# SONIC Inference as-a-Service for Point Clouds

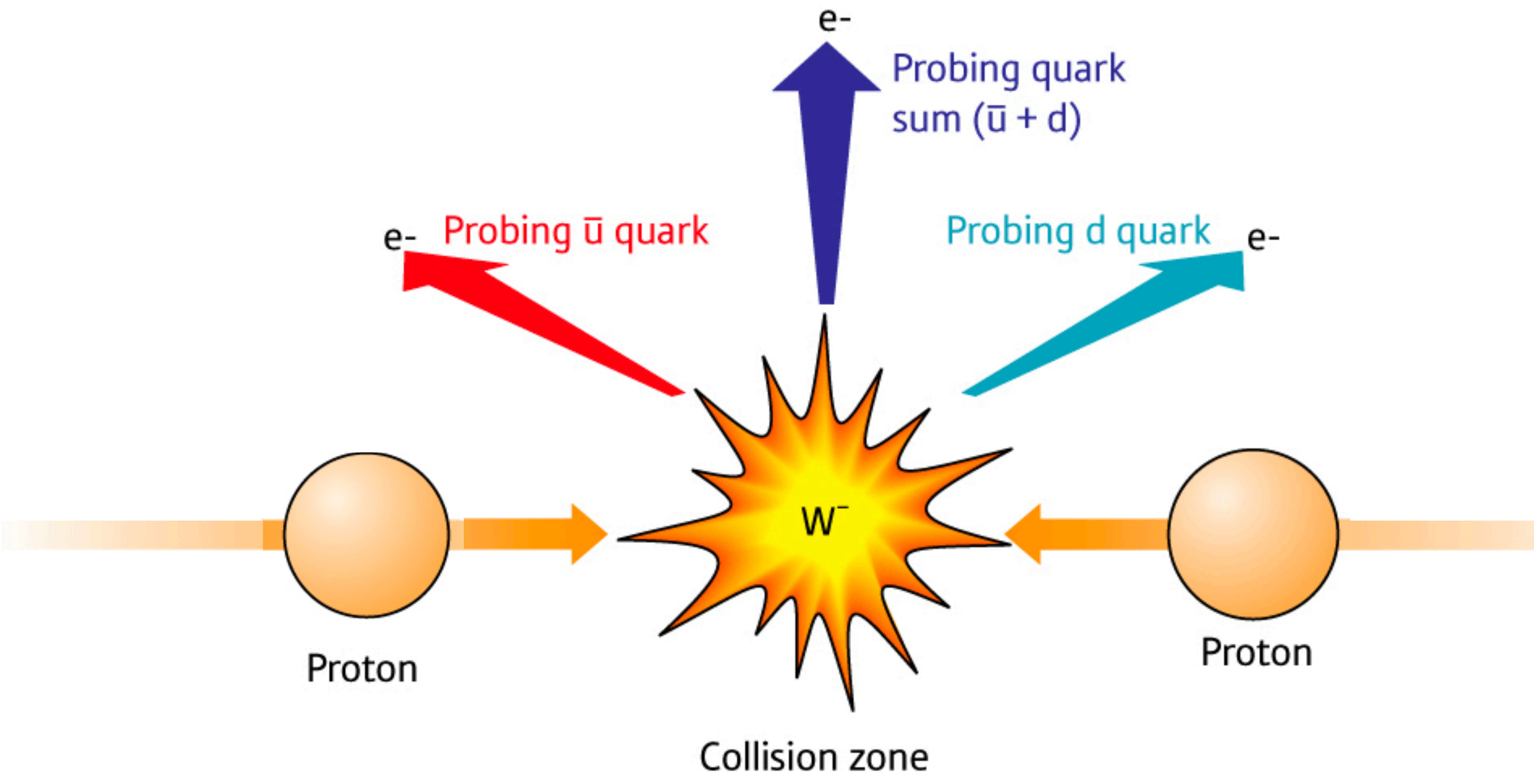
Yongbin Feng (Fermilab)

