# FPGAs as a Service to Accelerate Machine Learning Inference

Joint HSF/OSG/WLCG Workshop
March 20, 2019

Fermilab
Javier Duarte
Burt Holzman
Sergo Jindariani
Benjamin Kreis
Mia Liu
**Kevin Pedro**
Nhan Tran
Aristeidis Tsaris

Massachusetts Institute of Technology
Philip Harris
Dylan Rankin

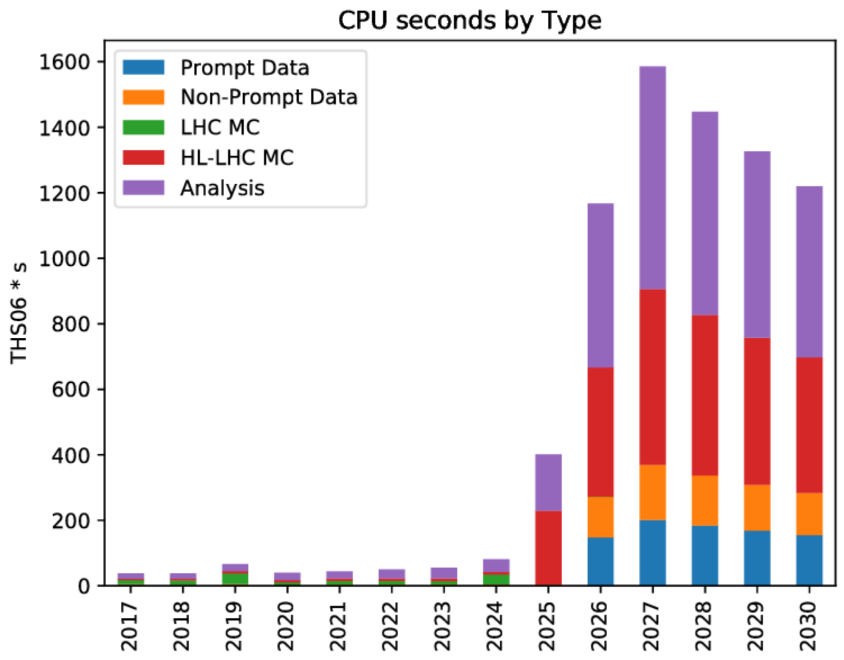
UNIVERSITY of WASHINGTON
Scott Hauck
Shih-Chieh Hsu
Matthew Trahms
Dustin Werran

UIC UNIVERSITY OF ILLINOIS AT CHICAGO
Zhenbin Wu

Microsoft
Suffian Khan
Brandon Perez
Colin Versteeg
Ted W. Way

CERN
Vladimir Loncar
Jennifer Ngadiuba
Maurizio Pierini

# Computing Challenges



HSF Community White Paper
arXiv:1712.06982

Energy frontier: HL-LHC

- 10× data vs. Run 2/3 → exabytes
- 200PU (vs. ~30PU in Run 2)
- CMS: 15× increase in pixel channels, 65× increase in calorimeter channels (similar for ATLAS)

Intensity frontier: DUNE

- Largest liquid argon detector ever designed
- ~1M channels, 1 ms integration time w/ MHz sampling → 30+ petabytes/year

➢ CPU needs for particle physics will increase by *more than an order of magnitude* in the next decade

# Development for Coprocessors

- Large speed improvement from hardware accelerated coprocessors
  - Architectures and tools are geared toward **machine learning**

**Option 1**

**re-write physics algorithms for new hardware**

Language: OpenCL, OpenMP, HLS, CUDA, …?

Hardware: FPGA, GPU

**Option 2**

**re-cast physics problem as machine learning problem**

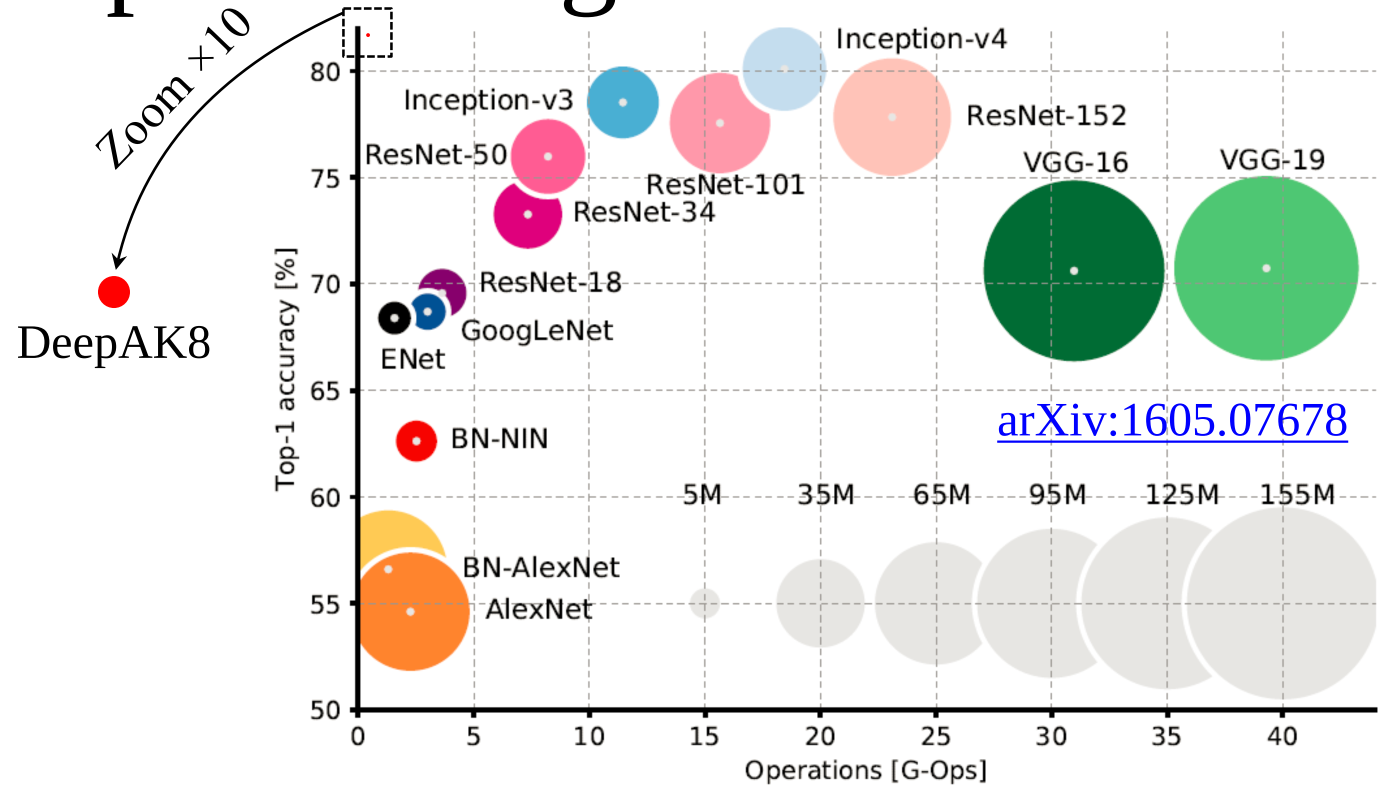
Language: C++, Python (TensorFlow, PyTorch,…)

Hardware: FPGA, GPU, ASIC

**Why (Deep) Machine Learning?**

- Common *language* for solving problems: simulation, reconstruction, analysis!
- Can be universally expressed on optimized computing hardware (follow industry trends)

# Deep Learning in Science and Industry



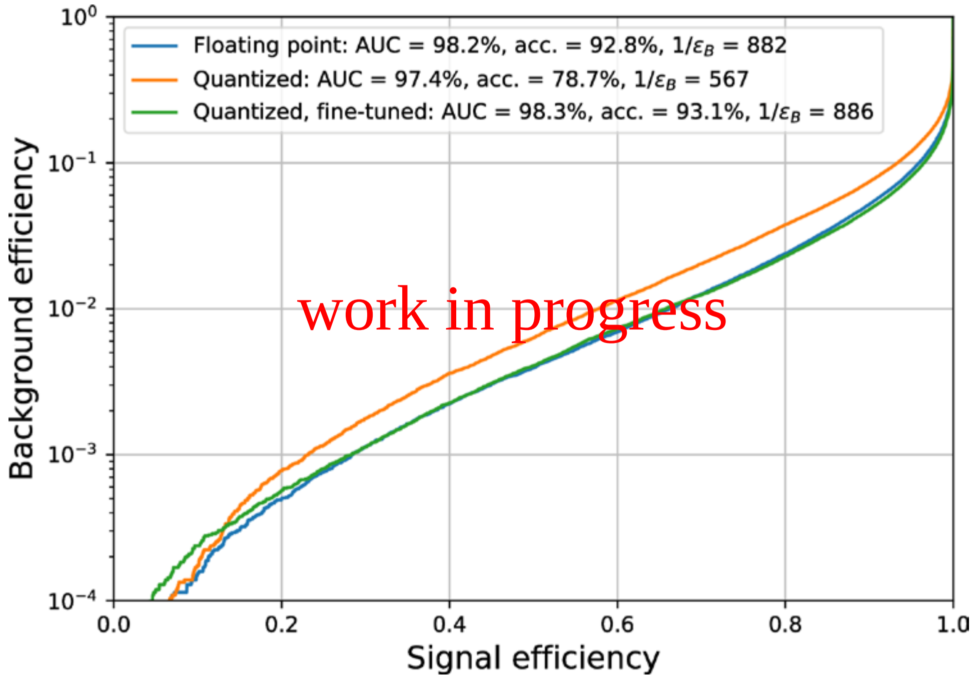
arXiv:1605.07678

- ResNet-50: 25M parameters, 7B operations
- Largest network currently used by CMS:
  - DeepAK8, 500K parameters, 15M operations
- Newer approaches w/ larger networks in development:
  - Particle cloud (arXiv:1902.08570), ResNet-like (arXiv:1902.09914)
  - Future: tracking (HEP.TrkX), HGCal clustering, …?

