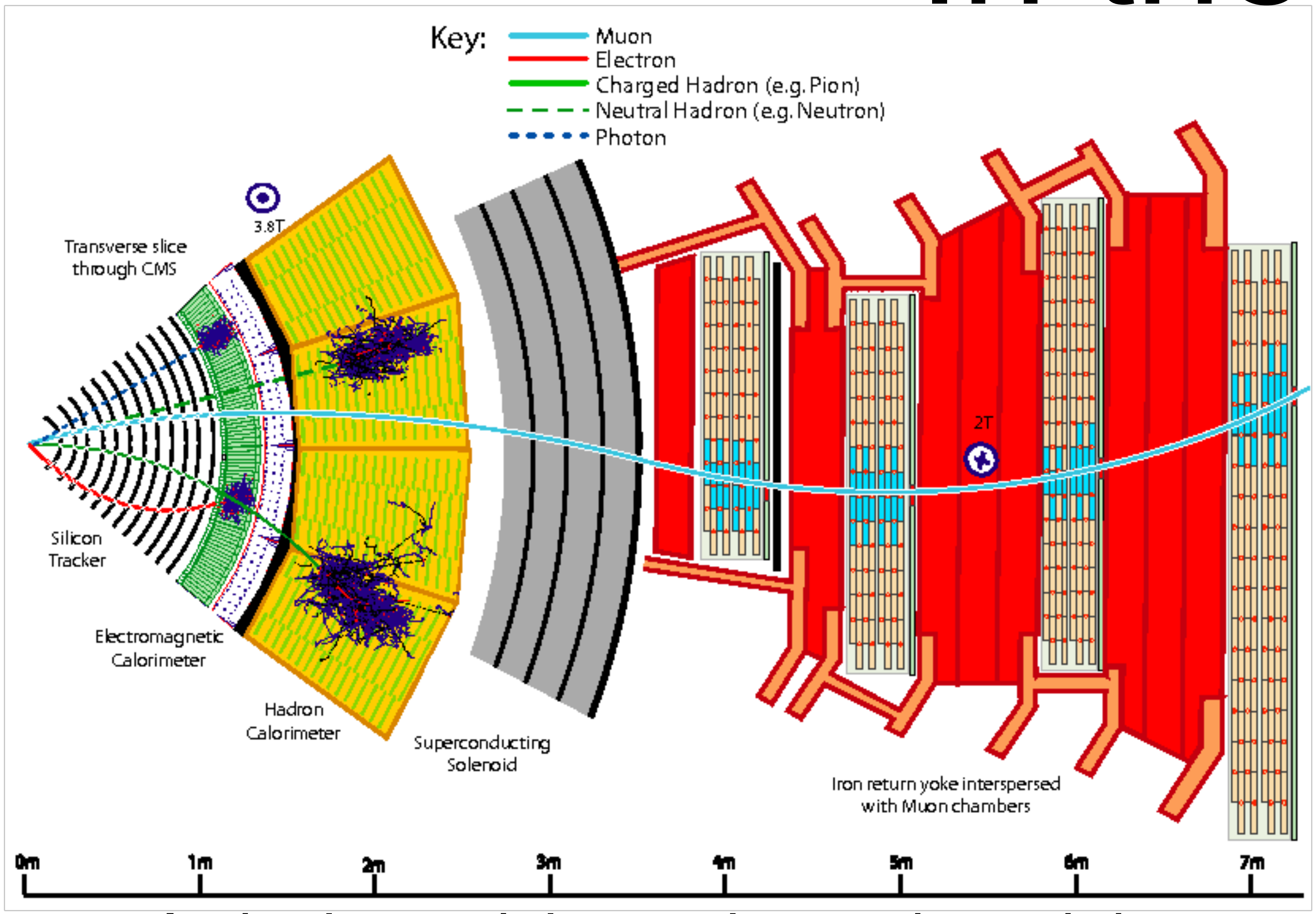
UCSD to MIT for GPU
FNAL to Azure for FPGA





