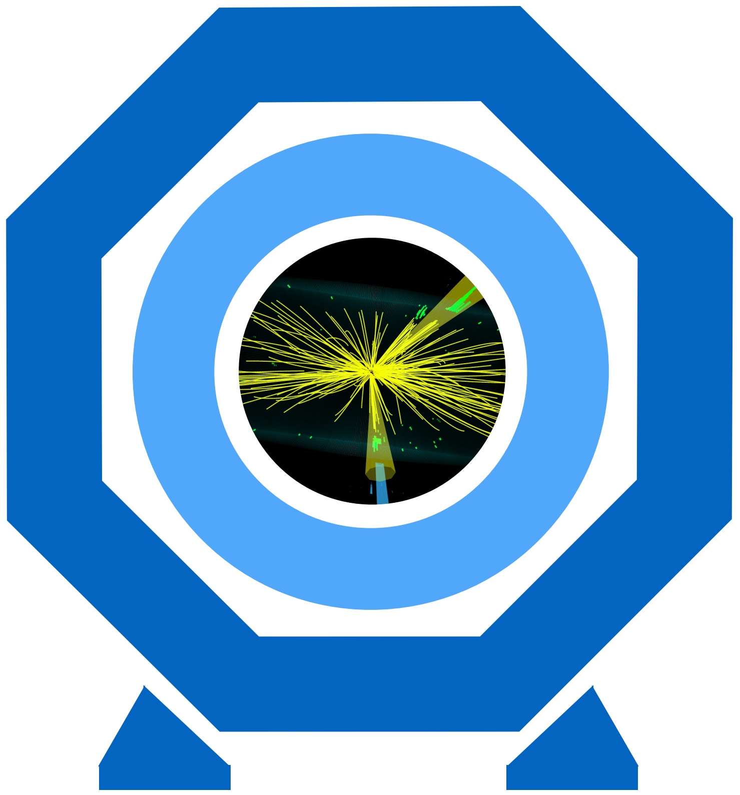
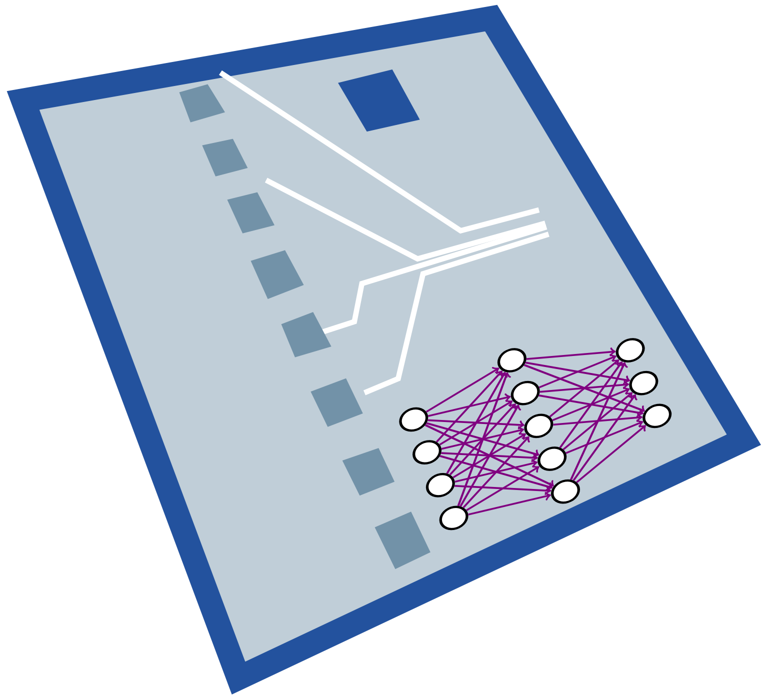
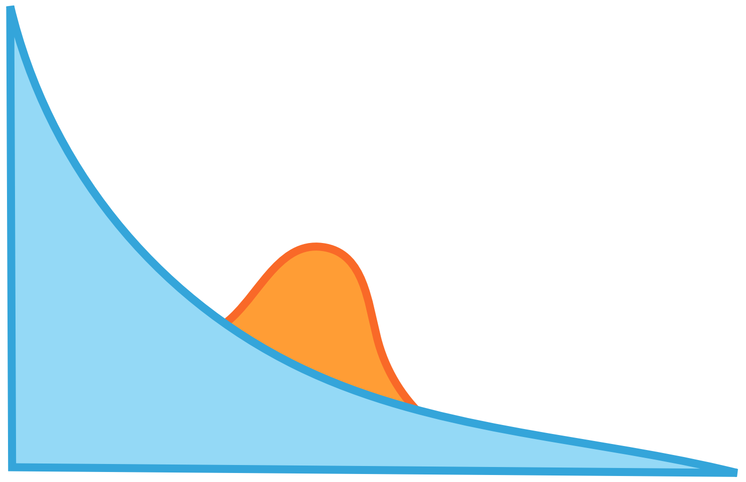




JAVIER DUARTE (UCSD)

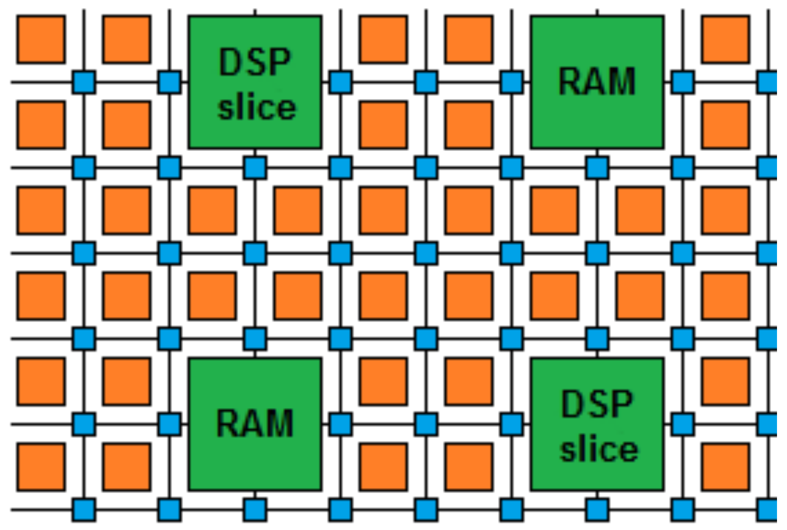
OCTOBER 23, 2019

WEST COAST LHC JAMBOREE, SLAC

NEW IDEAS & TECHNOLOGIES IN TRIGGER FOR HL-LHC (AND RUN 3)

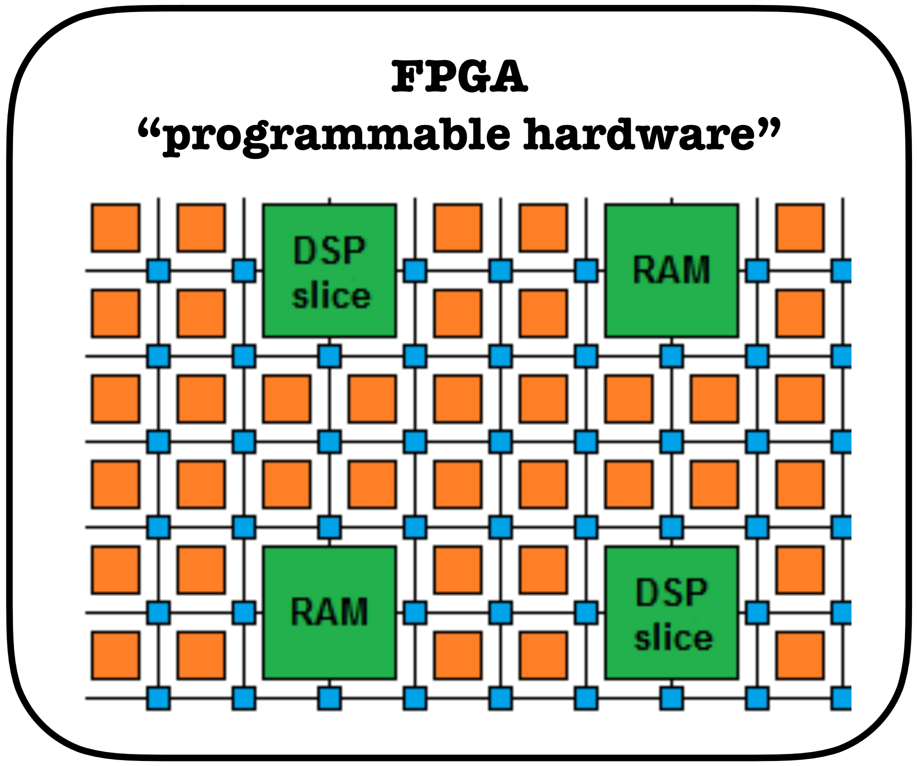
NEW IDEAS AND TECHNOLOGIES FOR TRIGGER

FPGA
“programmable hardware”

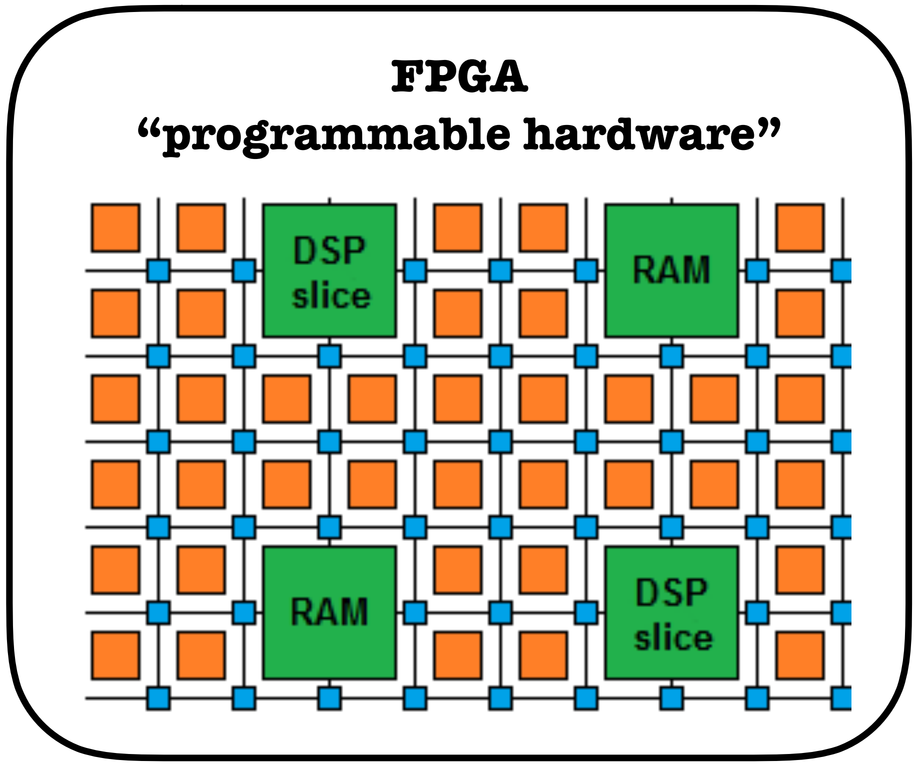


NEW IDEAS AND TECHNOLOGIES FOR TRIGGER

- **Modern FPGAs** with large amounts of embedded components that perform multiplication (DSPs), apply logical functions (LUTs), or store memory (BRAM)

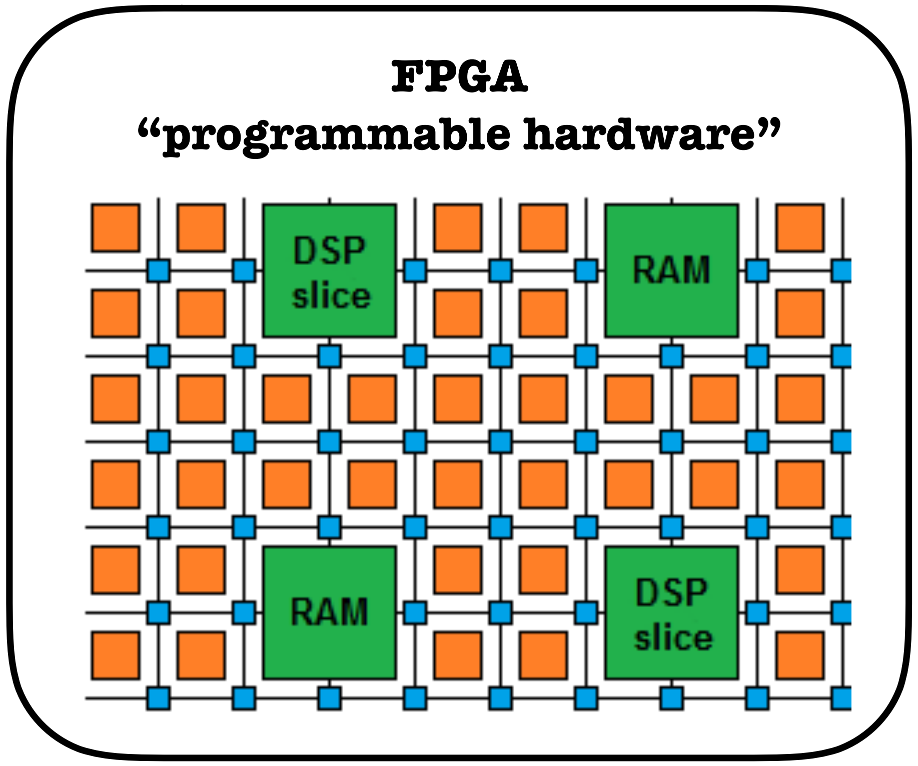


NEW IDEAS AND TECHNOLOGIES FOR TRIGGER



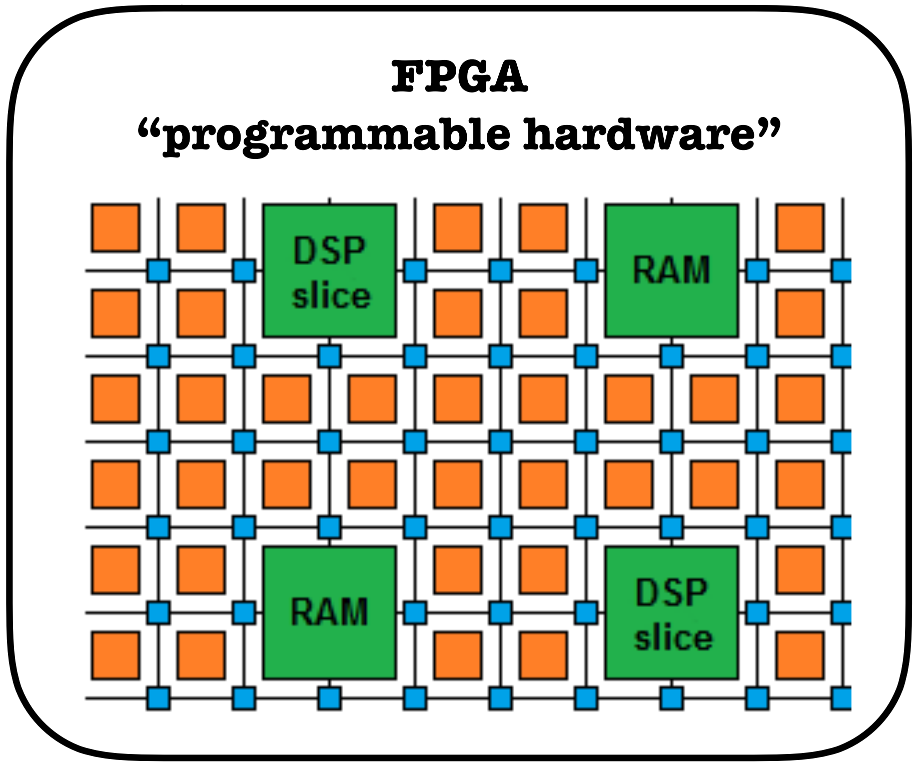
- **Modern FPGAs** with large amounts of embedded components that perform multiplication (DSPs), apply logical functions (LUTs), or store memory (BRAM)
- **High level synthesis** to more easily program FPGAs

NEW IDEAS AND TECHNOLOGIES FOR TRIGGER



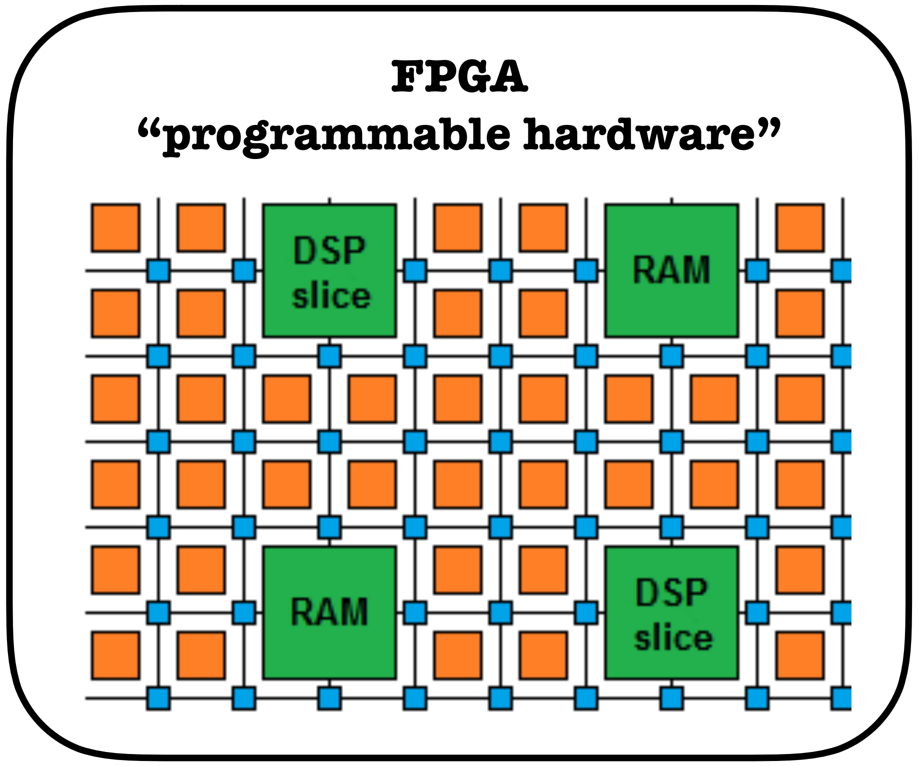
- **Modern FPGAs** with large amounts of embedded components that perform multiplication (DSPs), apply logical functions (LUTs), or store memory (BRAM)
- **High level synthesis** to more easily program FPGAs
- **Sophisticated algorithms**

NEW IDEAS AND TECHNOLOGIES FOR TRIGGER



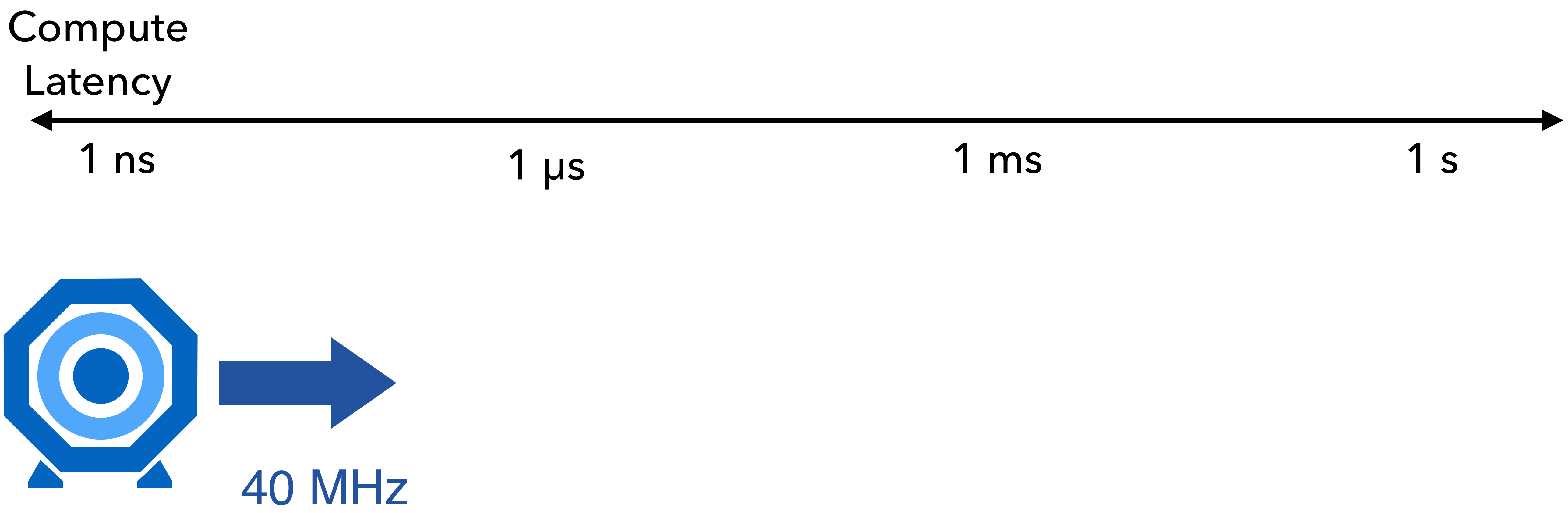
- **Modern FPGAs** with large amounts of embedded components that perform multiplication (DSPs), apply logical functions (LUTs), or store memory (BRAM)
- **High level synthesis** to more easily program FPGAs
- **Sophisticated algorithms**
- **Machine learning**

NEW IDEAS AND TECHNOLOGIES FOR TRIGGER



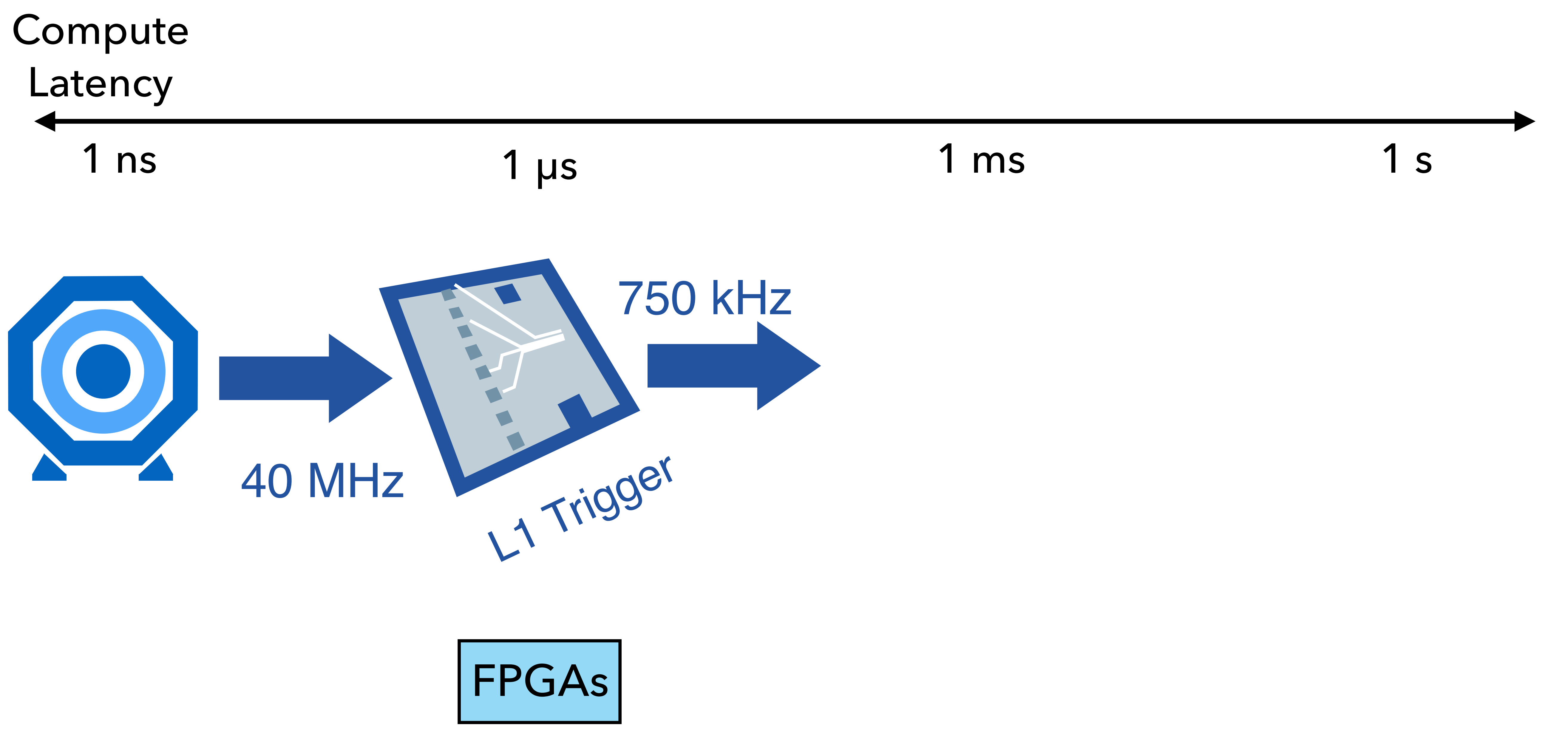
- **Modern FPGAs** with large amounts of embedded components that perform multiplication (DSPs), apply logical functions (LUTs), or store memory (BRAM)
- **High level synthesis** to more easily program FPGAs
- **Sophisticated algorithms**
- **Machine learning**
- GPUs or FPGAs or ASICs as **co-processors** for software trigger