# Top Tagging w/ ResNet-50

- Retrain ResNet-50 on publicly available top quark tagging dataset
  - Convert jets into images using constituent $p_T$, $\eta$, $\varphi$

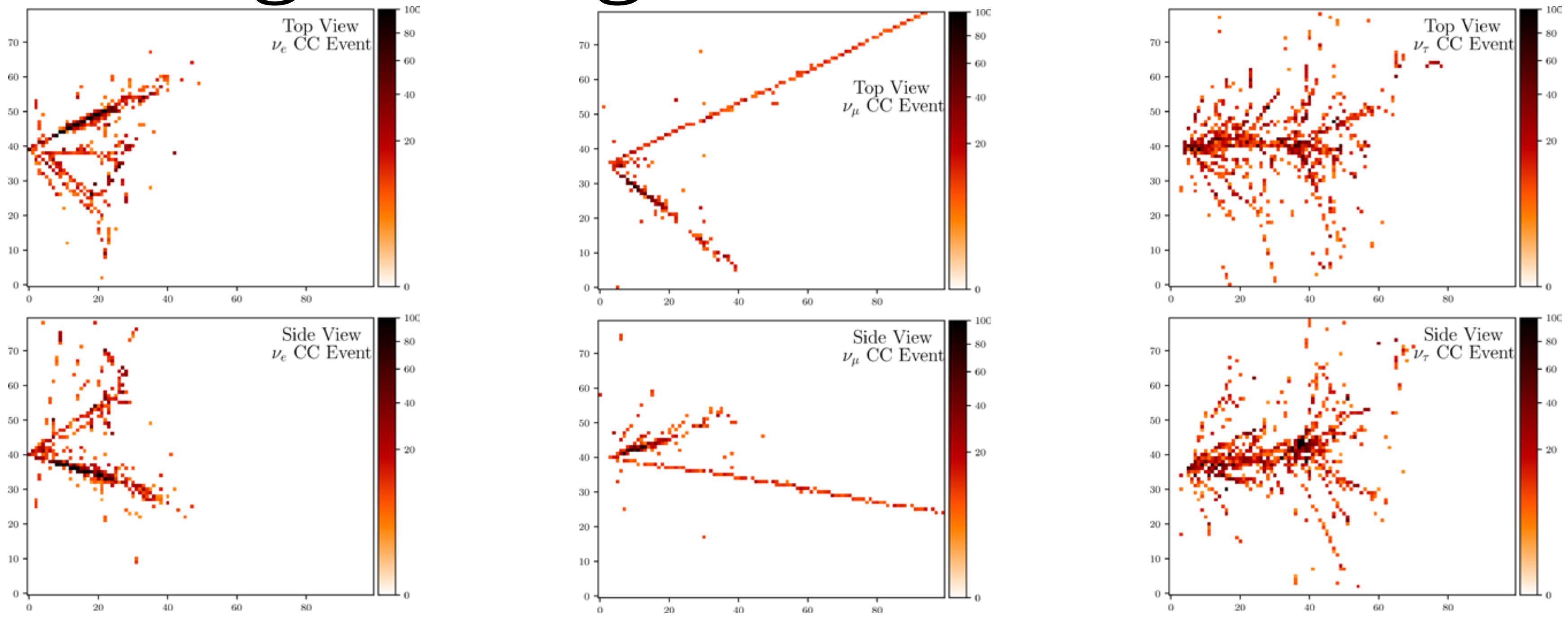
  $\rightarrow$ New set of weights, optimized for physics
  - Add custom classifier layers to interpret features from ResNet-50



- ResNet-50 model that runs on FPGAs is “quantized”
  - Tune weights to achieve similar performance

➢ State-of-the-art results vs. other leading algorithms

# Image Recognition for Neutrinos

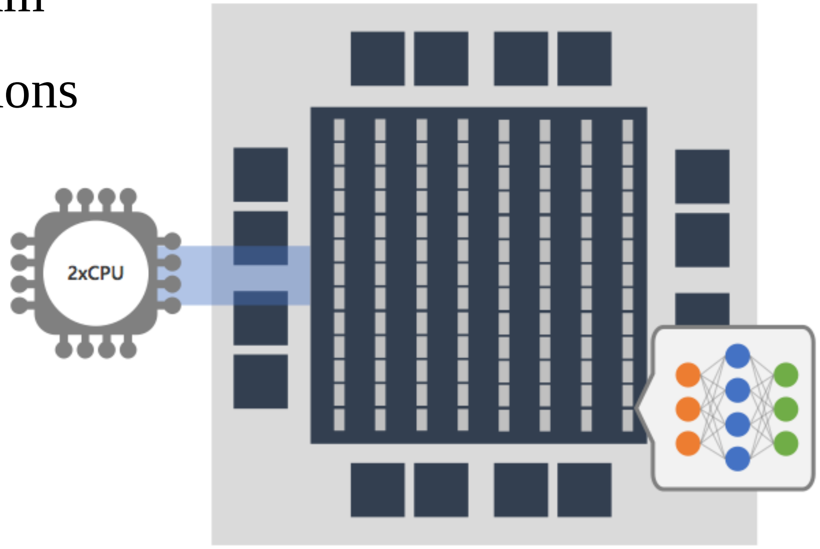


- ResNet-50 can also classify neutrino events to reject cosmic ray backgrounds
- Use *transfer learning*: keep default featurizer weights, retrain classifier layers
- Events above selected w/ probability > 0.9 in different categories
- NOvA was the first particle physics experiment to publish a result obtained using a CNN (arXiv:1604.01444, arXiv:1703.03328)
- CNN inference already a large fraction of neutrino reconstruction time

➢ Prime candidate for acceleration with coprocessors

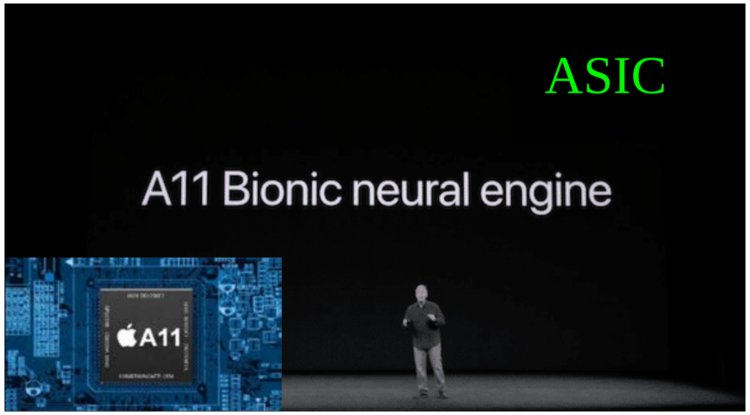
# Why Accelerate Inference?

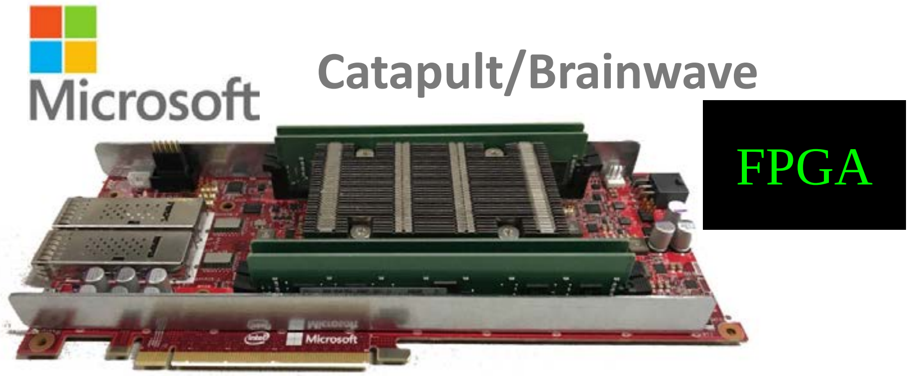
- DNN training happens ~once/year/algorithm
  - Cloud GPUs or new HPCs are good options
- Once DNN is in common use, inference will happens *billions* of times
  - MC production, analysis, prompt reconstruction, high level trigger…
- Inference as a service:
  - Minimize disruption to existing computing model
  - Minimize dependence on specific hardware
- Performance metrics:
  - Latency (time for a single request to complete)
  - Throughput (number of requests per unit time)

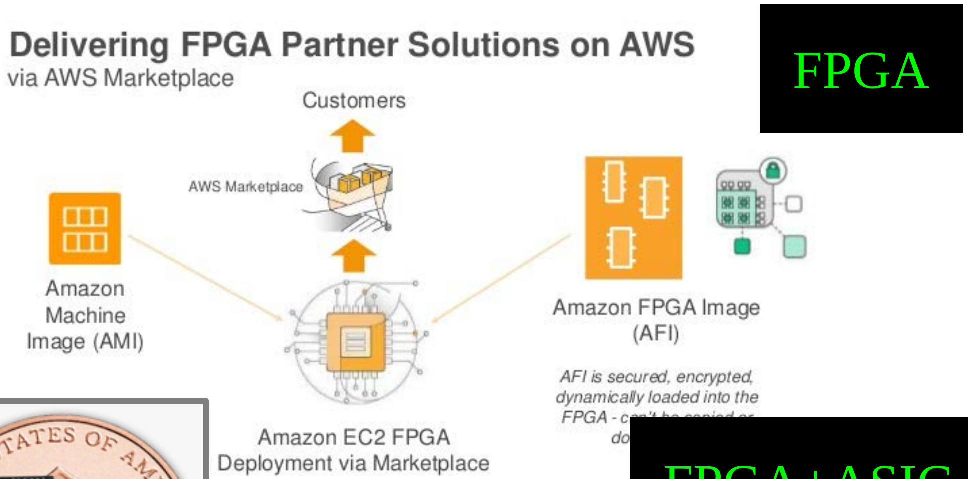


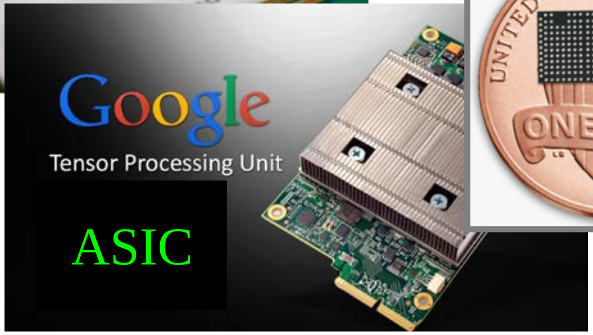
# Coprocessors: An Industry Trend

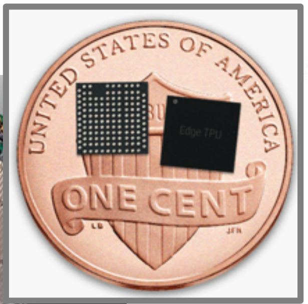
Specialized coprocessor hardware for machine learning inference