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

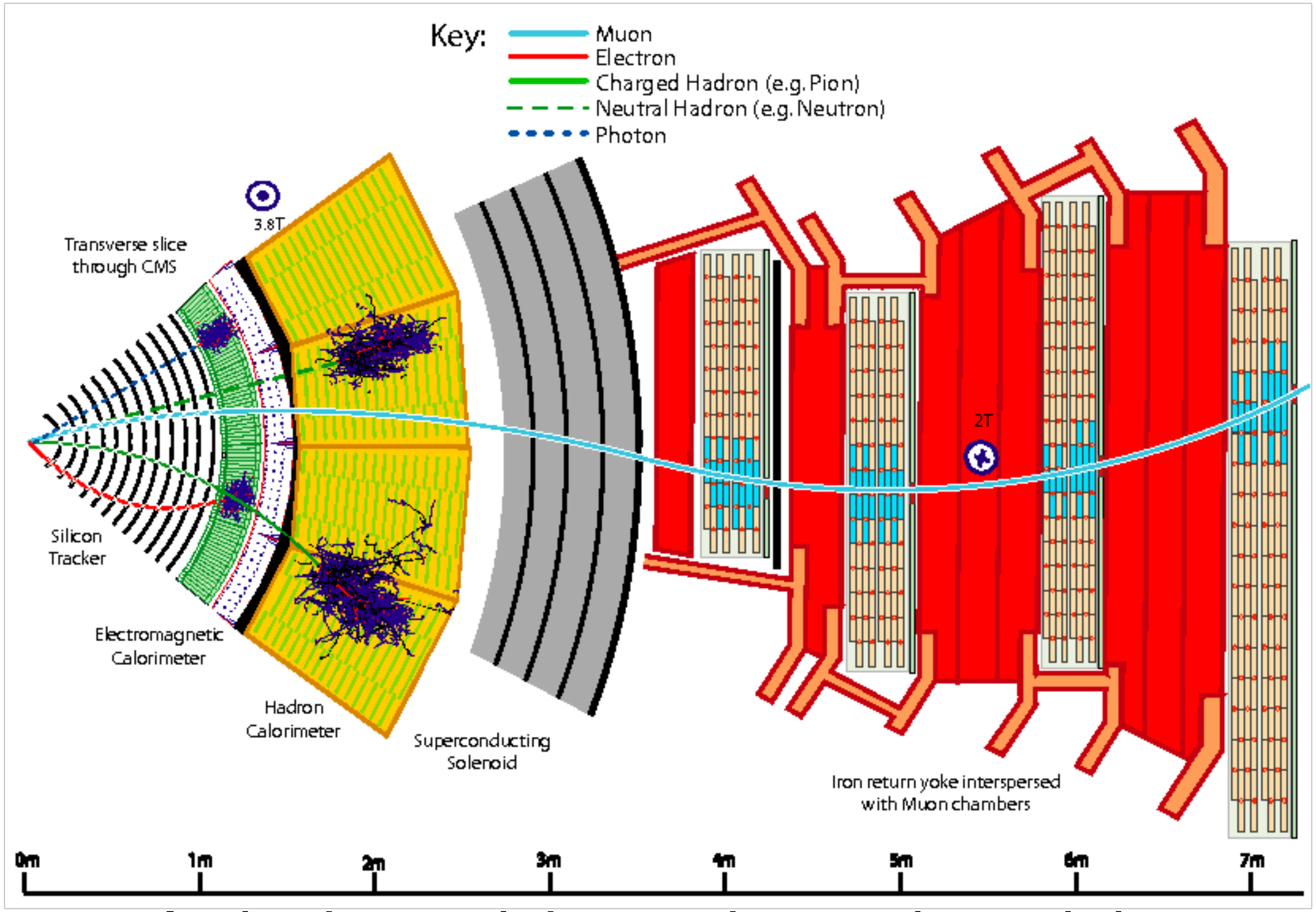
In the detector



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

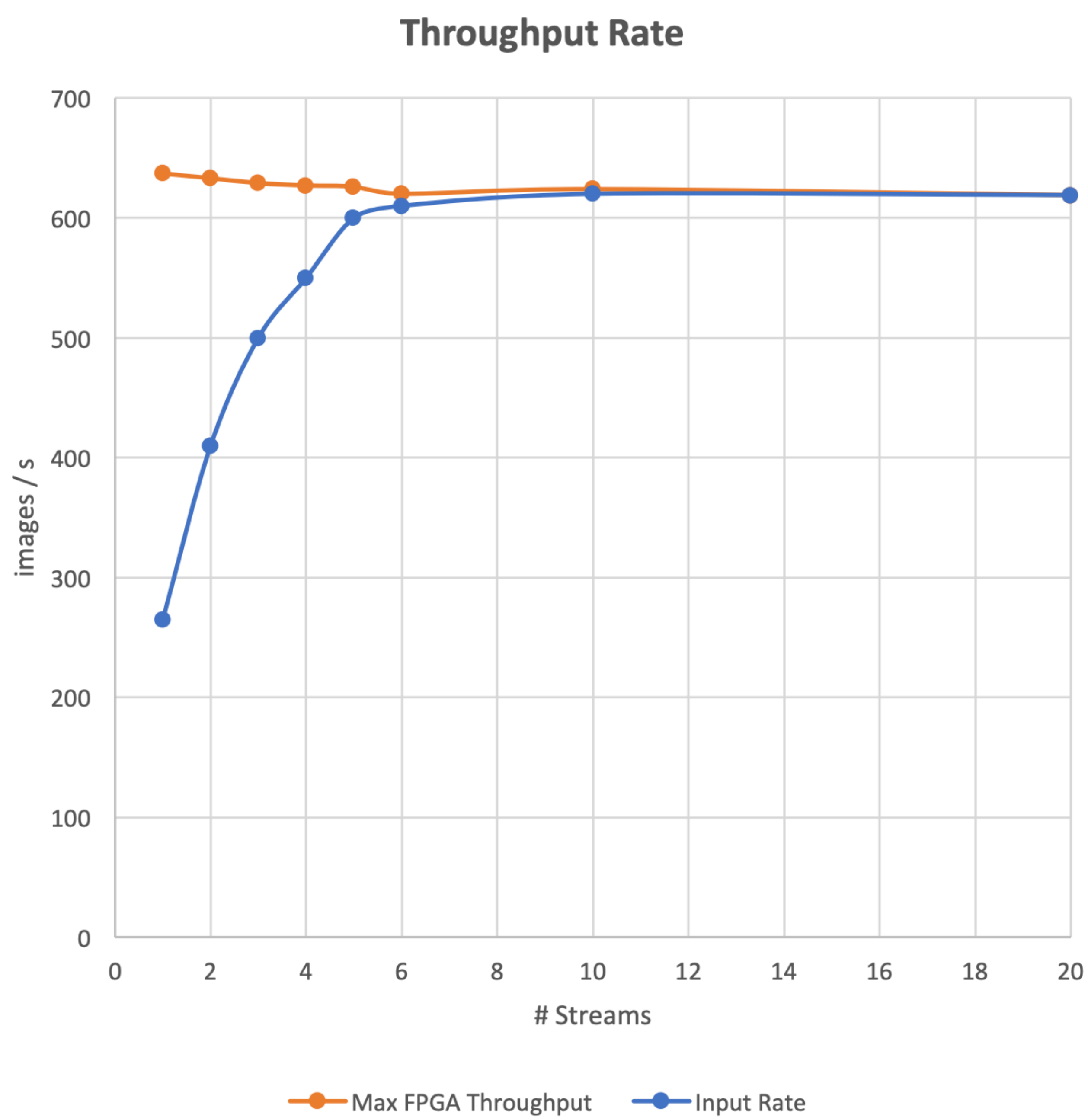
Batch N Per Particle

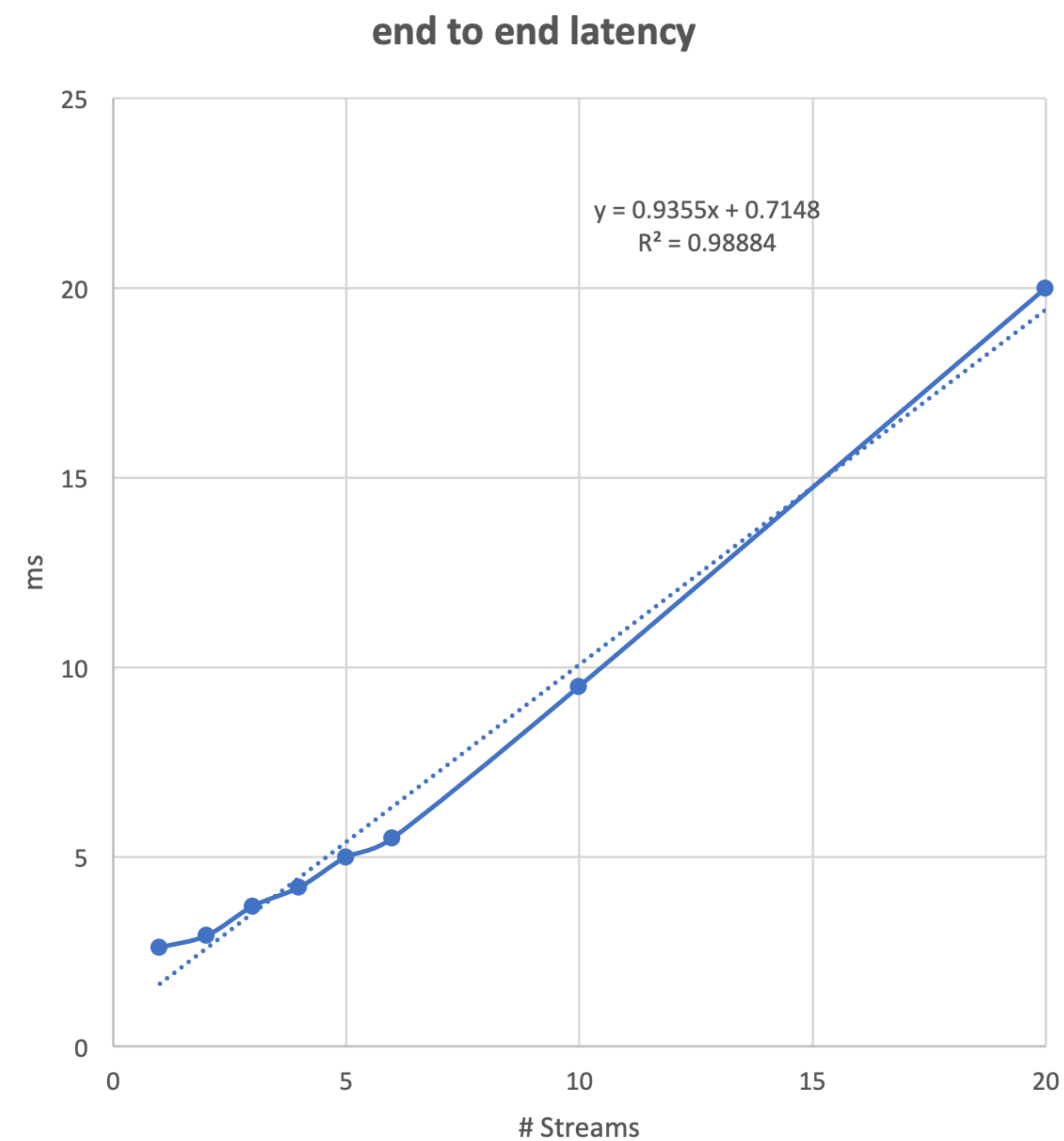
- Reconstruction of particles/detectors can be parallelized