LHC EVENT PROCESSING



Challenges:
Each collision produces $O(10^3)$ particles
The detectors have $O(10^8)$ sensors
Extreme data rates of O(100 TB/s)

LHC EVENT PROCESSING



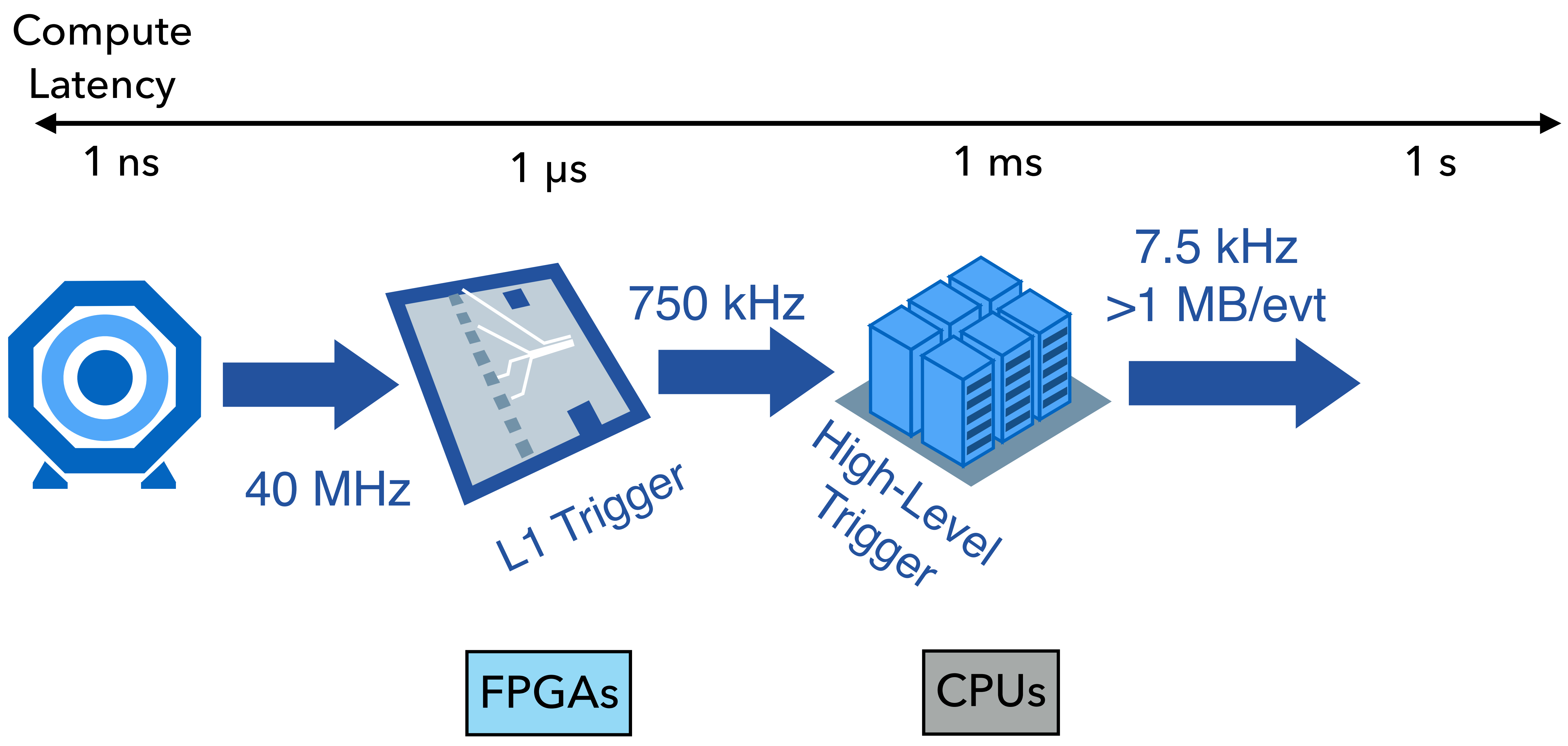
Challenges:

Each collision produces $O(10^3)$ particles

The detectors have $O(10^8)$ sensors

Extreme data rates of O(100 TB/s)

LHC EVENT PROCESSING



Challenges:

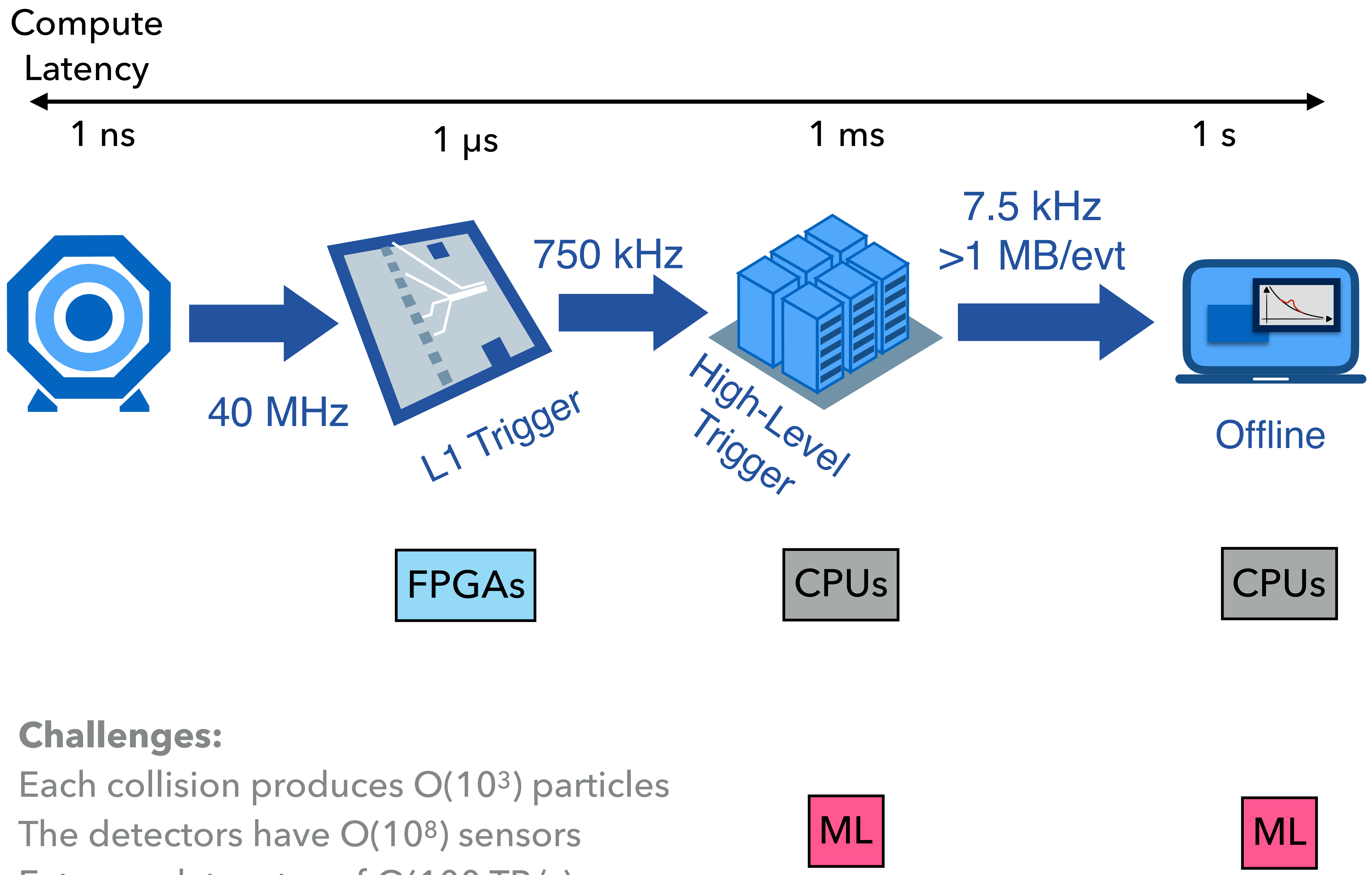
Each collision produces O(10^3) particles

The detectors have O(10^8) sensors

Extreme data rates of O(100 TB/s)

ML

LHC EVENT PROCESSING



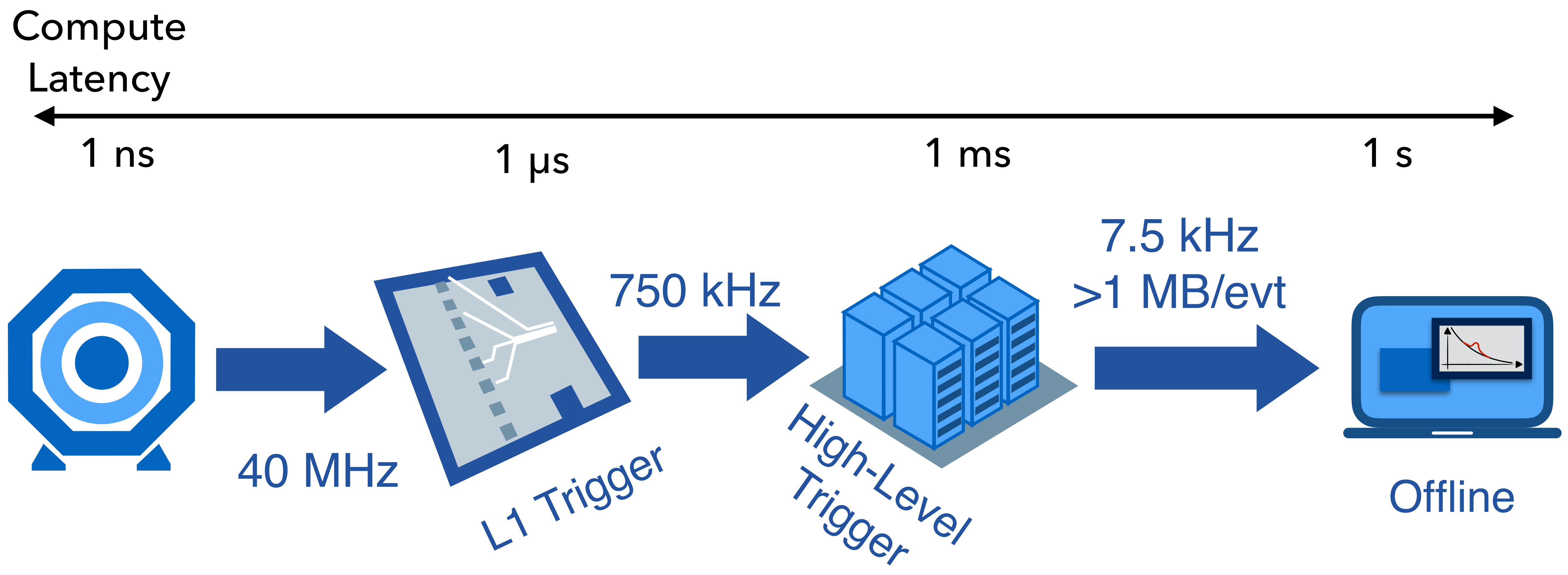
Challenges:

Each collision produces O(10^3) particles

The detectors have O(10^8) sensors

Extreme data rates of O(100 TB/s)

LHC EVENT PROCESSING



FPGAs
ML

CPUs
GPUs
FPGAs
ML

CPUs
GPUs
FPGAs
ML

Challenges:

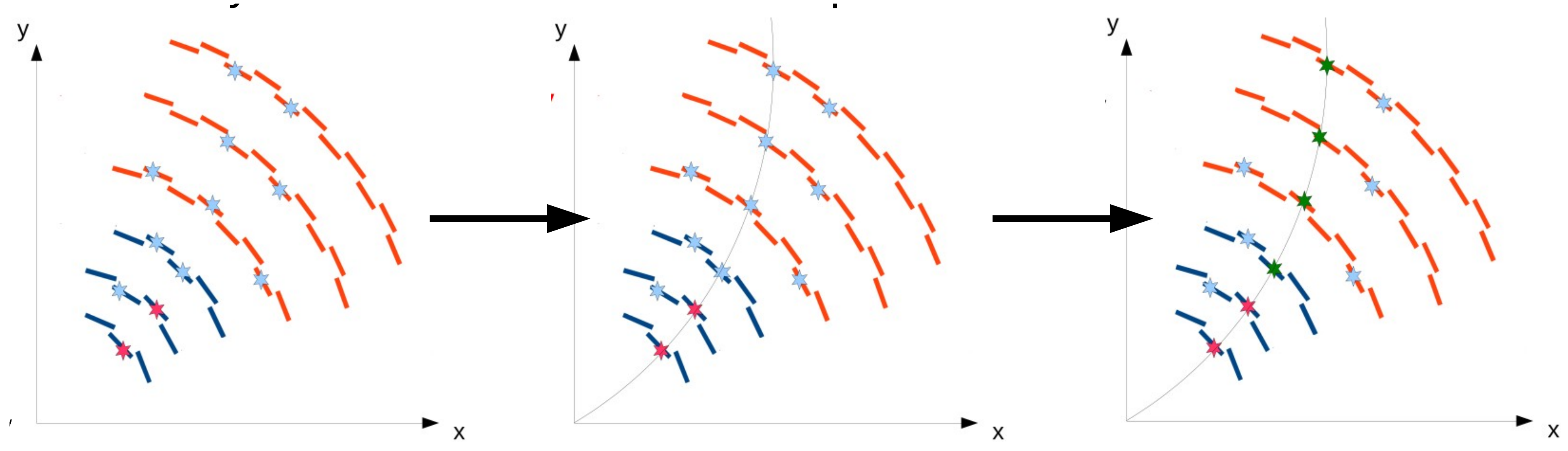
Each collision produces O(10^3) particles

The detectors have O(10^8) sensors

Extreme data rates of O(100 TB/s)

LEVEL-1 TRACK TRIGGER

- Algorithm approach: tracklet and Kalman filter hybrid algorithm written in Vivado HLS to expedite development
  - Tracks are seeded with pairs of stubs in adjacent layers
  - Projections to other layers are calculated (assuming beamline constraint)
  - Full tracks after duplicate removal are inputs to the final track fit (Kalman filter)
- R&D efforts: displaced tracking for long-lived particles, etc.



InputRouter

VMRouter

TrackletEngine

TrackletCalculator

ProjectionRouter

MatchEngine

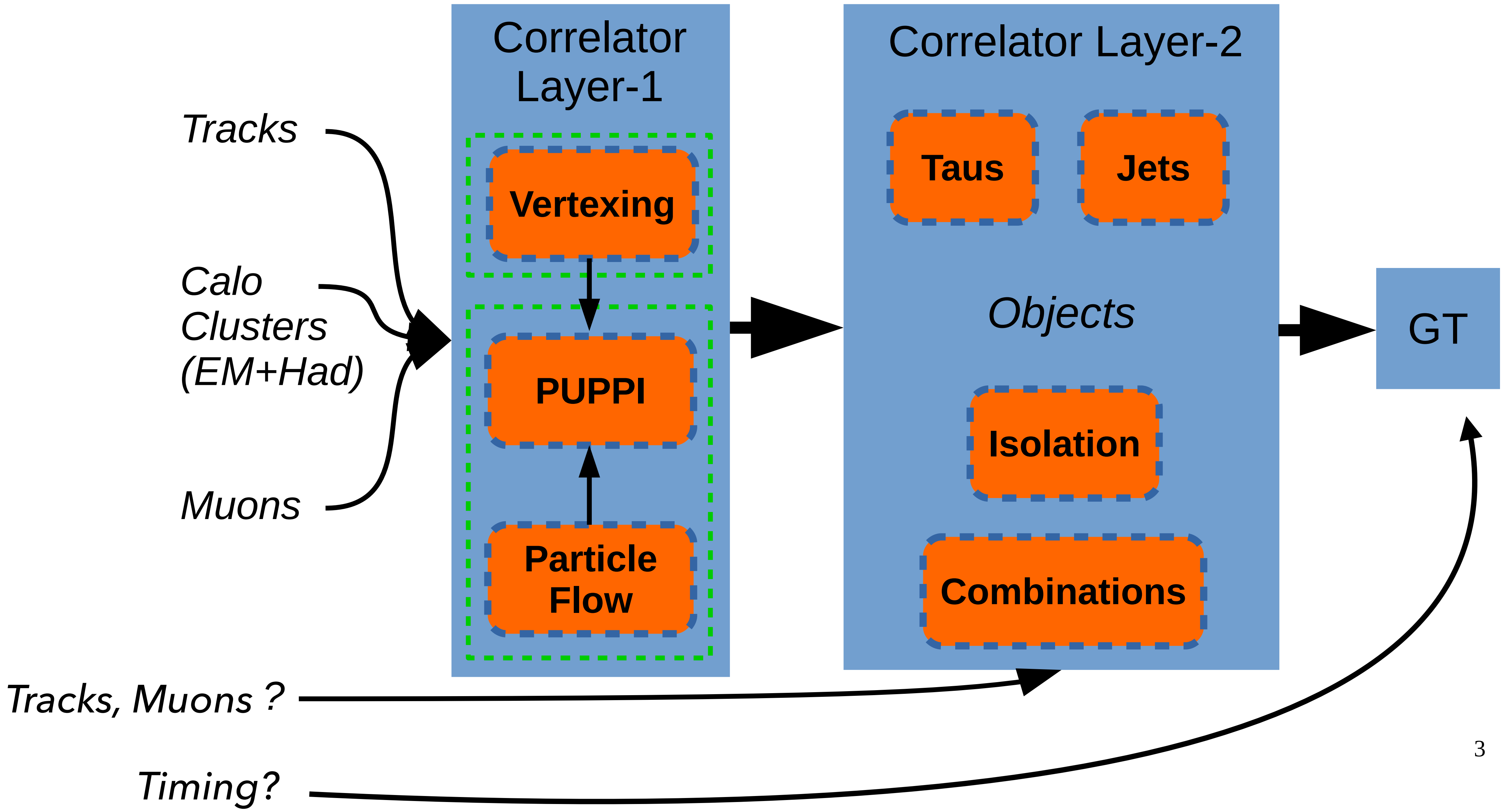
MatchCalculator

DuplicateRemoval

KalmanFilter

CORRELATOR TRIGGER

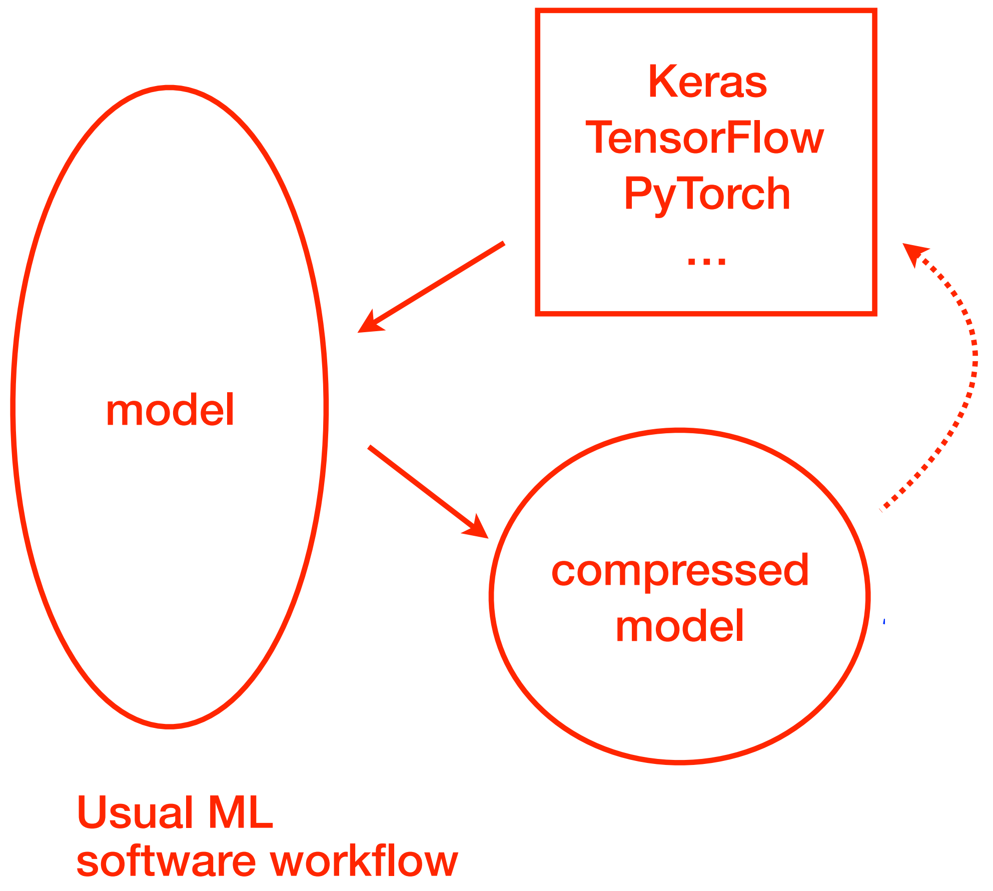
- Correlator layer 1 will process pileup mitigated candidates {μ,e,γ,h±,h0,vtx}
- Full correlator trigger must complete all processing & transmit trigger objects {μ,e,γ,τ,j,MET,etc.} to the GT within 2.5 μs



MACHINE LEARNING IN FPGAS WITH HLS4ML

JINST 13 (2018) P07027

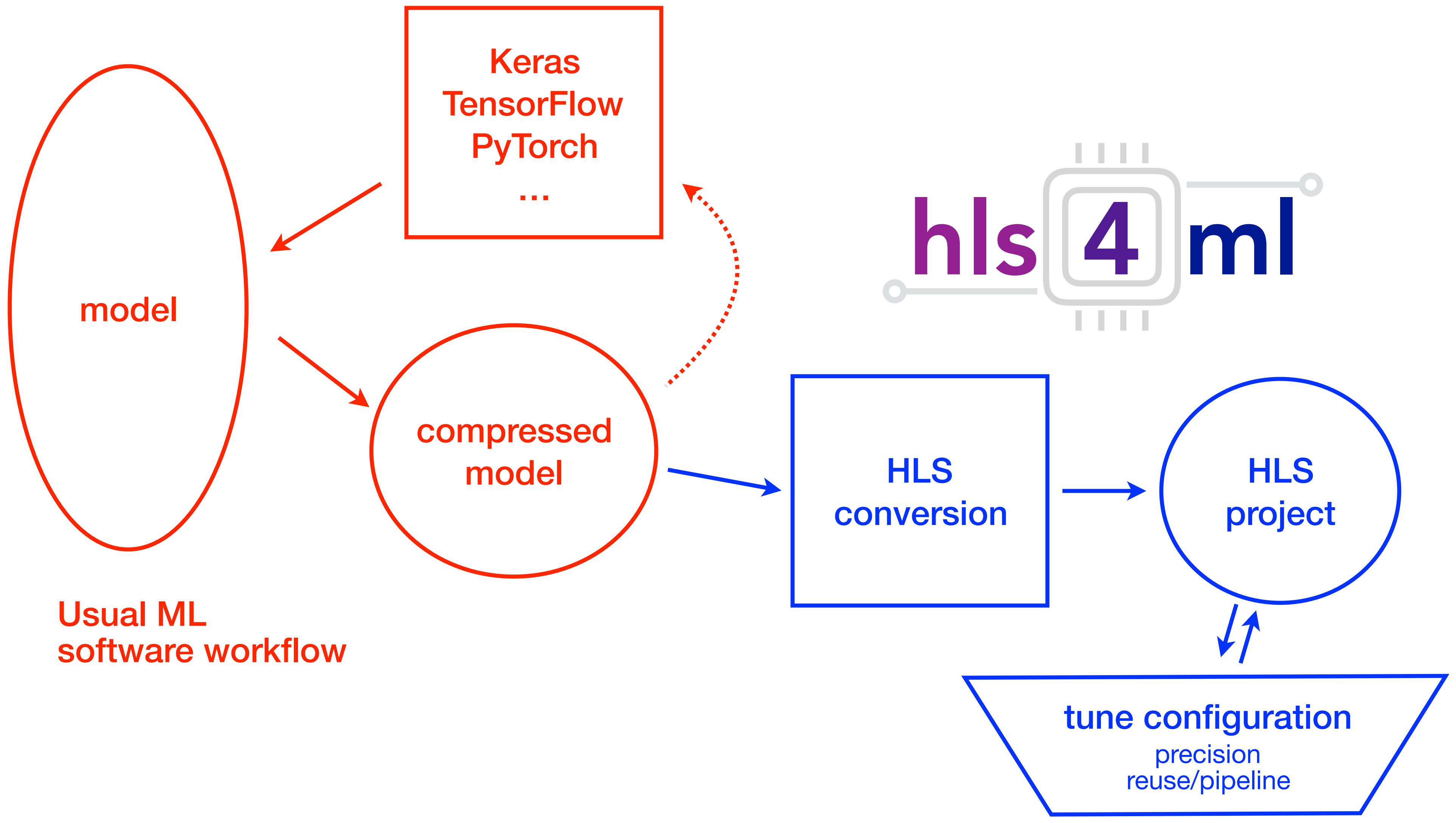
- **hls4ml** for physicists or ML experts to translate **ML algorithms** into **FPGA firmware**