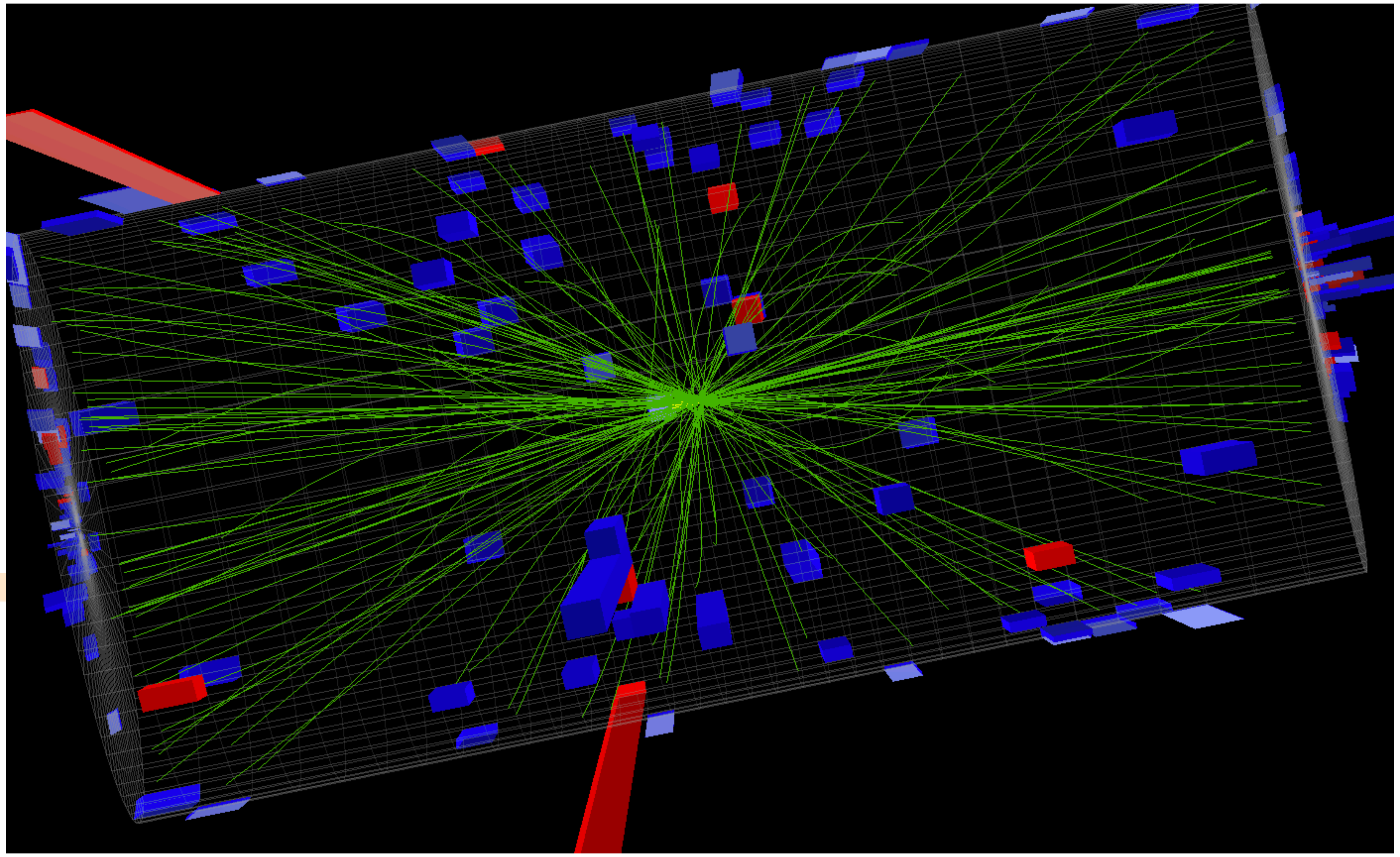
Accelerating Physics with ML @ MIT

Cambridge, MA

Jan. 30th, 2023

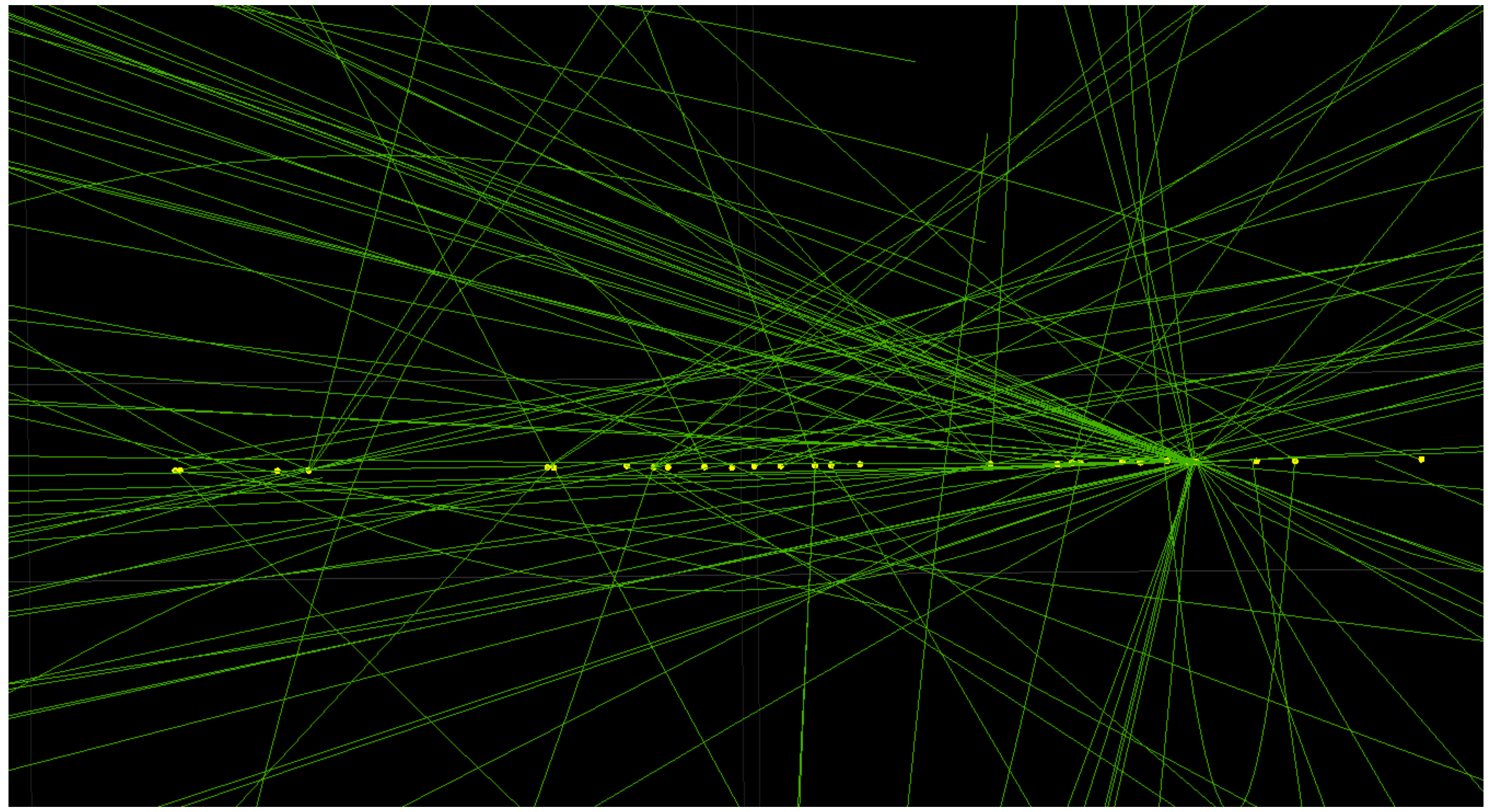
# Proton Collisions

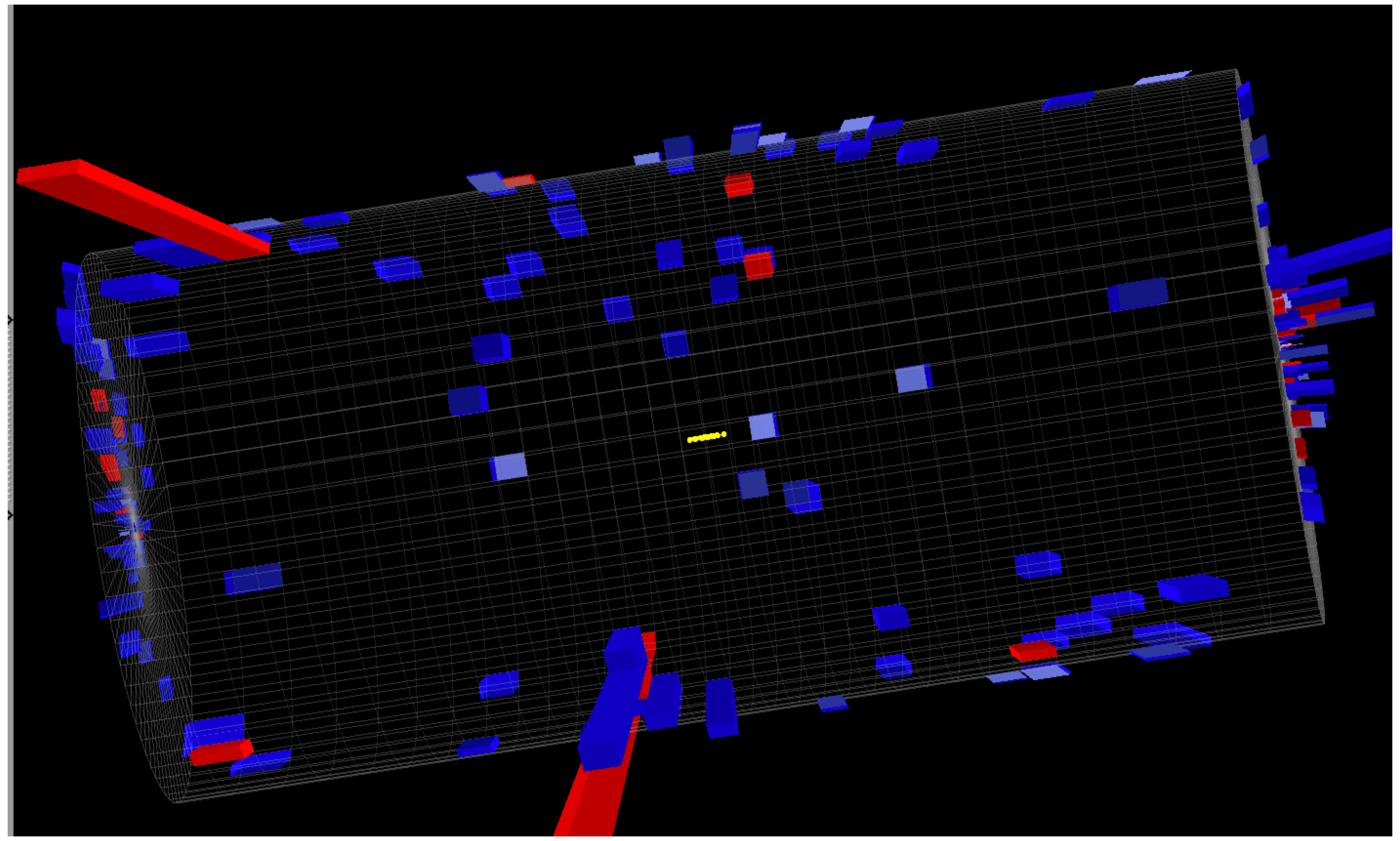


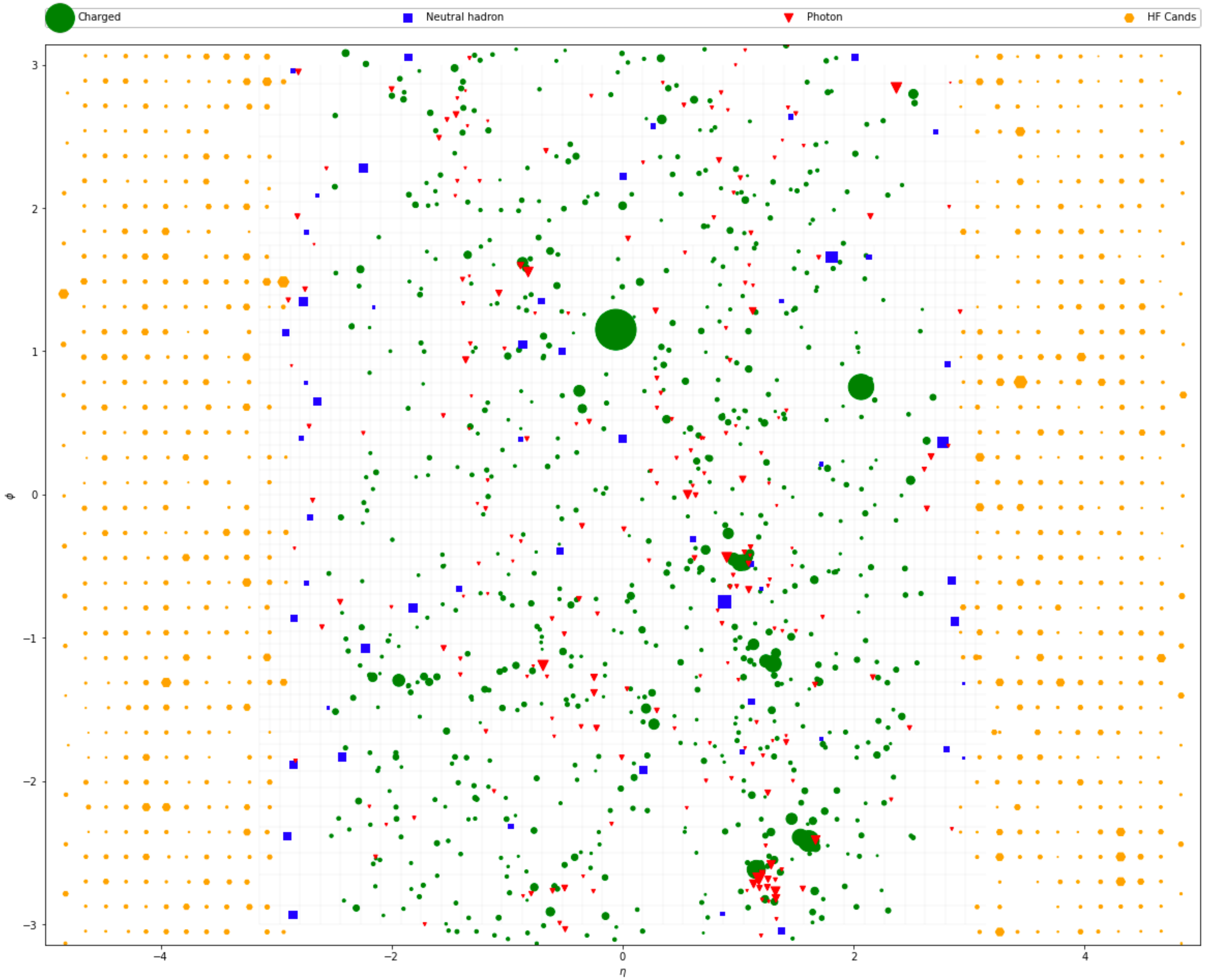



- Proton-proton collisions
- Different particles produced from the collisions:
  - charged, neutral particles (neutral hadrons, photons, etc)
  - Each particle carries its own position and kinematic information