Xilinx ML Suite

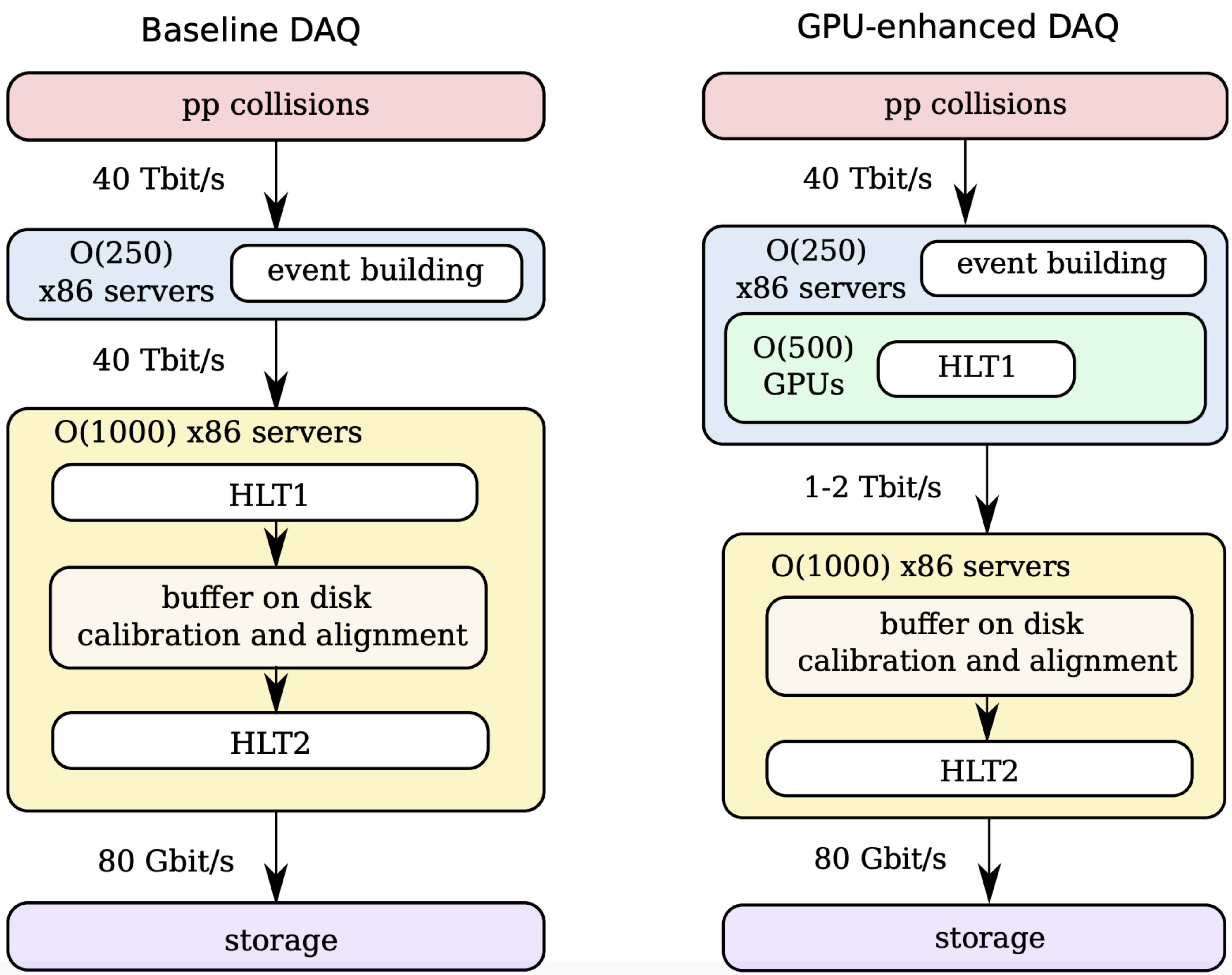
- Consider Googlenet example

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



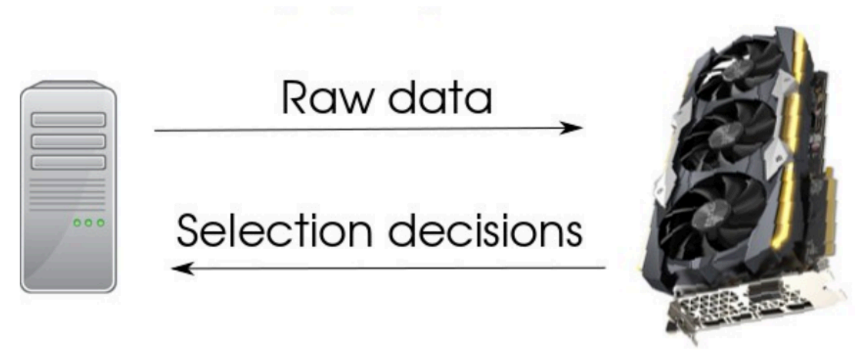


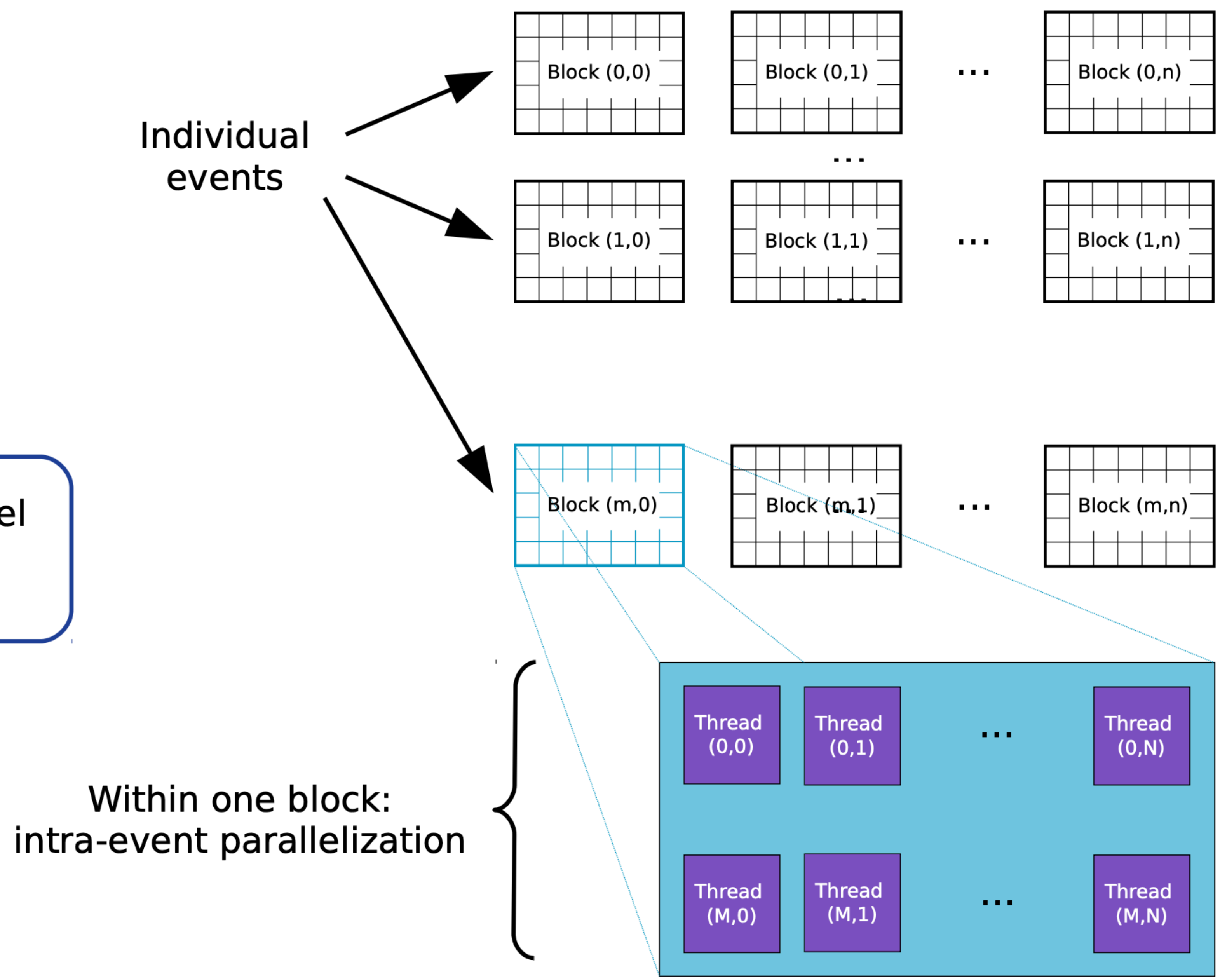
Alternative GPU Model



Full Reconstruction algorithm ported to GPU

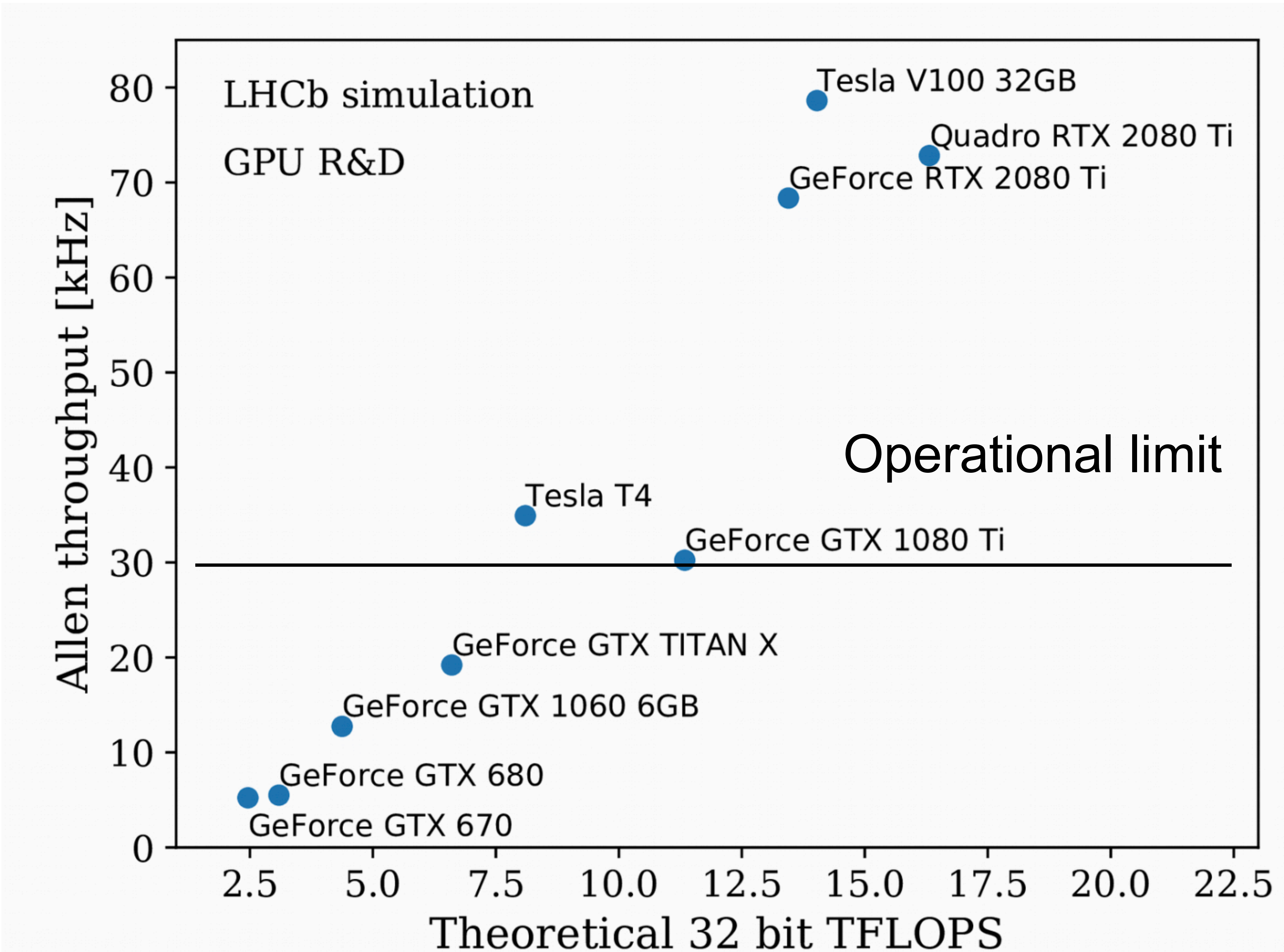
Alternative GPU Model





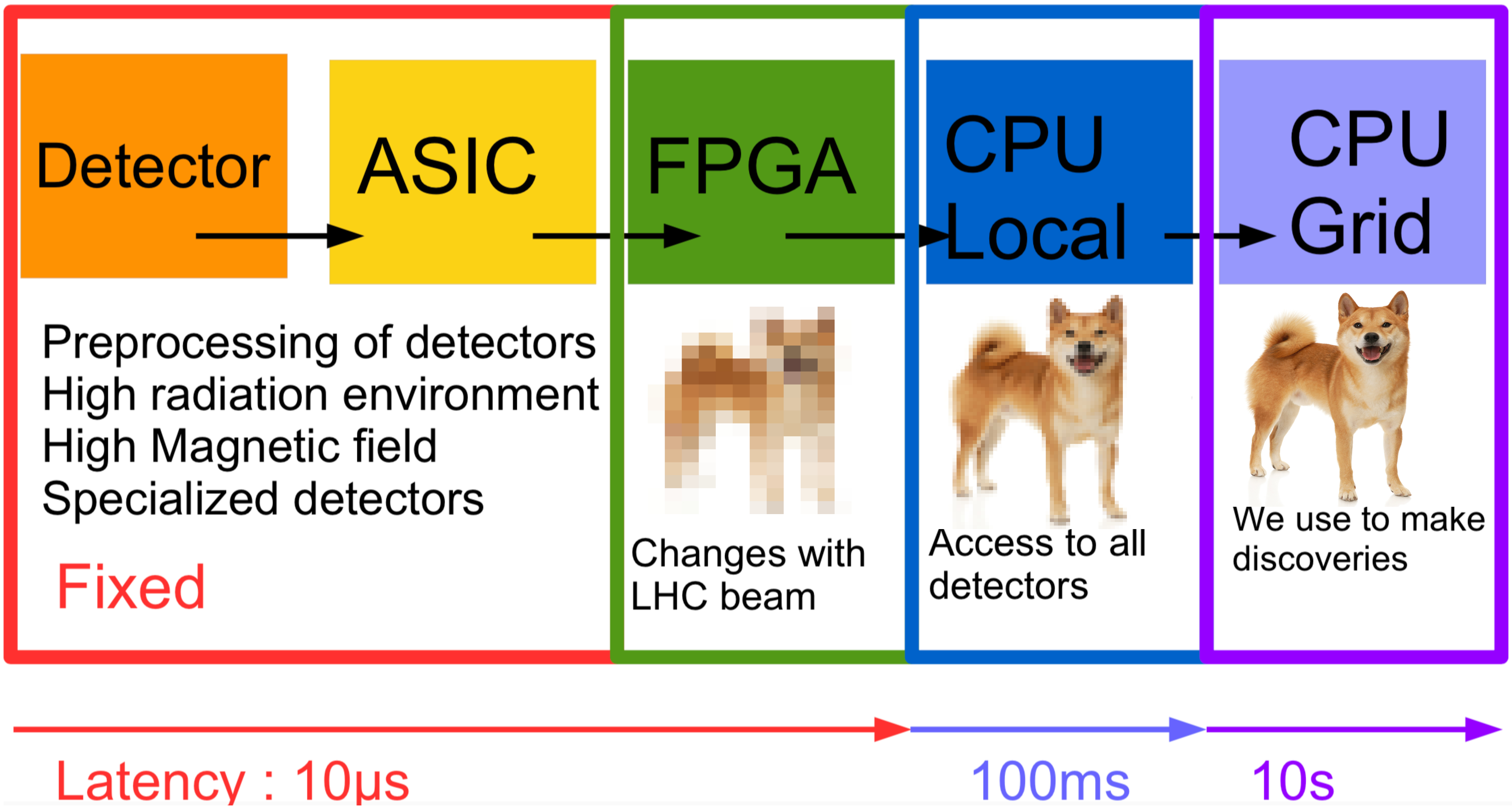
- Process thousands of events in parallel
- Single precision only