MACHINE LEARNING IN FPGAS WITH HLS4ML

JINST 13 (2018) P07027

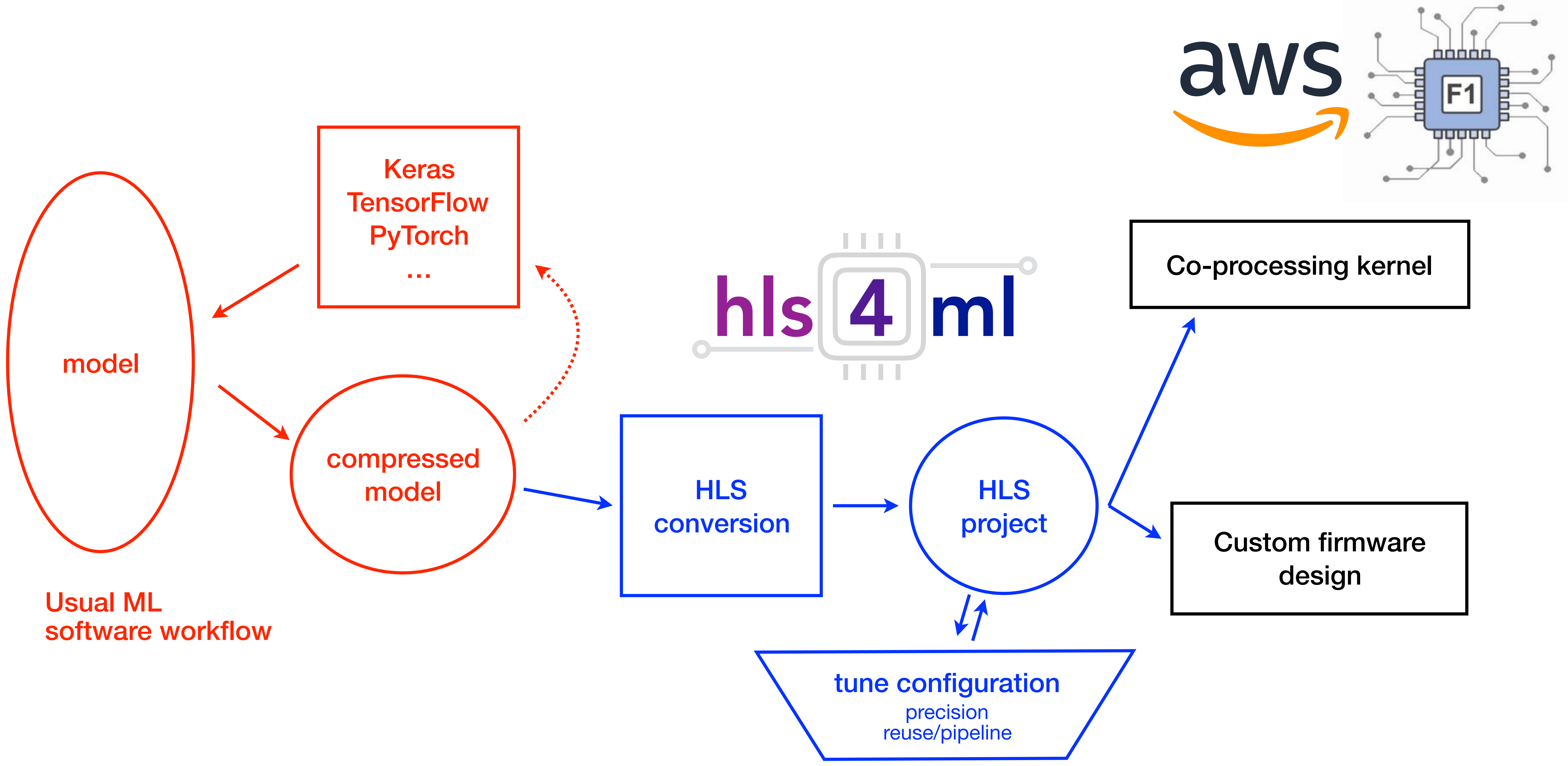
- **hls4ml** for physicists or ML experts to translate **ML algorithms** into **FPGA firmware**



MACHINE LEARNING IN FPGAS WITH HLS4ML

JINST 13 (2018) P07027

- **hls4ml** for physicists or ML experts to translate **ML algorithms** into **FPGA firmware**



TRANSLATION OF ML MODELS

TRANSLATION OF ML MODELS

Translation $\longrightarrow$

```
hls4ml convert -c keras-config.yml
```

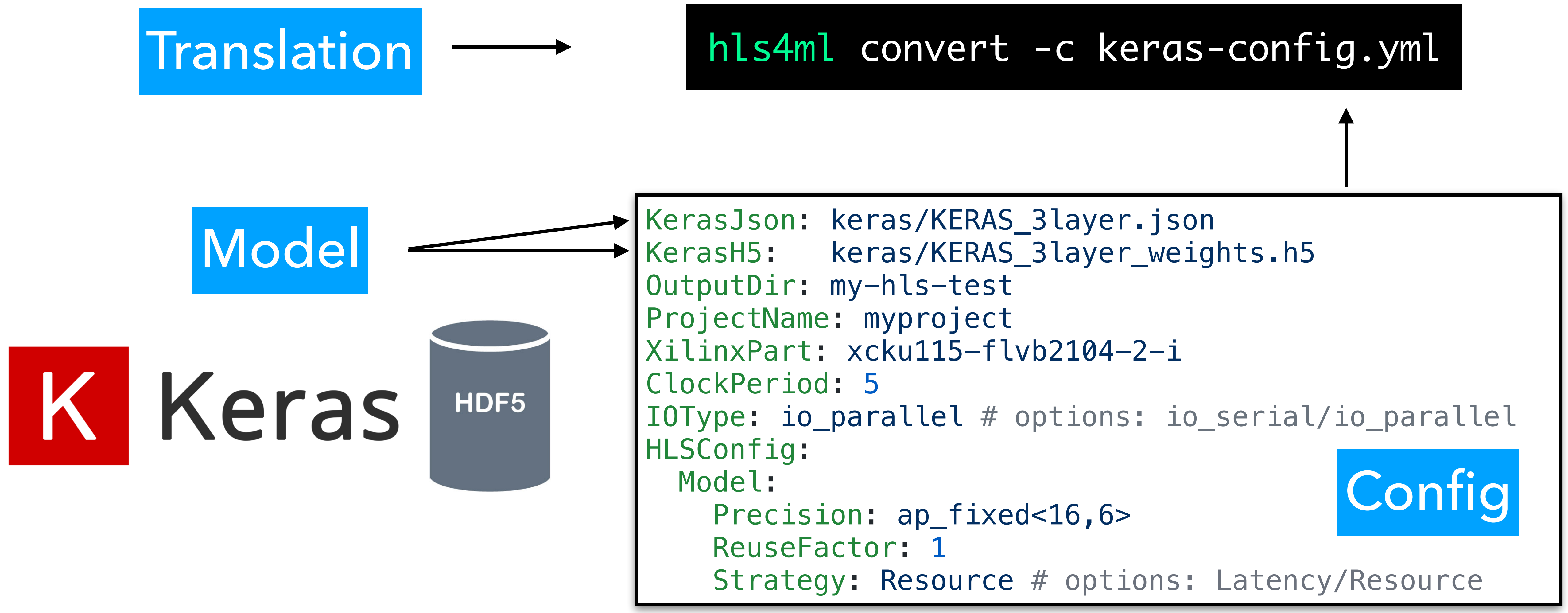
TRANSLATION OF ML MODELS

Translation →

```
hls4ml convert -c keras-config.yml
```

```
KerasJson: keras/KERAS_3layer.json
KerasH5:   keras/KERAS_3layer_weights.h5
OutputDir: my-hls-test
ProjectName: myproject
XilinxPart: xcku115-flvb2104-2-i
ClockPeriod: 5
IOType: io_parallel # options: io_serial/io_parallel
HLSConfig:
  Model:
    Precision: ap_fixed<16,6>
    ReuseFactor: 1
    Strategy: Resource # options: Latency/Resource
```

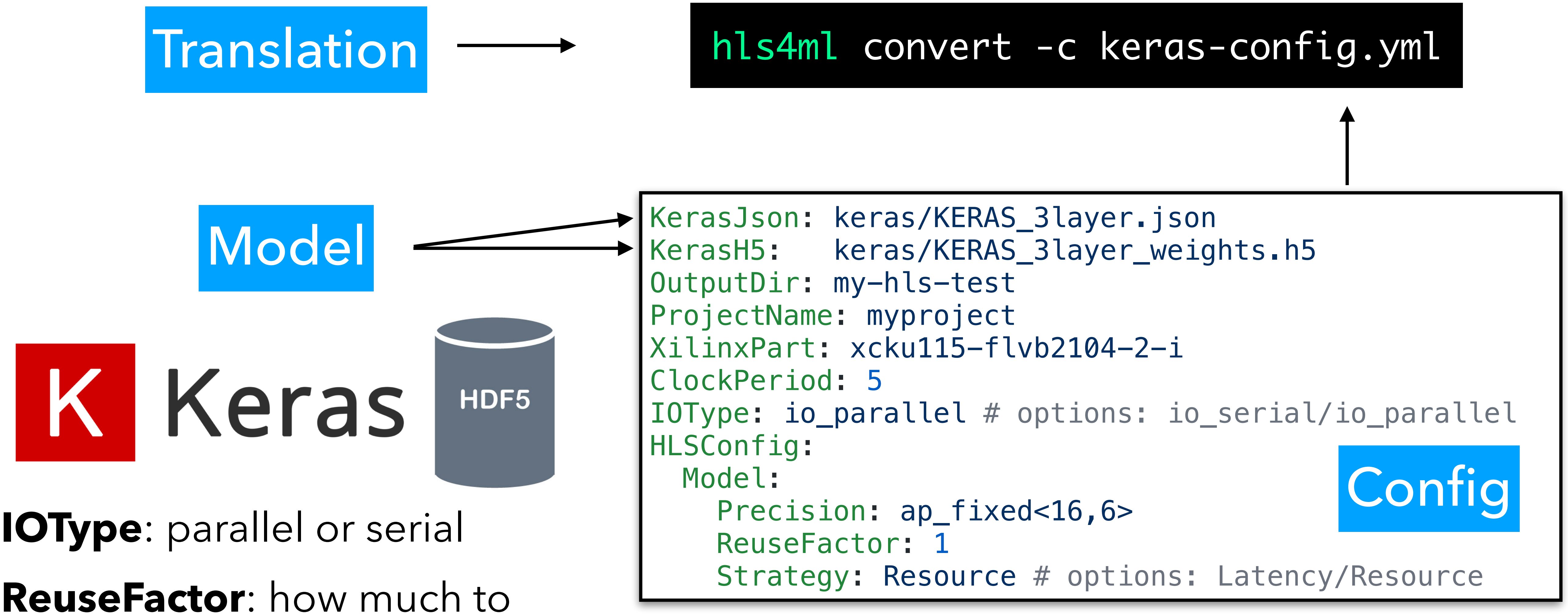
TRANSLATION OF ML MODELS



```
hls4ml convert -c keras-config.yml
```

```
KerasJson: keras/KERAS_3layer.json
KerasH5:   keras/KERAS_3layer_weights.h5
OutputDir: my-hls-test
ProjectName: myproject
XilinxPart: xcku115-flvb2104-2-i
ClockPeriod: 5
IOType: io_parallel # options: io_serial/io_parallel
HLSConfig:
  Model:
    Precision: ap_fixed<16,6>
    ReuseFactor: 1
    Strategy: Resource # options: Latency/Resource
```

TRANSLATION OF ML MODELS



```
hls4ml convert -c keras-config.yml
```

```
KerasJson: keras/KERAS_3layer.json
KerasH5:   keras/KERAS_3layer_weights.h5
OutputDir: my-hls-test
ProjectName: myproject
XilinxPart: xcku115-flvb2104-2-i
ClockPeriod: 5
IOType: io_parallel # options: io_serial/io_parallel
HLSConfig:
  Model:
    Precision: ap_fixed<16,6>
    ReuseFactor: 1
    Strategy: Resource # options: Latency/Resource
```

- **IOType**: parallel or serial
- **ReuseFactor**: how much to parallelize
- **Precision**: inputs, weights, biases
- **Strategy**:
  - Resource for large NN
  - Latency for small NN (fully pipelined)

TRANSLATION OF ML MODELS

Translation

```
hls4ml convert -c keras-config.yml
```

Model

```
KerasJson: keras/KERAS_3layer.json
KerasH5:   keras/KERAS_3layer_weights.h5
OutputDir: my-hls-test
ProjectName: myproject
XilinxPart: xcku115-flvb2104-2-i
ClockPeriod: 5
IOType: io_parallel # options: io_serial/io_parallel
HLSConfig:
  Model:
    Precision: ap_fixed<16,6>
    ReuseFactor: 1
    Strategy: Resource # options: Latency/Resource
```

Config

Keras

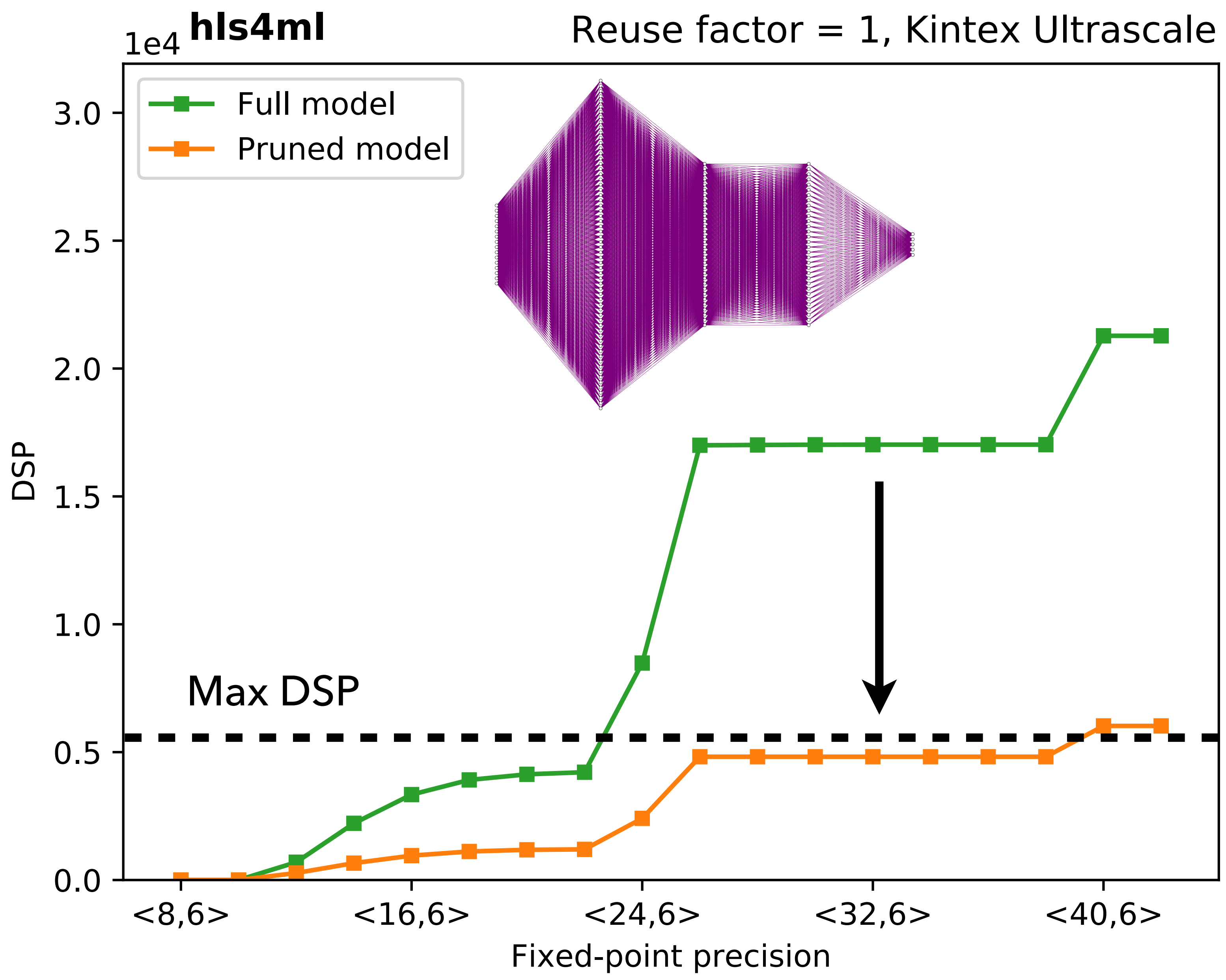
HDF5

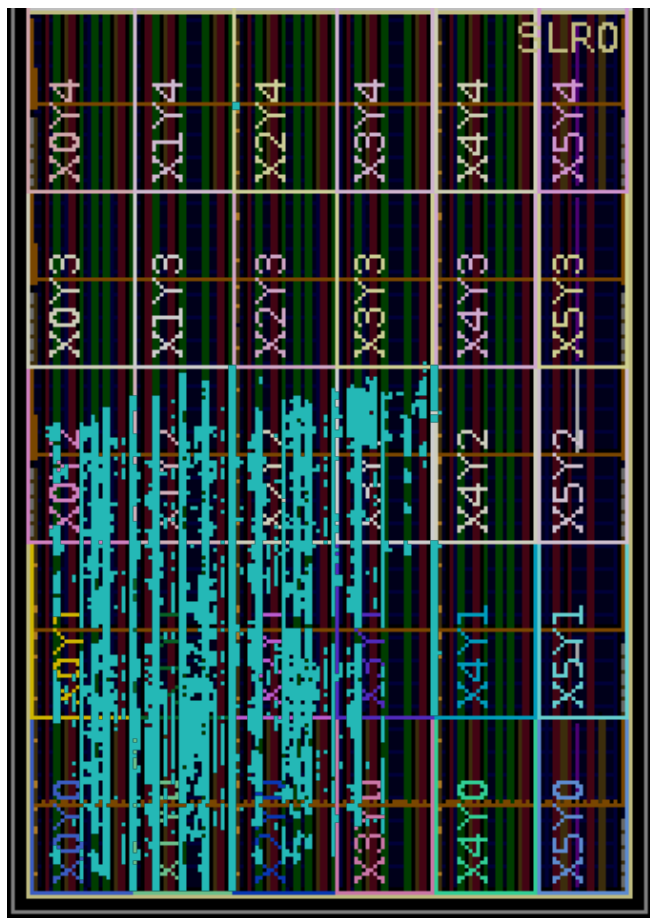
- **IOType**: parallel or serial
- **ReuseFactor**: how much to parallelize
- **Precision**: inputs, weights, biases
- **Strategy**:
  - Resource for large NN
  - Latency for small NN (fully pipelined)

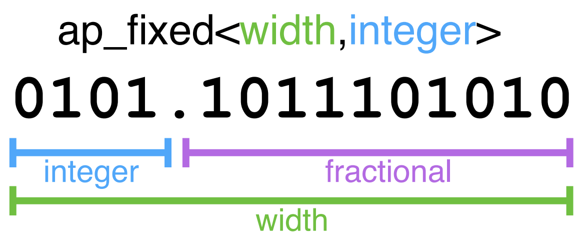
Build HLS project

```
hls4ml build -p my-hls-test -a
```