# Point Cloud Data

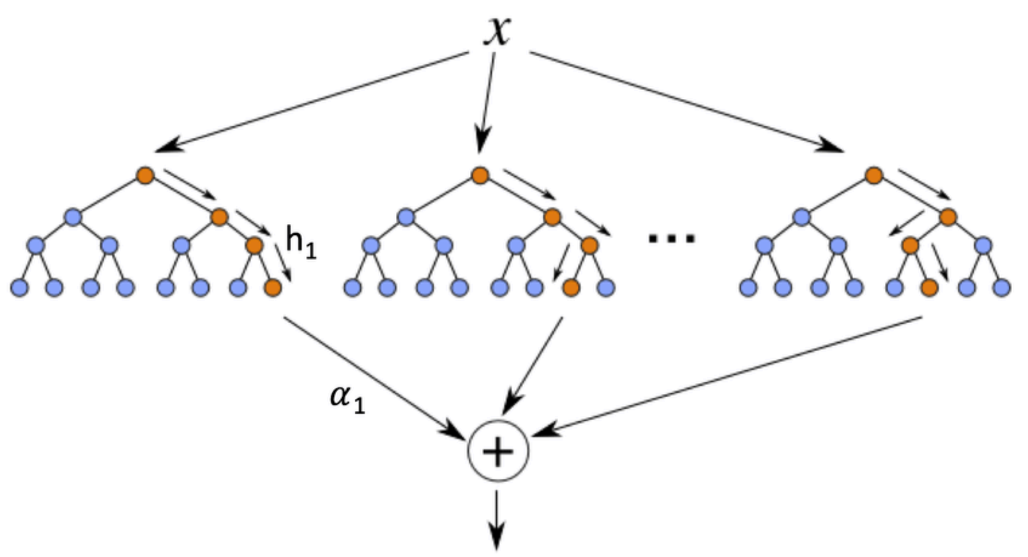


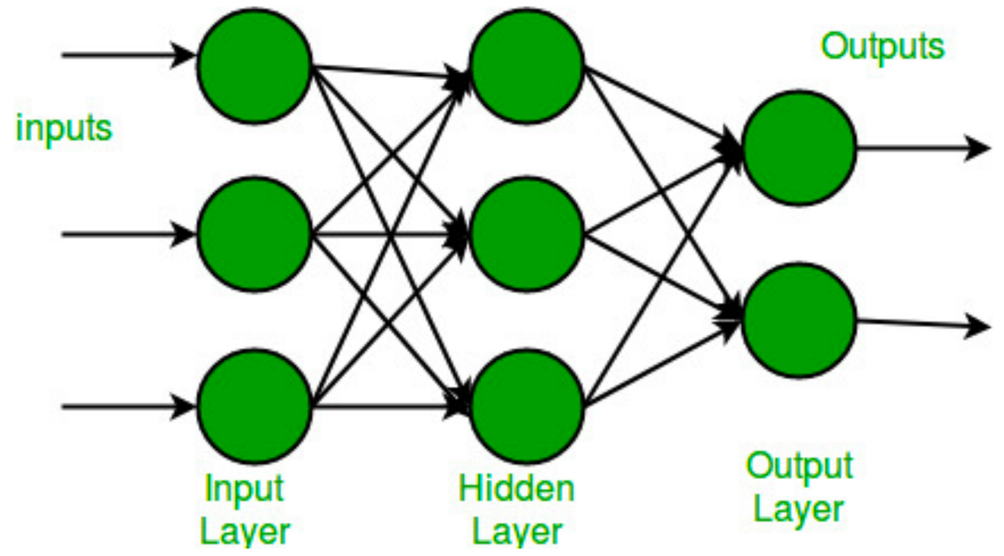


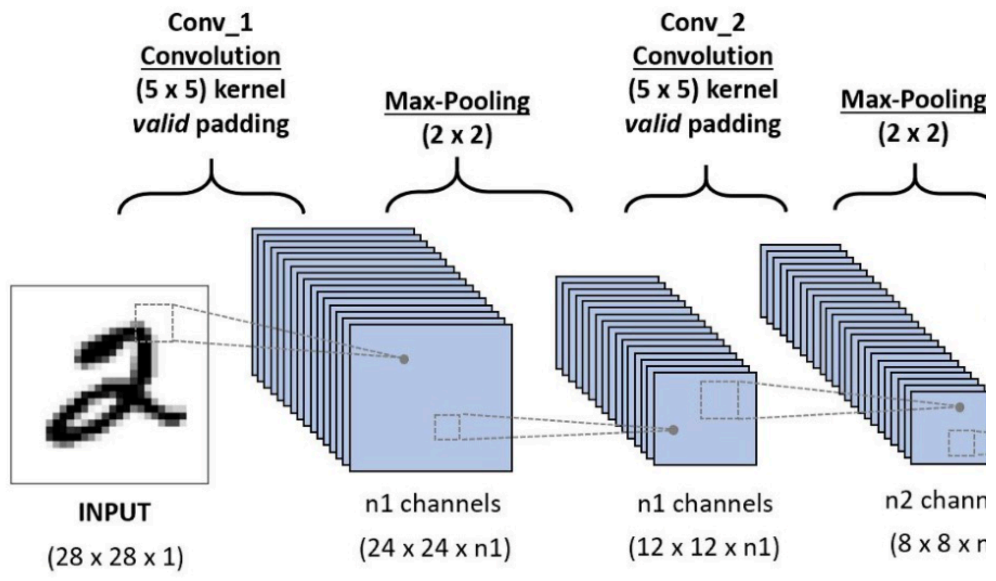


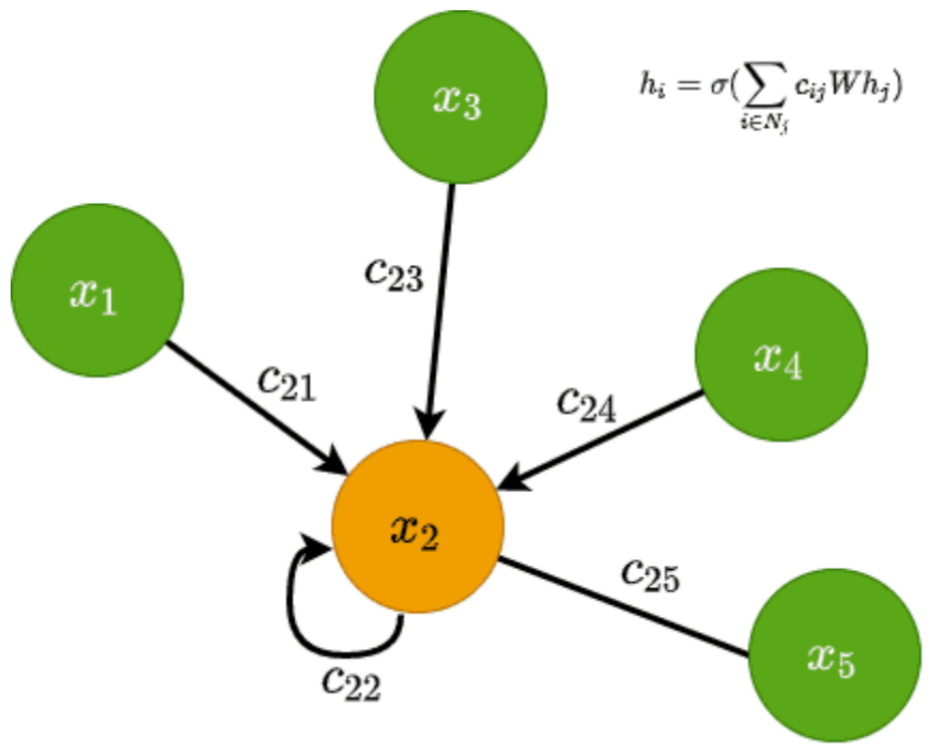
# ML Algorithm Developments

Boost Decision Trees → (Fully-Connected) Deep-Neural-Network (Multi Layer Perceptron) → Convolutional Neural Network → Graph Neural Network



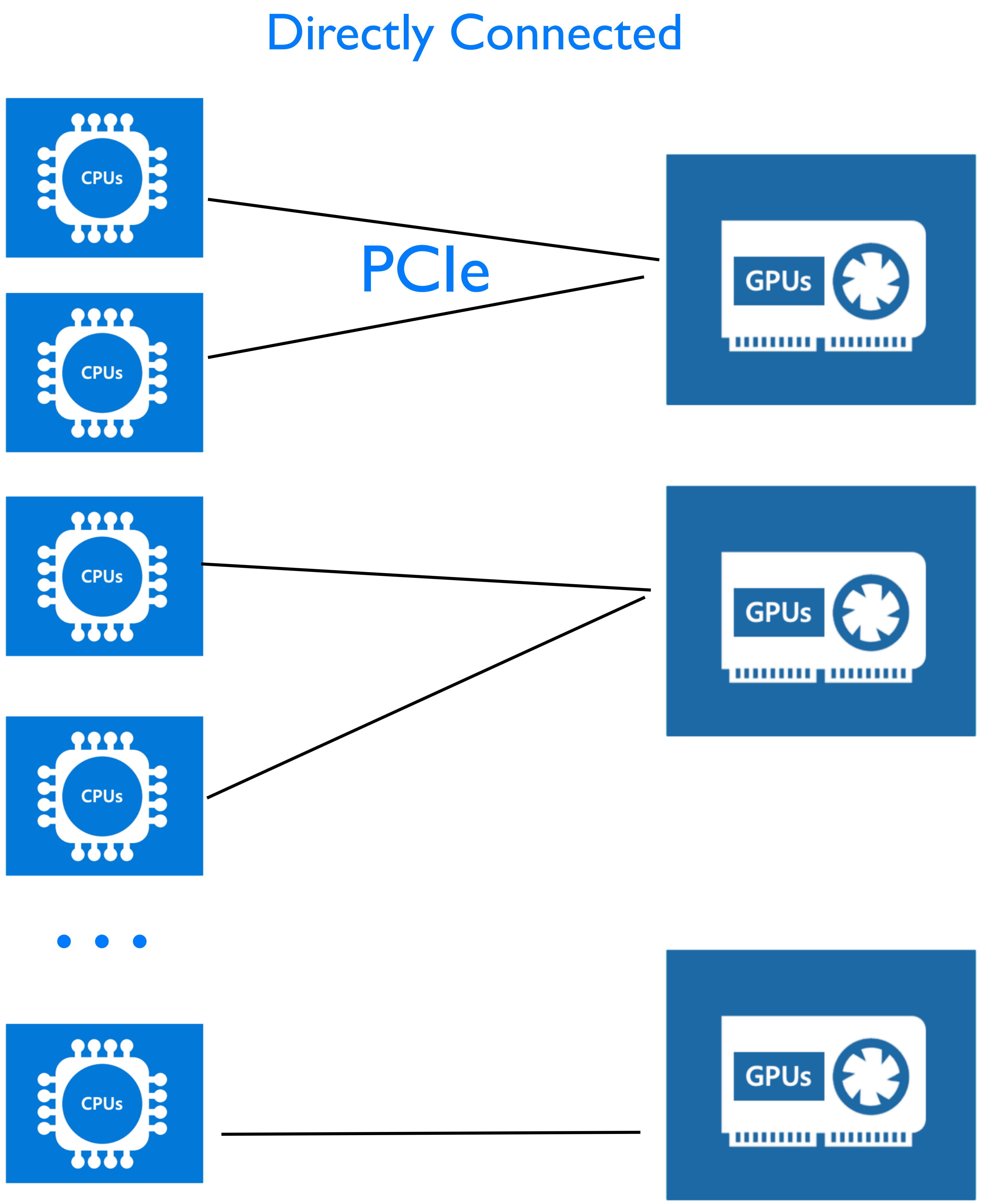



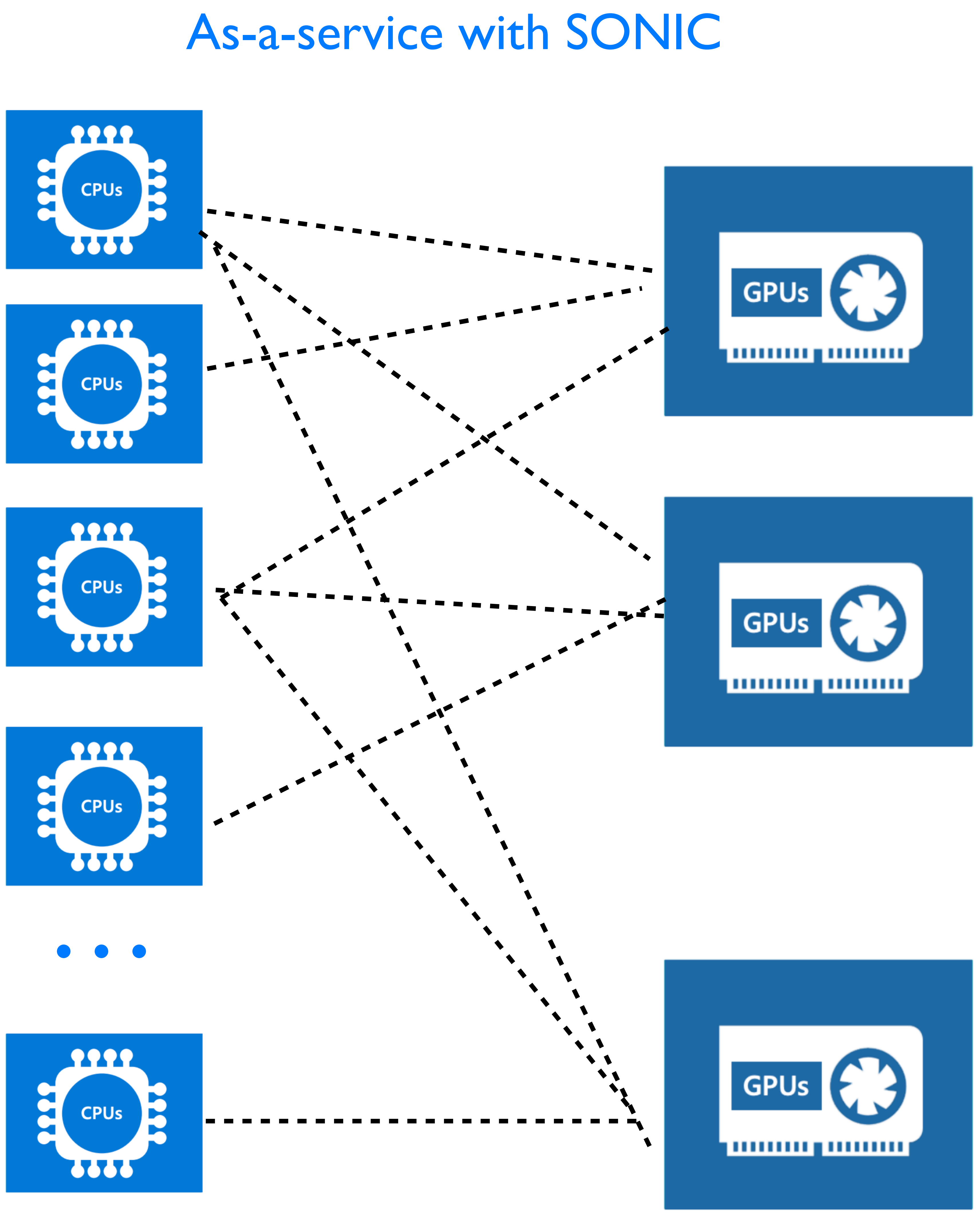




- ML algorithms explored and deployed in LHC/HEP experiments:
  - Object (jet/tau/etc) tagging, signal/background discrimination, track/calorimeter reconstruction, trigger, etc
  - Architectures get more complicated; networks get deeper; and the performances get better and better
  - So does the computing time