Alternative GPU Model



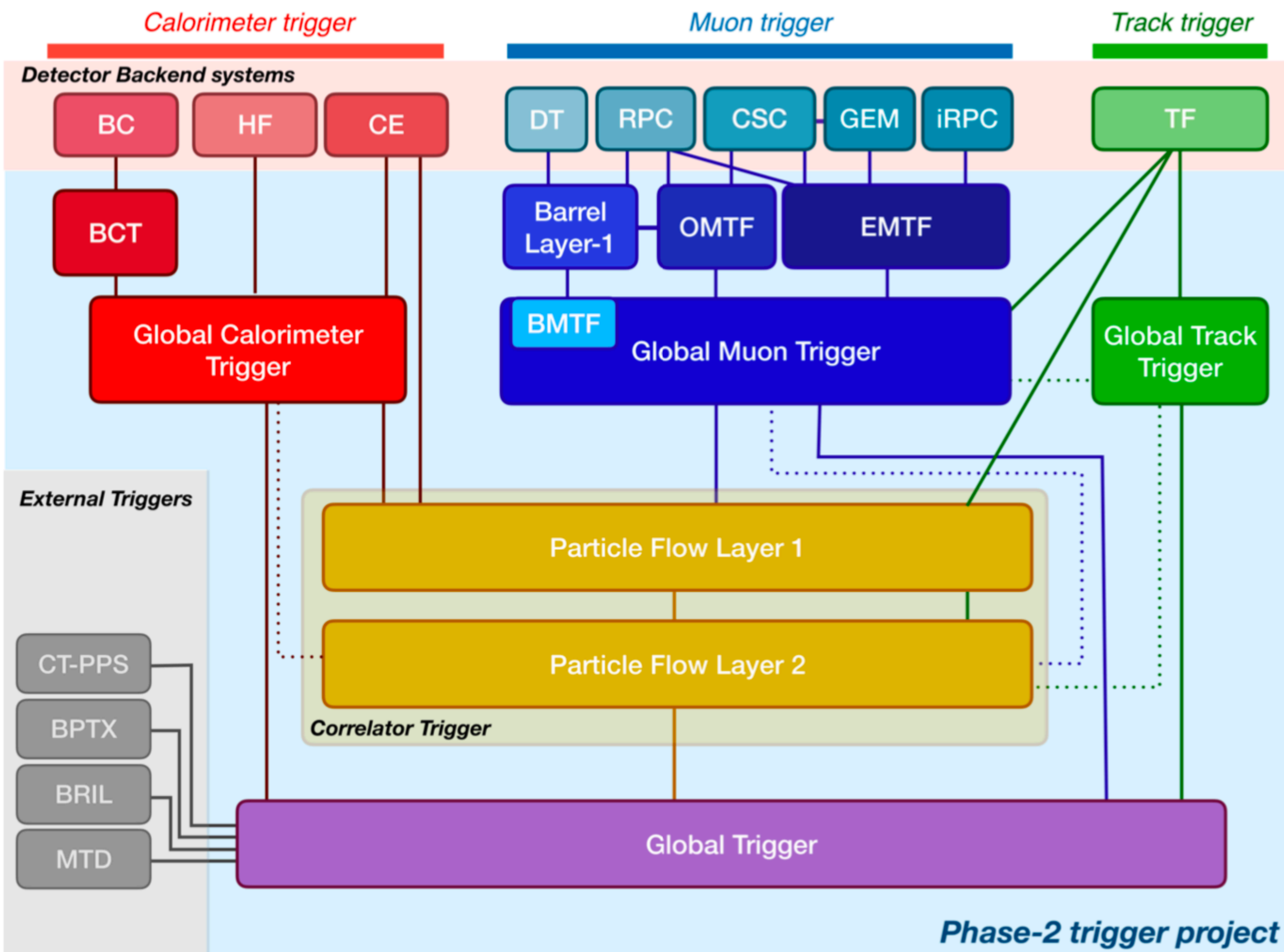
Another View of Same

Collision rate is 40 MHz
A new collision every 25ns



40 MHz (10μs)

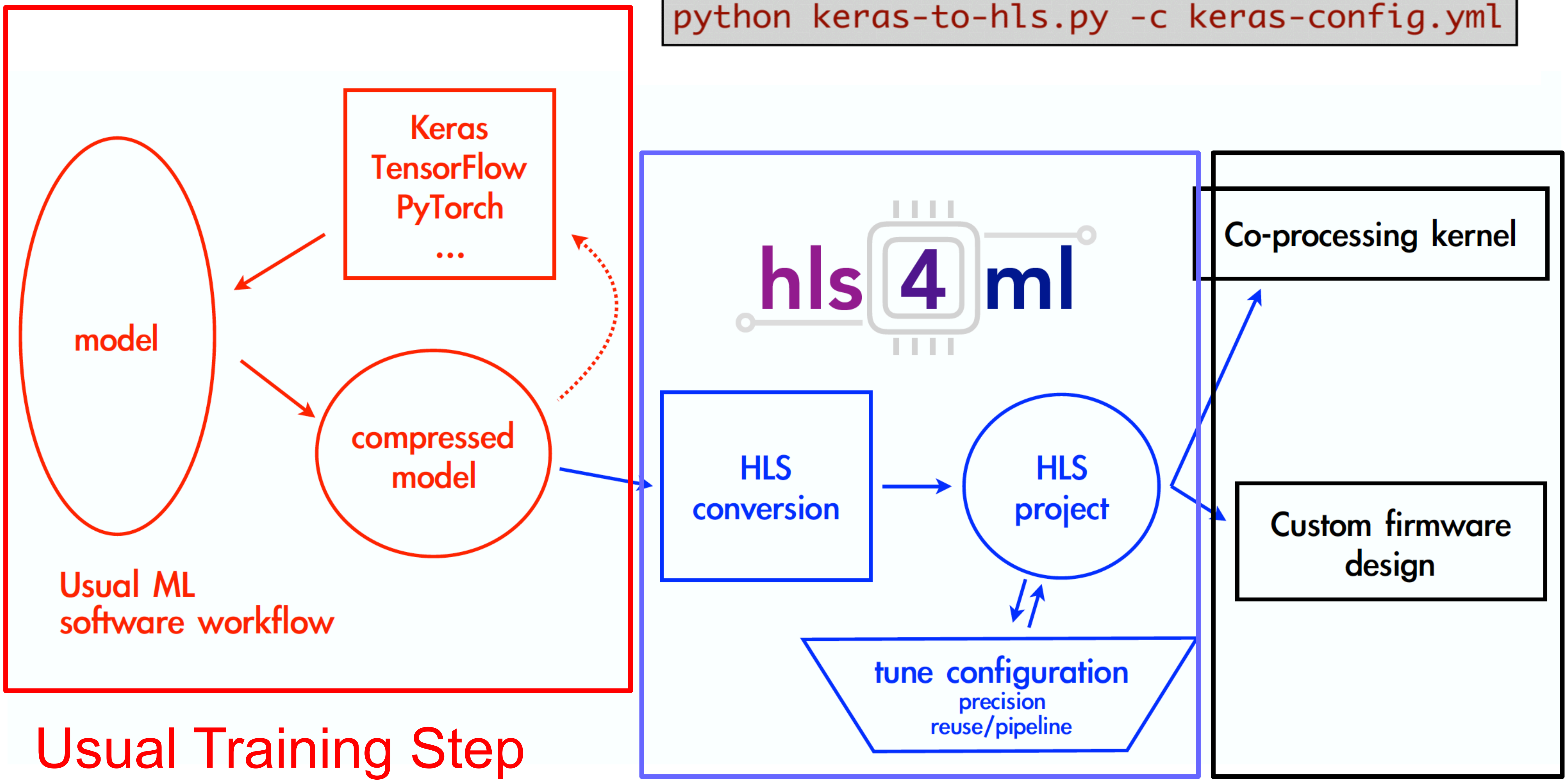
Systems



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
```

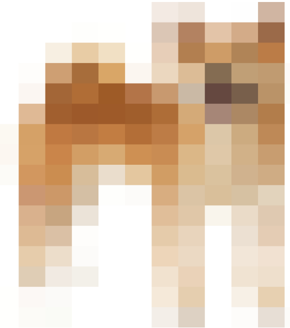


Usual Training Step

Targeting Ultra low latency applications

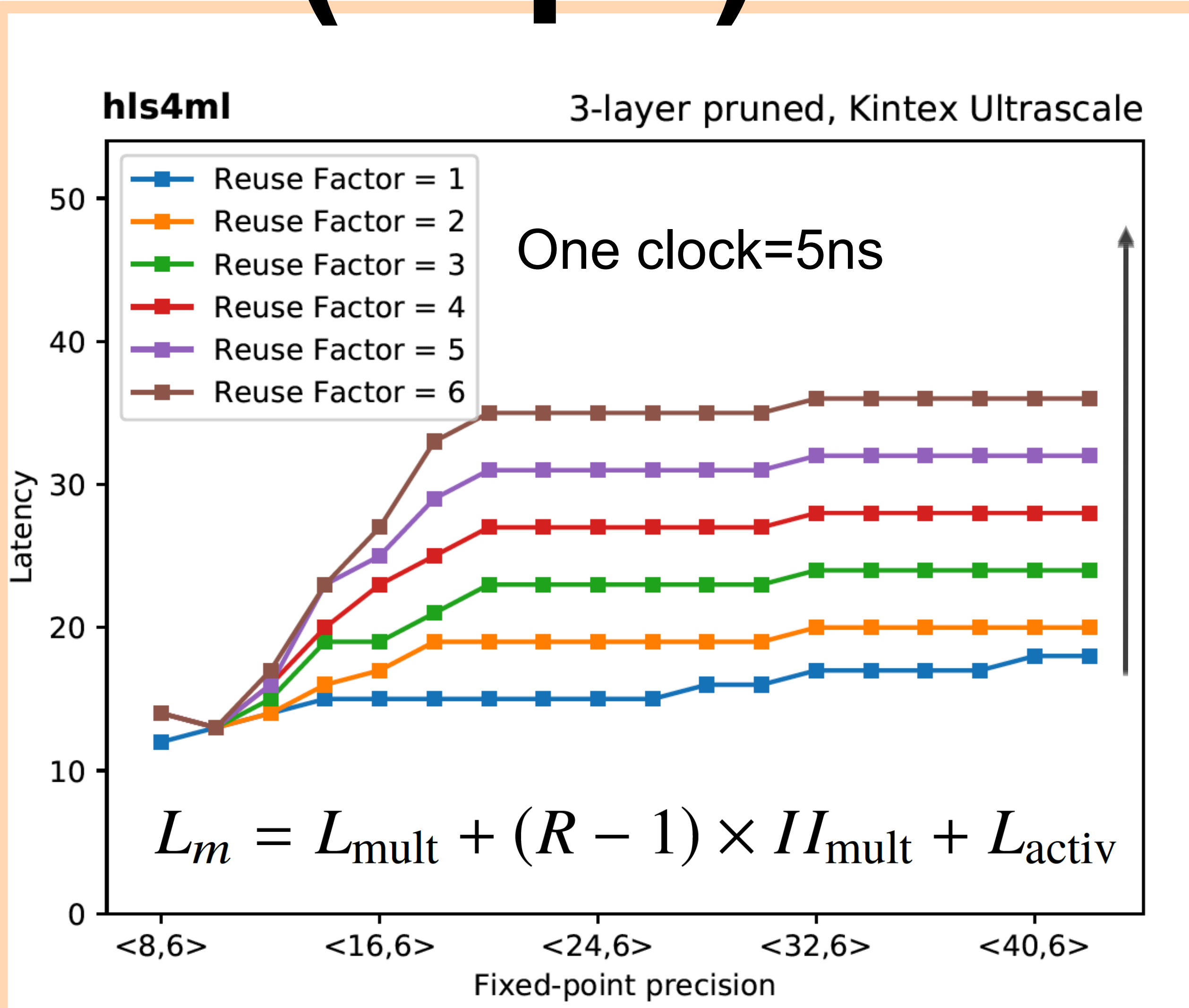
HLS tuning