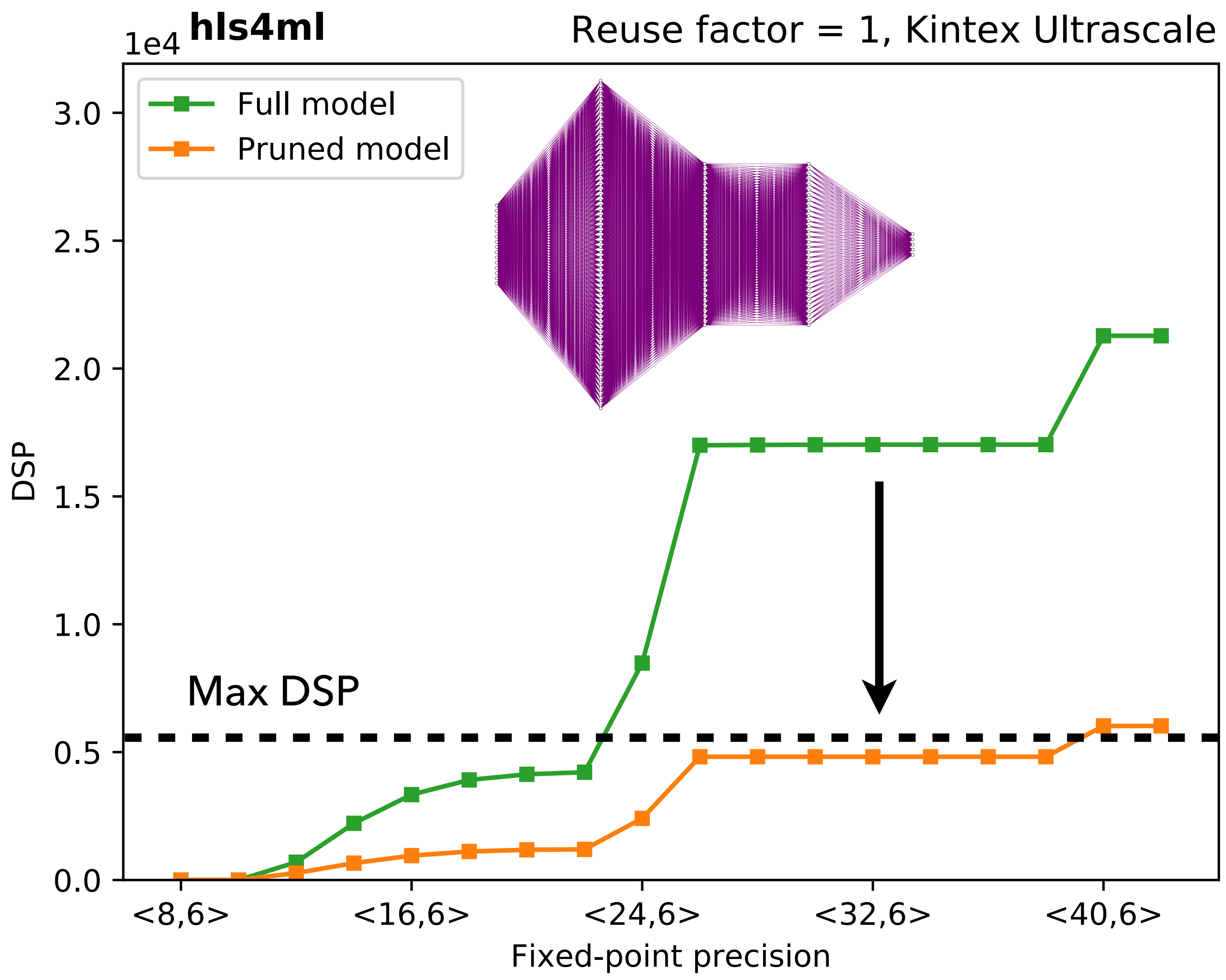
NETWORK TUNING: COMPRESSION & RESOURCES

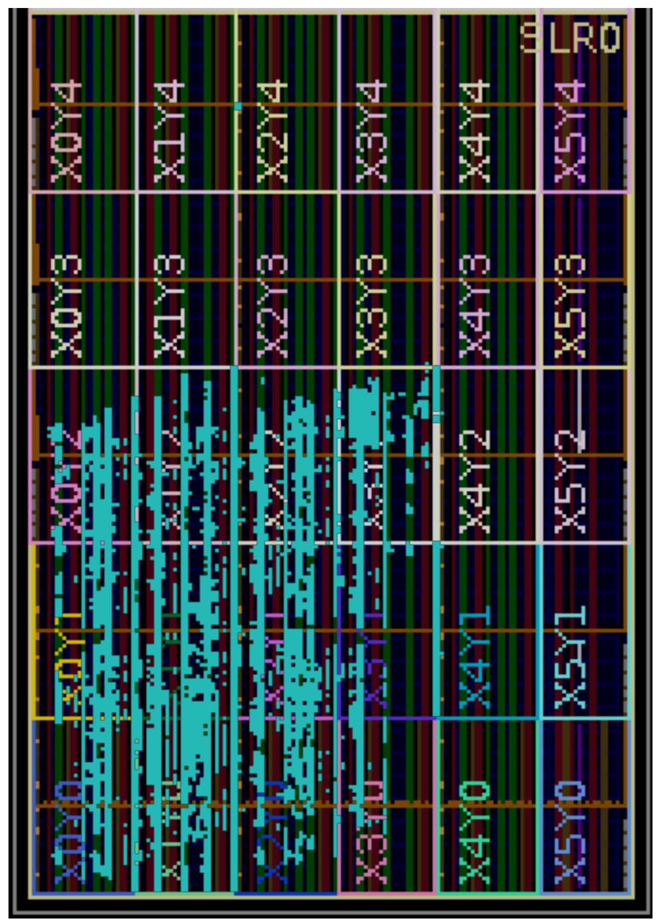


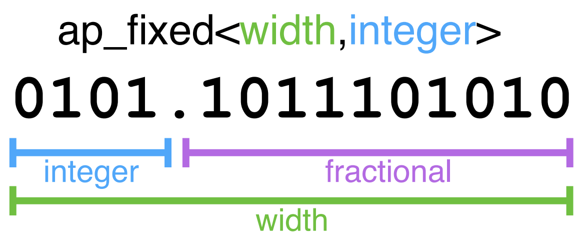




NETWORK TUNING: COMPRESSION & RESOURCES

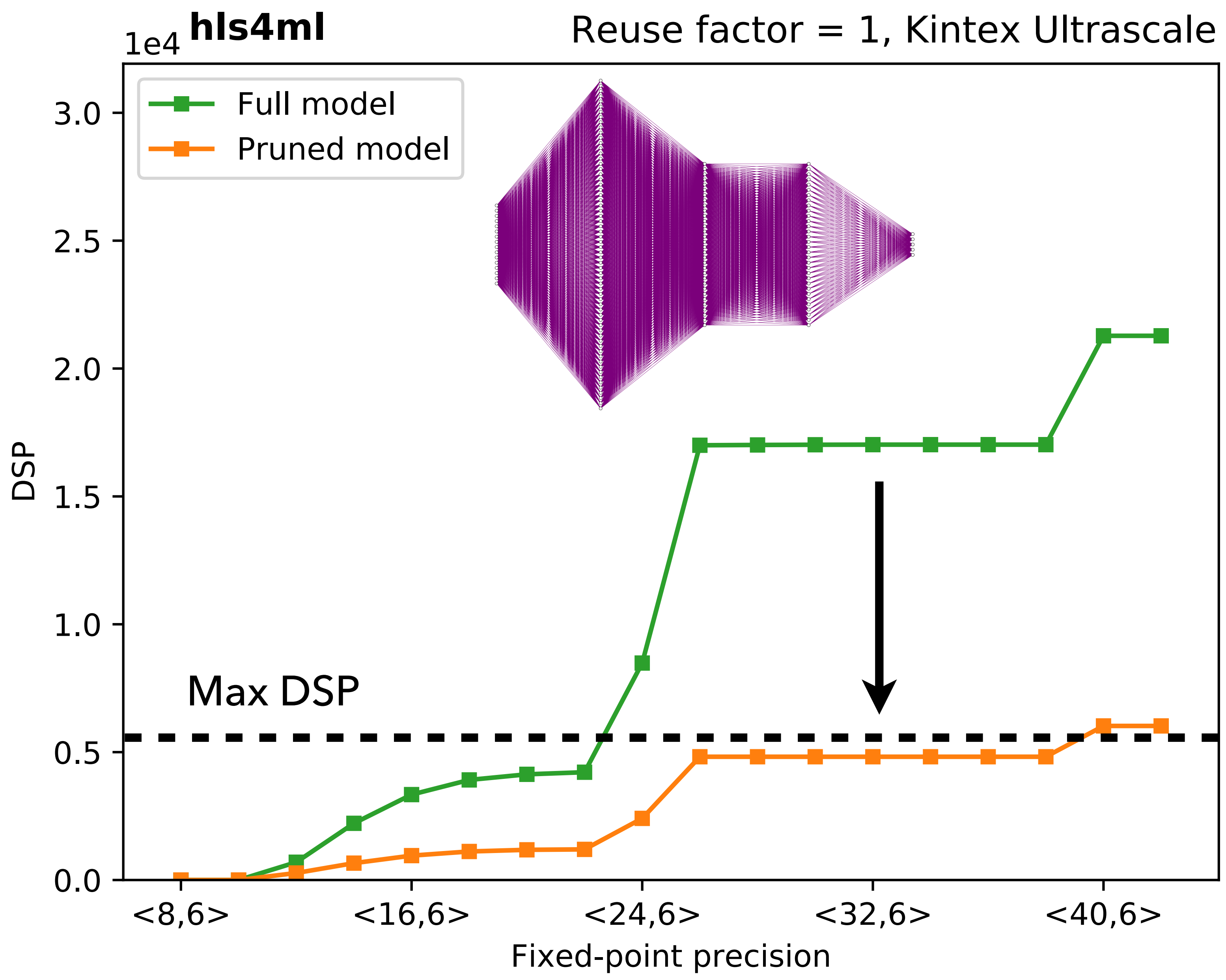


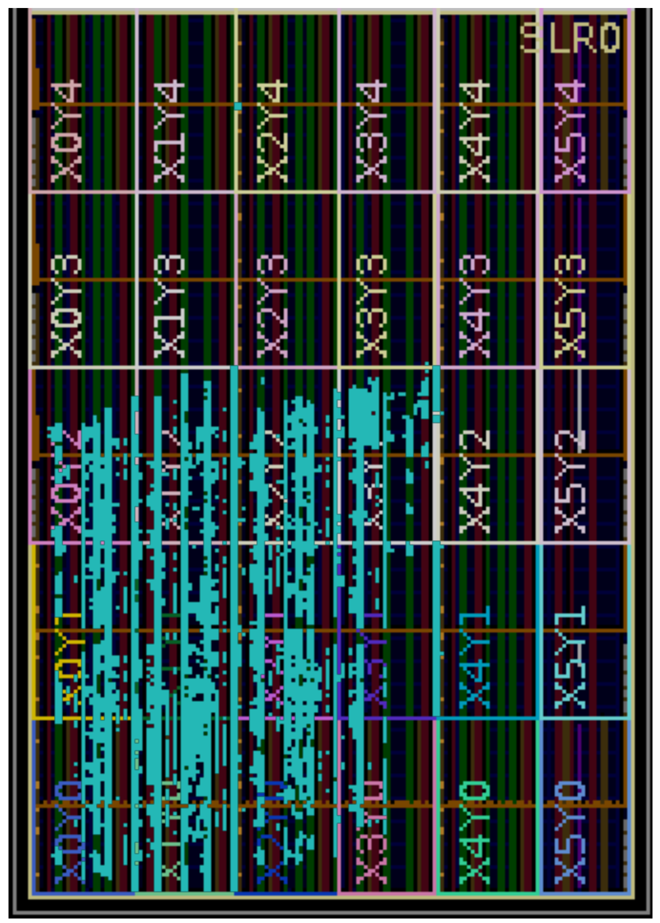


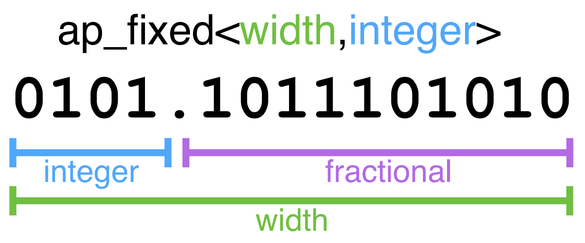


- Big reduction in DSPs (multipliers) with compression

NETWORK TUNING: COMPRESSION & RESOURCES



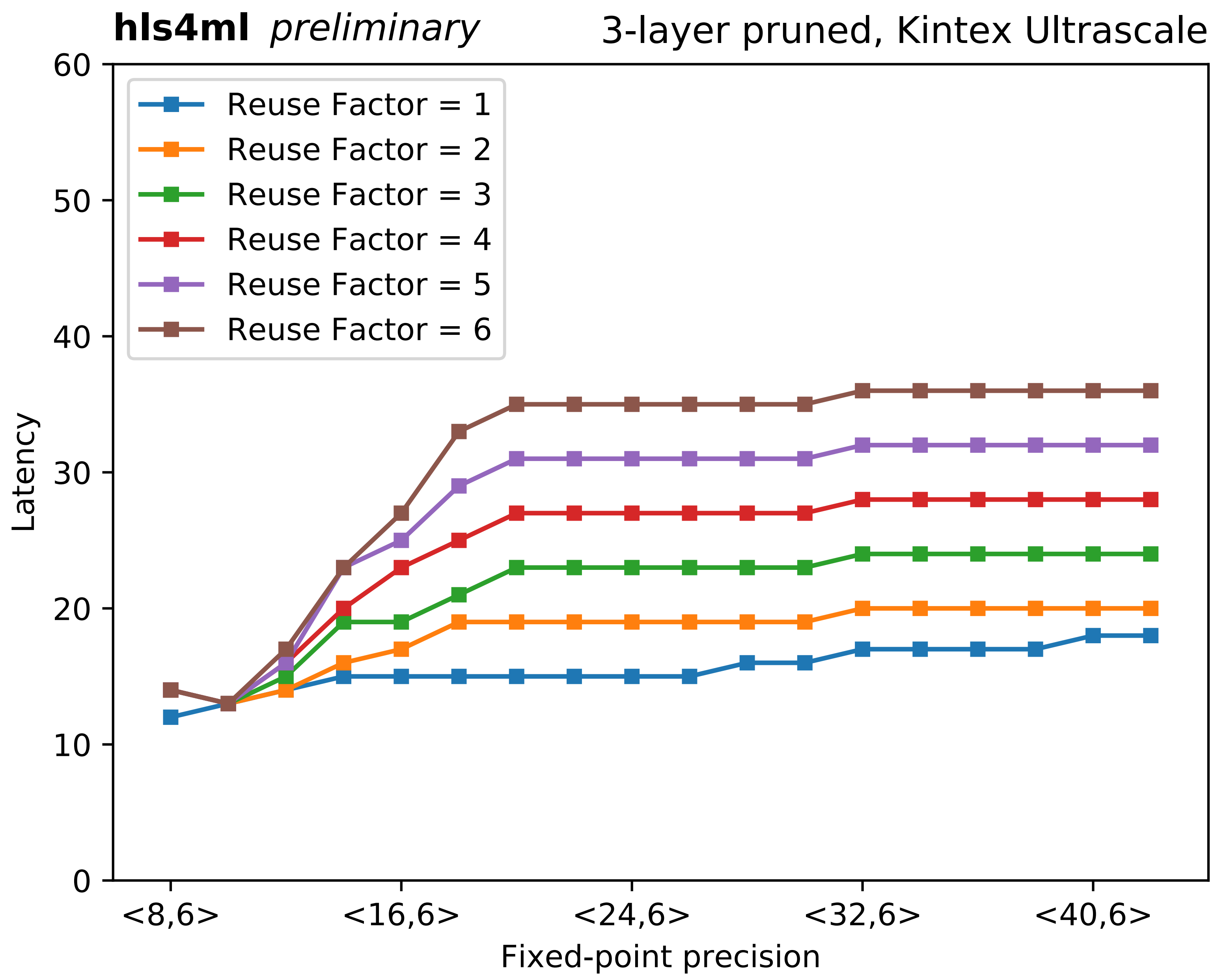




- Big reduction in DSPs (multipliers) with compression
- Easily fits on 1 FPGA **after compression**

NETWORK TUNING: PARALLELIZATION & TIMING

- **Increasing** reuse factor, **increases** latency

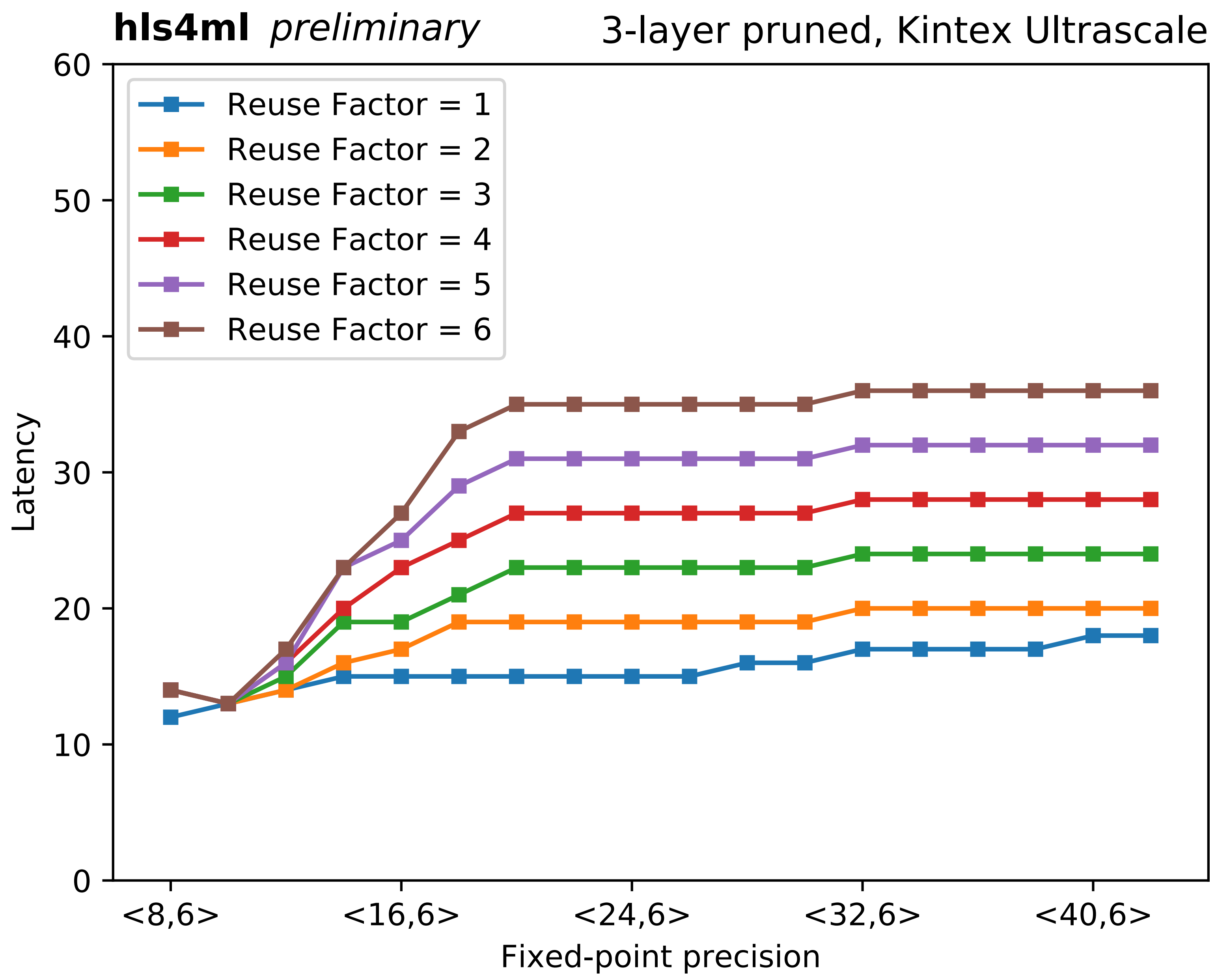


~35 clocks
@ 200 MHz
= 175 ns

~15 clocks
@ 200 MHz
= 75 ns

NETWORK TUNING: PARALLELIZATION & TIMING

- **Increasing** reuse factor, **increases** latency



~35 clocks
@ 200 MHz
= 175 ns

~15 clocks
@ 200 MHz
= 75 ns

For low-latency, small reuse factor, inference in O(100 ns)!
What if we have O(ms)? Can go to **bigger networks!**

MACHINE LEARNING IN FPGAS

MACHINE LEARNING IN FPGAS

- Inference of ML algorithms possible in **O(100 ns)** on **1 FPGA** with **hls4ml**!

MACHINE LEARNING IN FPGAS

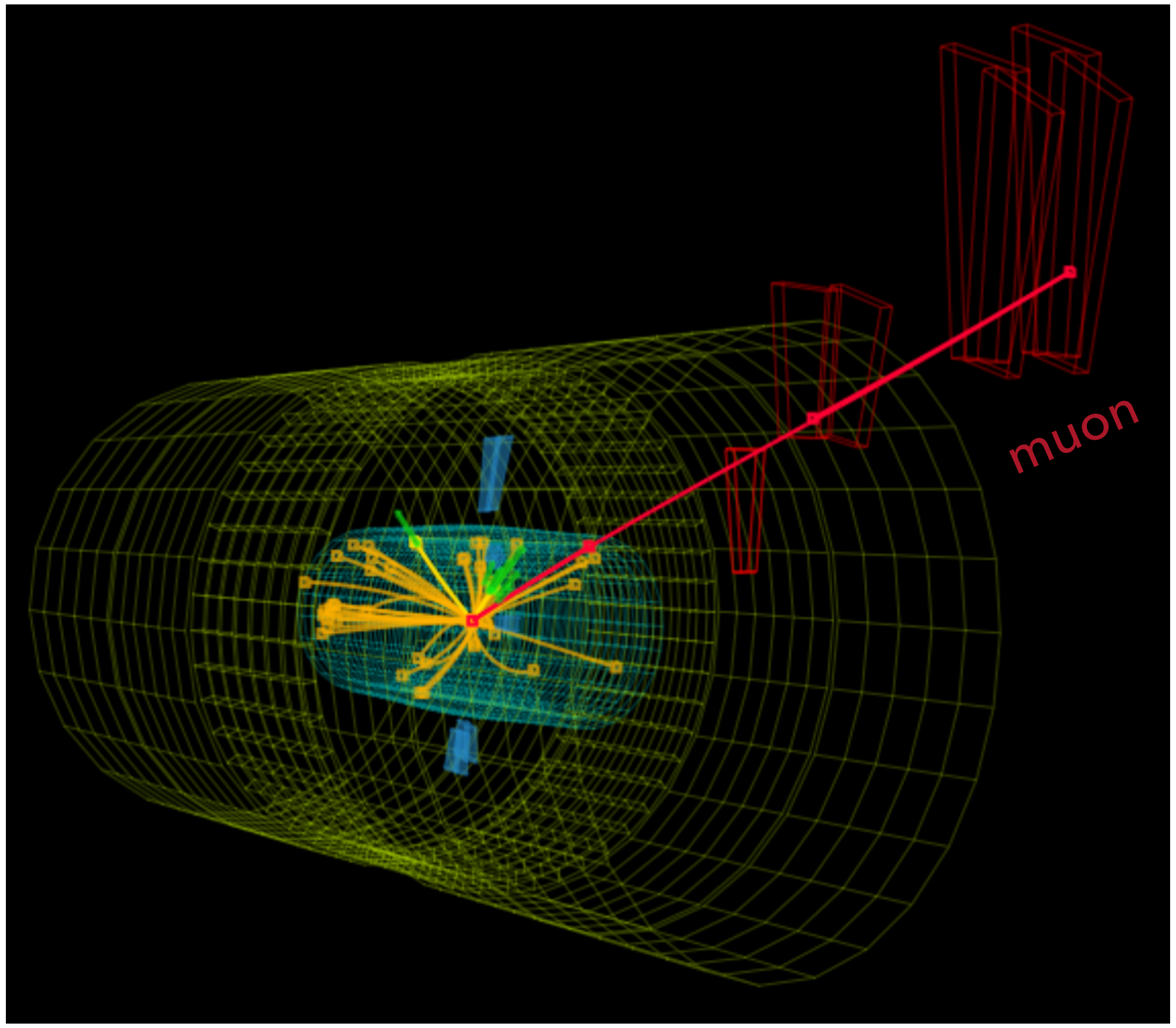
- Inference of ML algorithms possible in **O(100 ns)** on **1 FPGA** with **hls4ml**!
  - Applications across CMS, ATLAS, DUNE, and accelerator controls:

MACHINE LEARNING IN FPGAS

- Inference of ML algorithms possible in **O(100 ns)** on **1 FPGA** with **hls4ml**!
  - Applications across CMS, ATLAS, DUNE, and accelerator controls:
    - Muon p_T determination in the CMS endcap with a DNN: **runs in 160 ns** on an FPGA and **reduces** the **fake muon rate** by **up to 80%**

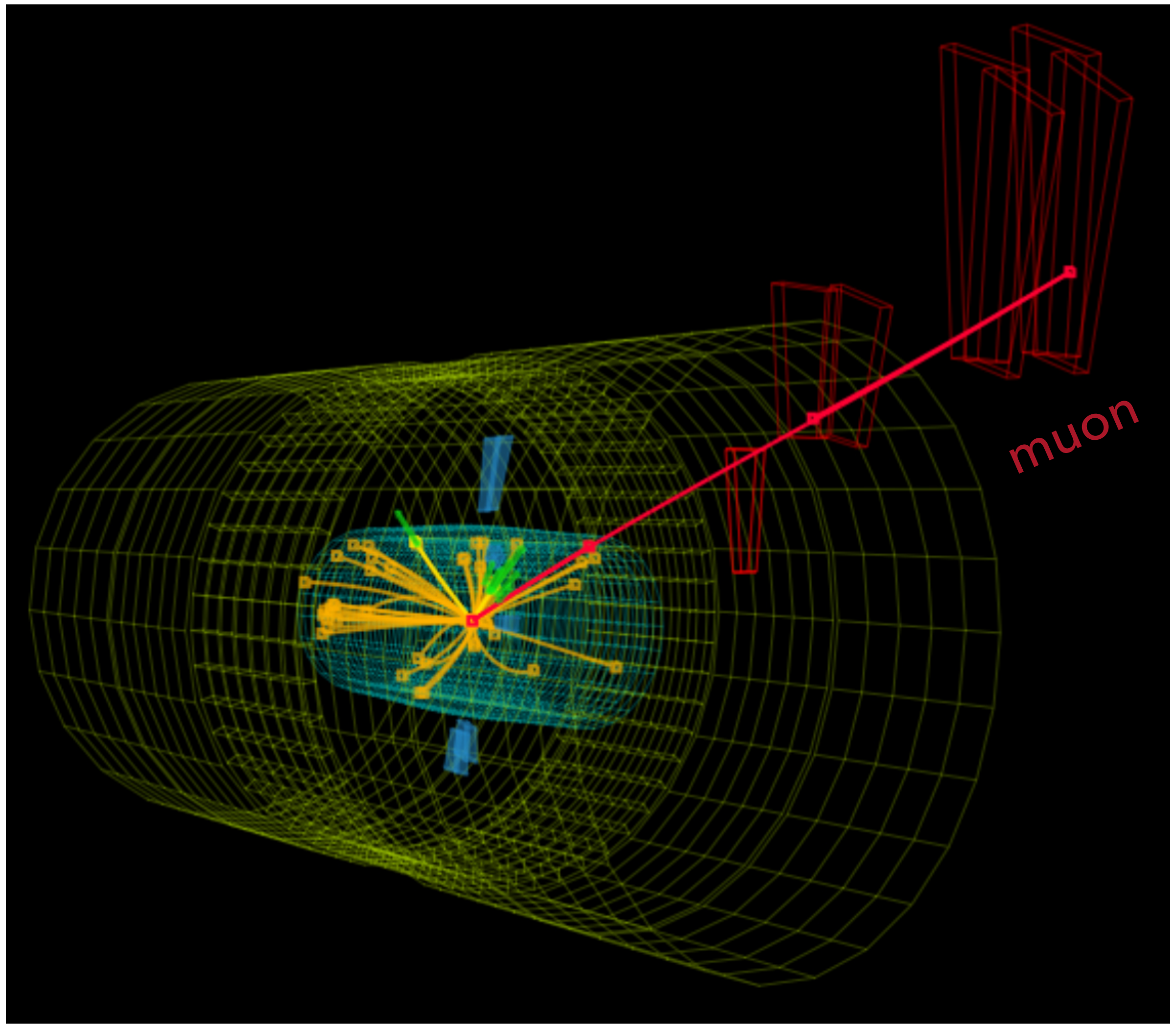
MACHINE LEARNING IN FPGAS

- Inference of ML algorithms possible in **O(100 ns)** on **1 FPGA** with **hls4ml**!
  - Applications across CMS, ATLAS, DUNE, and accelerator controls:
    - Muon p_T determination in the CMS endcap with a DNN: **runs in 160 ns** on an FPGA and **reduces** the **fake muon rate** by **up to 80%**
    - **Variational autoencoder** for anomaly detection



MACHINE LEARNING IN FPGAS

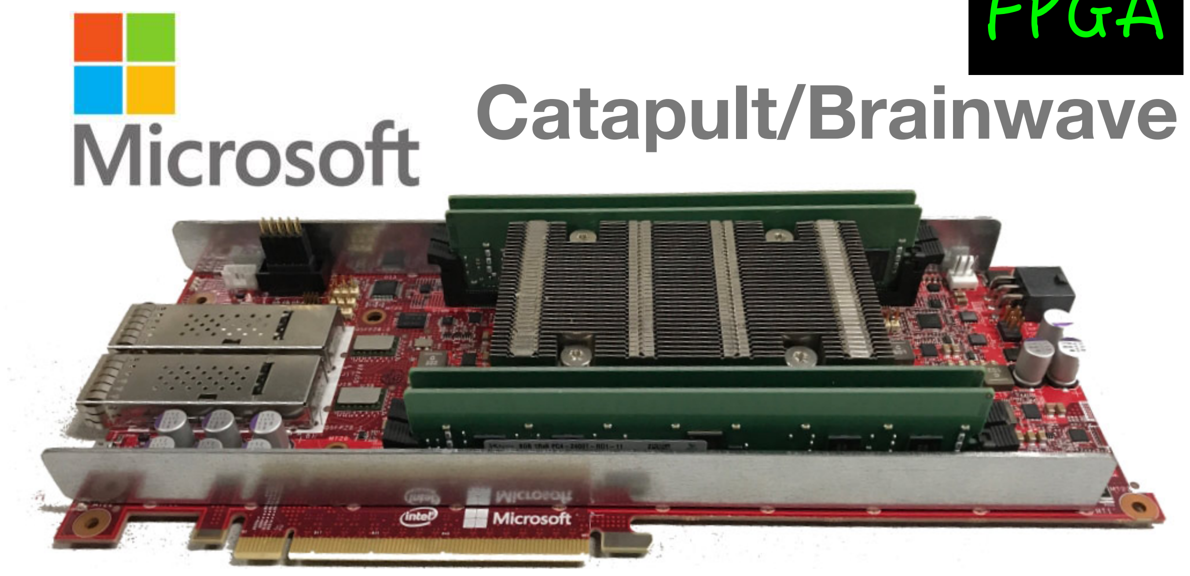
- Inference of ML algorithms possible in **O(100 ns)** on **1 FPGA** with **hls4ml**!
  - Applications across CMS, ATLAS, DUNE, and accelerator controls:
    - Muon p_T determination in the CMS endcap with a DNN: **runs in 160 ns** on an FPGA and **reduces** the **fake muon rate** by **up to 80%**
    - **Variational autoencoder** for anomaly detection