# Deploying Coprocessors for High Throughput

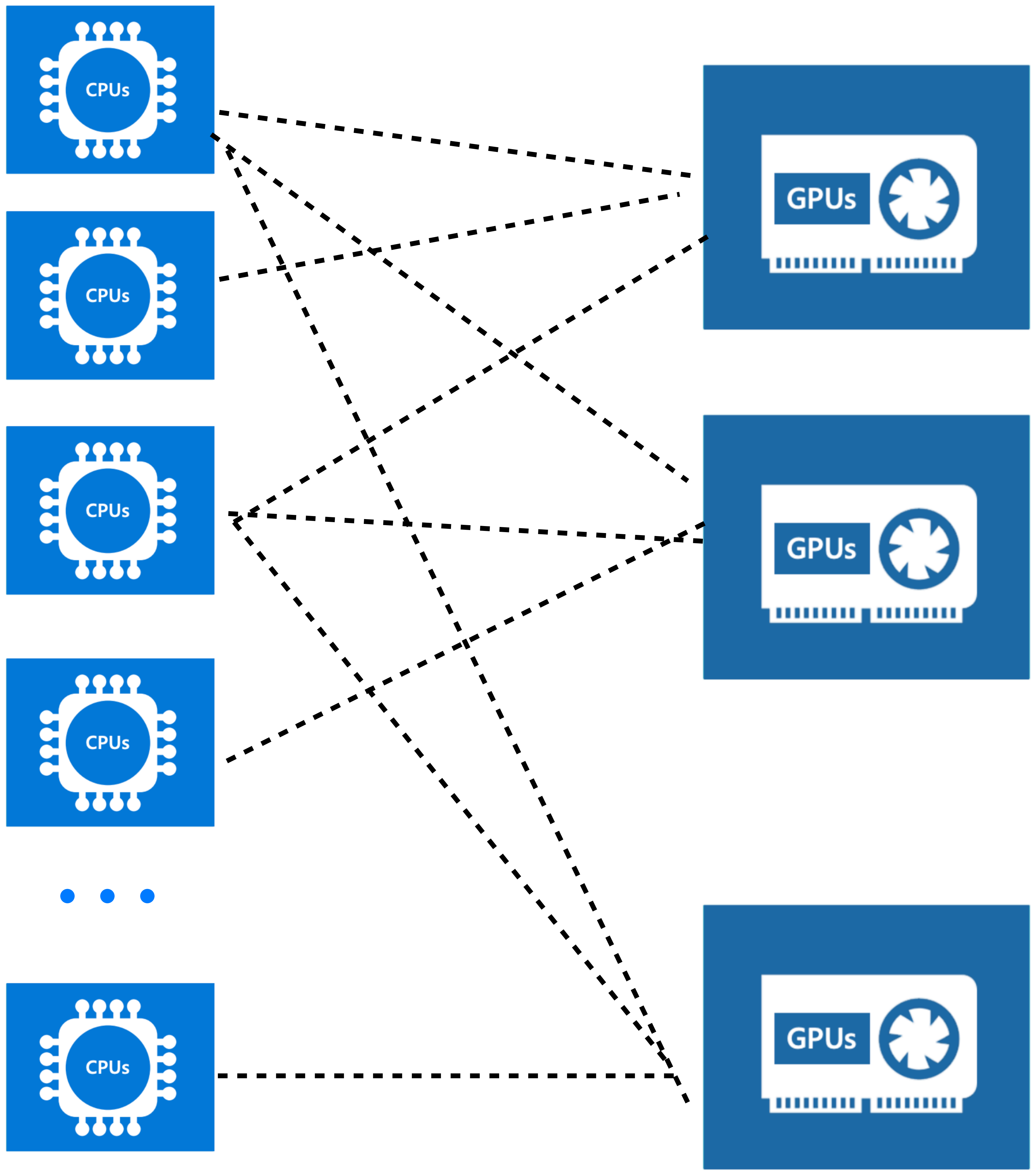


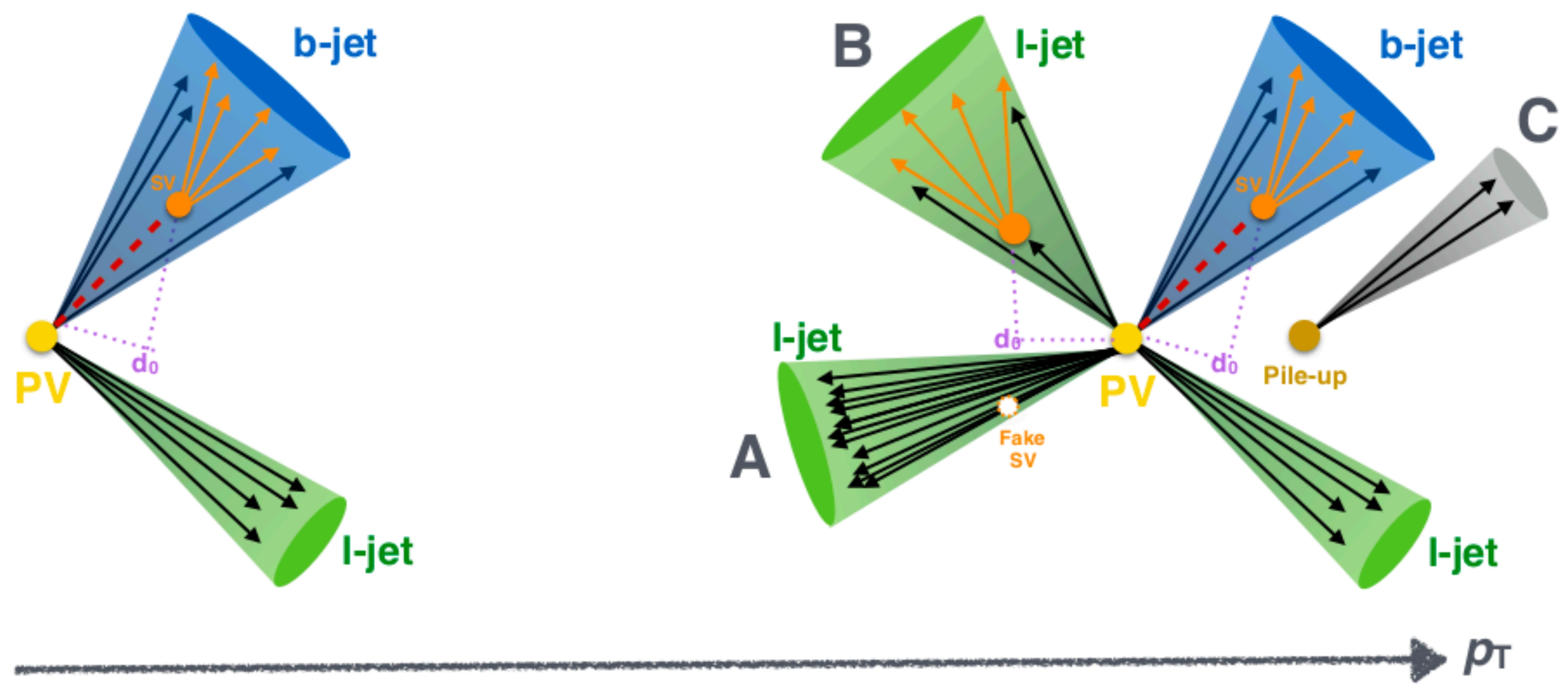
# Deploying Coprocessors for High Throughput

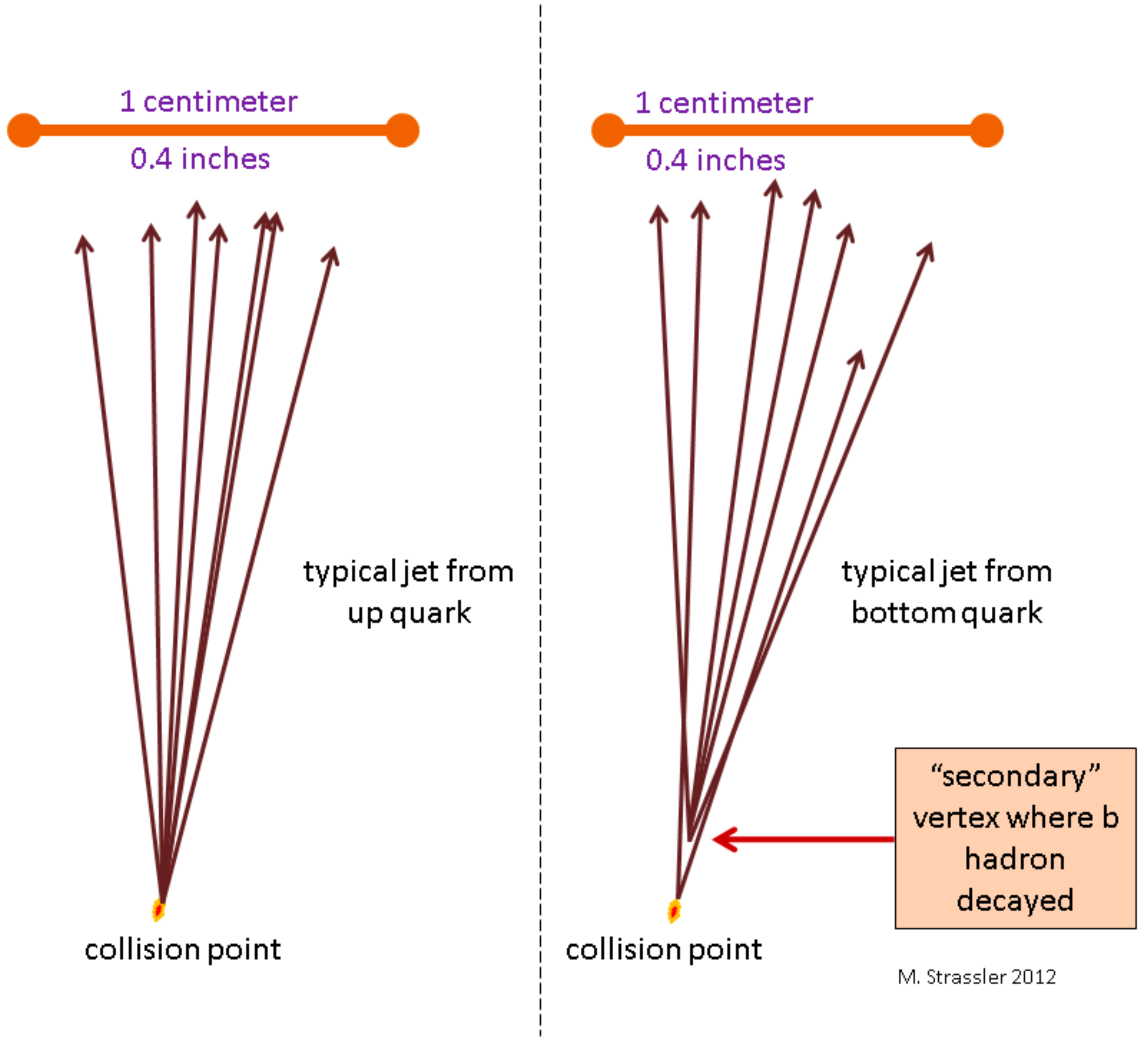
- Some of the benefits with as-a-service model:
  - CPU and GPU ratios are dynamic depending on the inferecen task. GPUs can batch inference requests from different CPU clients together, such that the throughput can be increased and the GPU utilizations can be increased. (Dynamic Batching)