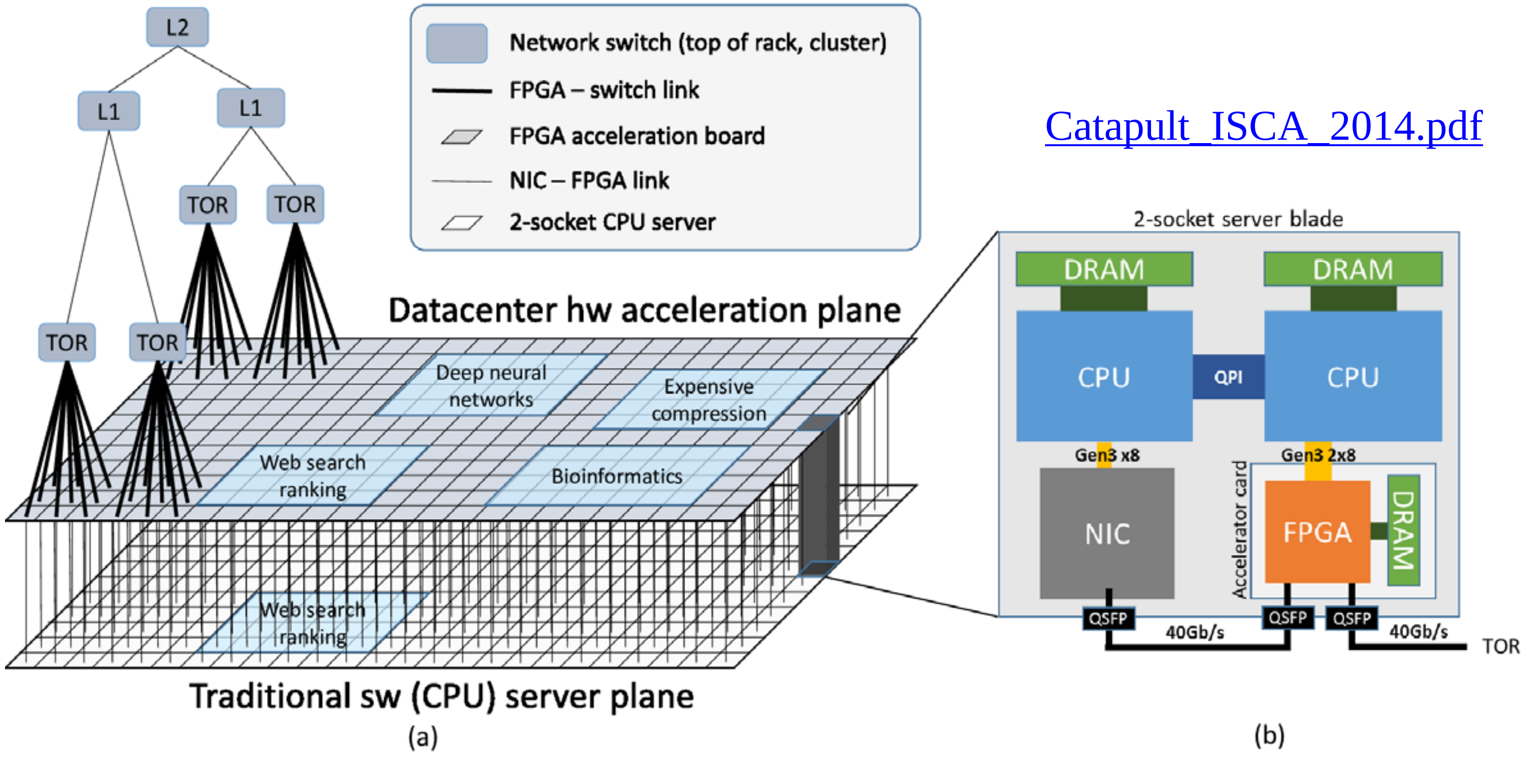



Delivering FPGA Partner Solutions on AWS
via AWS Marketplace
FPGA
Customers
AWS Marketplace
Amazon Machine Image (AMI)
Amazon FPGA Image (AFI)
AFI is secured, encrypted, dynamically loaded into the FPGA
Amazon EC2 FPGA Deployment via Marketplace





# Microsoft Brainwave

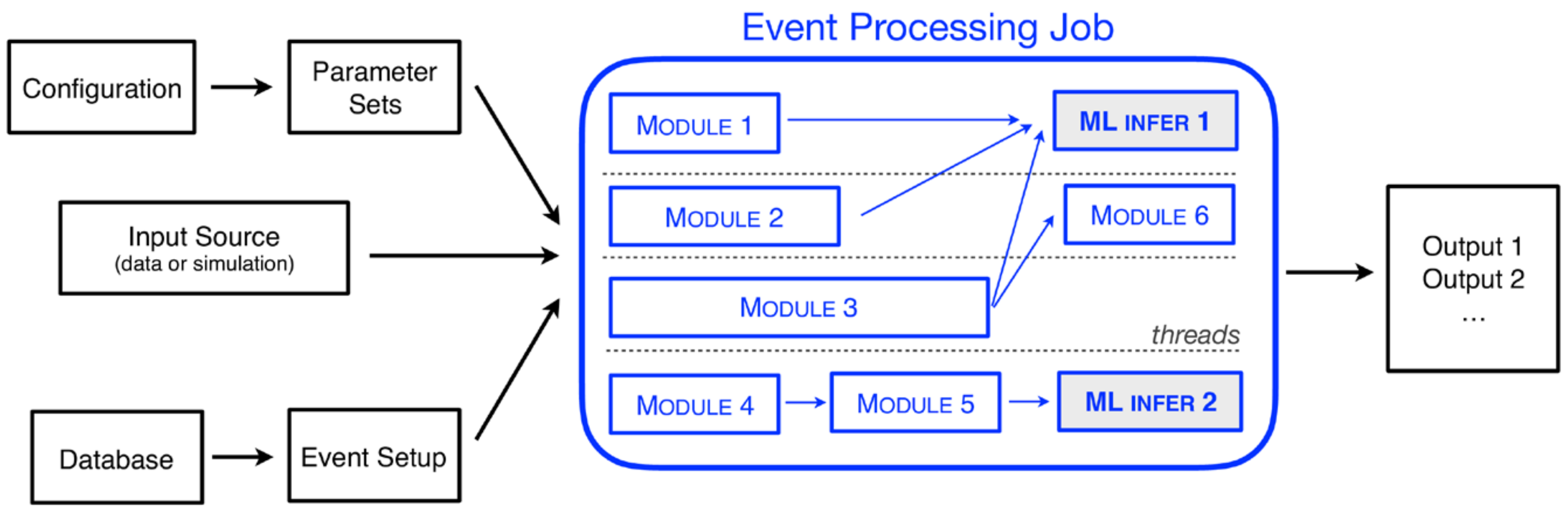


[Catapult_ISCA_2014.pdf]

- Provides a full service at scale (more than just a single co-processor)
- Multi-FPGA/CPU fabric accelerates both computing and network
- Weight retuning available: retrain supported networks to optimize for a different problem

Brainwave supports:
- ResNet50
- ResNet152
- DenseNet121
- VGGNet16

# Particle Physics Computing Model

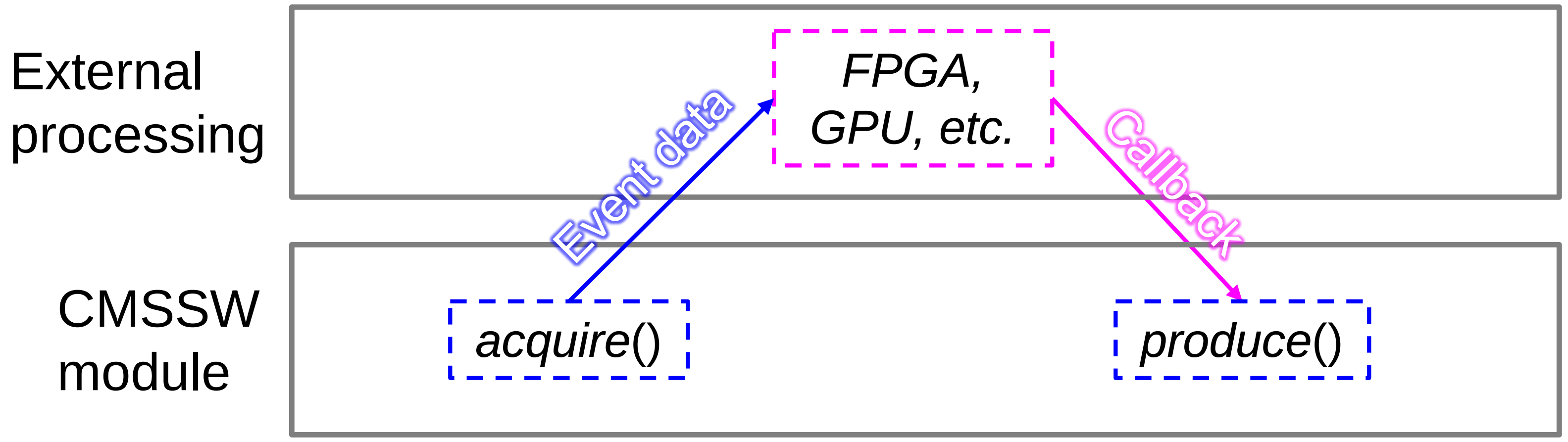


- Event-based processing
  - Events are very complex with hundreds of products
  - Load one event into memory, then execute all algorithms on it

➢ Most applications not a good fit for large batches, which are required for best GPU performance

# Accessing Heterogeneous Resources

- New CMSSW feature called ExternalWork:
  - Asynchronous task-based processing



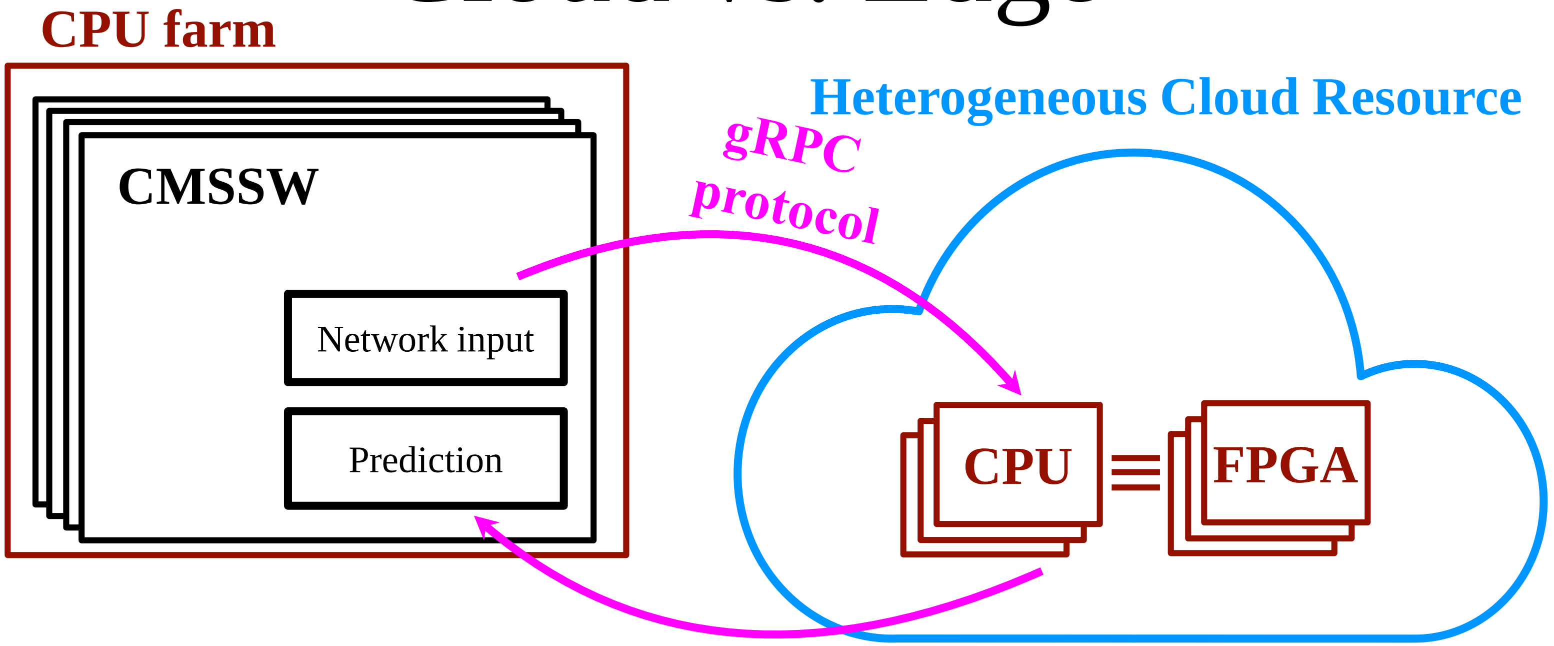
  - Non-blocking: schedule other tasks while waiting for external processing
- Can be used with GPUs, FPGAs, cloud, …
  - Even other software running on CPU that wants to schedule its own tasks