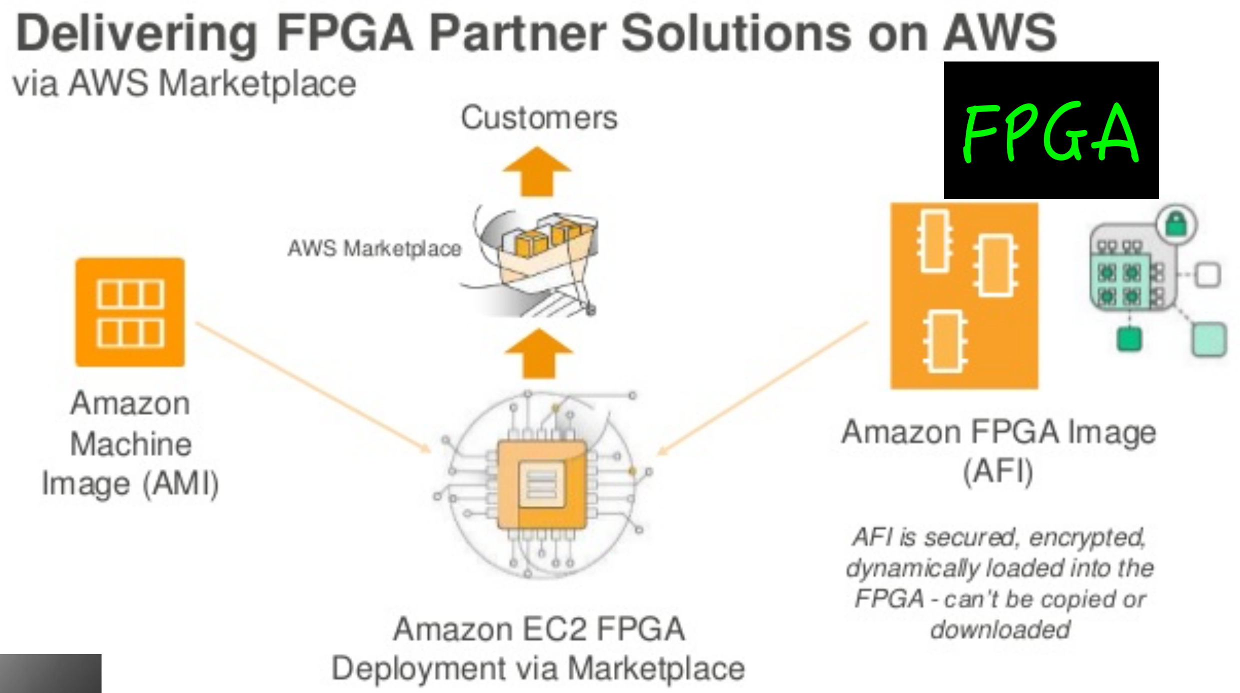


- Currently supported:
  - Small and large dense NNs
  - Binary and ternary NNs
  - Small 1D/2D CNNs
- Planned support
  - Big 1D/2D CNNs
  - Graph NNs
  - Other HLS/RTL backends

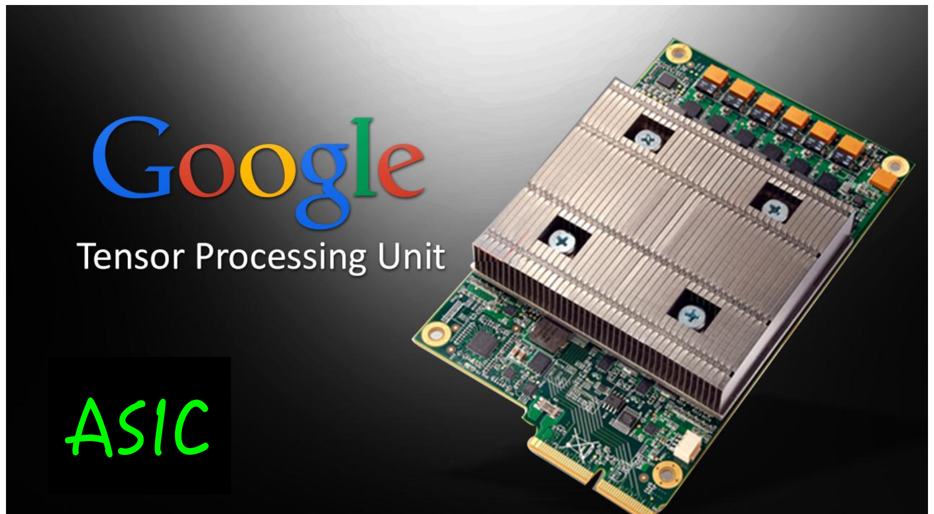
CO-PROCESSORS

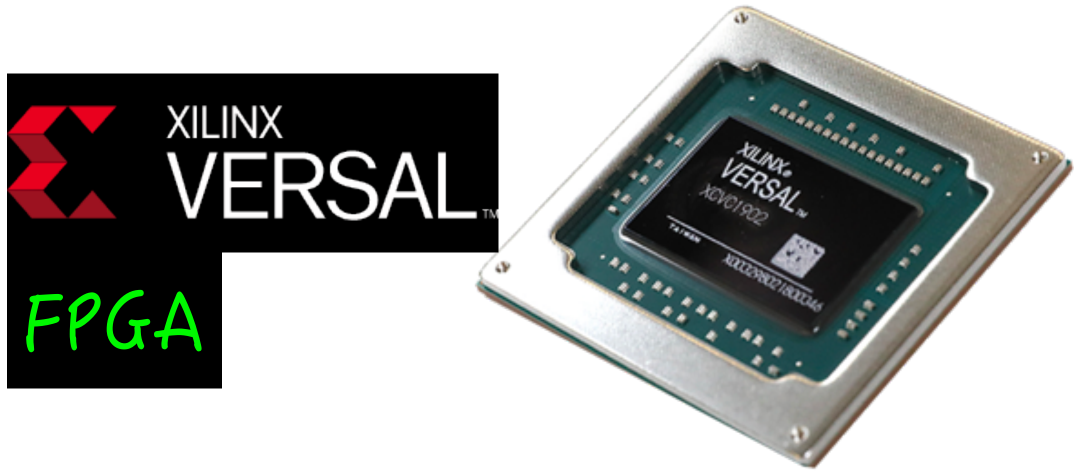
Specialized co-processor hardware for machine learning inference




Delivering FPGA Partner Solutions on AWS
via AWS Marketplace
FPGA
Customers
AWS Marketplace
Amazon Machine Image (AMI)
Amazon FPGA Image (AFI)
AFI is secured, encrypted, dynamically loaded into the FPGA - can't be copied or downloaded
Amazon EC2 FPGA Deployment via Marketplace

INTEL® FPGA ACCELERATION HUB





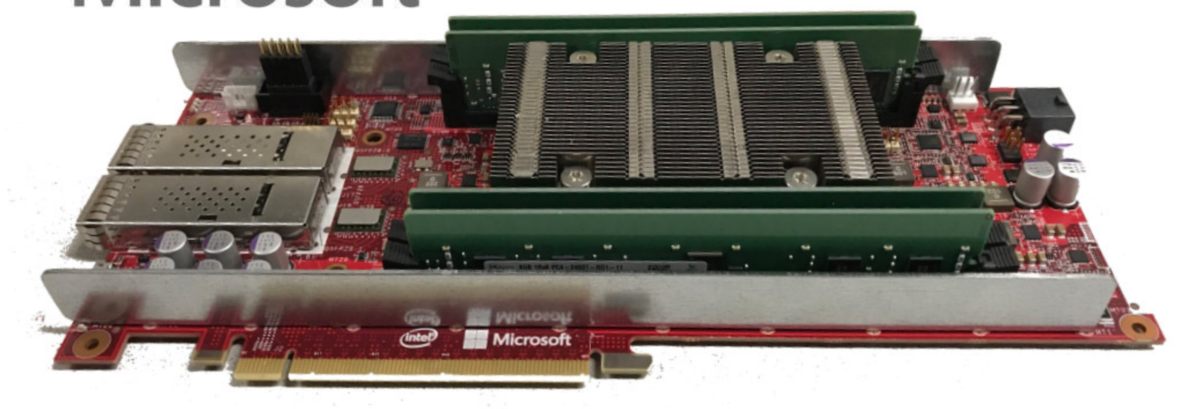
CO-PROCESSORS

Specialized co-processor hardware for machine learning inference

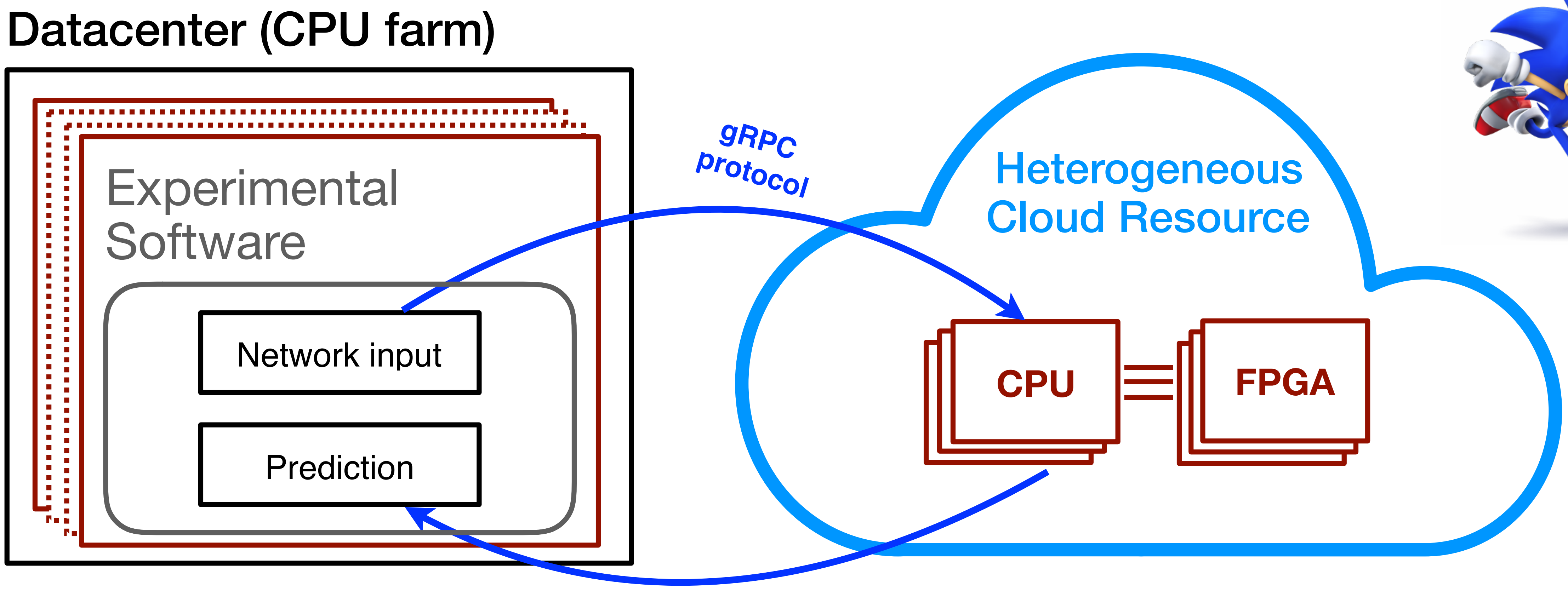
FPGA

Microsoft

Catapult/Brainwave

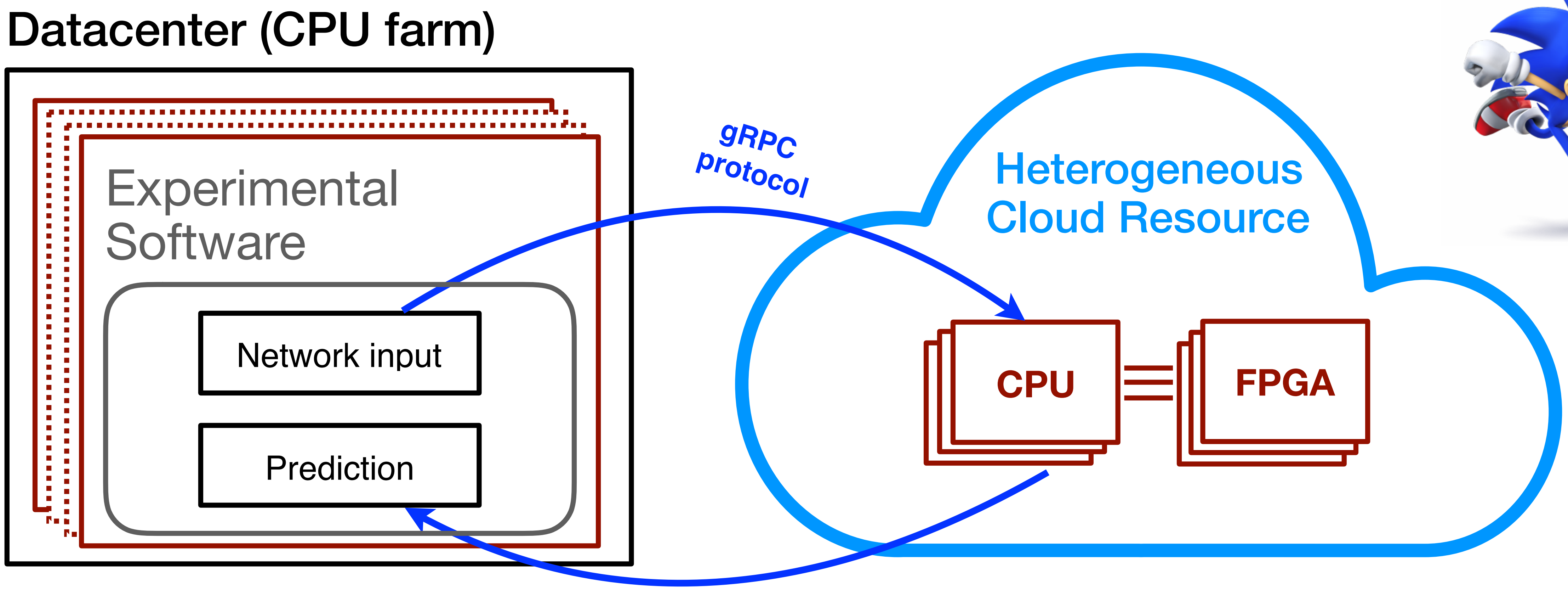


SONIC



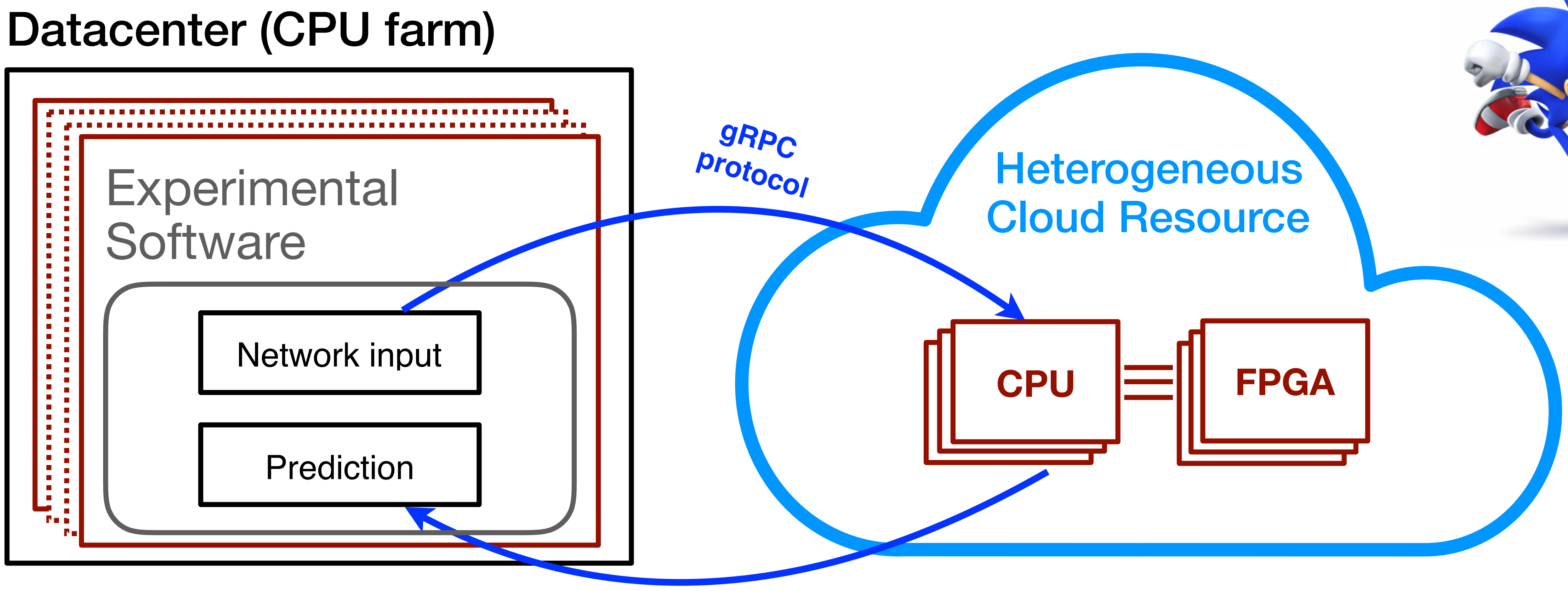
Datacenter (CPU farm)
Experimental Software
Network input
Prediction
gRPC protocol
Heterogeneous Cloud Resource
CPU
FPGA

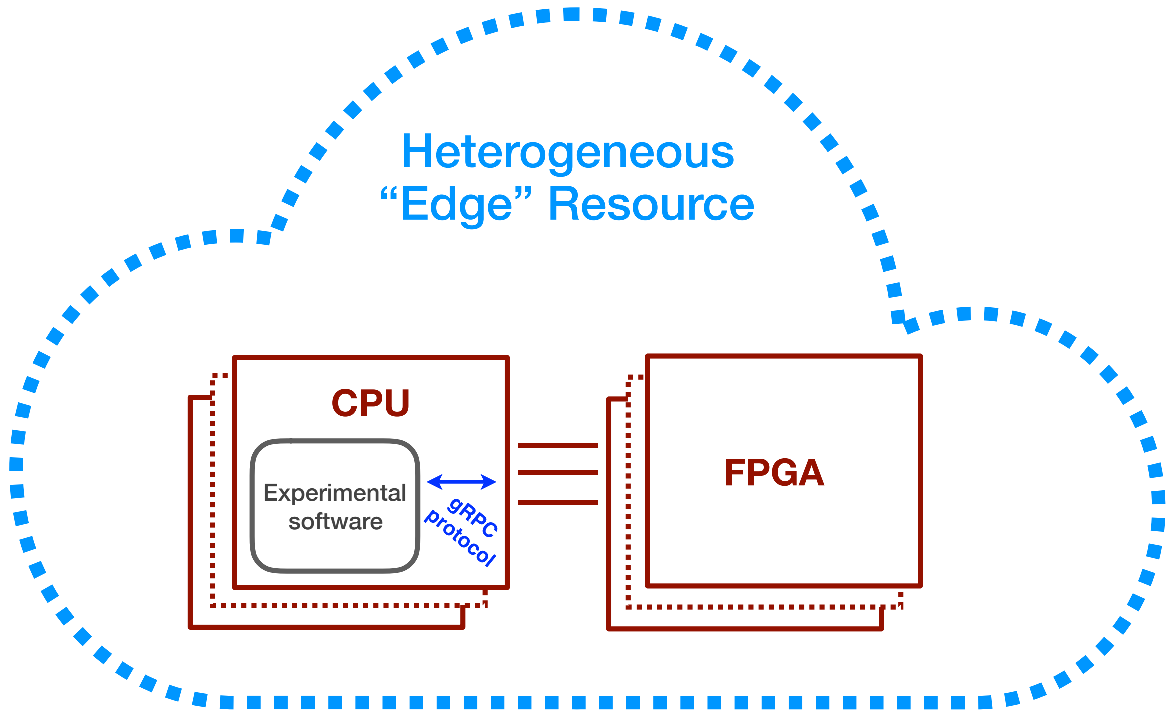
SONIC



- Services for Optimized Network Inference on Coprocessors (SONIC)
  - Send jet images from CMSSW to Microsoft Brainwave FPGA

SONIC

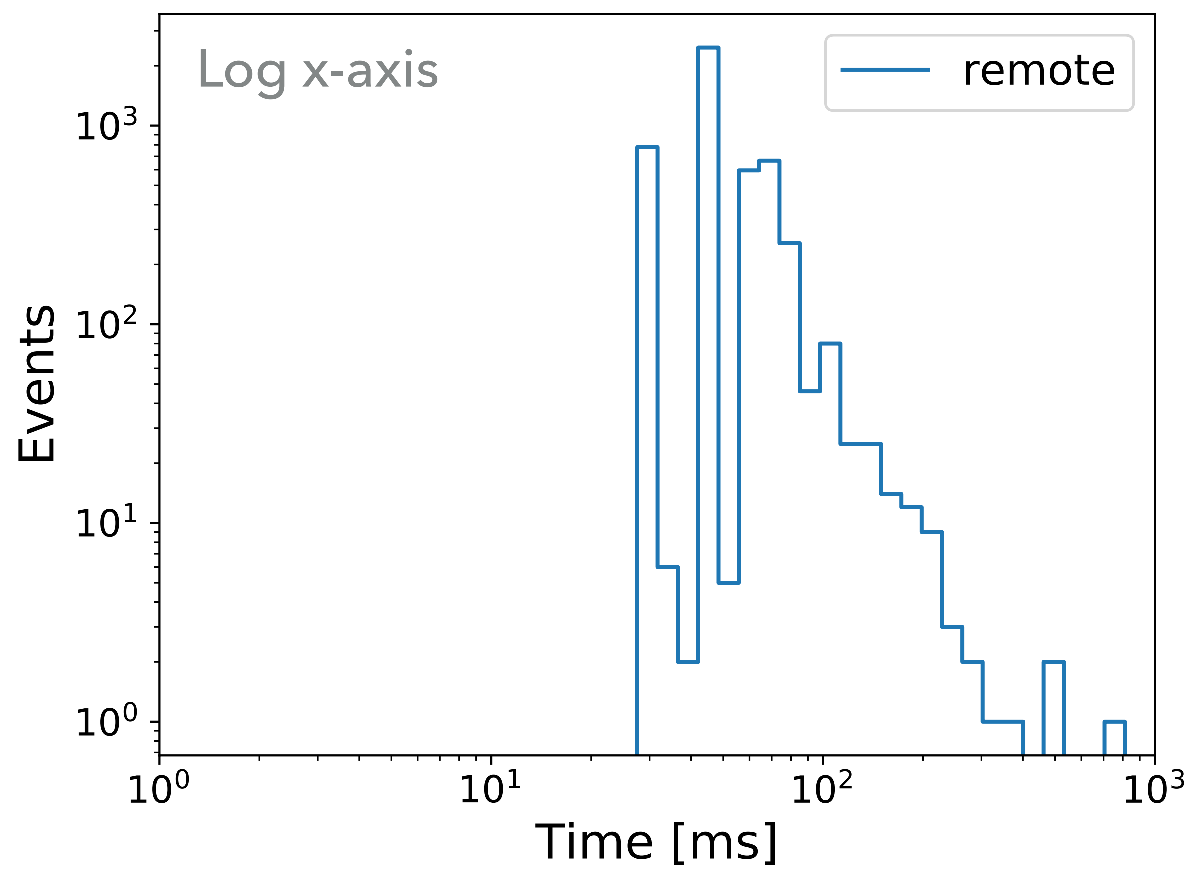


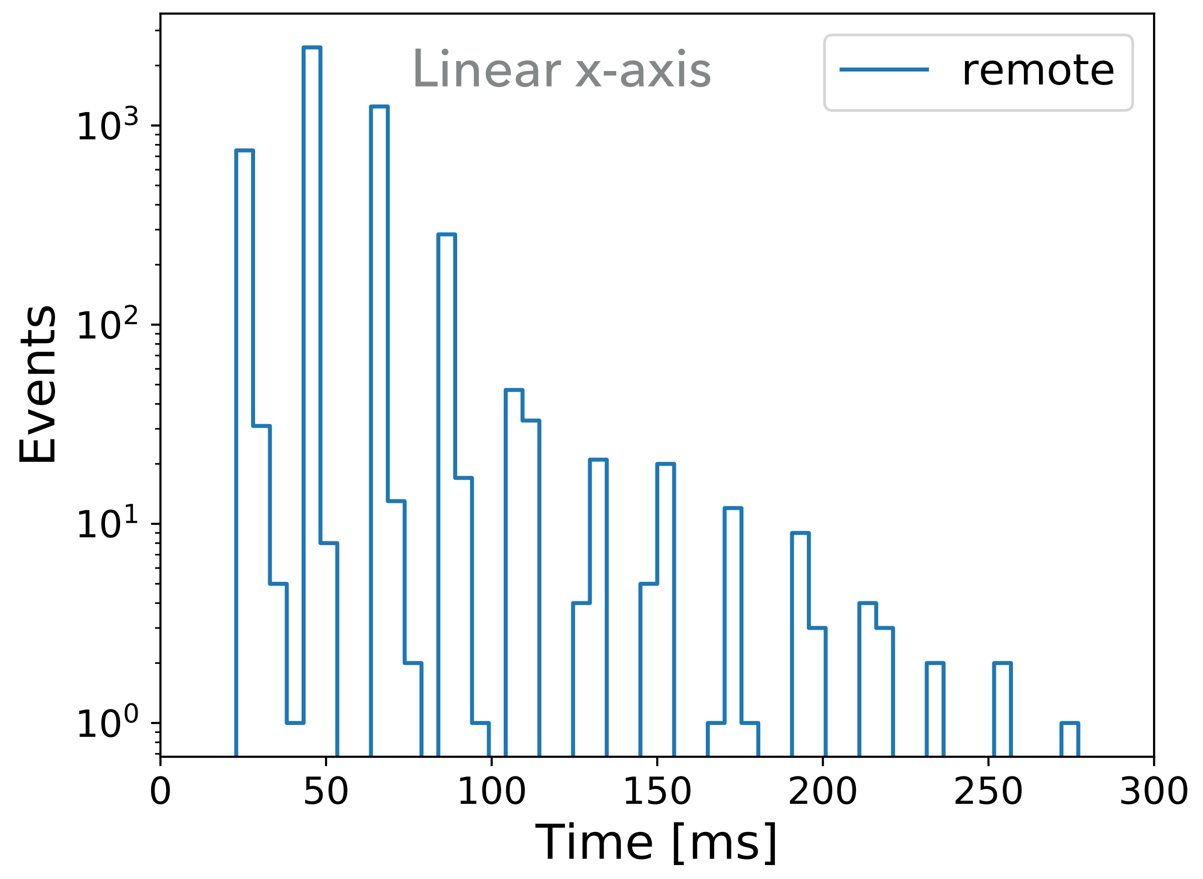


- Services for Optimized Network Inference on Coprocessors (SONIC)
  - Send jet images from CMSSW to Microsoft Brainwave FPGA
- Two modes: cloud service and on premises