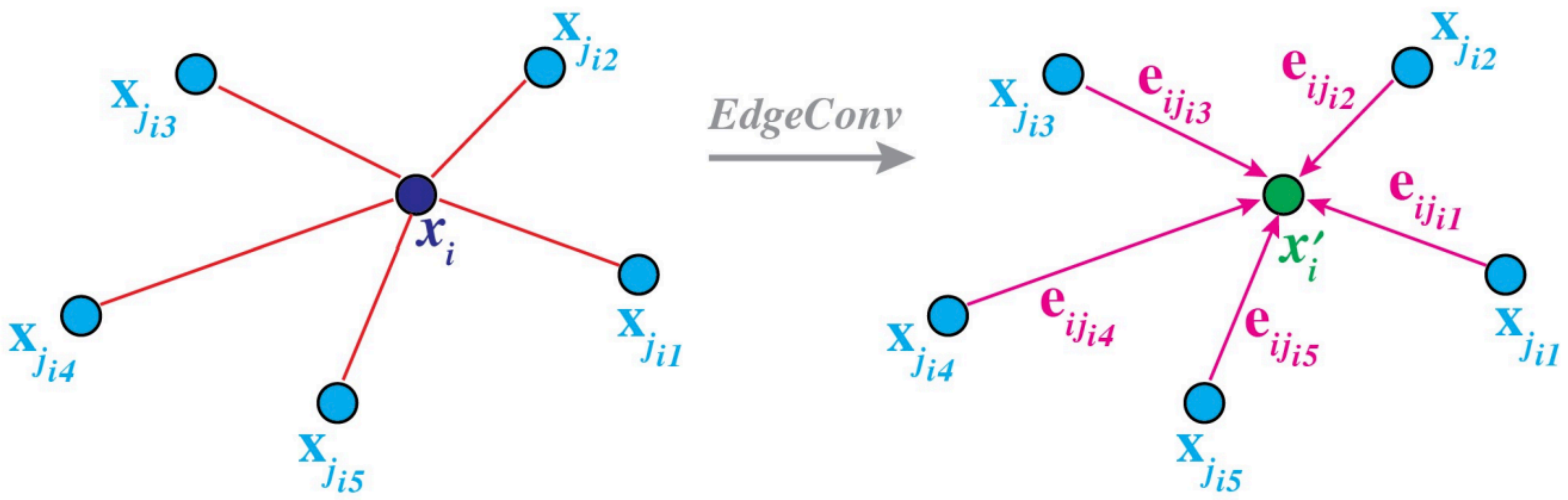
As-a-service with SONIC



# Example: Jet Flavor Tagging



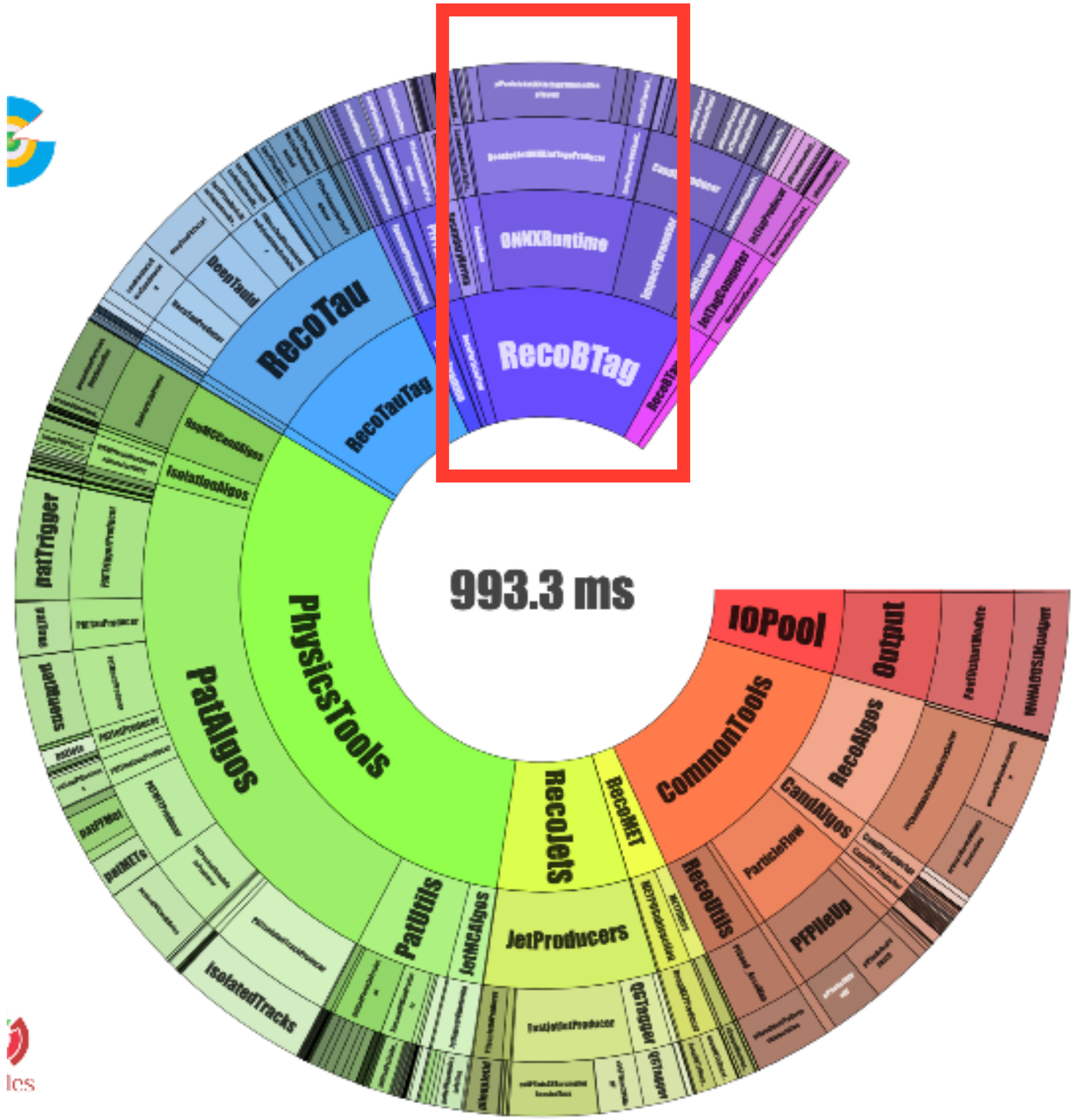



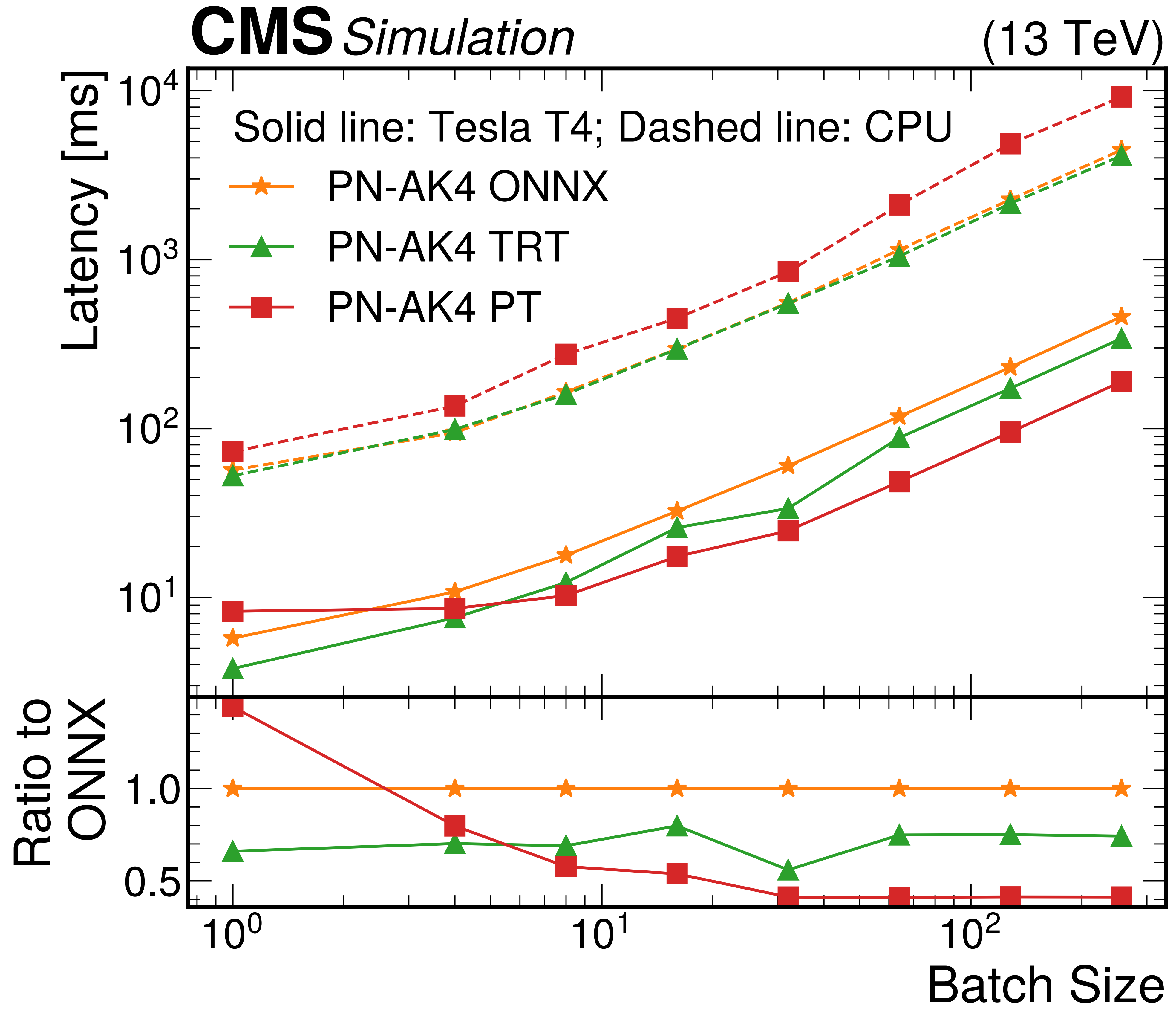

- One example: jet flavor tagging task
- With new GNN models the mistag rate can drop by one order of magnitude without tagging efficiency decrease

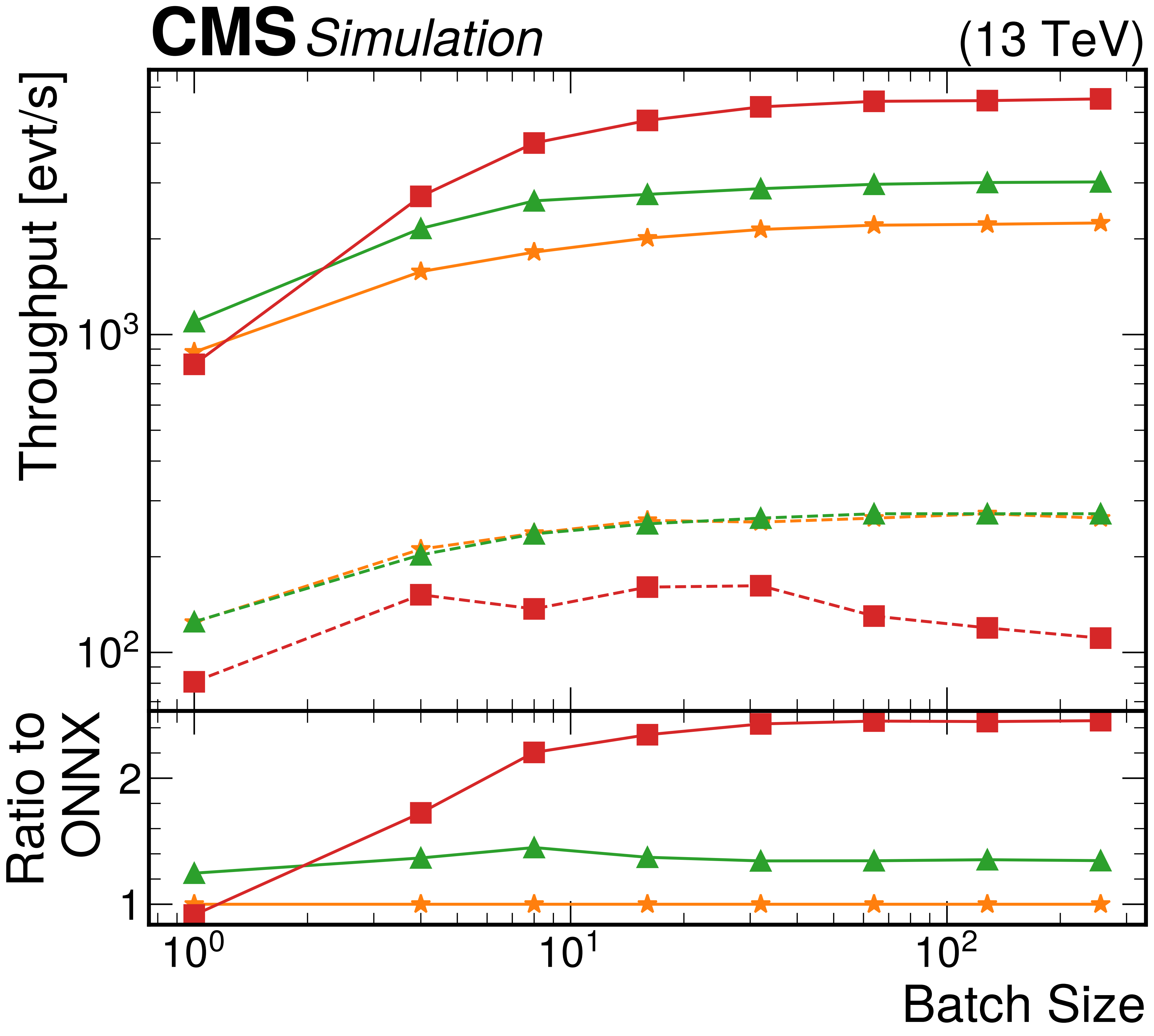
# Example: Jet Flavor Tagging

- Inferences take times: currently takes about 5-10% of the CMS Mini-AOD processing time for the inferences
- Expect these to increase dramatically in the next decade, as more and more algorithms get integrated, and they are more and more complicated