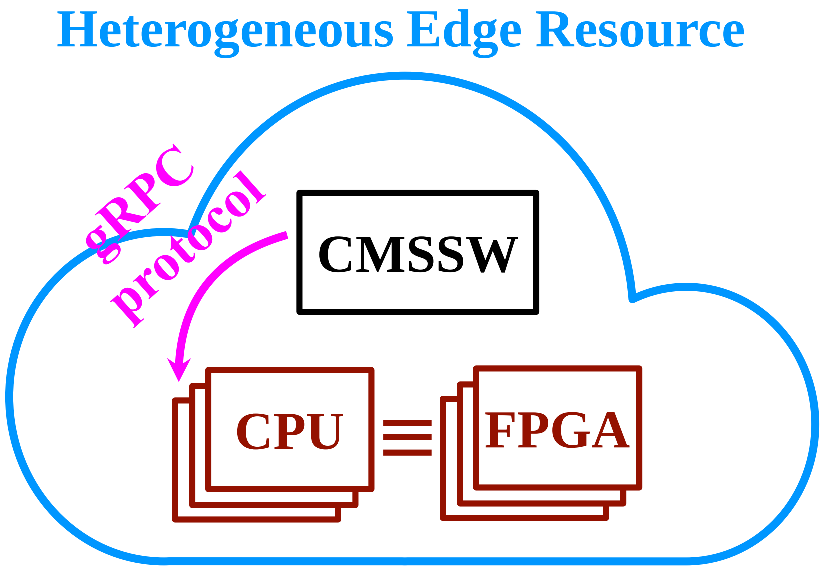
➢ Now demonstrated to work with Microsoft Brainwave!

# SONIC in CMSSW

- **S**ervices for **O**ptimized **N**etwork **I**nference on **C**oprocessors
  - Convert experimental data into neural network input
  - Send neural network input to coprocessor using communication protocol
  - Use ExternalWork mechanism for asynchronous requests
- Currently supports:
  - gRPC communication protocol
    - Callback interface for C++ API in development
      → wait for return in lightweight std::thread
  - TensorFlow w/ inputs sent as TensorProto (protobuf)
- Tested w/ Microsoft Brainwave service (cloud FPGAs)
- gRPC SonicCMS repository on GitHub

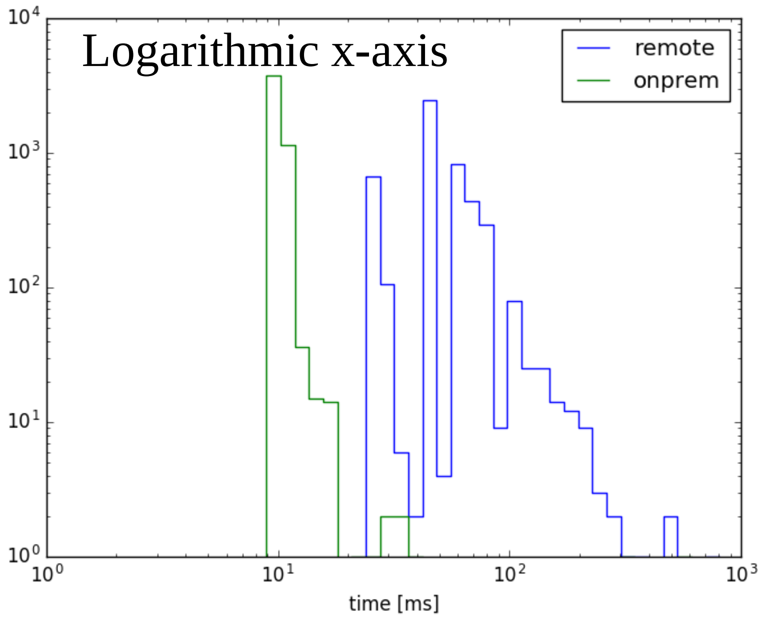
# Cloud vs. Edge

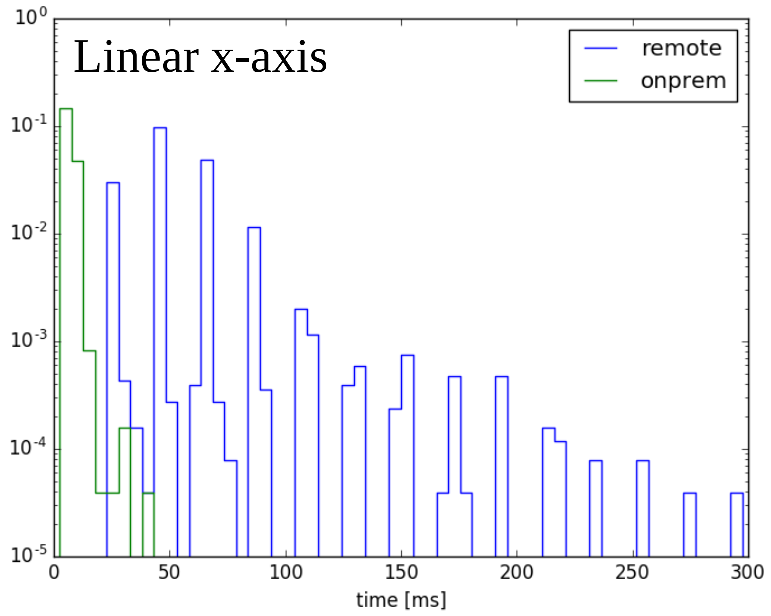




- Cloud service has latency
- Run CMSSW on Azure cloud machine → simulate local installation of FPGAs (“on-prem” or “edge”)
- Provides test of ultimate performance
- Use gRPC protocol either way

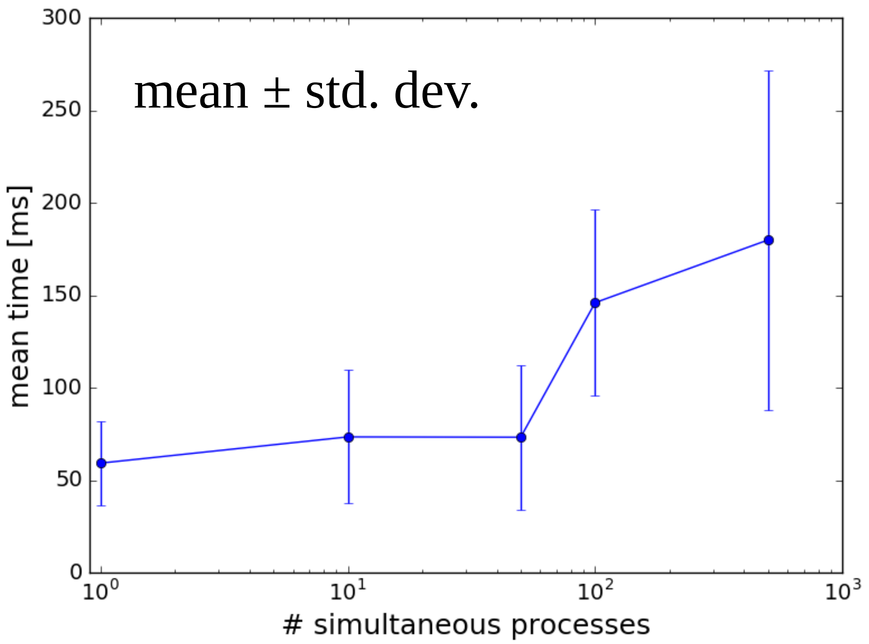
# SONIC Latency

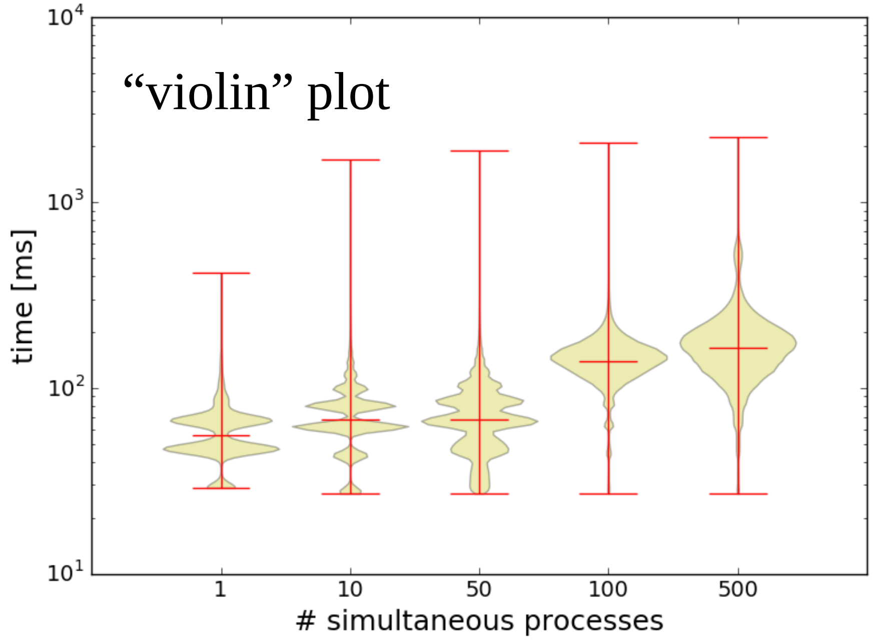




- Remote: cmslpc @ FNAL to Azure (VA), ‹time› = 60 ms
  - Highly dependent on network conditions
- On-prem: run CMSSW on Azure VM, ‹time› = 10 ms
  - FPGA: 1.8 ms for inference
  - Remaining time used for classifying and I/O