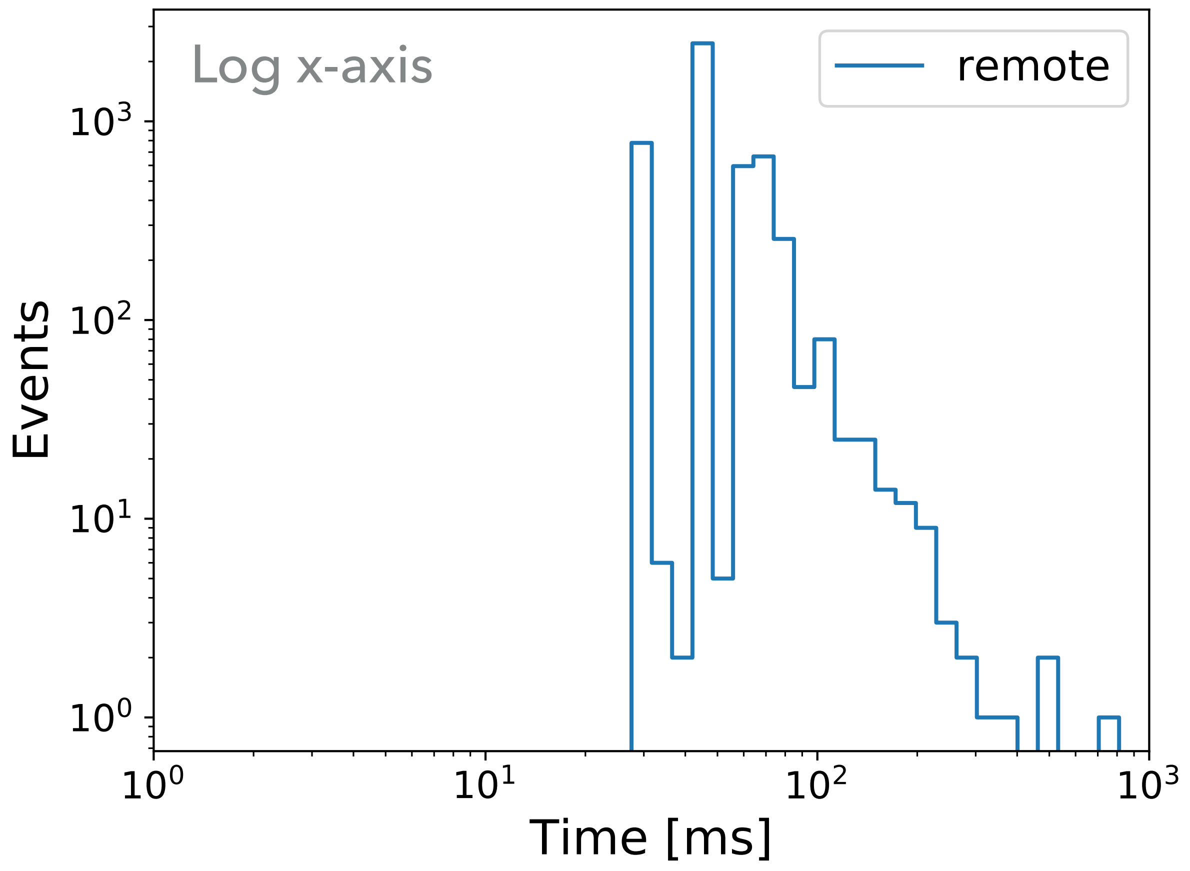
SONIC LATENCY

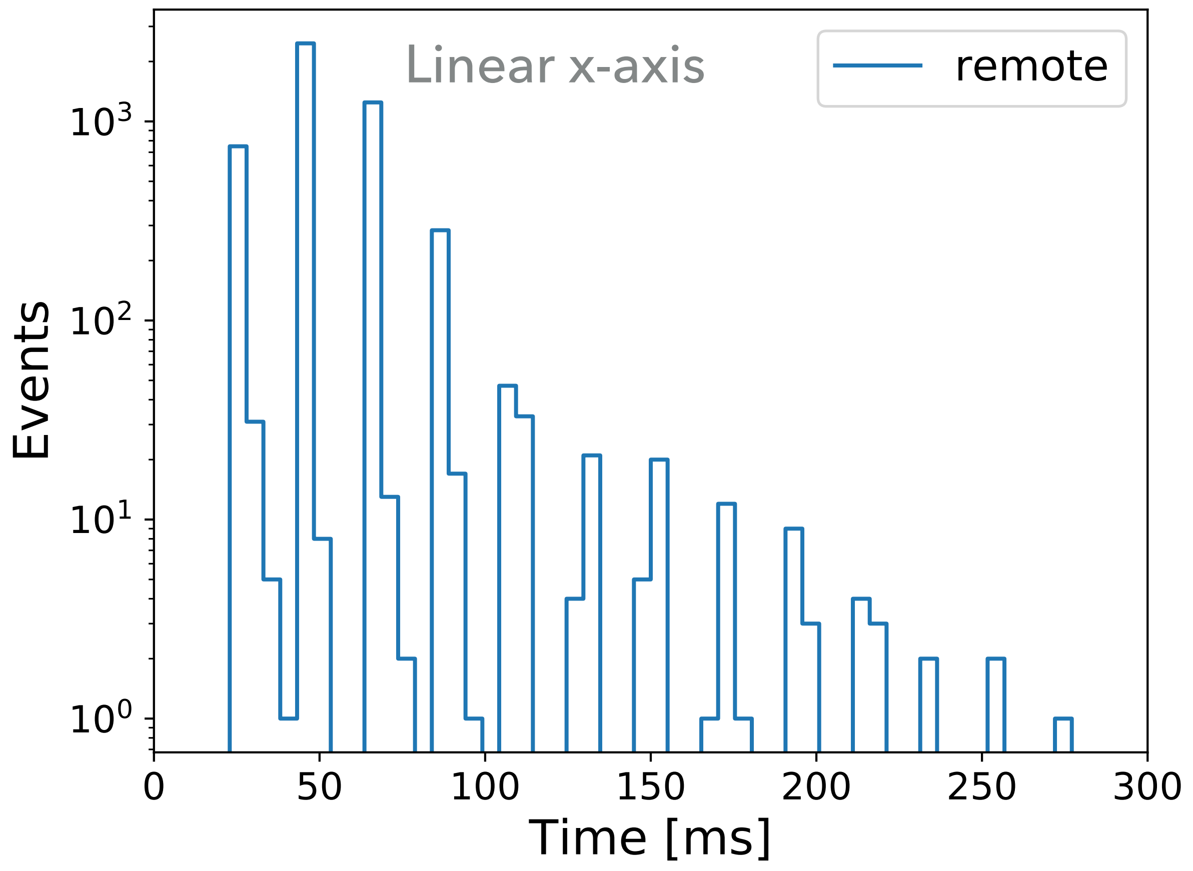
arXiv:1904.08986





SONIC LATENCY

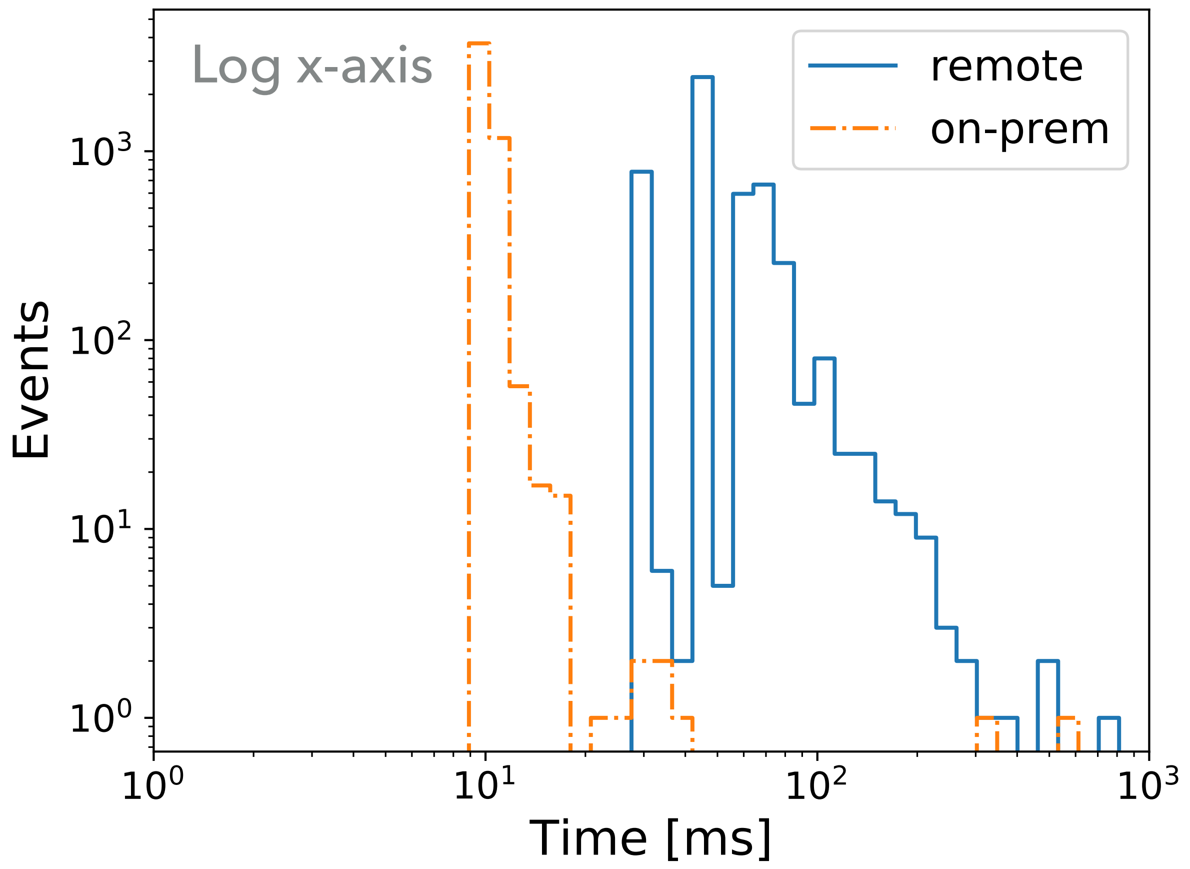


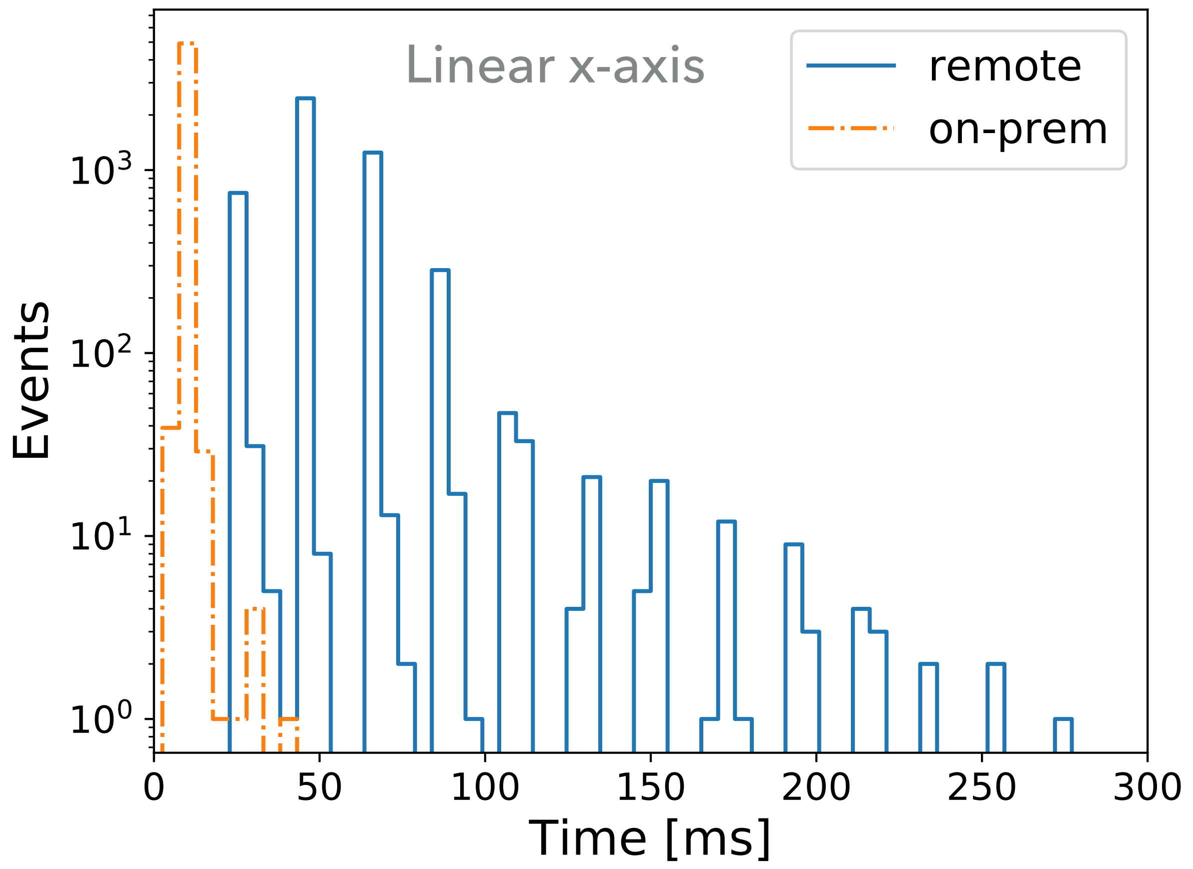


- Remote: FNAL (IL) to Azure (VA) ‹time› = 60 ms
  - Highly dependent on network conditions

SONIC LATENCY

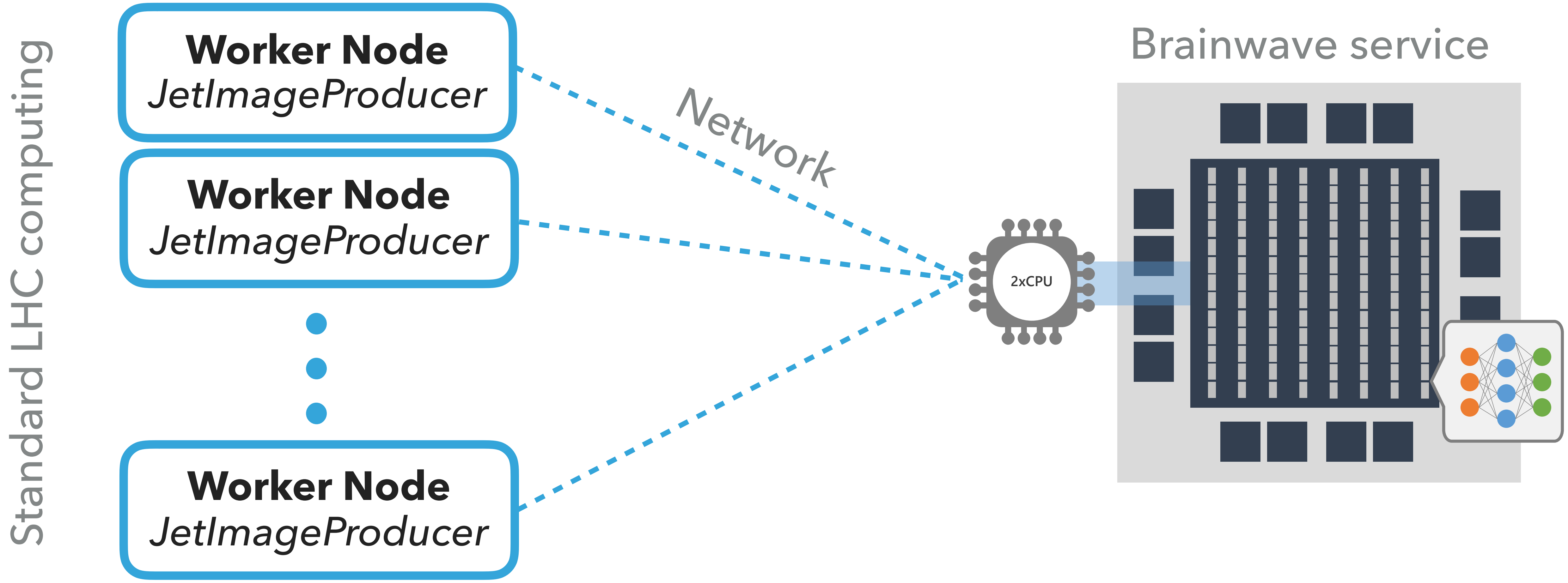
arXiv:1904.08986





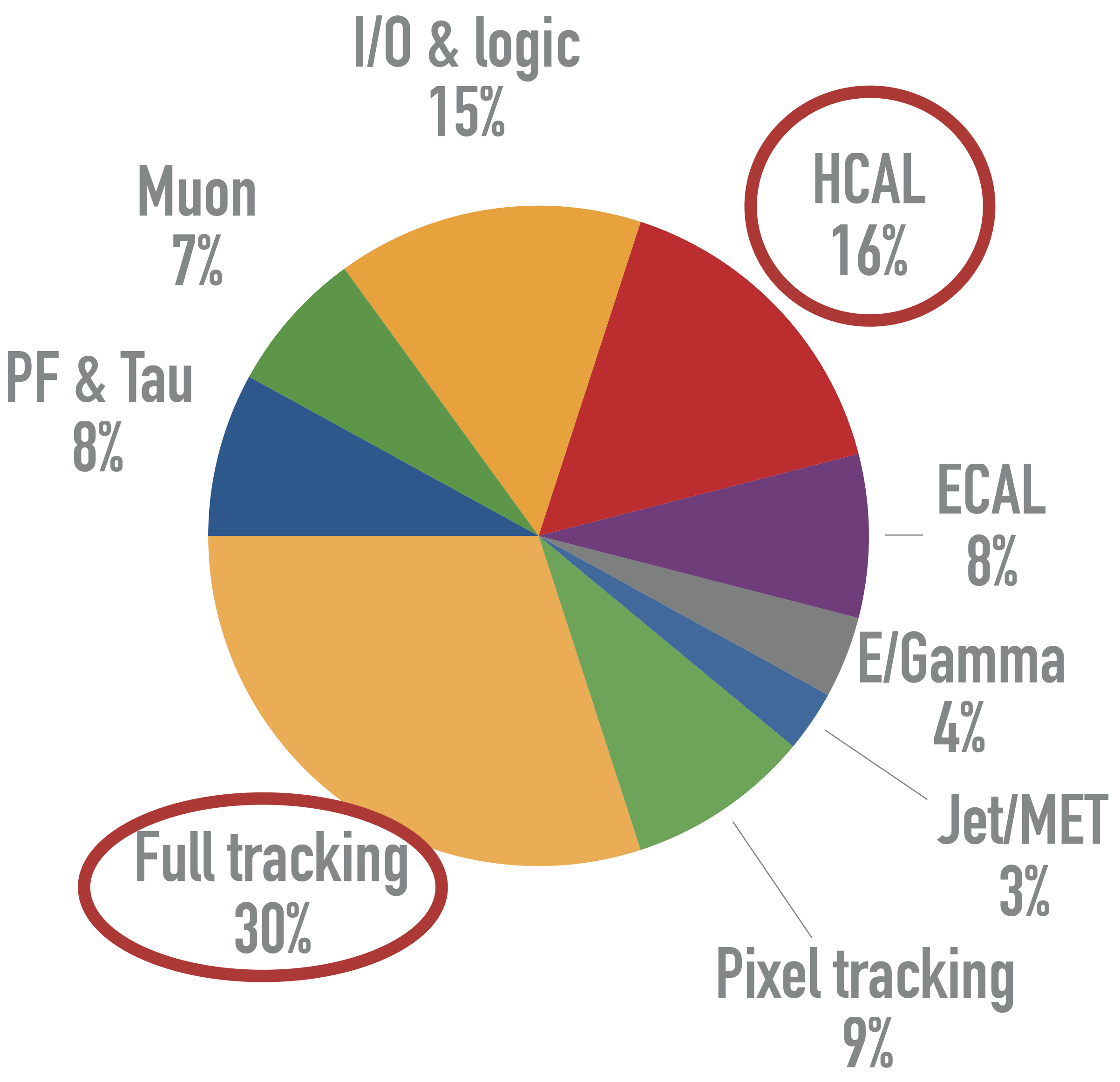
- Remote: FNAL (IL) to Azure (VA) ‹time› = 60 ms
  - Highly dependent on network conditions
- On-prem: run CMSSW on Azure ‹time› = 10 ms
  - on FPGA: 1.8 ms for inference
  - Remaining time used for classifying and I/O