# Inference Comparisons

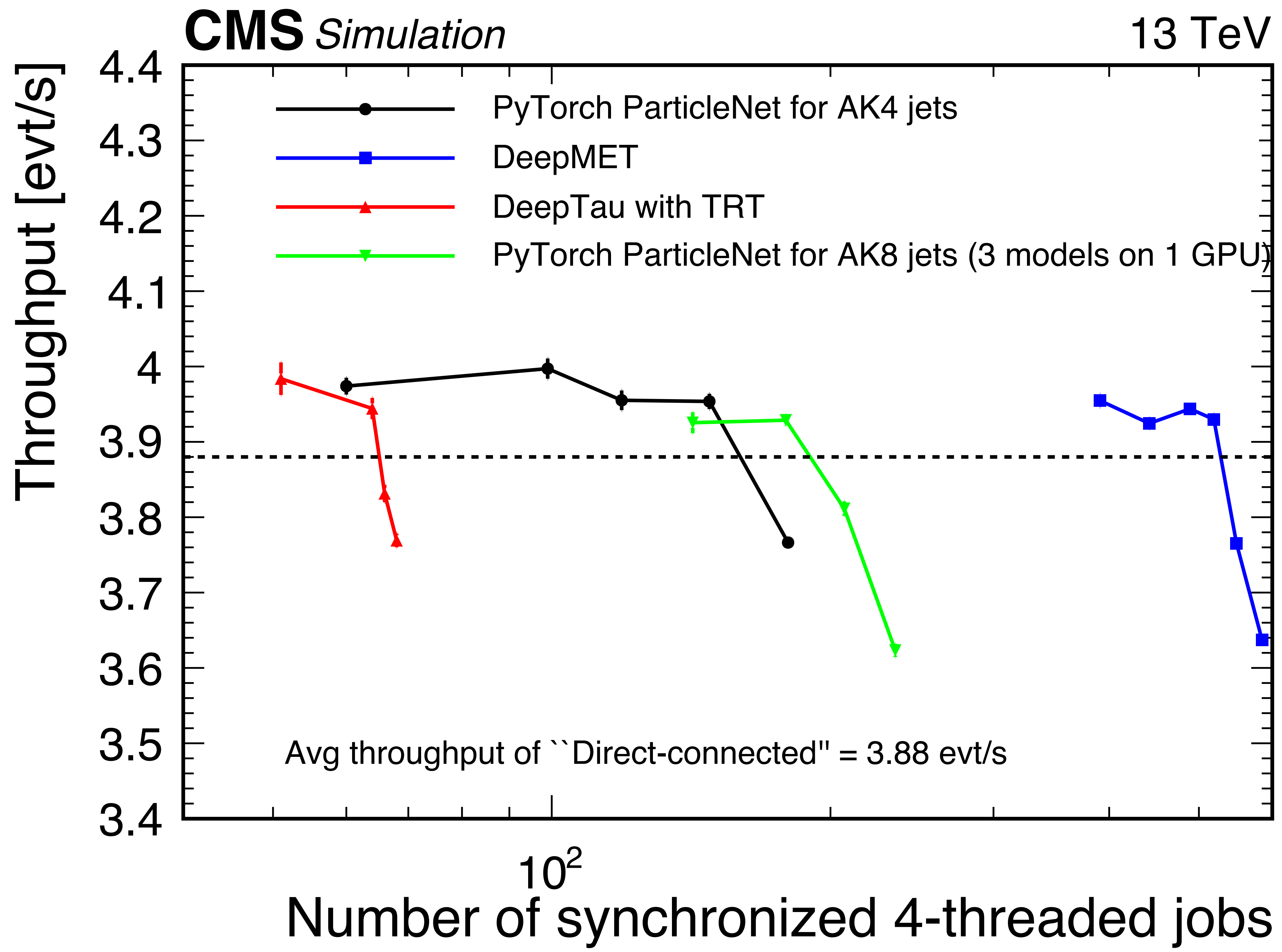


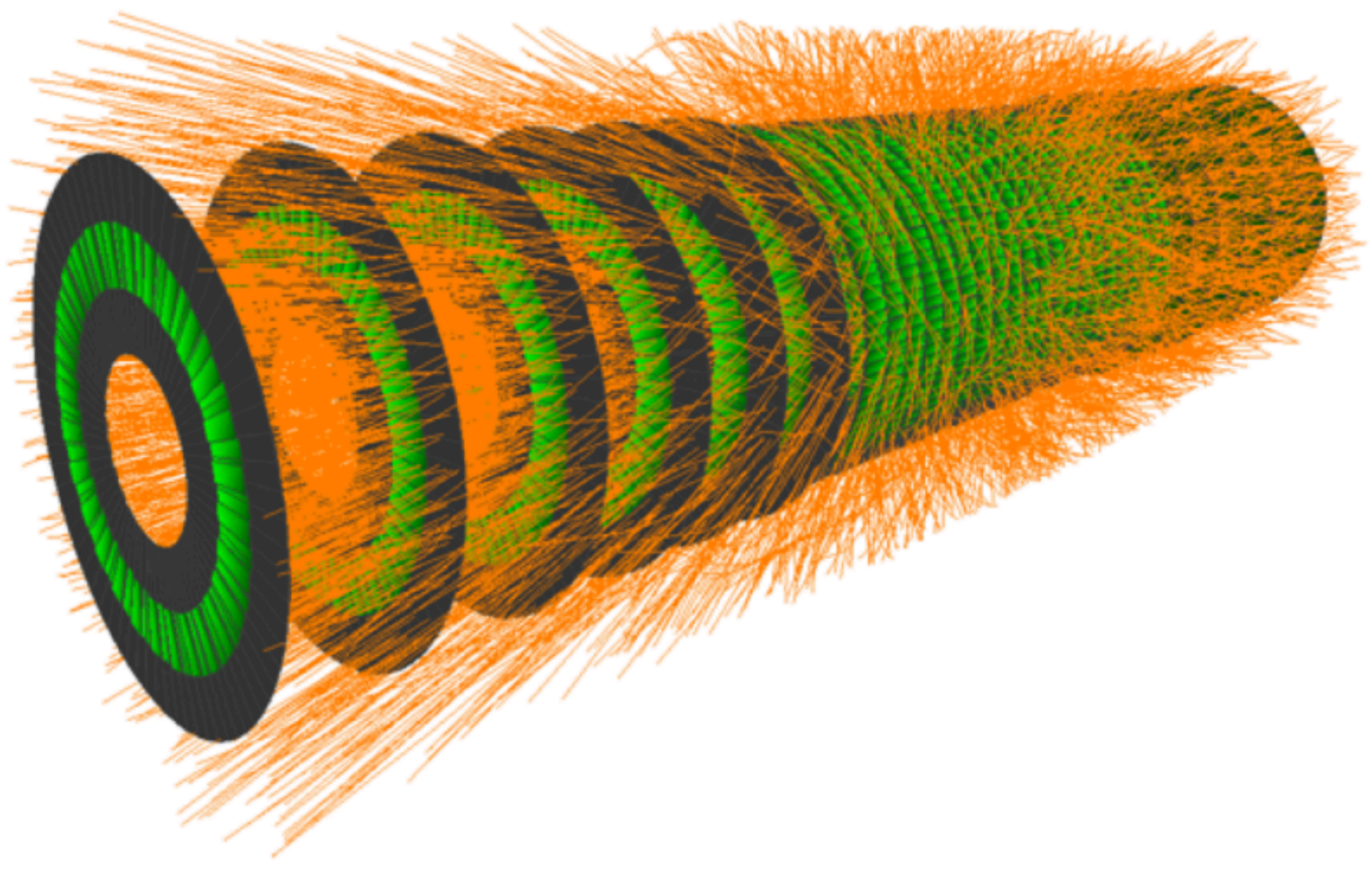

- O(10) times faster running on the GPUs (dashed) compared with CPUs (solid)
- Large batch sizes bring to larger throughputs
- Can explore different ML backends and see which one is faster

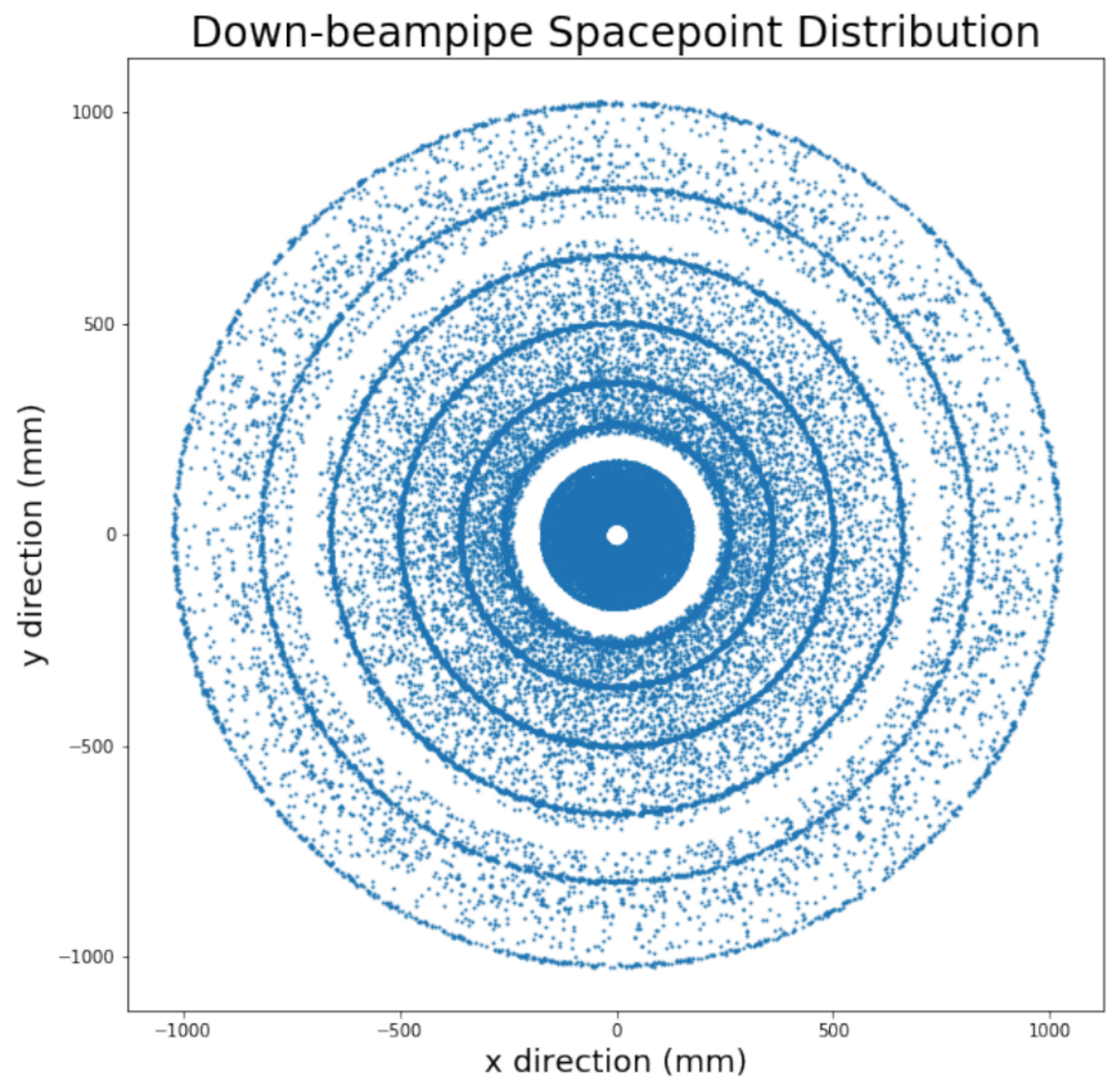
# Inference Comparisons

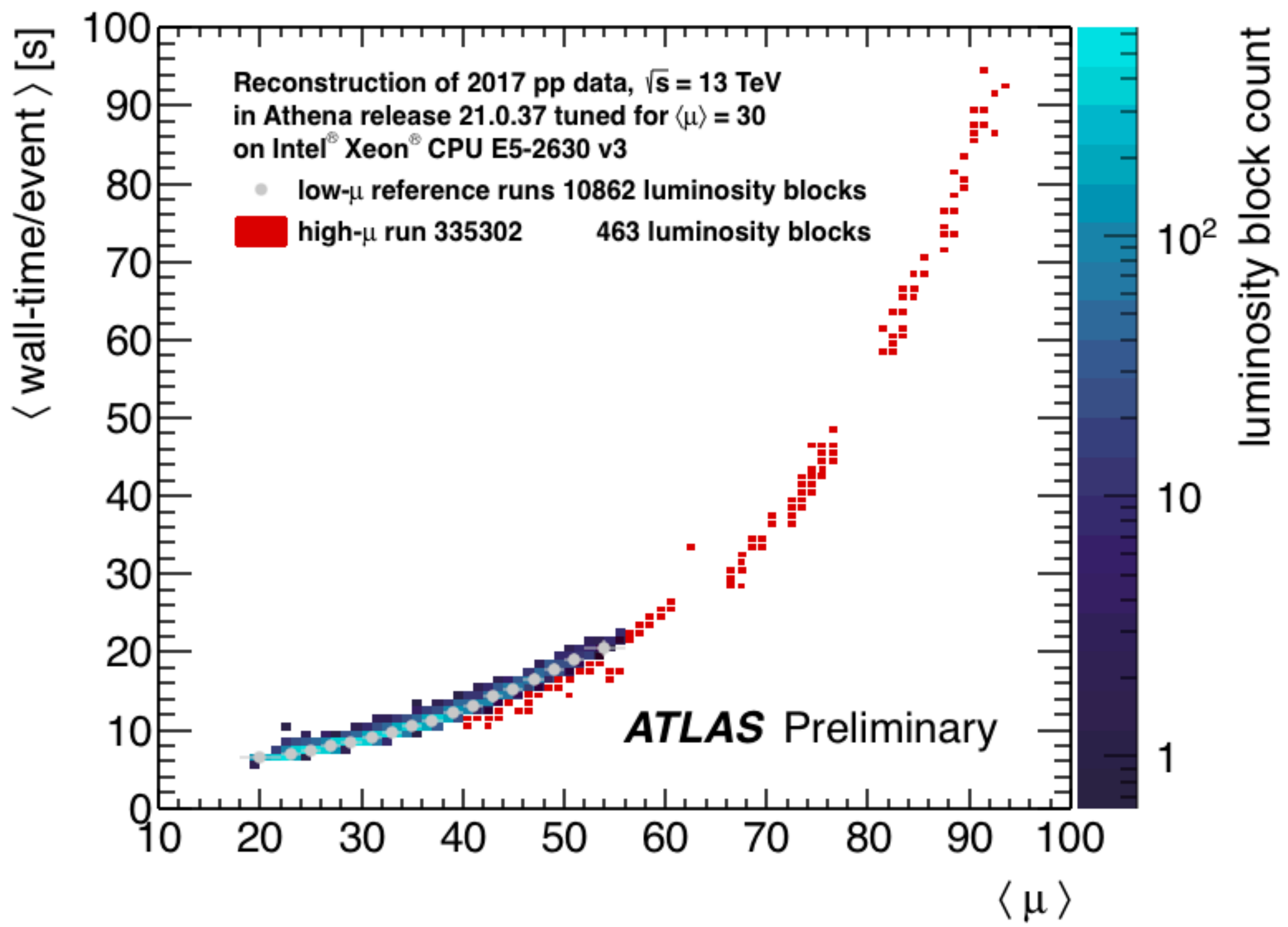
- One server can serve many CPU clients: O(10) - O(100) CPU clients pinging one server, without any performance decrease