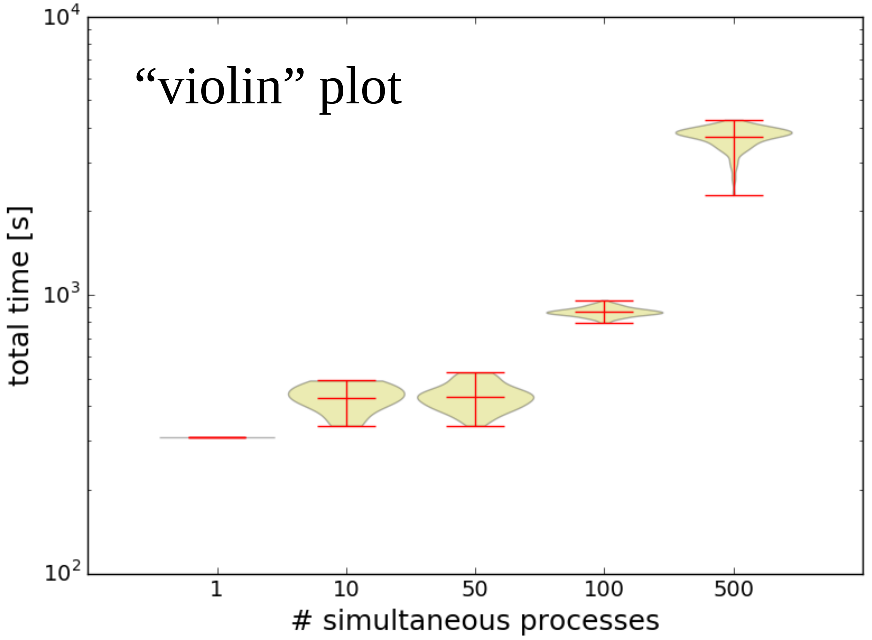
# SONIC Latency: Scaling

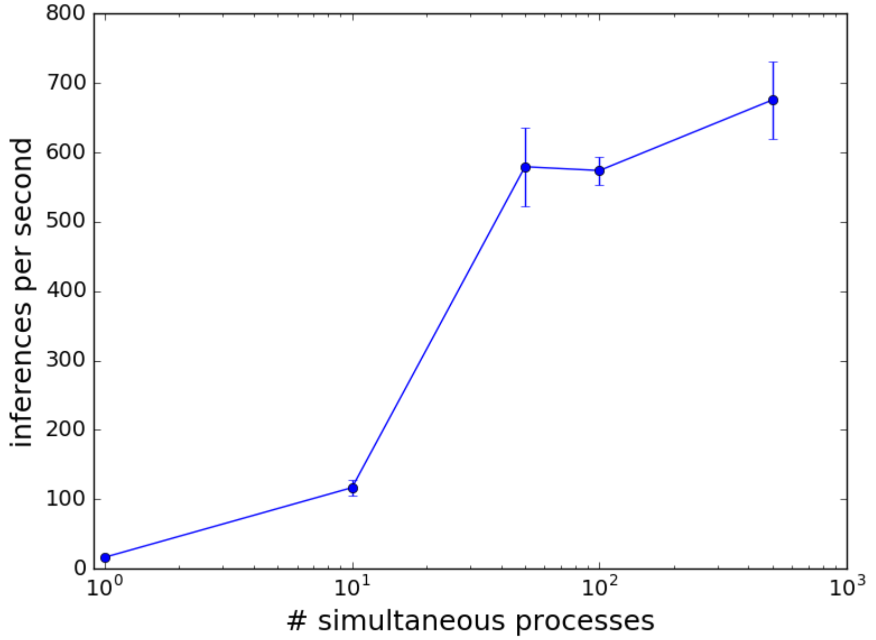




- Run N simultaneous processes, all sending requests to 1 BrainWave service
- Processes only run JetImageProducer from SONIC → "worst case" scenario
  - o Standard reconstruction process would have many other non-SONIC modules
- Only moderate increases in mean, standard deviation, and long tail for latency
  - o Fairly stable up to N = 50

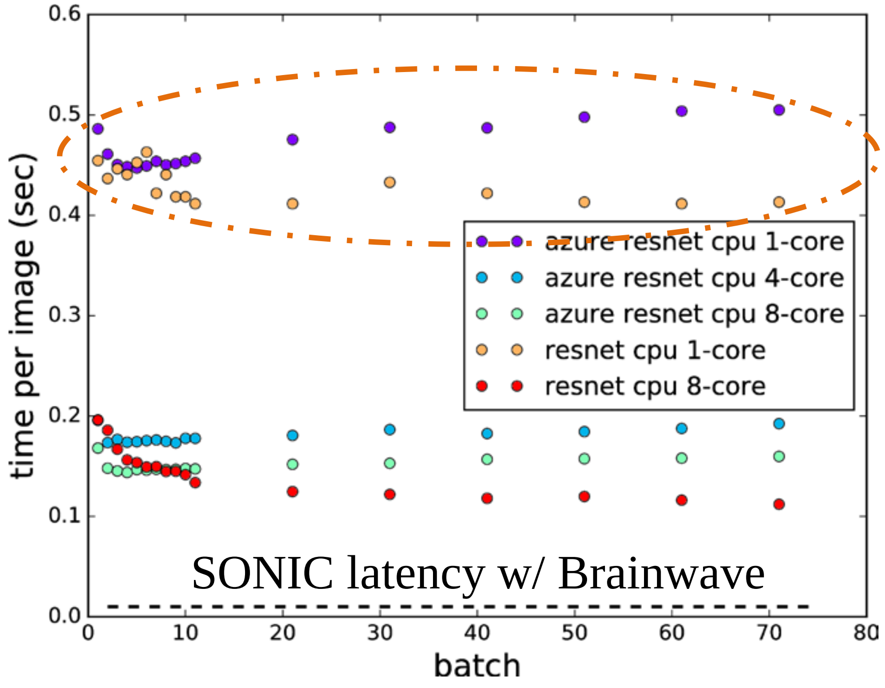
# SONIC Throughput

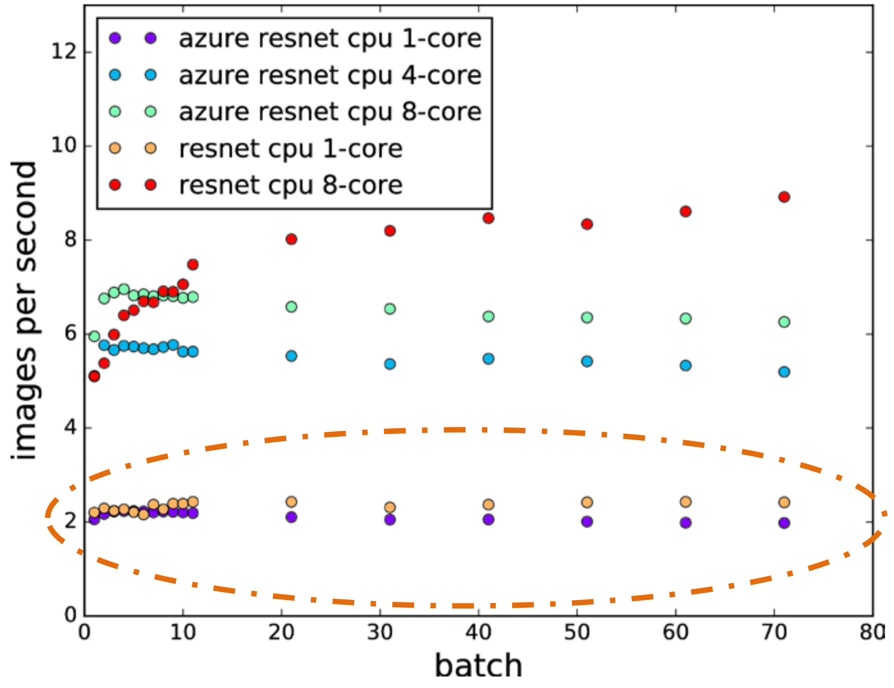




- Each process evaluates 5000 jet images in series
- Remarkably consistent total time for each process to complete
  - o Brainwave load balancer works well
- Compute inferences per second as (5000 · N)/(total time)
- N = 50 ~fully occupies FPGA:
  - o Throughput up to 600 inferences per second (max ~650)