SONIC+BRAINWAVE IN LHC COMPUTING



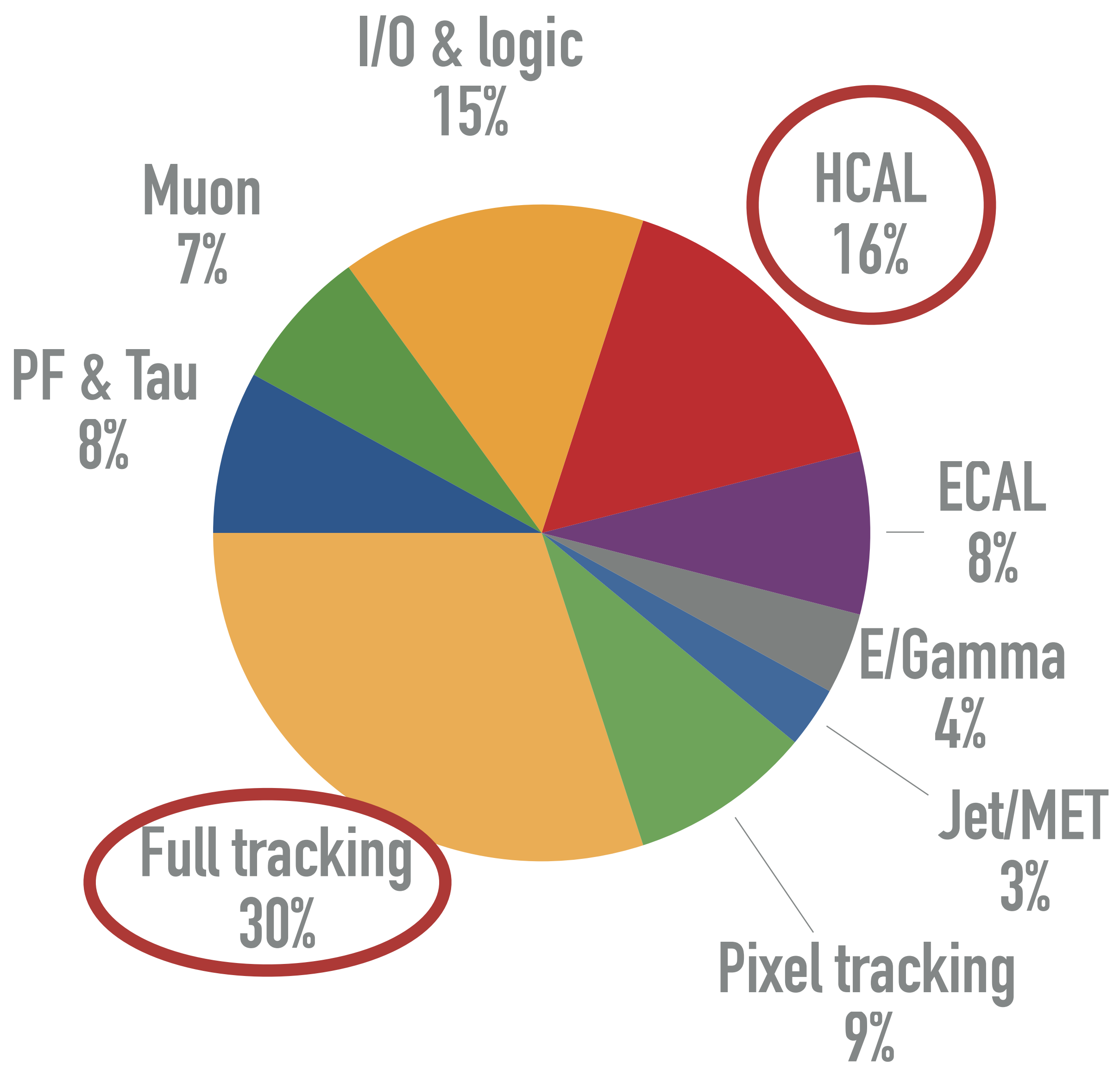
- **Brainwave + SONIC achieves**
  - **175× (30×)** on-prem (remote) **better latency** vs. CMS CPU
  - 1 FPGA service can serve 100s of CPU worker nodes
    - **Competitive throughput** vs. GPU as a service

ACCELERATING HIGH-LEVEL TRIGGER



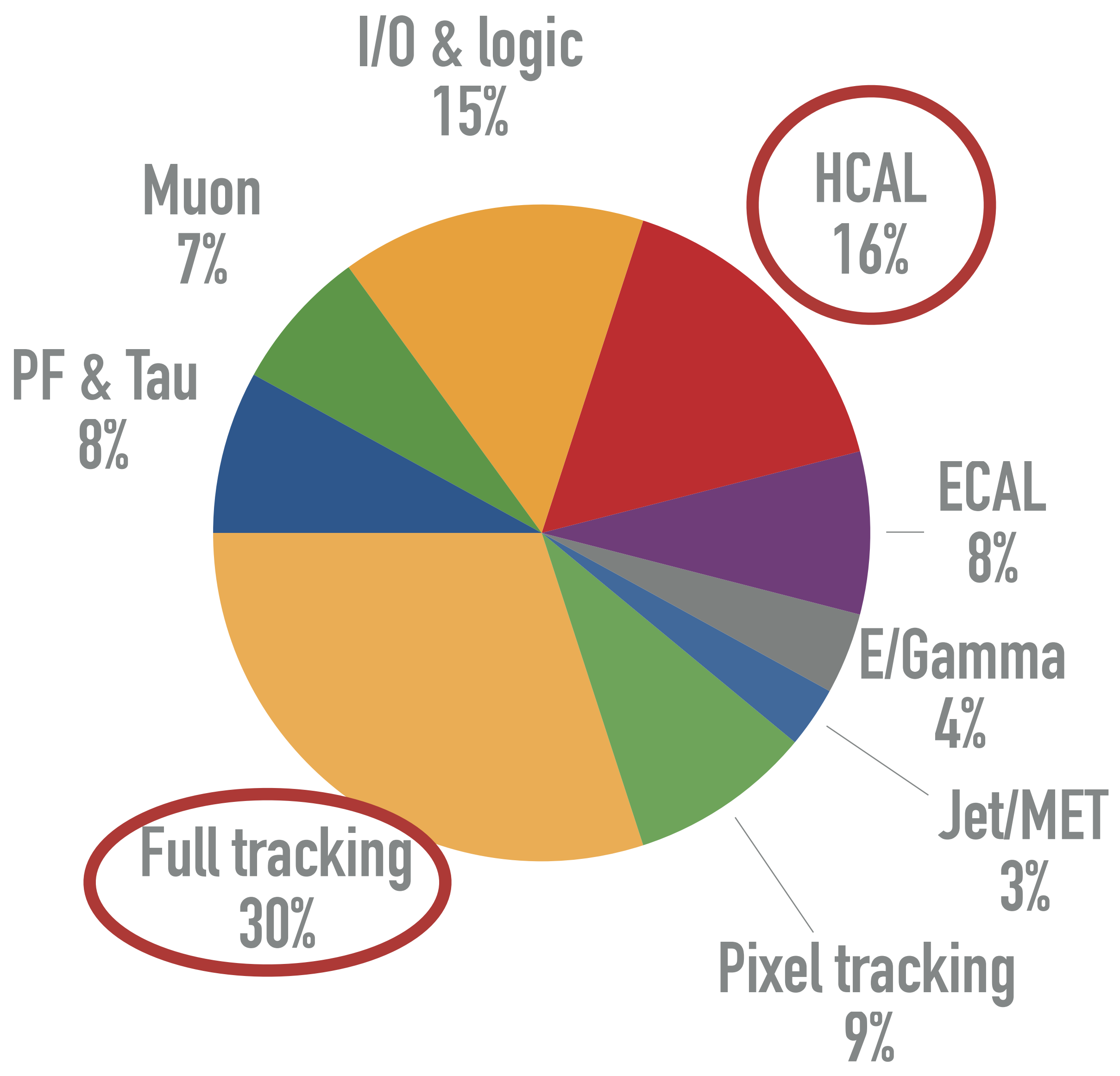
ACCELERATING HIGH-LEVEL TRIGGER

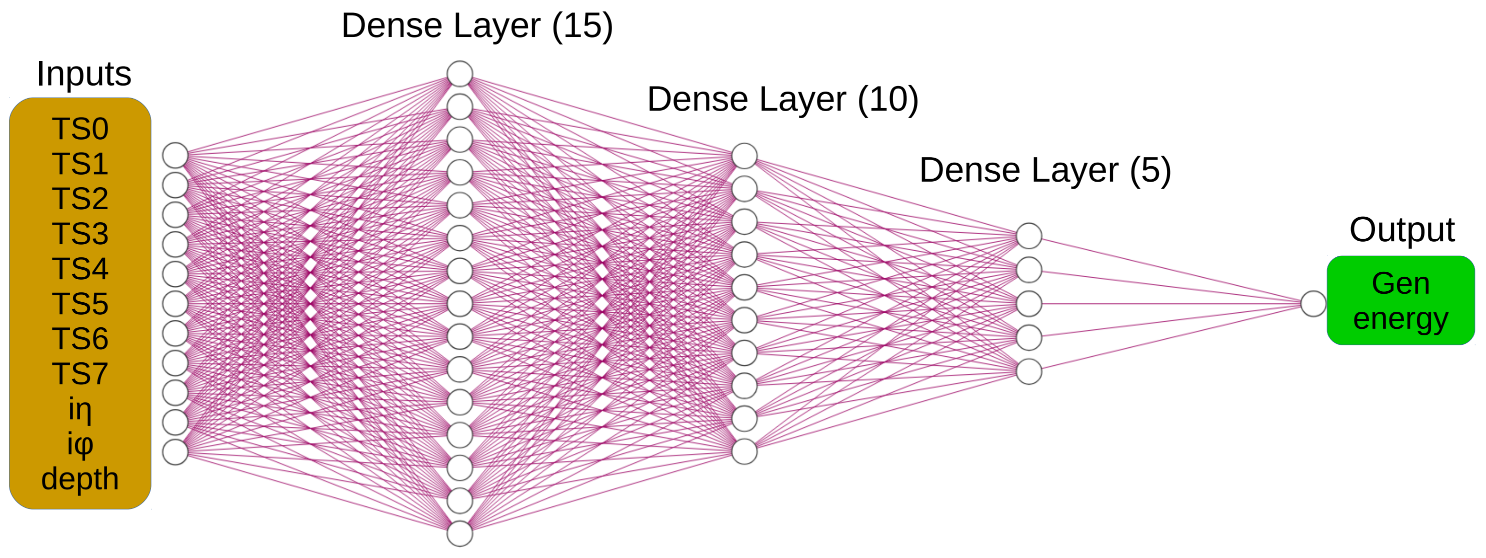
- HCAL reconstruction and tracking contribute significantly to HLT compute time



ACCELERATING HIGH-LEVEL TRIGGER

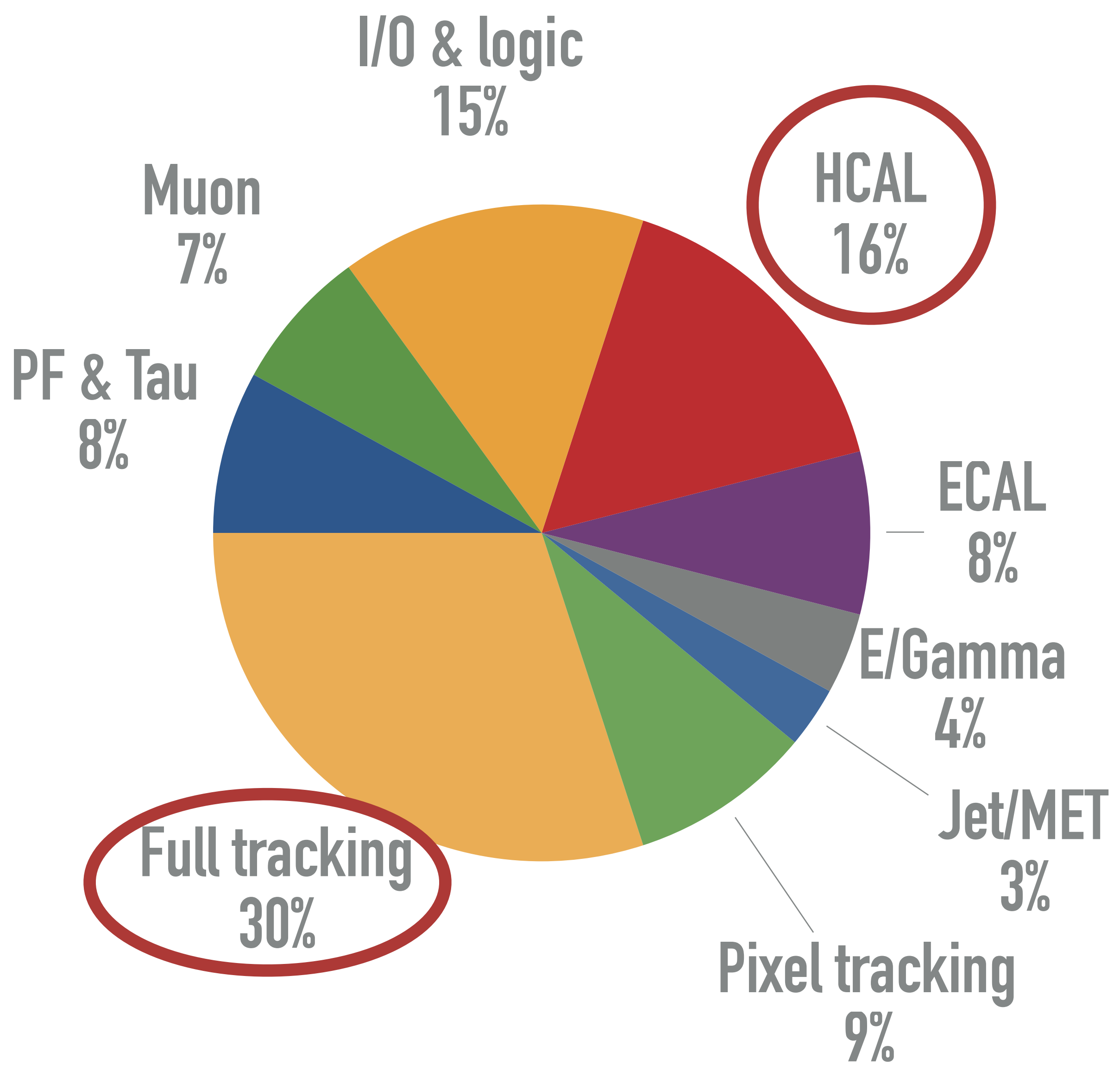
- HCAL reconstruction and tracking contribute significantly to HLT compute time
- GPU/FPGA as co-processor can reduce compute time

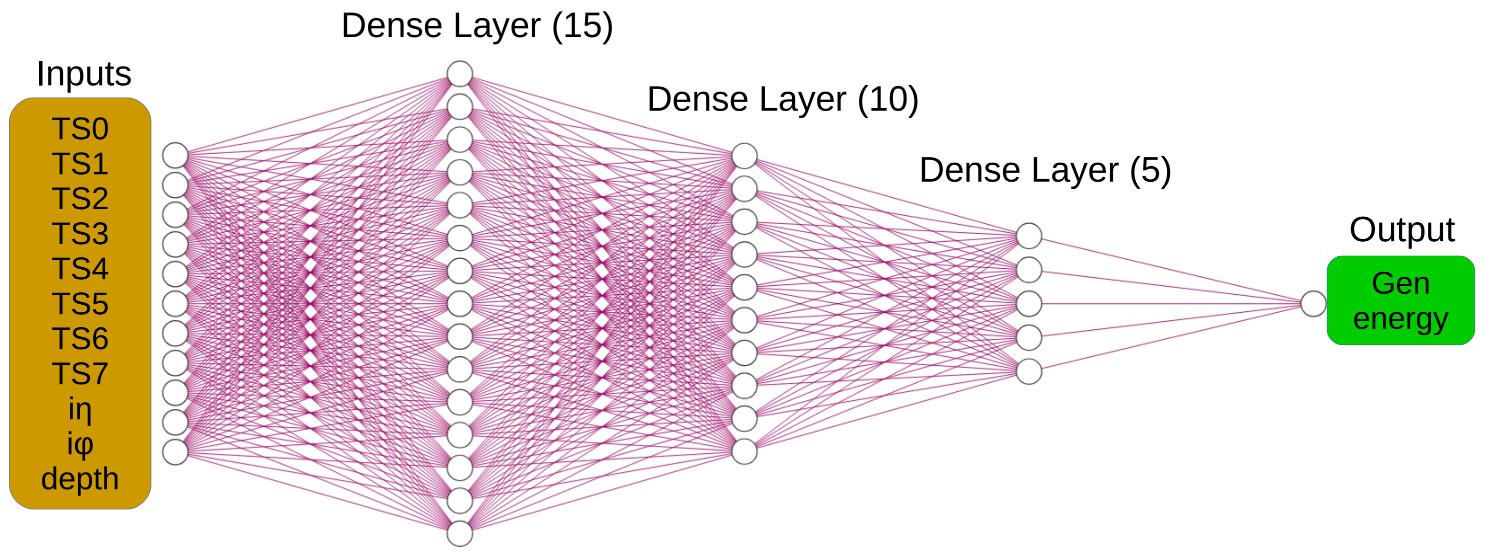




ACCELERATING HIGH-LEVEL TRIGGER

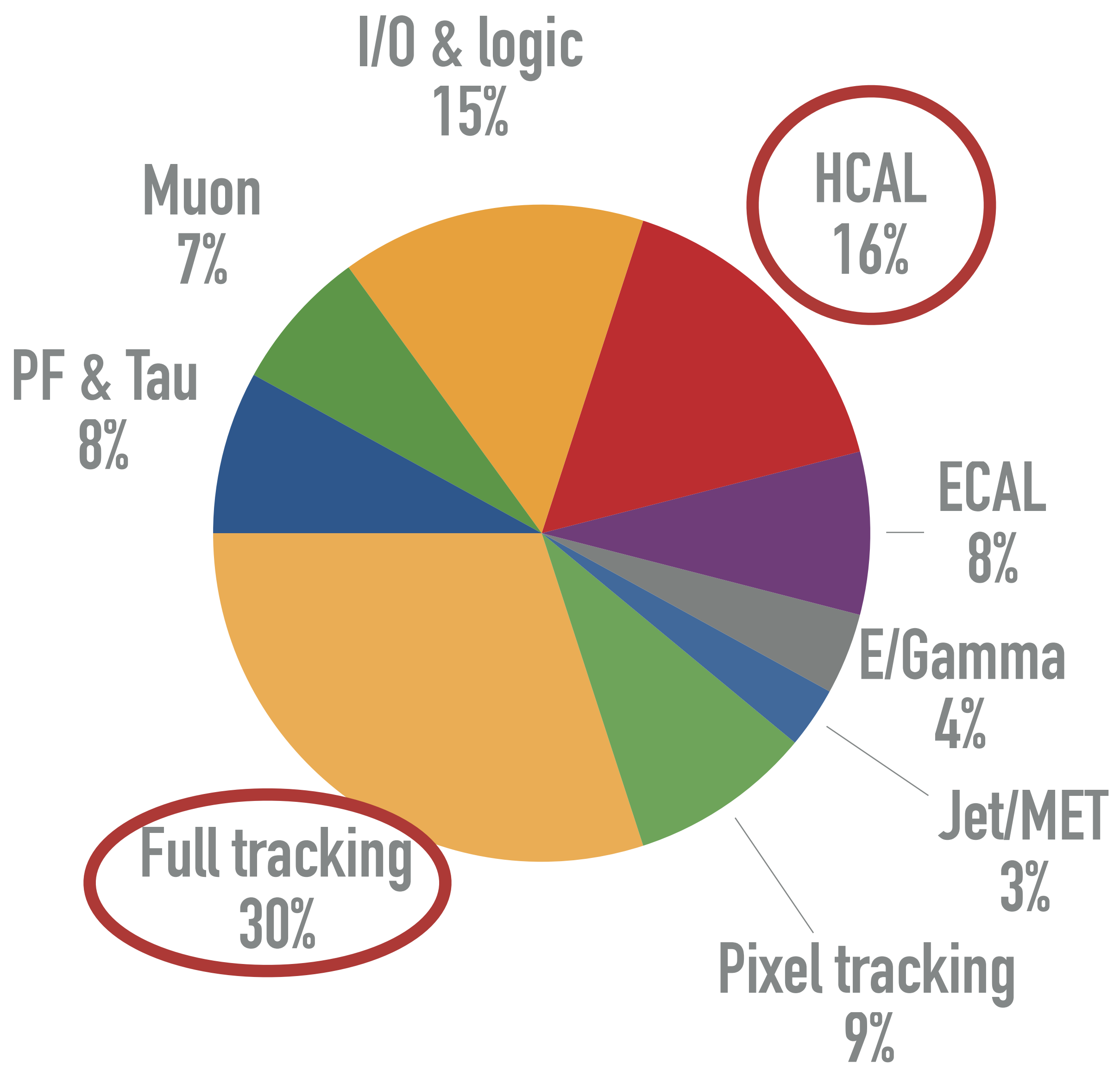
- HCAL reconstruction and tracking contribute significantly to HLT compute time
- GPU/FPGA as co-processor can reduce compute time
  - Patatrack pixel reconstruction on GPUs

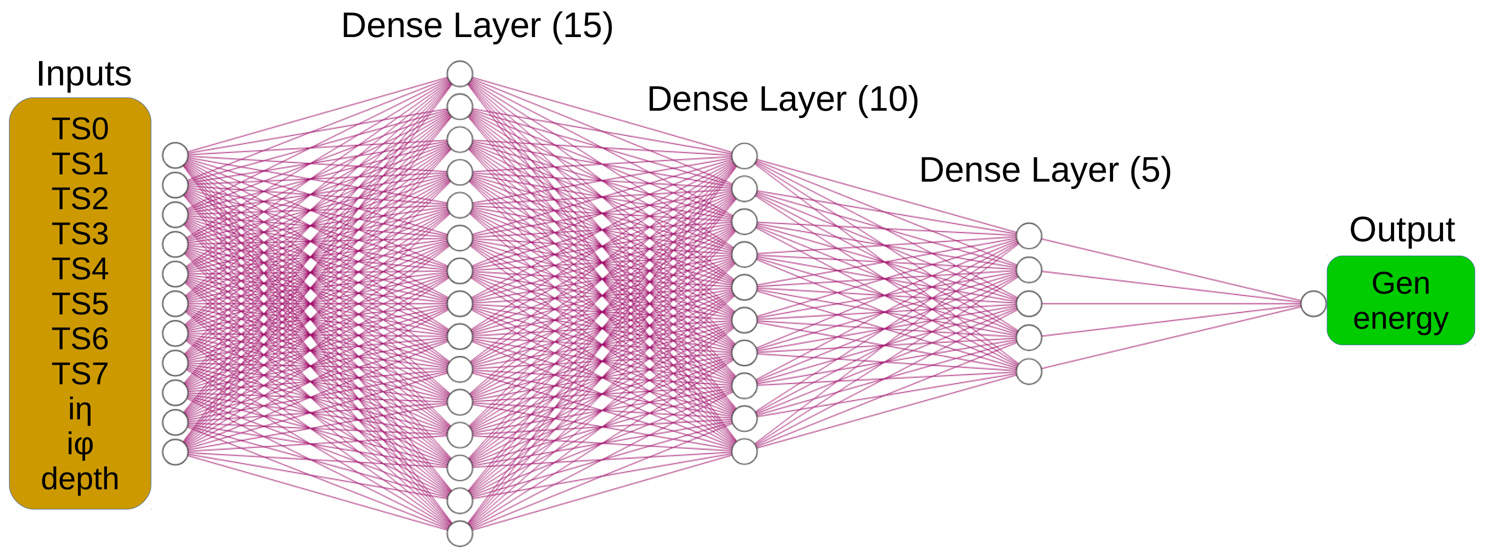




ACCELERATING HIGH-LEVEL TRIGGER

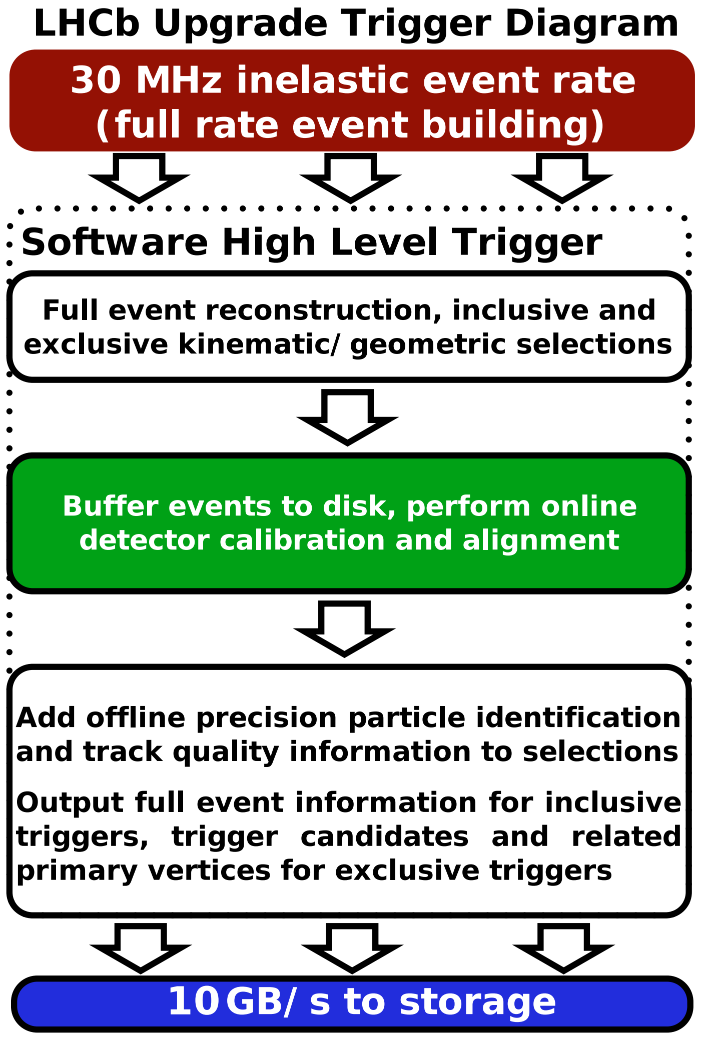
- HCAL reconstruction and tracking contribute significantly to HLT compute time
- GPU/FPGA as co-processor can reduce compute time
  - Patatrack pixel reconstruction on GPUs
  - HCAL reconstruction with ML on GPUs/FPGAs (as a service)