# Track Reconstruction at the HL-LHC

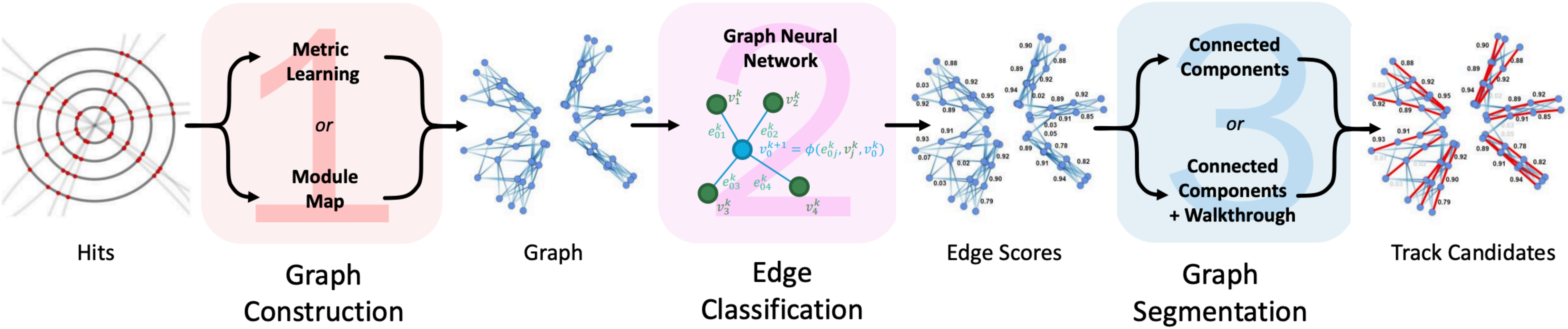




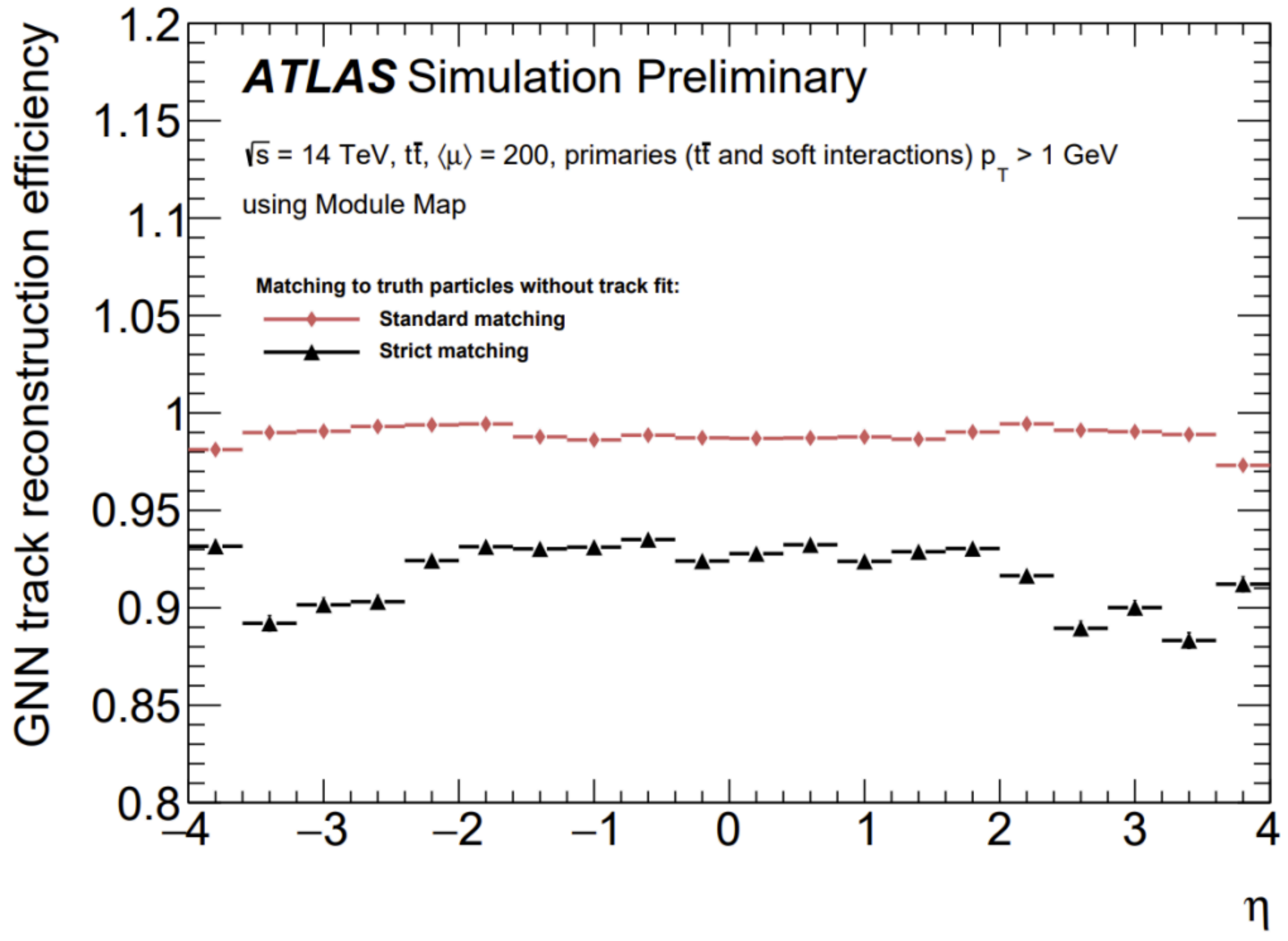




- Track reconstruction is expected to be very challenging in the future, especially at the HL-LHC:
  - A ttbar event with 150-200 pileup at the HL-LHC will produce O(5K) charged particles, and O(100K) spacepoints
- Computing cost does not scale linearly with number of pileup. Track reconstruction takes the major fraction of time among all the reconstruction steps