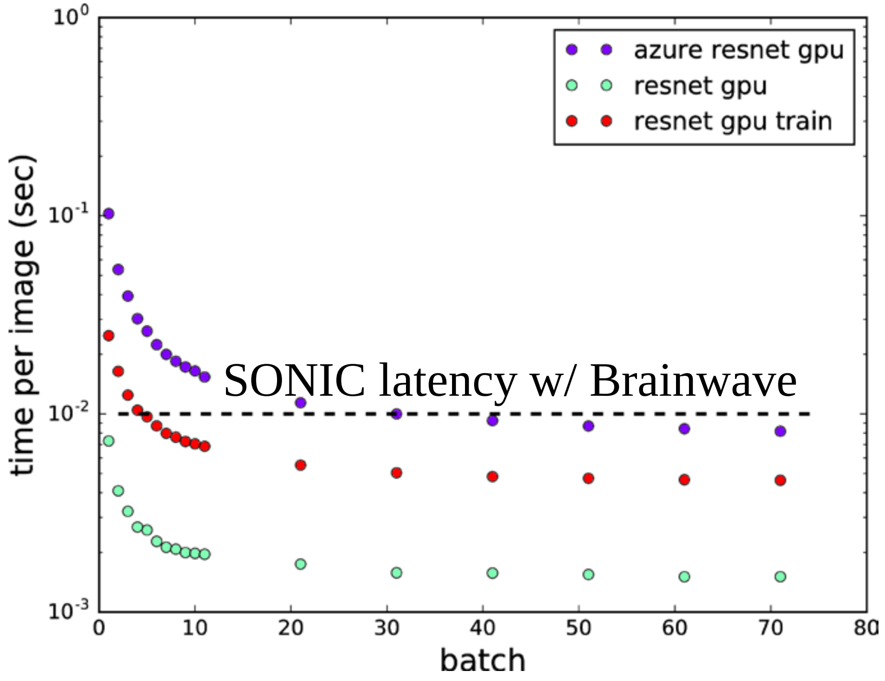
# CPU Performance

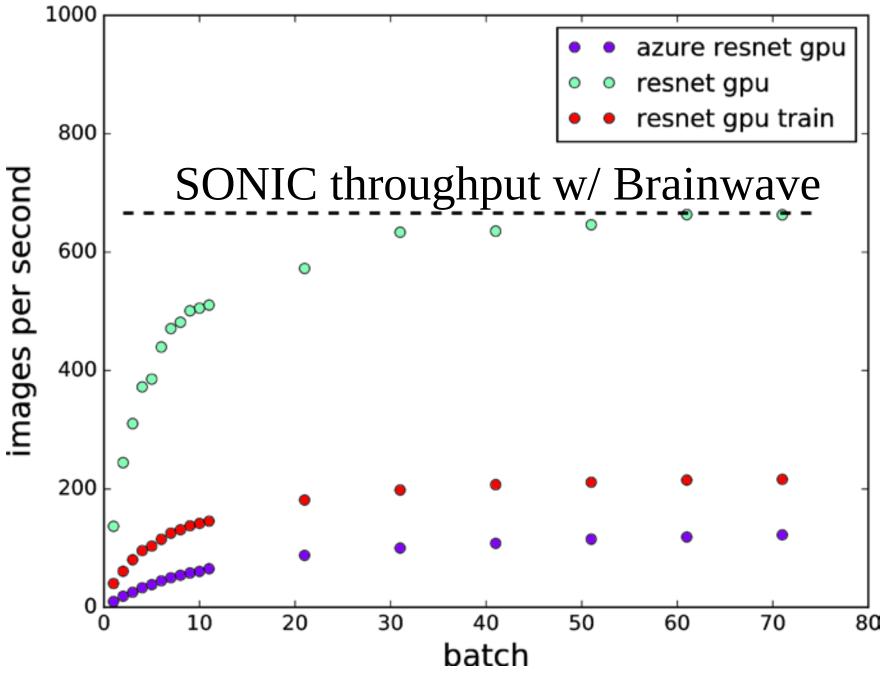




- Above plots use i7 3.6 GHz, TensorFlow v1.10
- Local test with CMSSW on cluster @ FNAL:
  - Xeon 2.6 GHz, TensorFlow v1.06
  - 5 min to import Brainwave version of ResNet-50
  - 1.75 sec/inference subsequently

# GPU Performance





- Above plots use NVidia GTX 1080, TensorFlow v1.10
- GPU directly connected to CPU via PCIe
- TF built-in version of ResNet-50 performs better on GPU than quantized version used in Brainwave

# Performance Comparisons

| Type | Note | Latency [ms] | Throughput [img/s] |
|---|---|---|---|
| CPU* | Xeon 2.6 GHz | 1750 | 0.6 |
| | i7 3.6 GHz | 500 | 2 |
| GPU** | batch = 1 | 7 | 143 |
| | batch = 32 | 1.5 | 667 |
| Brainwave | remote | 60 | 660 |
| | on-prem | 10 (1.8 on FPGA) | 660 |

- *CPU performance depends on:
  - clock speed, TensorFlow version, # threads (=1 here)
- **GPU caveats:
  - Directly connected to CPU via PCIe – not a service
  - Performance depends on batch size & optimization of ResNet-50 network
- SONIC achieves:
  - 175× (30×) on-prem (remote) improvement in latency vs. CMSSW CPU!
  - Competitive throughput vs. GPU, w/ single-image batch as a service!

# Summary

- Particle physics experiments face extreme computing challenges
  - More data, more complex detectors, more pileup
- Growing interest in machine learning for reconstruction and analysis
  - As networks get larger, inference takes longer
- FPGAs are a promising option to accelerate neural network inference
  - Can achieve order of magnitude improvement in latency over CPU
  - Comparable throughput to GPU, without batching
    - ➢ Better fit for event-based computing model
- SONIC infrastructure developed and tested
  - Compatible with any service that uses gRPC and TensorFlow

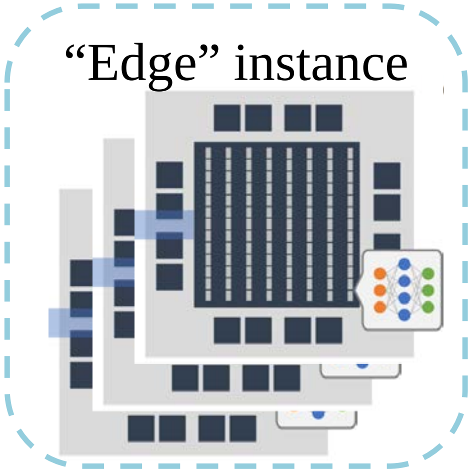
➢ Paper with these results in preparation

- Thanks to Microsoft for lots of help and advice!
  - Azure Machine Learning, Bing, Project Brainwave teams
  - Doug Burger, Eric Chung, Jeremy Fowers, Kalin Ovtcharov, Andrew Putnam

# Continuing Work

- **Continue to translate particle physics algorithms into machine learning**
  - Easier to accelerate inference w/ commercial coprocessors
- **Develop tools for generic model translation**
  - E.g. graph NNs used for HEP.TrkX and other projects
- **Explore broad offering of potential hardware**
  - Google TPUs, Xilinx ML suite on AWS, Intel OpenVINO, …
- **Continue to build infrastructure and study scalability/cost**
  - Adapt SONIC to handle other protocols, other network architectures and ML libraries, other experiments (e.g. neutrinos)

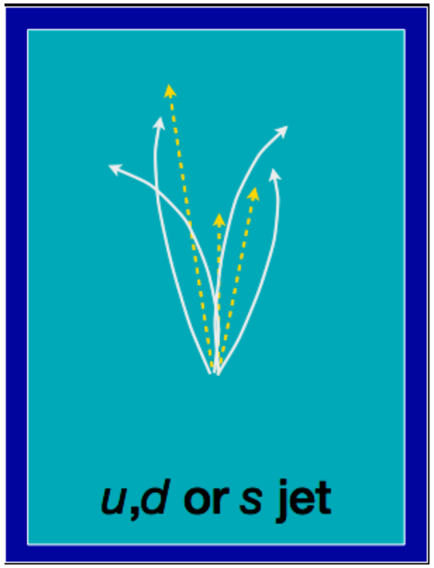
# A Vision of the Future

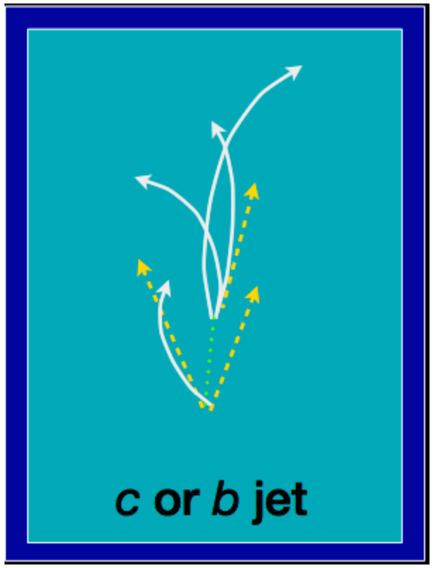


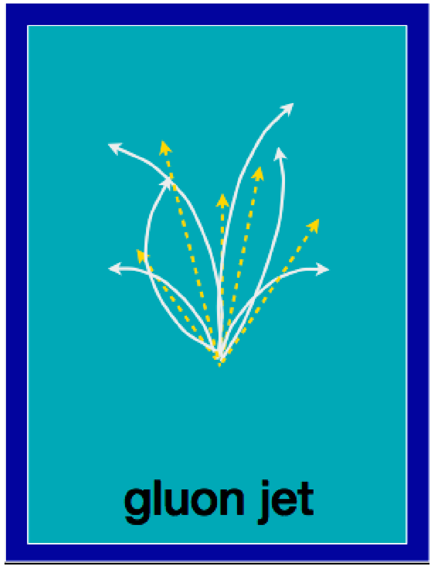
- A single FPGA can support many CPUs → cost-effective
  - SONIC throughput results indicate 1 FPGA for 100–1000 CPUs running realistic processes (many algorithms, only some ML inferences)
- Install small "edge" instances at T1s and T2s
  - Can also install a dedicated instance for CMS HLT farm at CERN