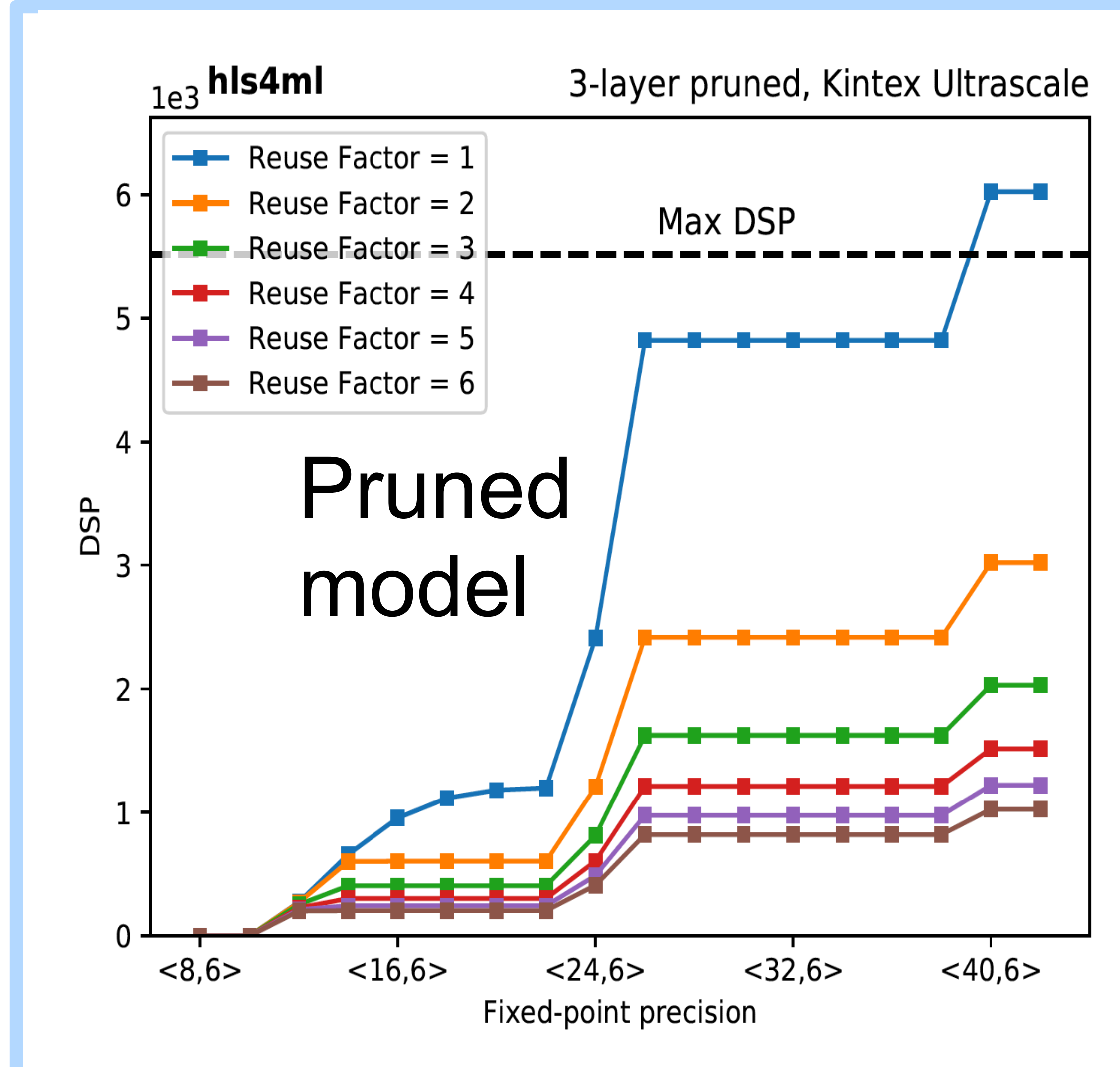
Final Product



40 MHz (10μs)

Example Performance





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

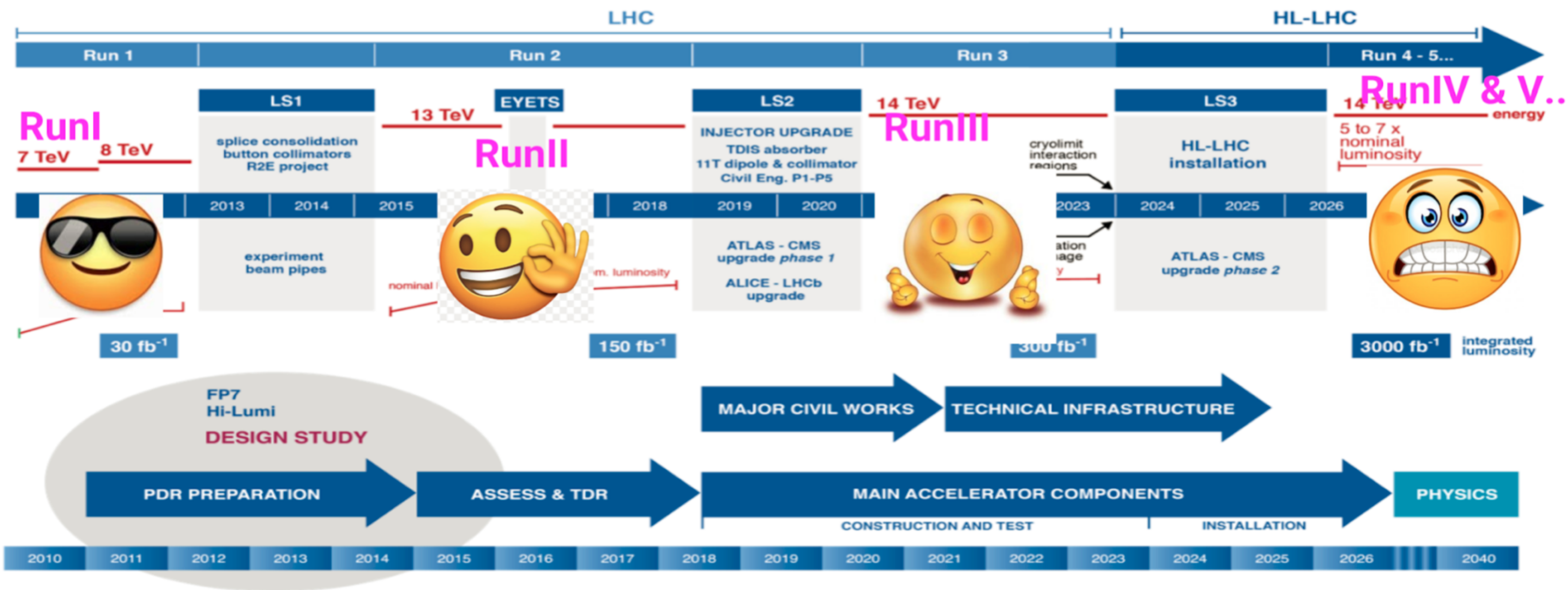
Latency (in clocks) gets worse
With reuse factor
Consistent with sharing resources

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
    )
```

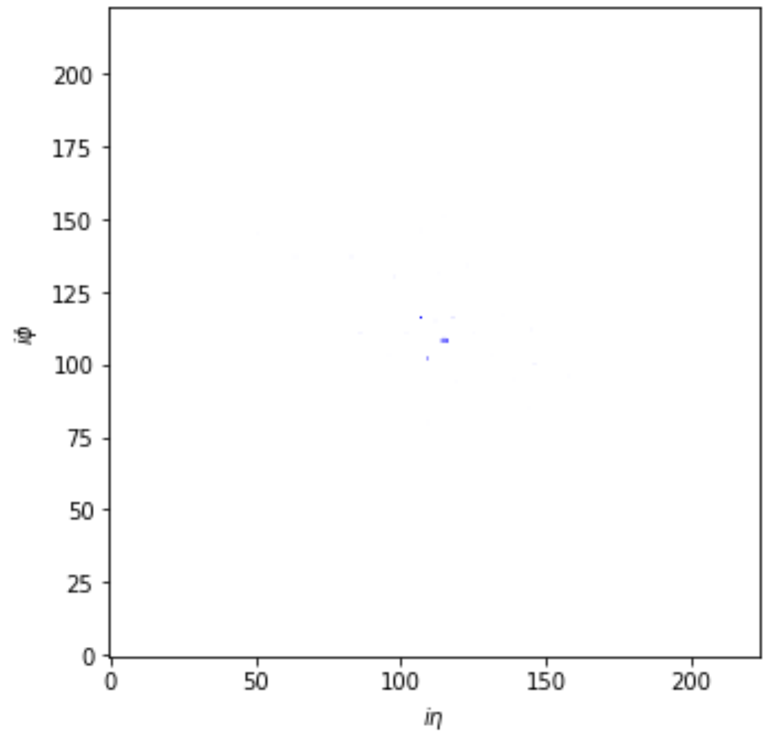
- **Deploy pre-trained NNs** using a CLI or a python SDK