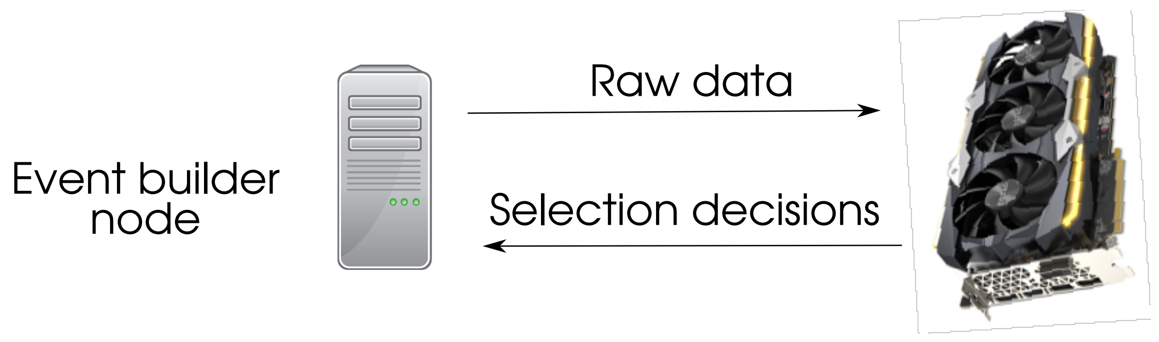


LHCB HIGH-LEVEL TRIGGER ON GPUS

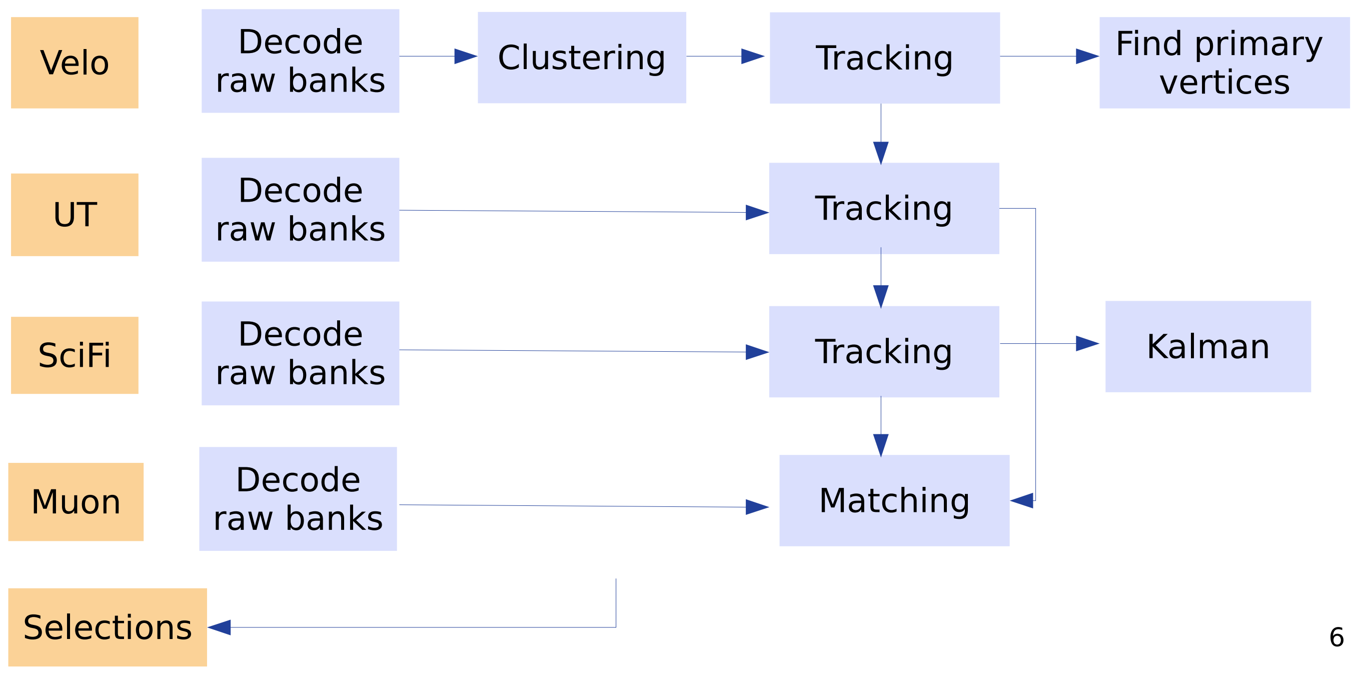
- By 2021, full LHCb trigger chain in software (HLT)
- Run full first stage of HLT (HLT1) on GPUs
- One GPU has to process 30/60 k events/s
- The current sequence of full Velo, primary vertices, full UT, and SciFi decoding runs on an NVIDIA V100 at 112 kHz



LHCb Upgrade Trigger Diagram
30 MHz inelastic event rate (full rate event building)
Software High Level Trigger
Full event reconstruction, inclusive and exclusive kinematic/ geometric selections
Buffer events to disk, perform online detector calibration and alignment
Add offline precision particle identification and track quality information to selections
Output full event information for inclusive triggers, trigger candidates and related primary vertices for exclusive triggers
10 GB/ s to storage

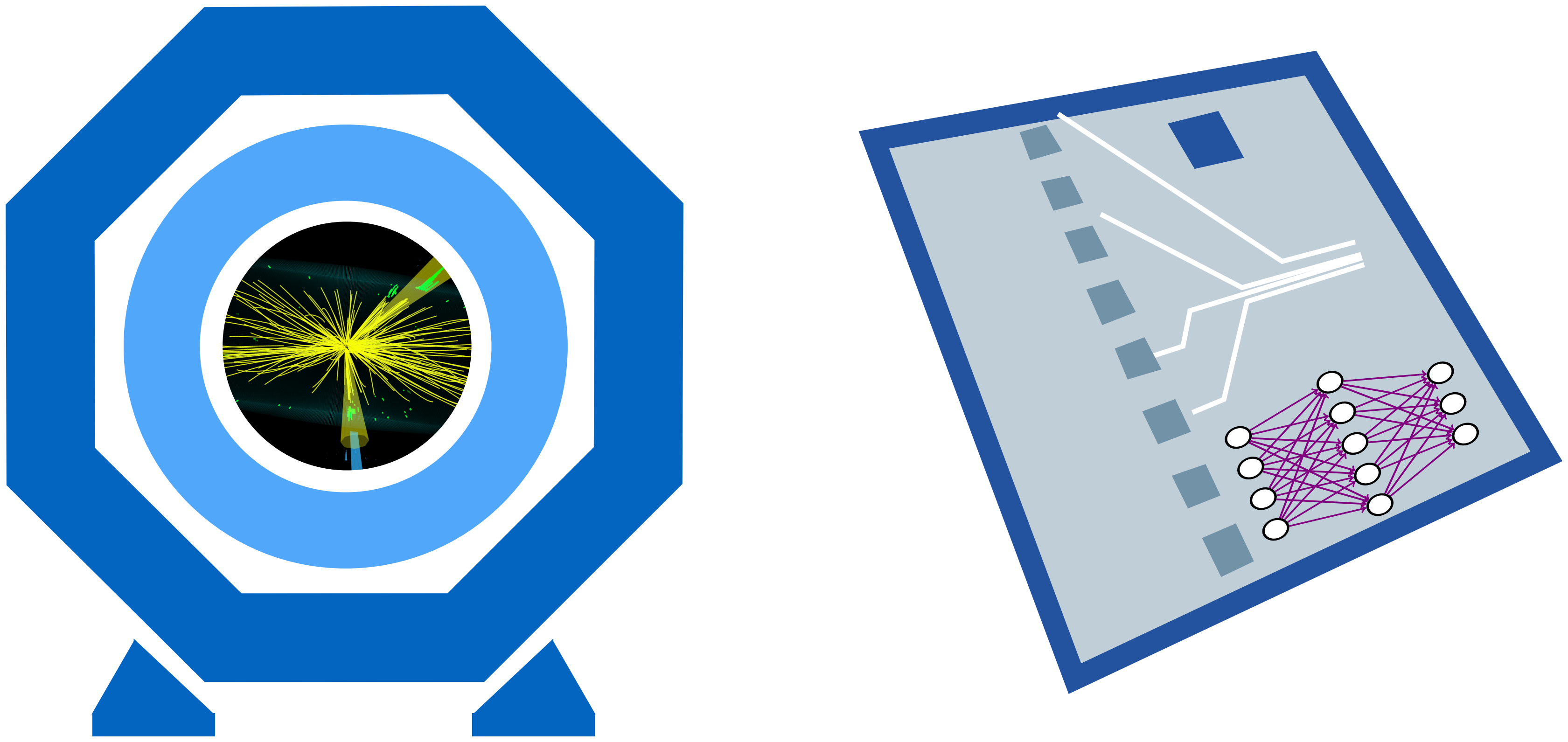


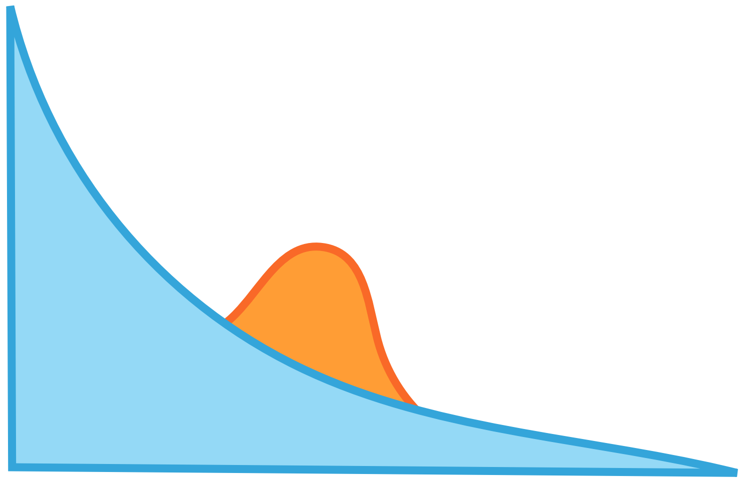
Raw data is decoded on the GPU



SUMMARY AND OUTLOOK

- Particle physics experiments face **extreme trigger challenges** in the coming years
- Exploiting **new algorithms**, **new hardware**, and **machine learning** will be key to the success of next-gen experiments
- Open questions:
    - With more sophisticated algorithms at earlier trigger, how do we ensure performance/safety? backup triggers?
    - What community tools do we need to deploy ML at the trigger?
    - Which co-processors are best suited to which tasks for the high-level trigger?
    - How do we incorporate timing information at the trigger level?
    - What are the physics use-cases for L1 scouting at 40 MHz?
    - What can we do with the new trigger hardware capabilities which we aren't thinking about?
    - L1 gives us a fundamental limitation but is there more we can exploit at the HLT?
    - Can we make a (realistic) wish-list for triggerable characteristics of events?





JAVIER DUARTE (UCSD)

OCTOBER 23, 2019

WEST COAST LHC JAMBOREE, SLAC

BACKUP

MUON RECONSTRUCTION WITH KALMAN FILTER

$$x_n = \begin{pmatrix} k \\ \phi \\ \phi_b \end{pmatrix}_n = \underbrace{\boxed{\begin{pmatrix} 1 & 0 & 0 \\ a & 1 & b \\ c & 0 & d \end{pmatrix}}}_{F} \begin{pmatrix} k \\ \phi \\ \phi_b \end{pmatrix}_{n-1}$$

multiple scattering

$$P_{n+1} = F P_n F^T + \boxed{Q}$$

$$z_k = \begin{pmatrix} \phi_s \\ \phi_{bs} \end{pmatrix} = \overset{H}{\boxed{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}} \begin{pmatrix} k \\ \phi \\ \phi_b \end{pmatrix}$$

$$y_n = z - H x_n$$

$$S = HPH^T + \boxed{R}$$

position error

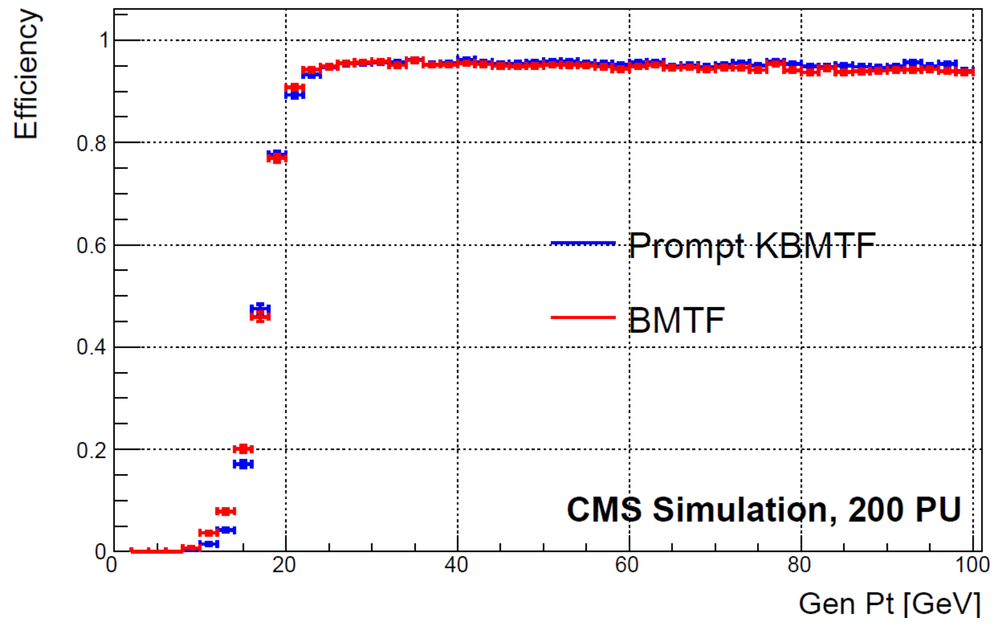
$$K = PH^T \boxed{S^{-1}}$$

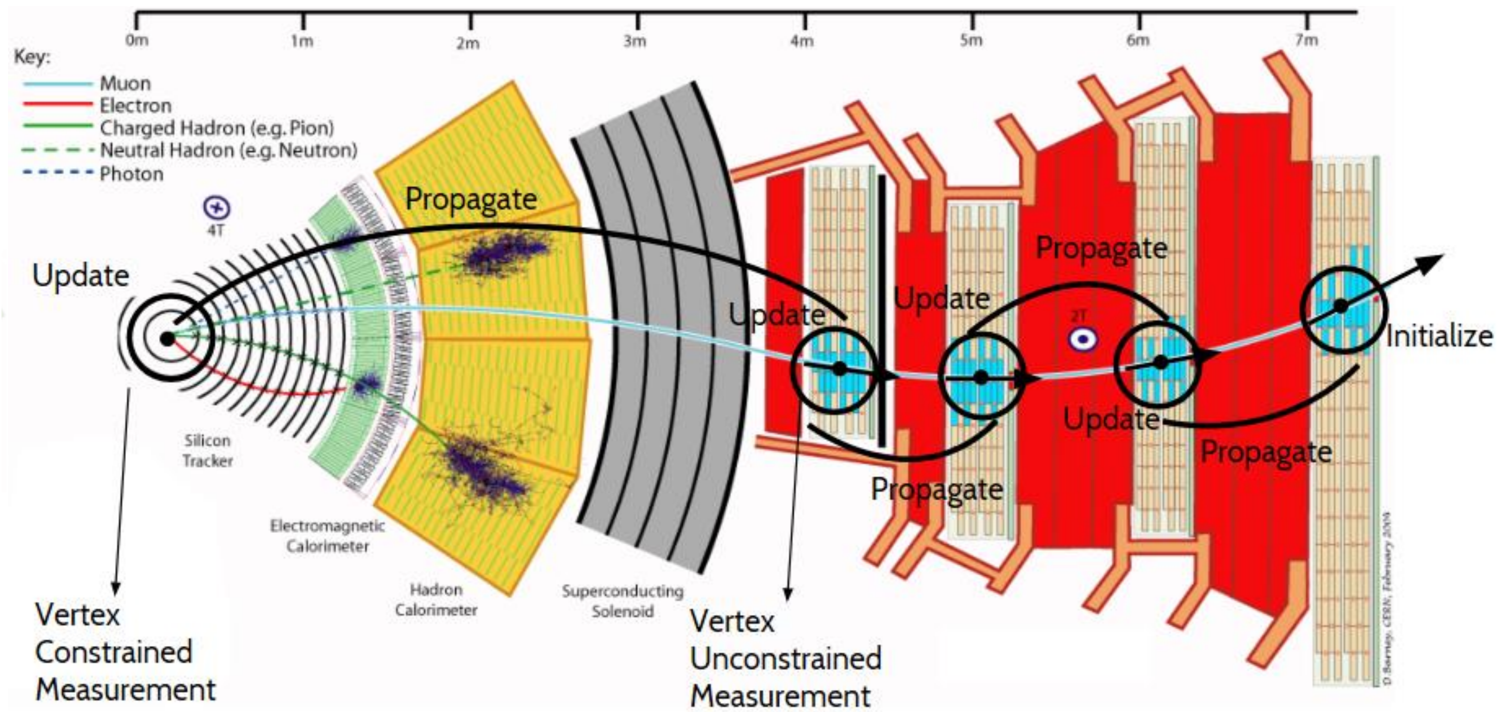
matrix inversion!

$$\boxed{x = x_n + \boxed{K} y_n}$$

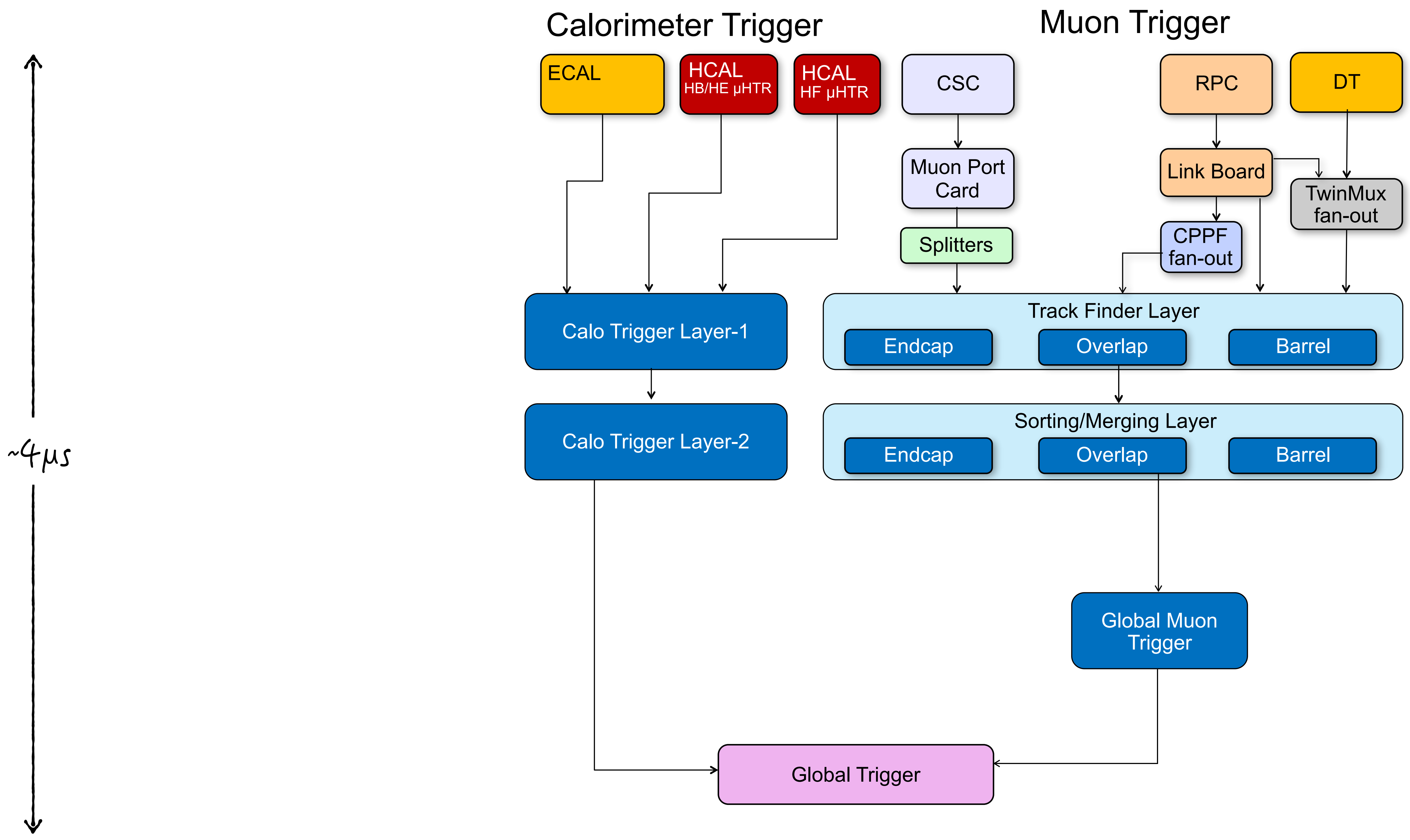
Kalman Gain

- Phase-2 : improved reconstruction using a Kalman filter
  - Iterative outer-inner tracking to reconstruct tracks and assign track p_T (as offline)
- Both PV constrained and unconstrained tracks: displaced standalone muons





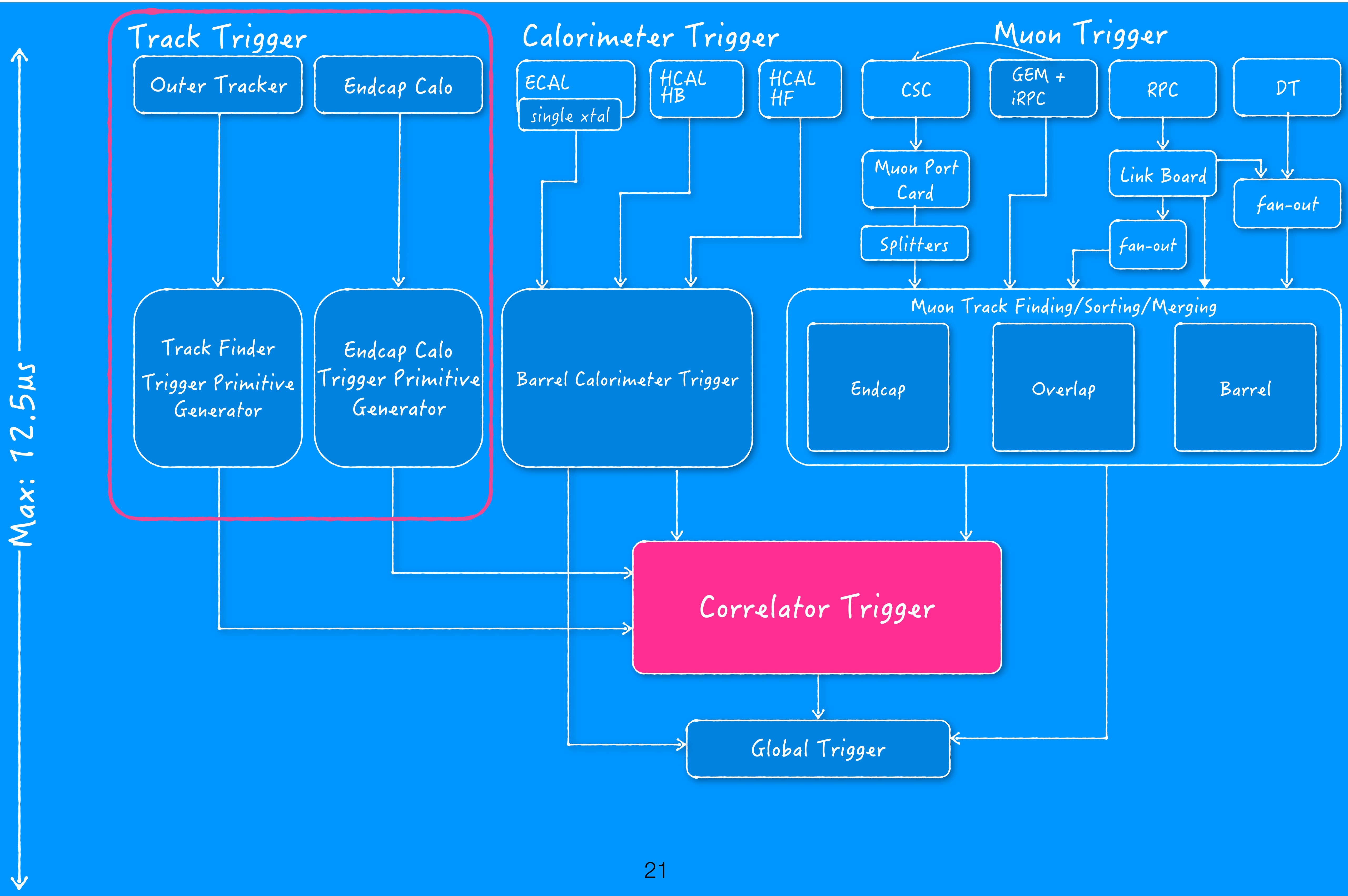
UPGRADING THE LEVEL-1 TRIGGER (BEFORE)



UPGRADING THE LEVEL-1 TRIGGER (AFTER)

More and better information available in the Level-1 trigger!

What can we do with it?


Max: 12.5μs
Track Trigger
Outer Tracker
Endcap Calo
Track Finder Trigger Primitive Generator
Endcap Calo Trigger Primitive Generator
Calorimeter Trigger
ECAL
single xtal
HCAL HB
HCAL HF
Barrel Calorimeter Trigger
Muon Trigger
CSC
GEM + iRPC
RPC
DT
Muon Port Card
Splitters
Link Board
fan-out
fan-out
Muon Track Finding/Sorting/Merging
Endcap
Overlap
Barrel
Correlator Trigger
Global Trigger