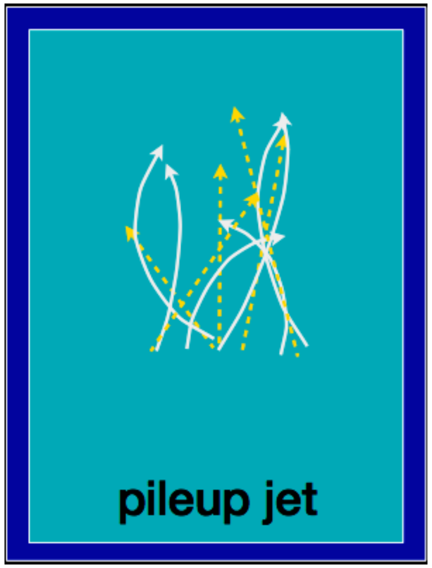
## Backup

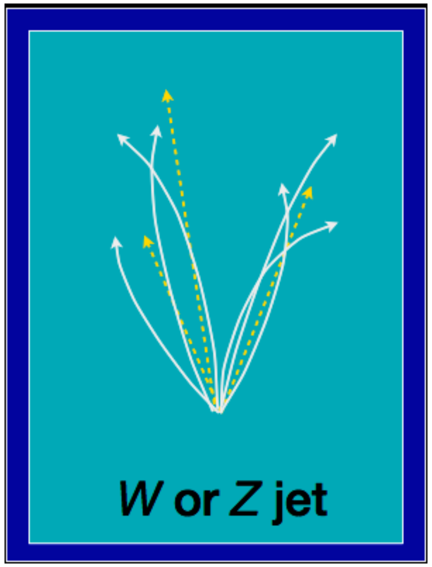
# Jet Substructure

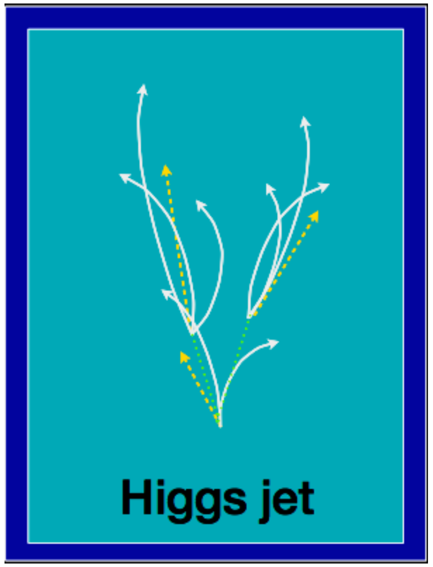


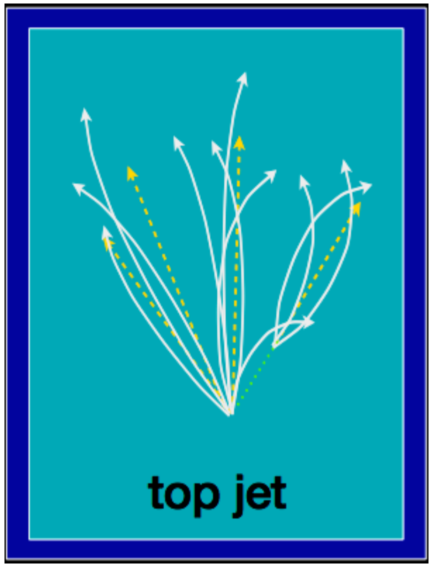


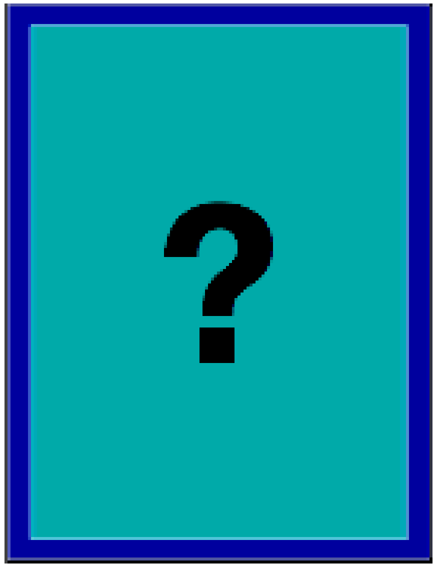




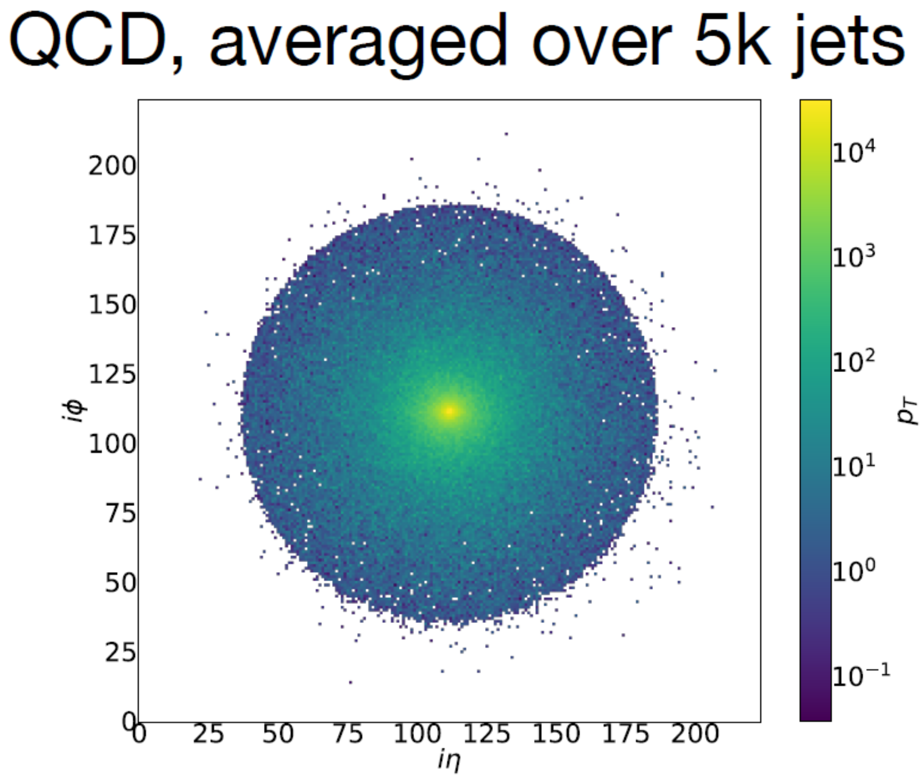


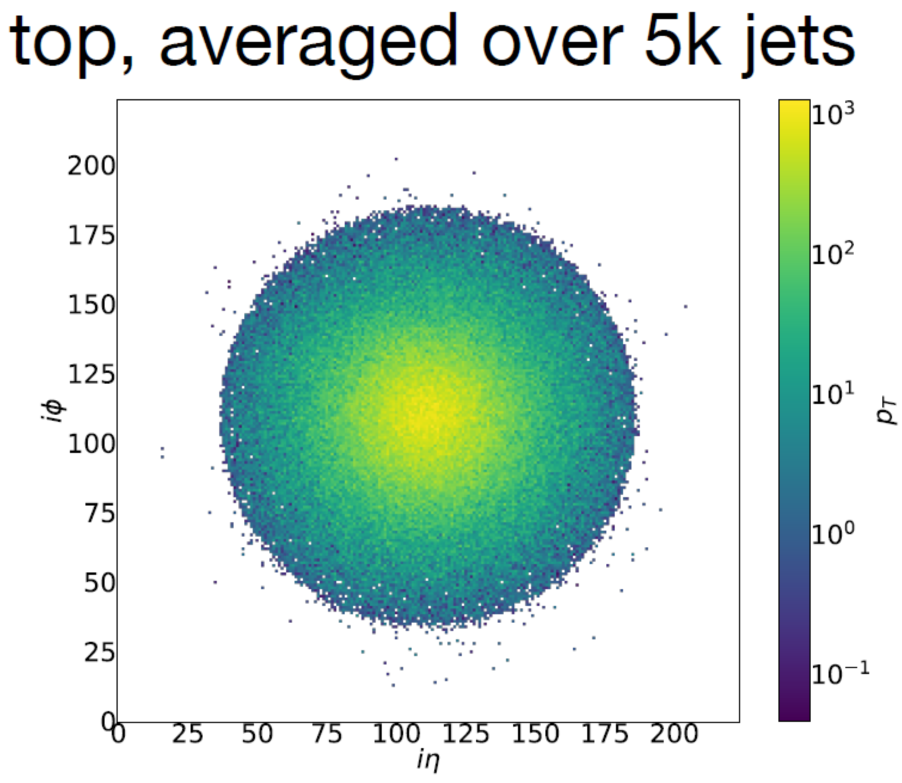






# Jet Images

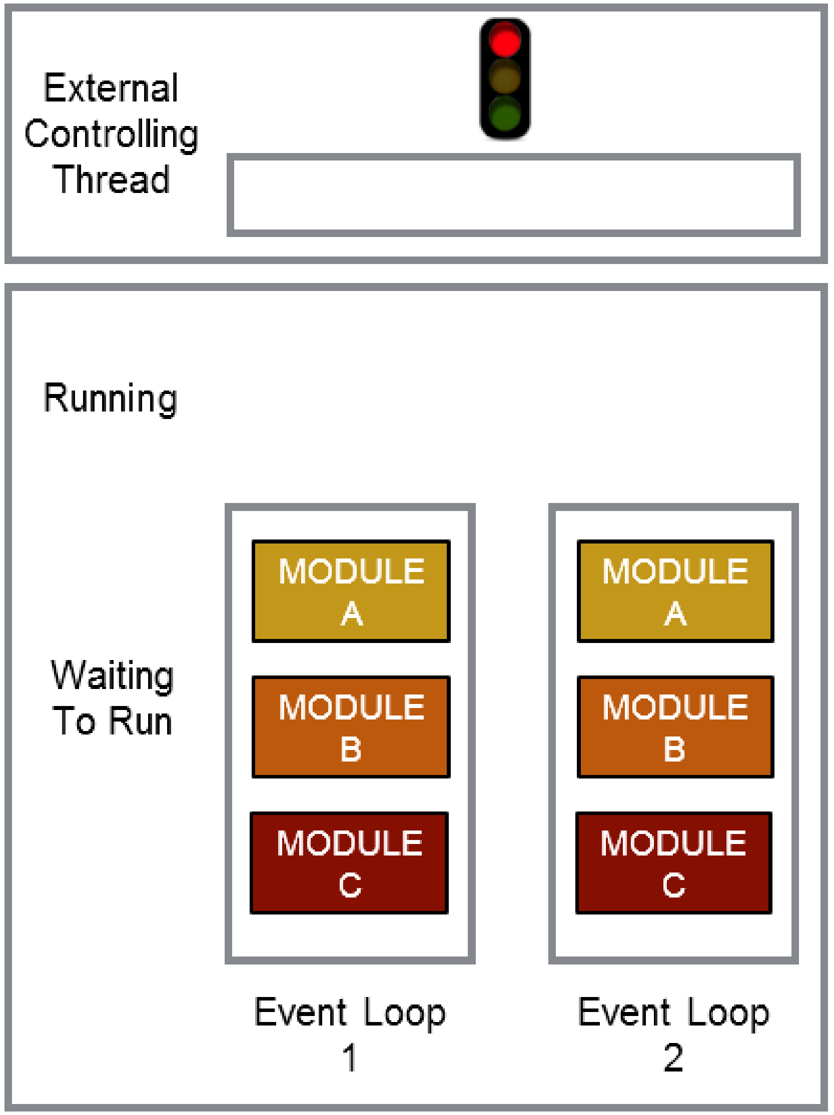

QCD, averaged over 5k jets
$i\phi$
$i\eta$
$p_T$
0
25
50
75
100
125
150
175
200
$10^{-1}$
$10^0$
$10^1$
$10^2$
$10^3$
$10^4$


top, averaged over 5k jets
$i\phi$
$i\eta$
$p_T$
0
25
50
75
100
125
150
175
200
$10^{-1}$
$10^0$
$10^1$
$10^2$
$10^3$

# External Work in CMSSW (1)

Setup:

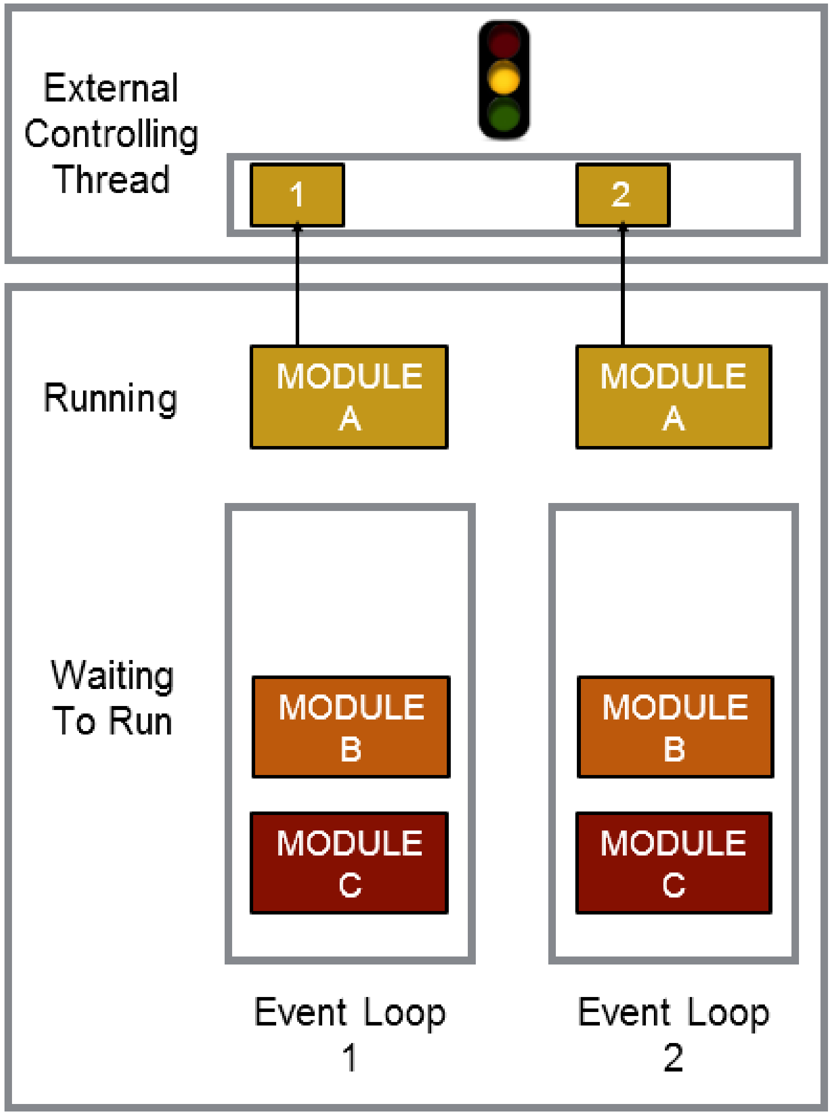
- TBB controls running modules
- Concurrent processing of multiple events
- Separate helper thread to control external
- Can wait until enough work is buffered before running external process