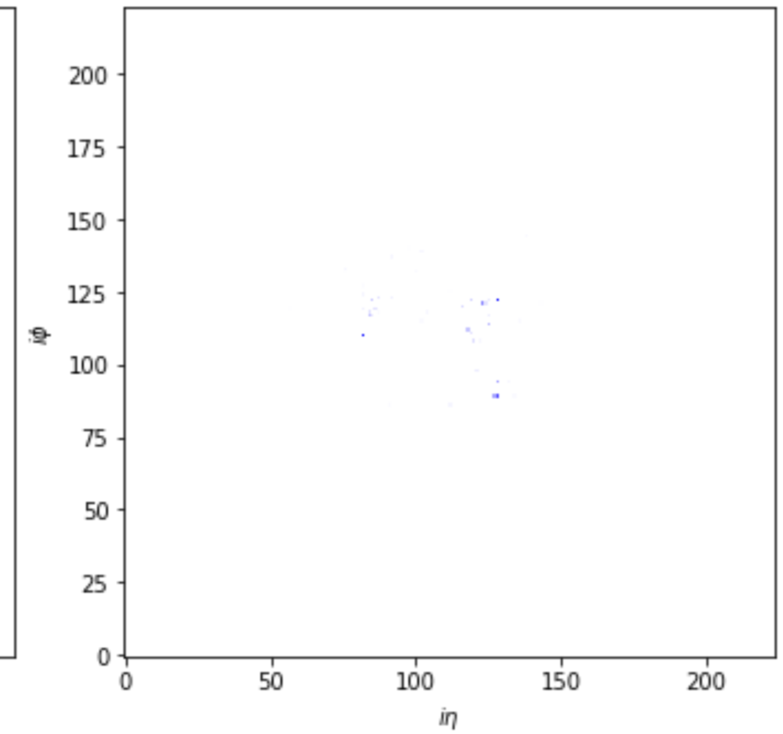
- **Inference from a client** by sending data over gRPC

```
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

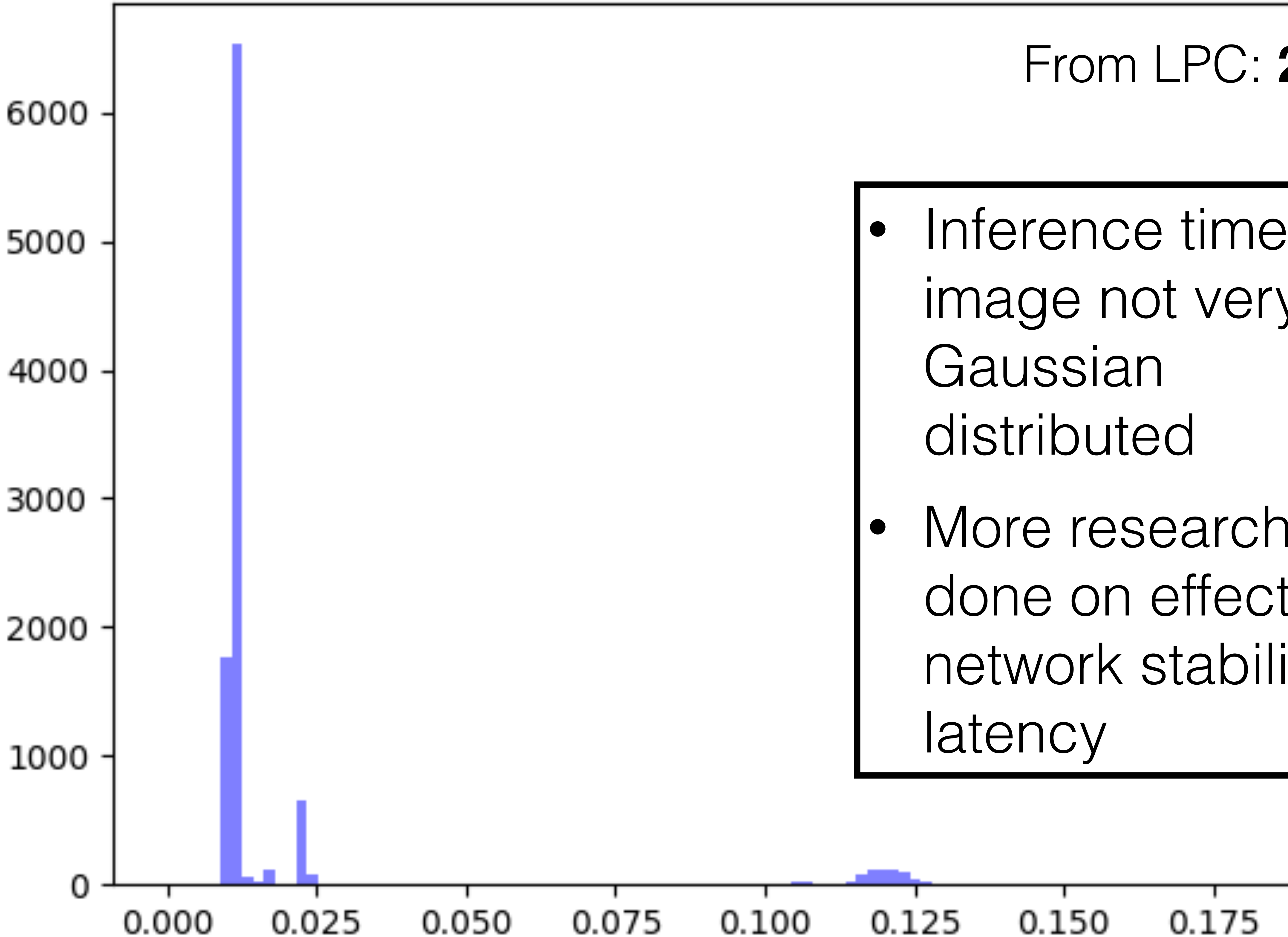
- On a single CPU: **~500 ms**
- On Azure Kubernetes Cloud Service: **~60-80 ms** (depending on distance)
- Deployed at Azure Data Center in Viriginia (2018): **~10 ms**

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**



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

time (s)

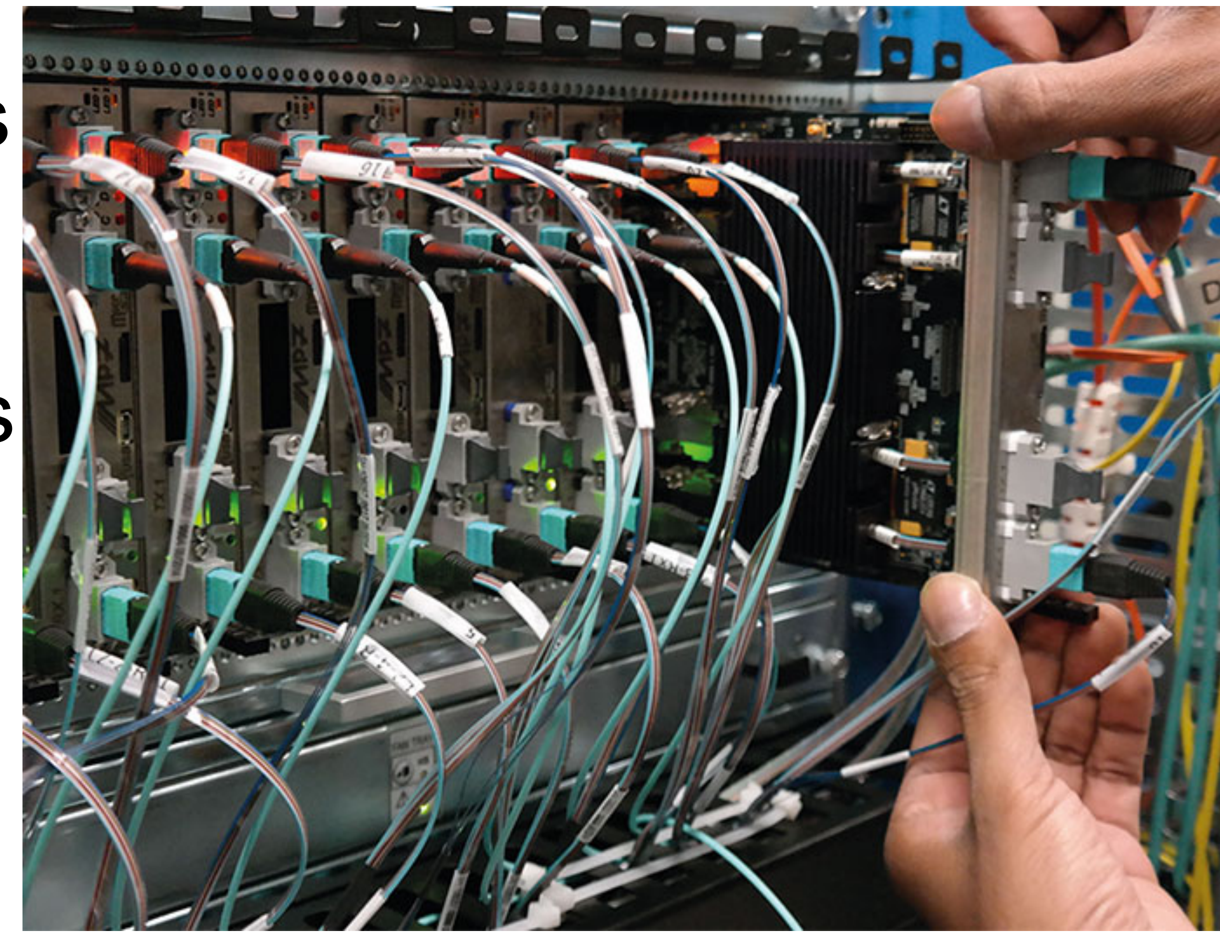
40 MHz (10µs)

L1 Trigger

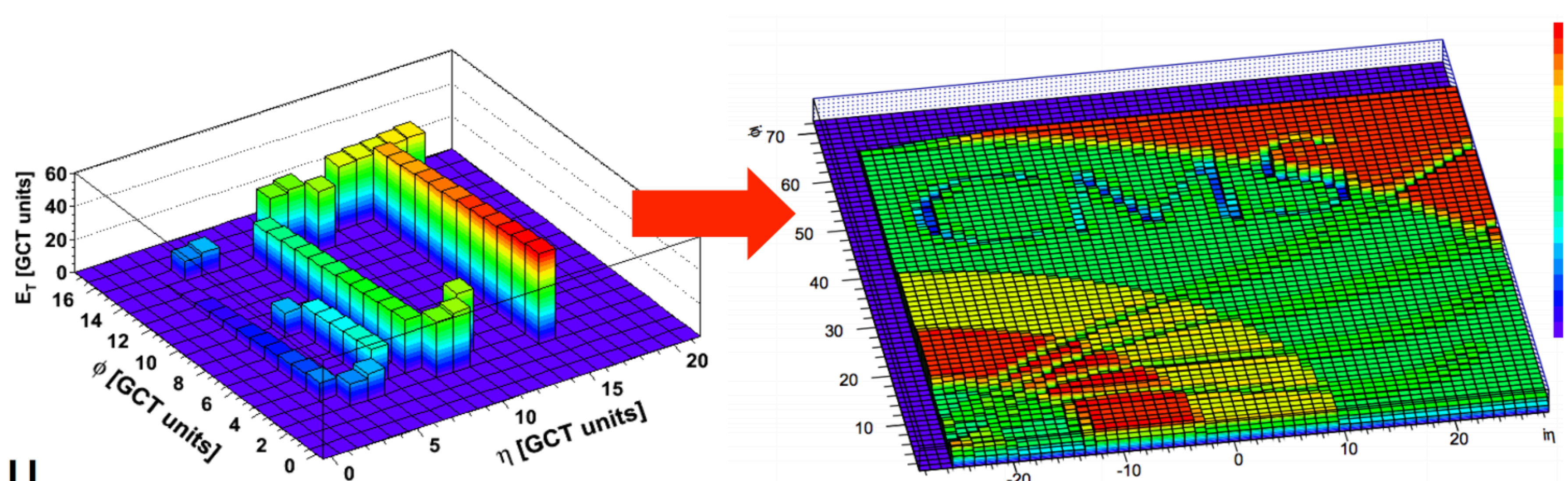
Have to take a new event every 25ns

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



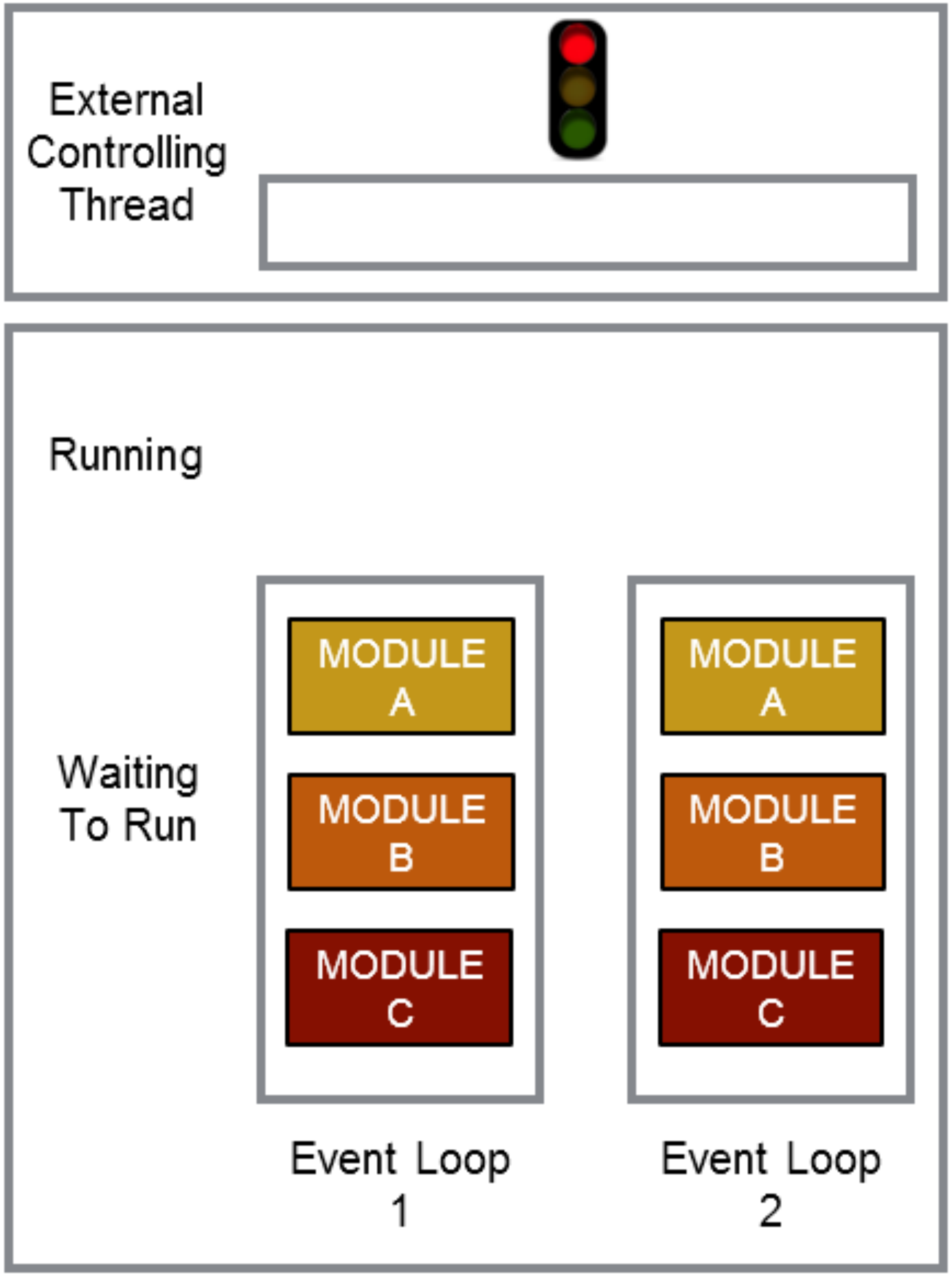
As FPGAs get larger so has the resolution of our detector



External Work in CMSSW (1)

Setup:

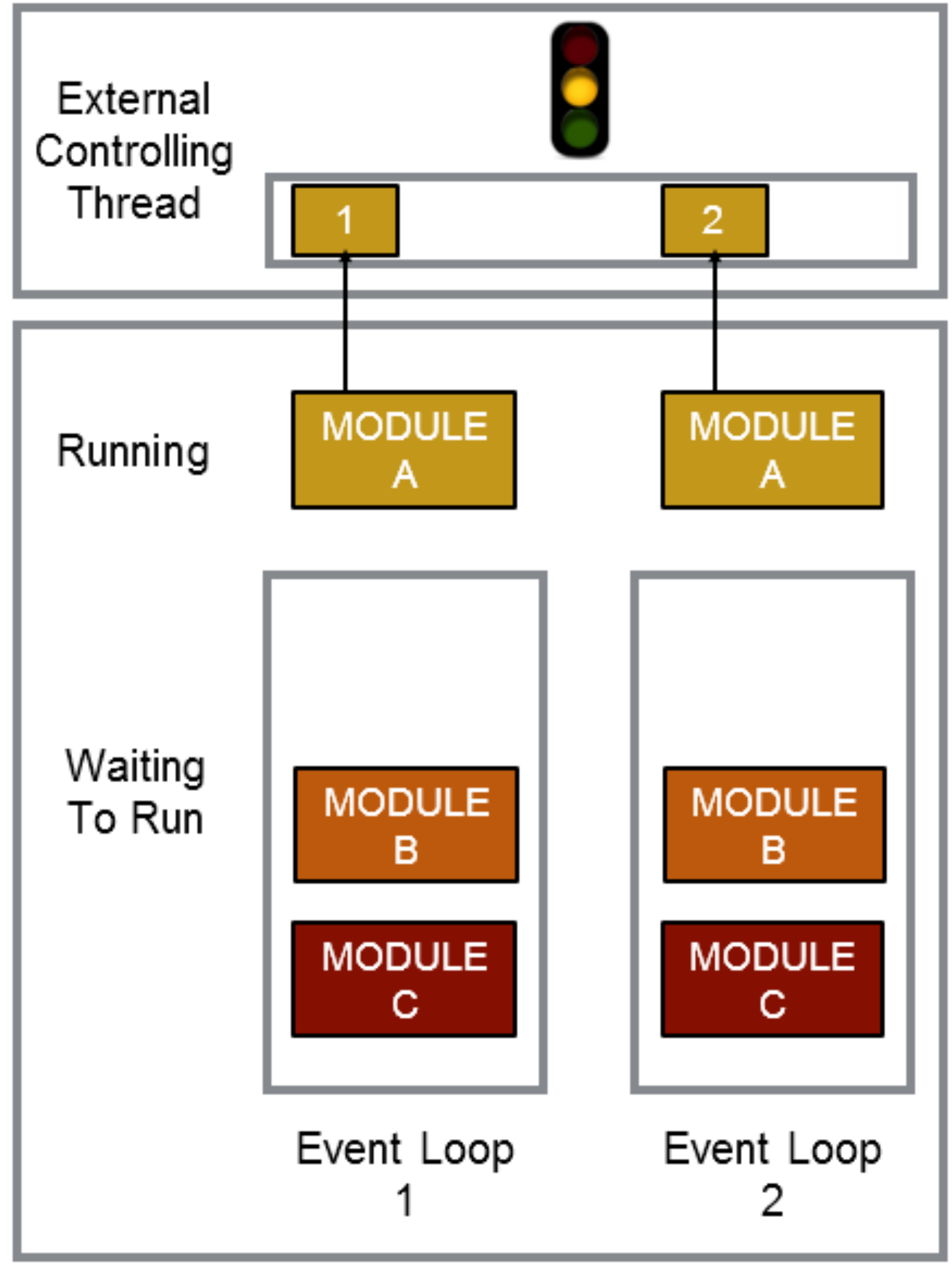
- TBB controls running modules
- Concurrent processing of multiple events
- Separate helper thread to control external
- Can wait until enough work is buffered before running external process



External Work in CMSSW (2)

Acquire:

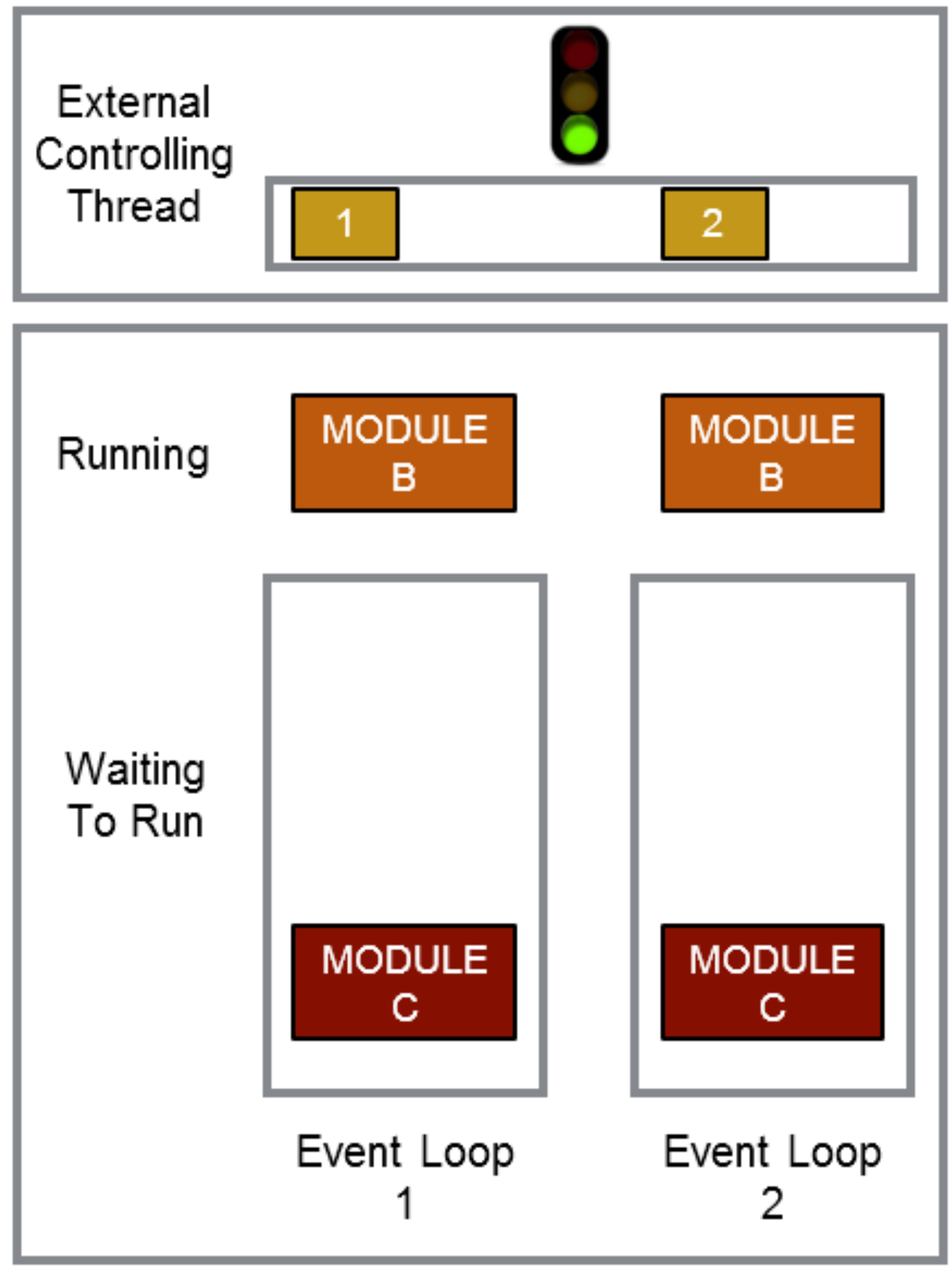
- Module *acquire*() method called
- Pulls data from event
- Copies data to buffer
- Buffer includes callback to start next phase of module running



External Work in CMSSW (3)

Work starts:

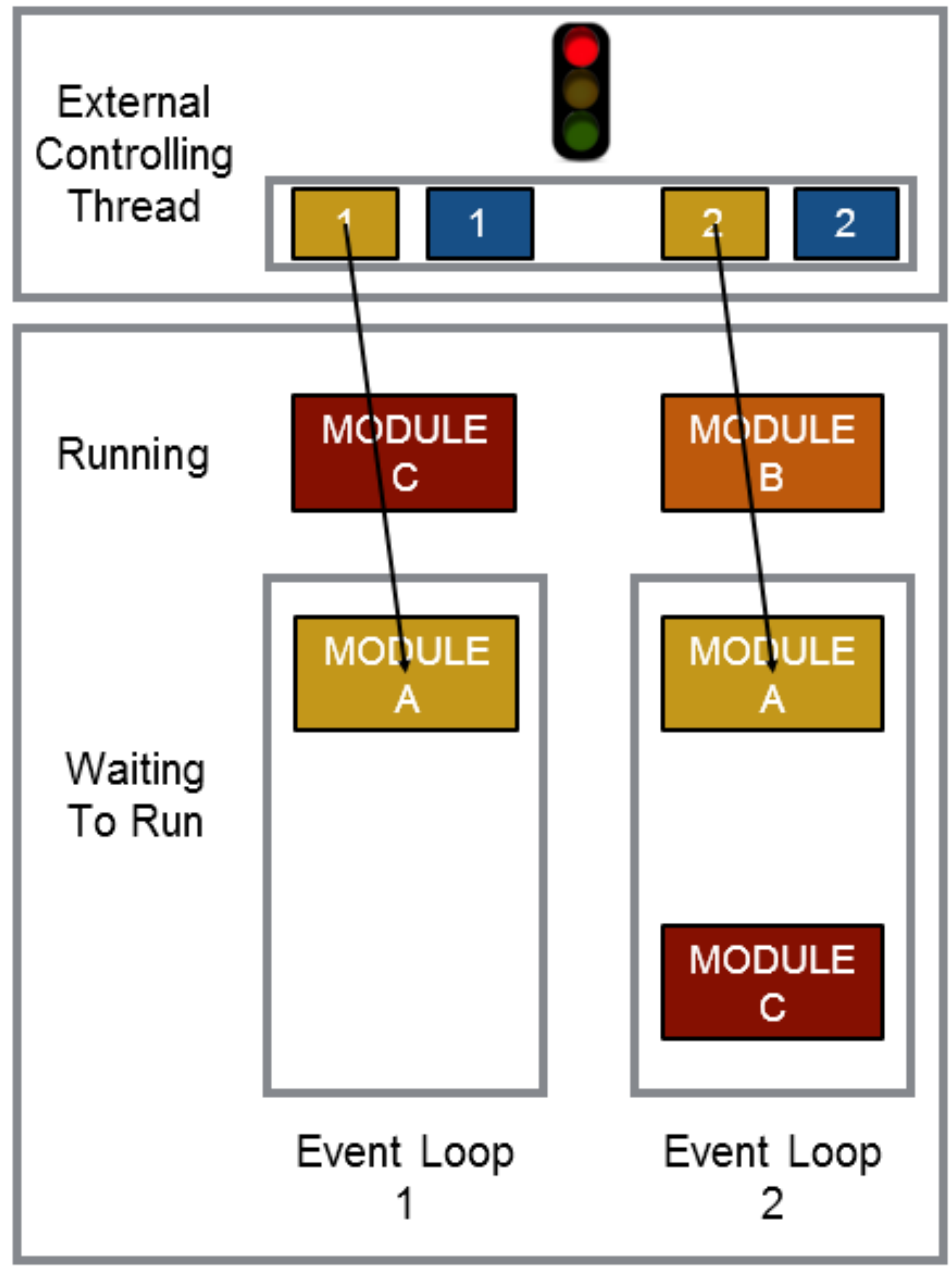
- External process runs
- Data pulled from buffer
- Next waiting modules can run (concurrently)



External Work in CMSSW (4)

Work finishes:

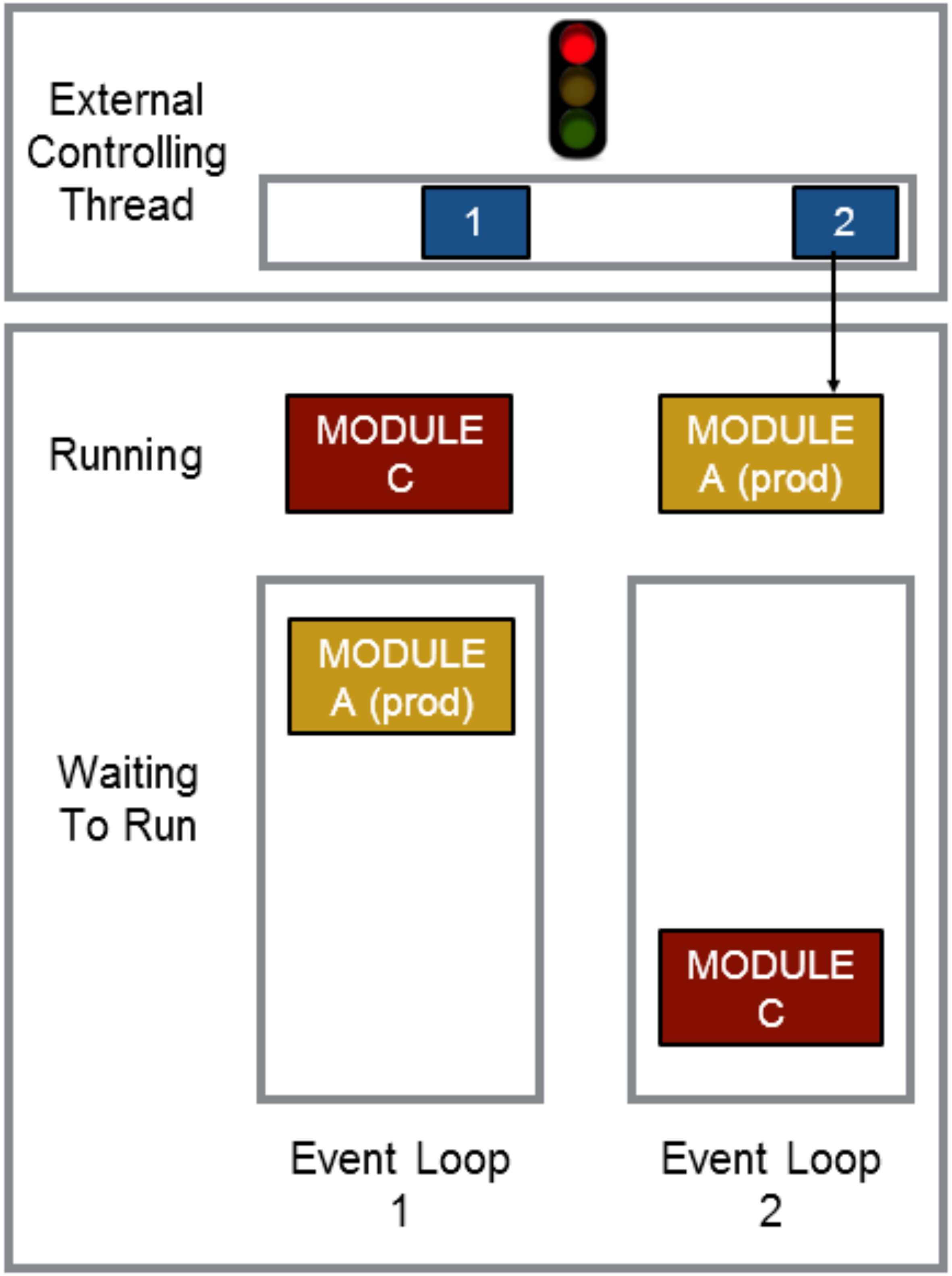
- Results copied to buffer
- Callback puts module back into queue



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