# External Work in CMSSW (2)

Acquire:

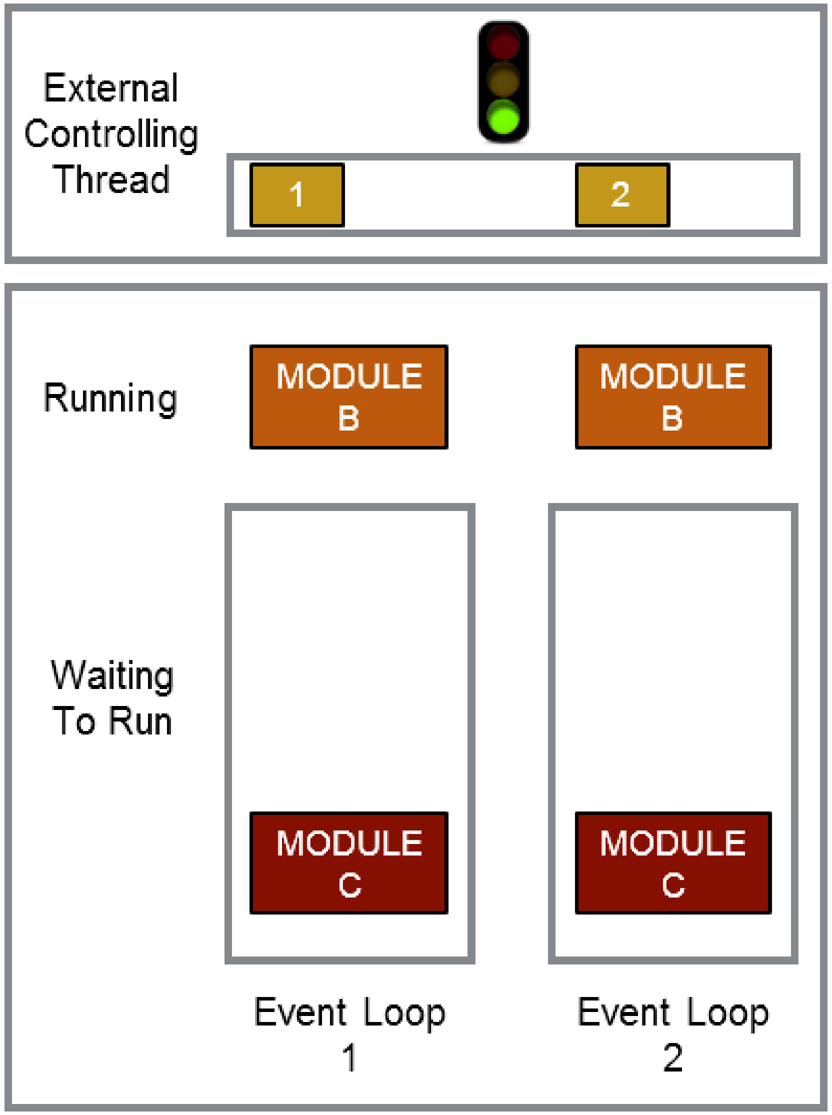
- Module *acquire*() method called
- Pulls data from event
- Copies data to buffer
- Buffer includes callback to start next phase of module running



# External Work in CMSSW (3)

Work starts:

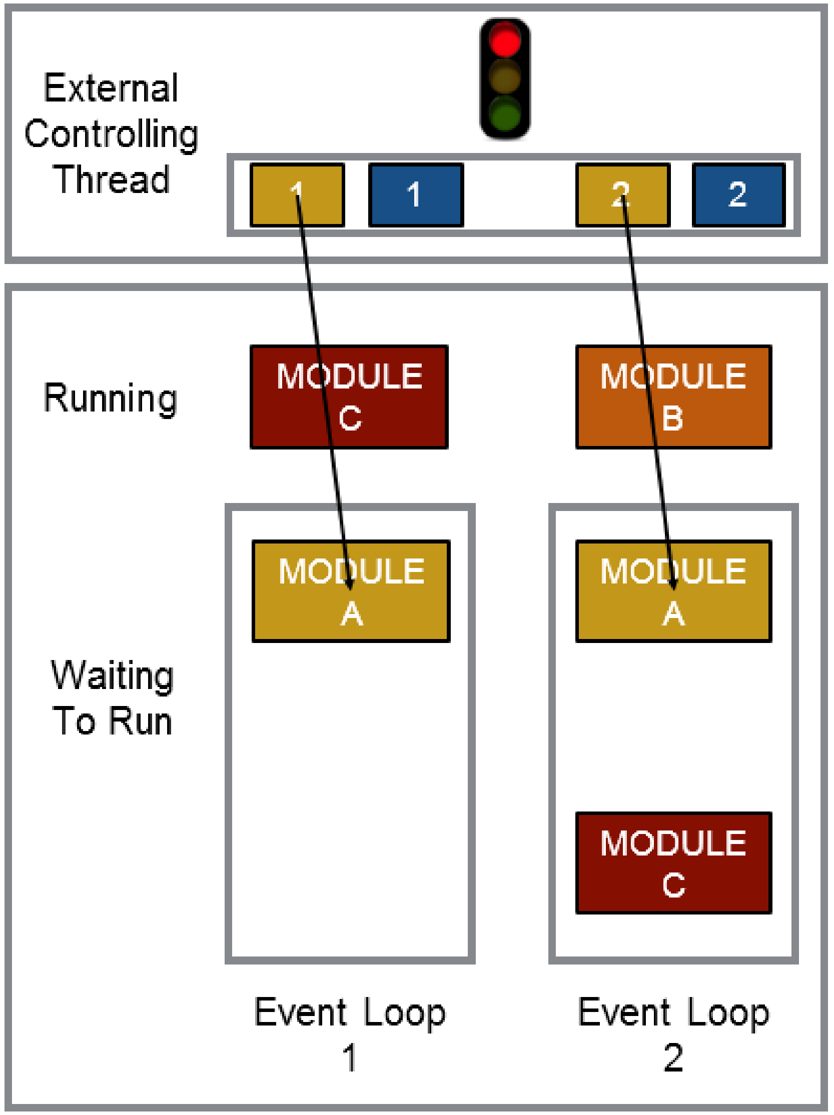
- External process runs
- Data pulled from buffer
- Next waiting modules can run (concurrently)



# External Work in CMSSW (4)

Work finishes:

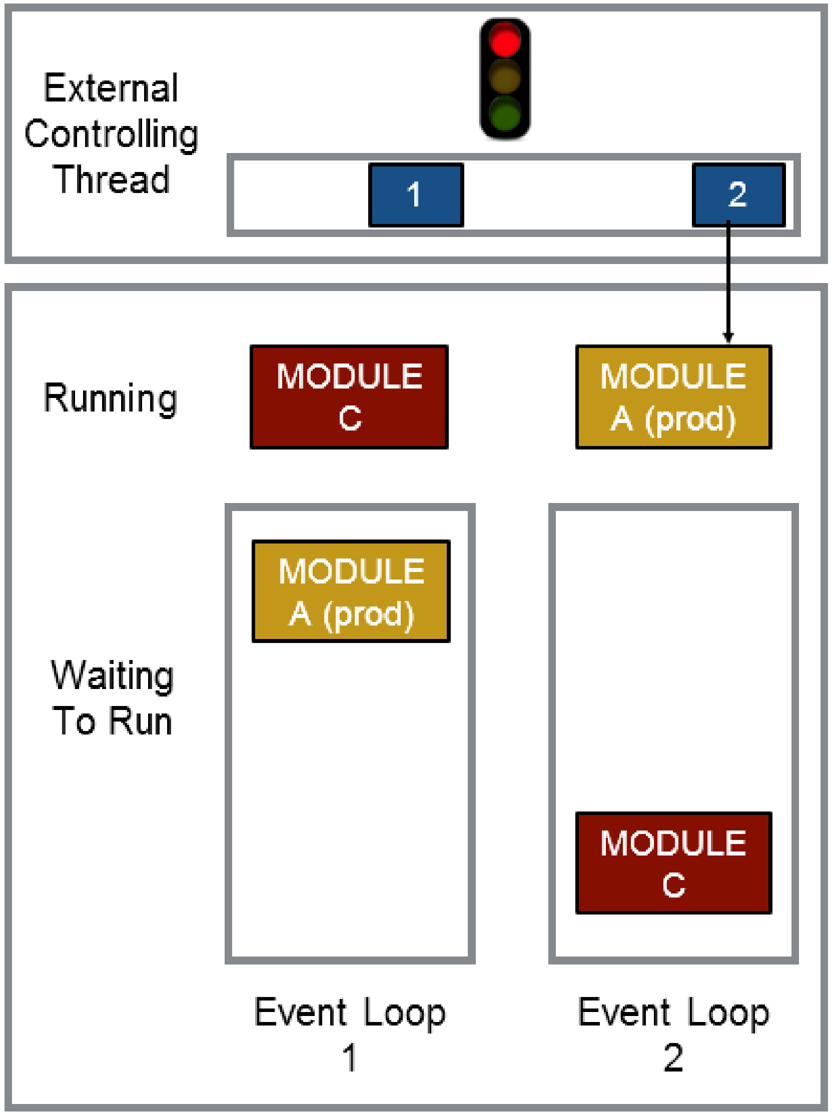
- Results copied to buffer
- Callback puts module back into queue



# External Work in CMSSW (5)

Produce:

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


External Controlling Thread
1
2
Running
MODULE C
MODULE A (prod)
Waiting To Run
MODULE A (prod)
MODULE C
Event Loop 1
Event Loop 2