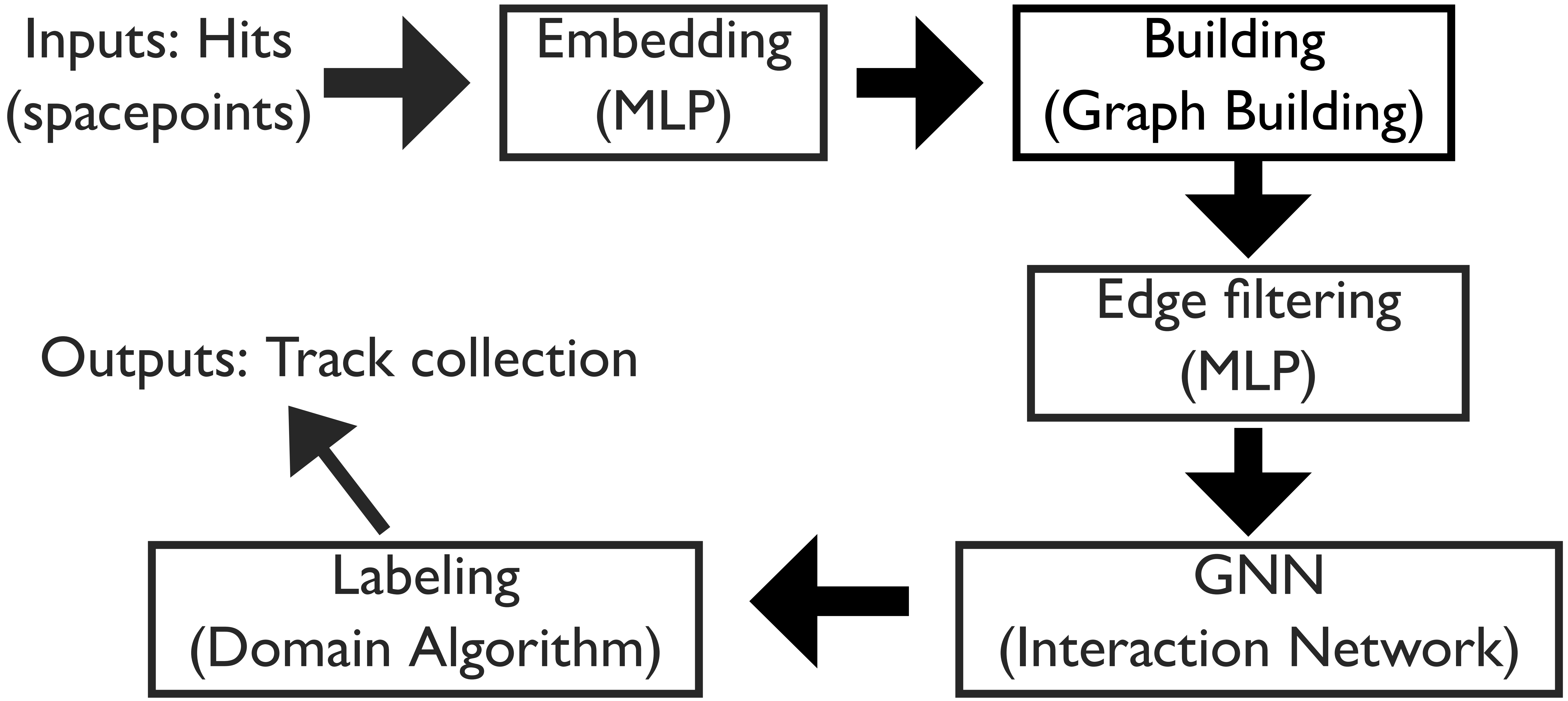
# ML-based Track Reconstruction

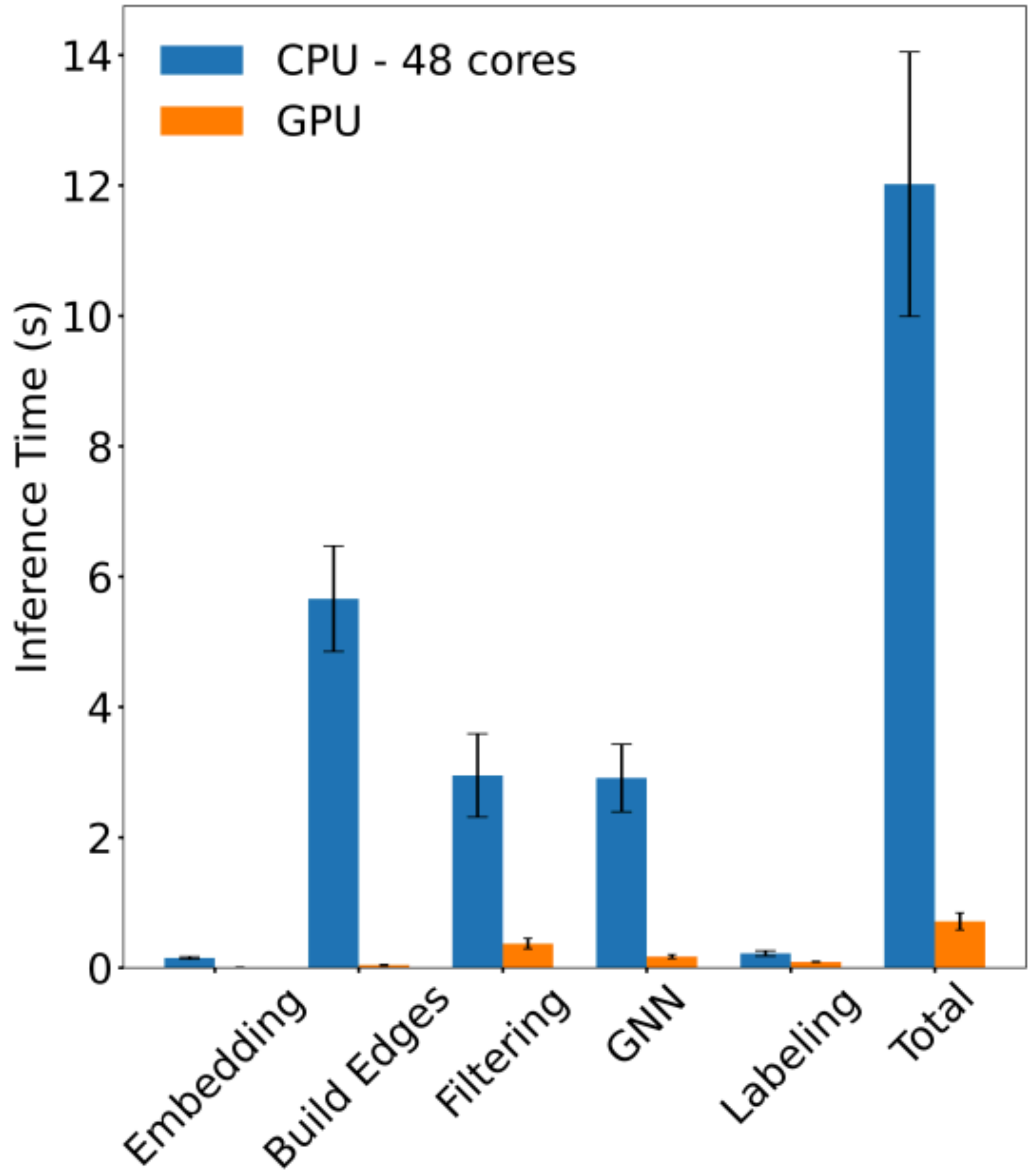



- ML-based track reconstruction with GraphNN could be a promising solution:
  - ML algorithms can run fast, easy to optimize, and easily accelerated on different coprocessors to get faster
- Good performances on the 200 pileup simulation datasets: similar efficiency as the classical algorithm, and O$(10^{-3})$ fake rates



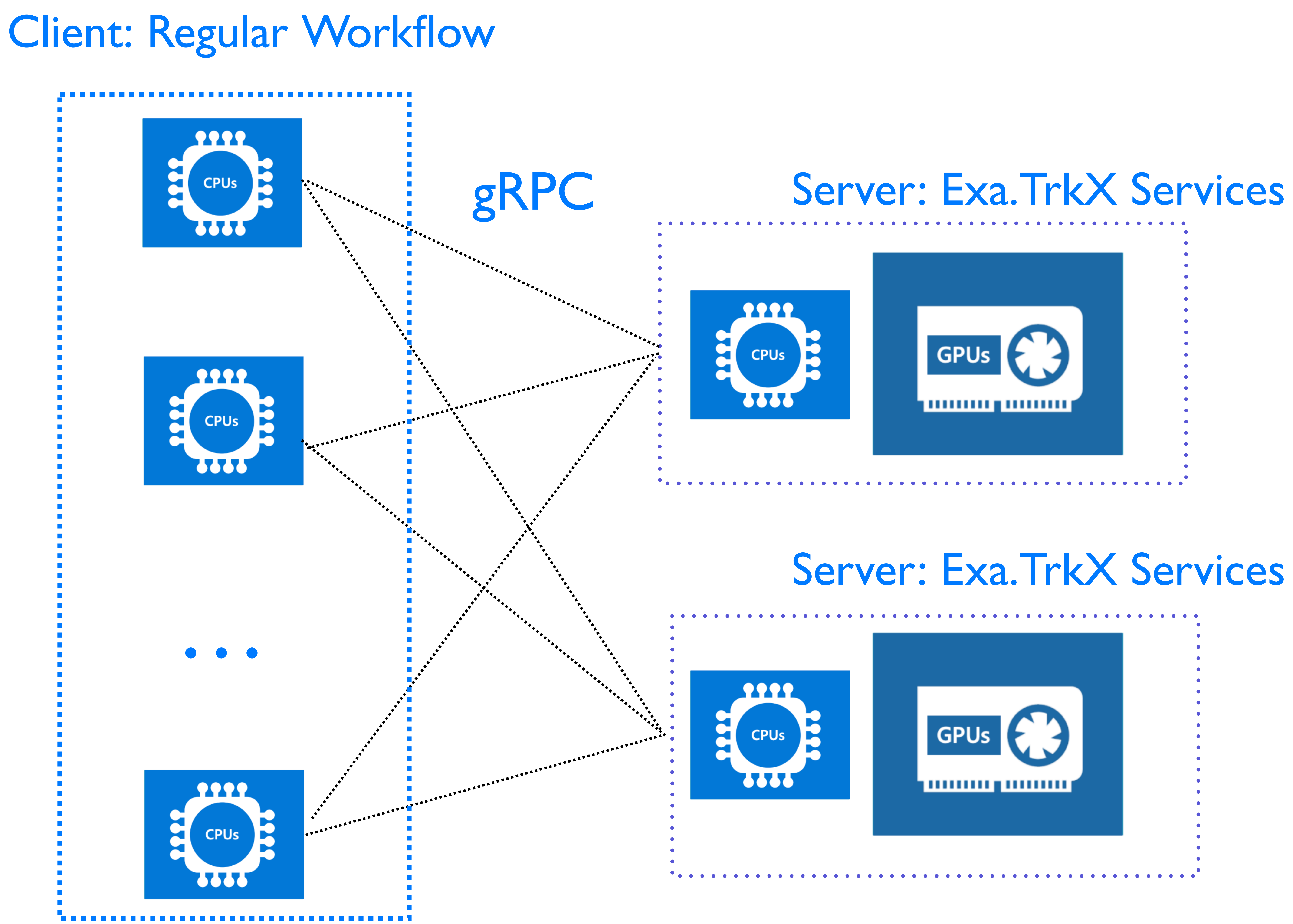
# Inference Costs



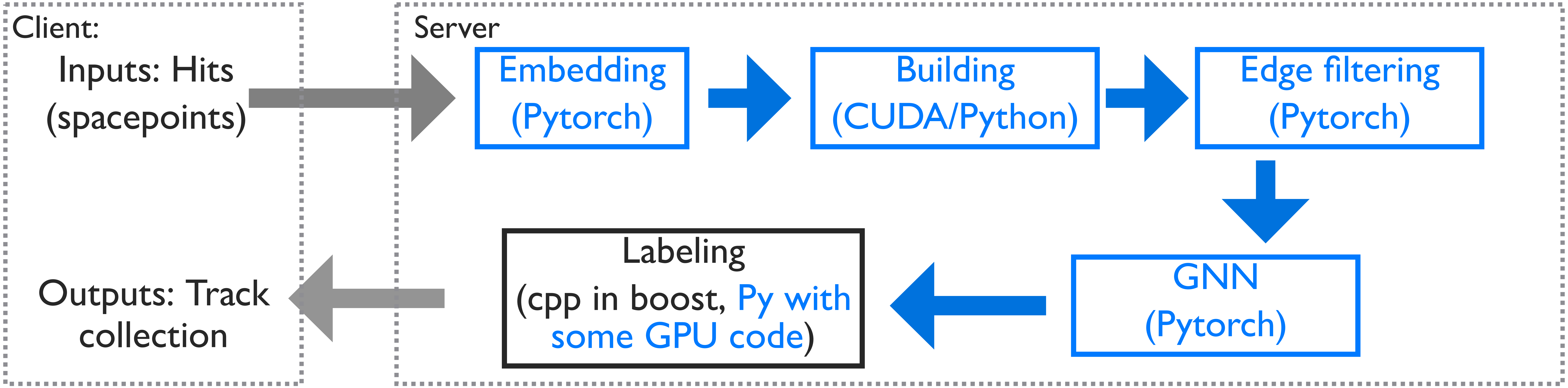

- Workflow runs much faster on GPUs compared with CPUs after optimizations: from O(20s) on 48-core Intel Xeon 8268s CPUs to <1s on NVIDIA V100. More details on Arxiv.2202.06929

# Inference As-a-Service




- Inference as-a-Service provides lots of benefits, e.g.:
  - Separate ML inferences out of the main software, easy to maintain
  - Enables access to remote GPUs;
  - more flexibility of the CPU/GPU ratios;
  - Easy deployment on different types of coprocessors
  - Etc
- More in Patrick's talk and Kevin's talk

# Current Exa.TrkX Workflow with as-a-Service




- Server side uses NVIDIA Triton Inference server. Various features and benefits:
  - Supports of different backends: ML including TF, Pytorch, ONNX; domain algorithms: CUDA, Python, Cpp
  - Ensemble model that can collect the whole inference modules together; reduce the IOs between client and server
- Pytorch models runs out of the box; CUDA and cpp implementations currently done with Python custom backend

# Preliminary Results

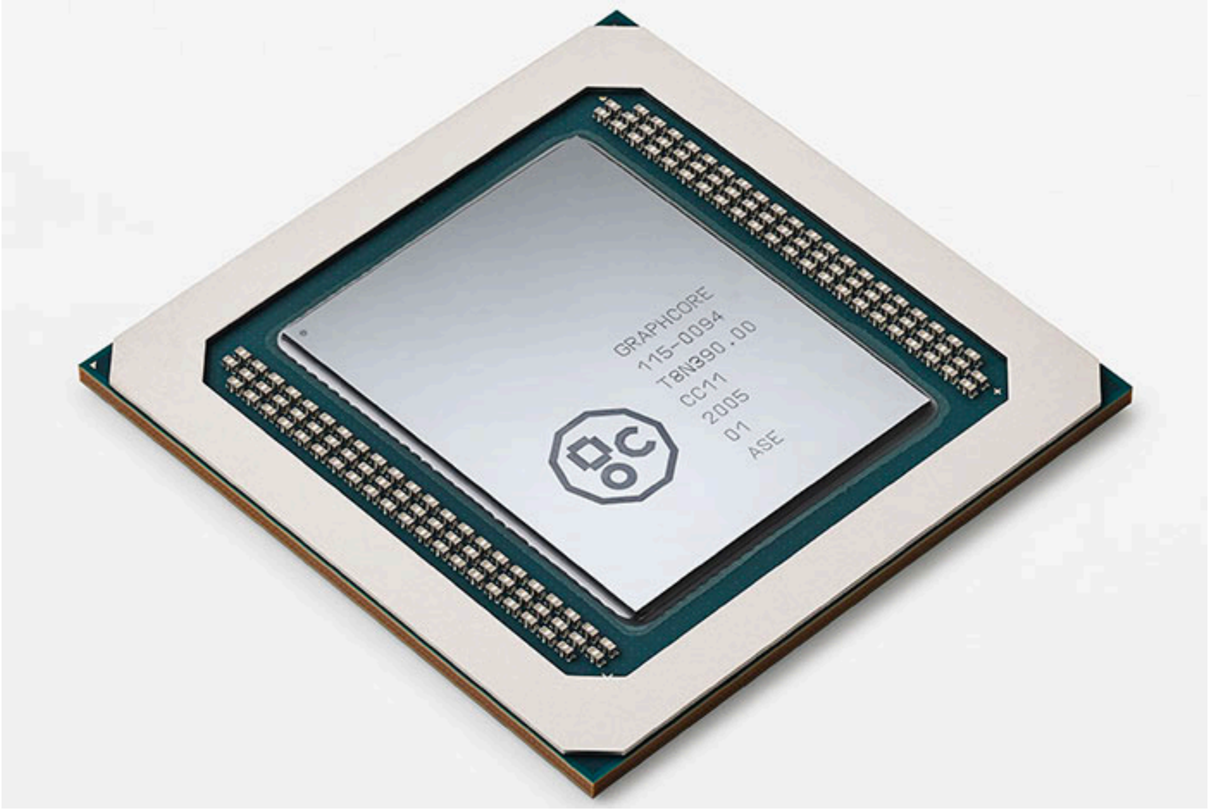
| Direct Inference | ms/evt |
|---|---|
| Embedding | 0.5 |
| Building | 2.2 |
| Filtering | 27.6 |
| GNN | 31.7 |
| Total | 62 |

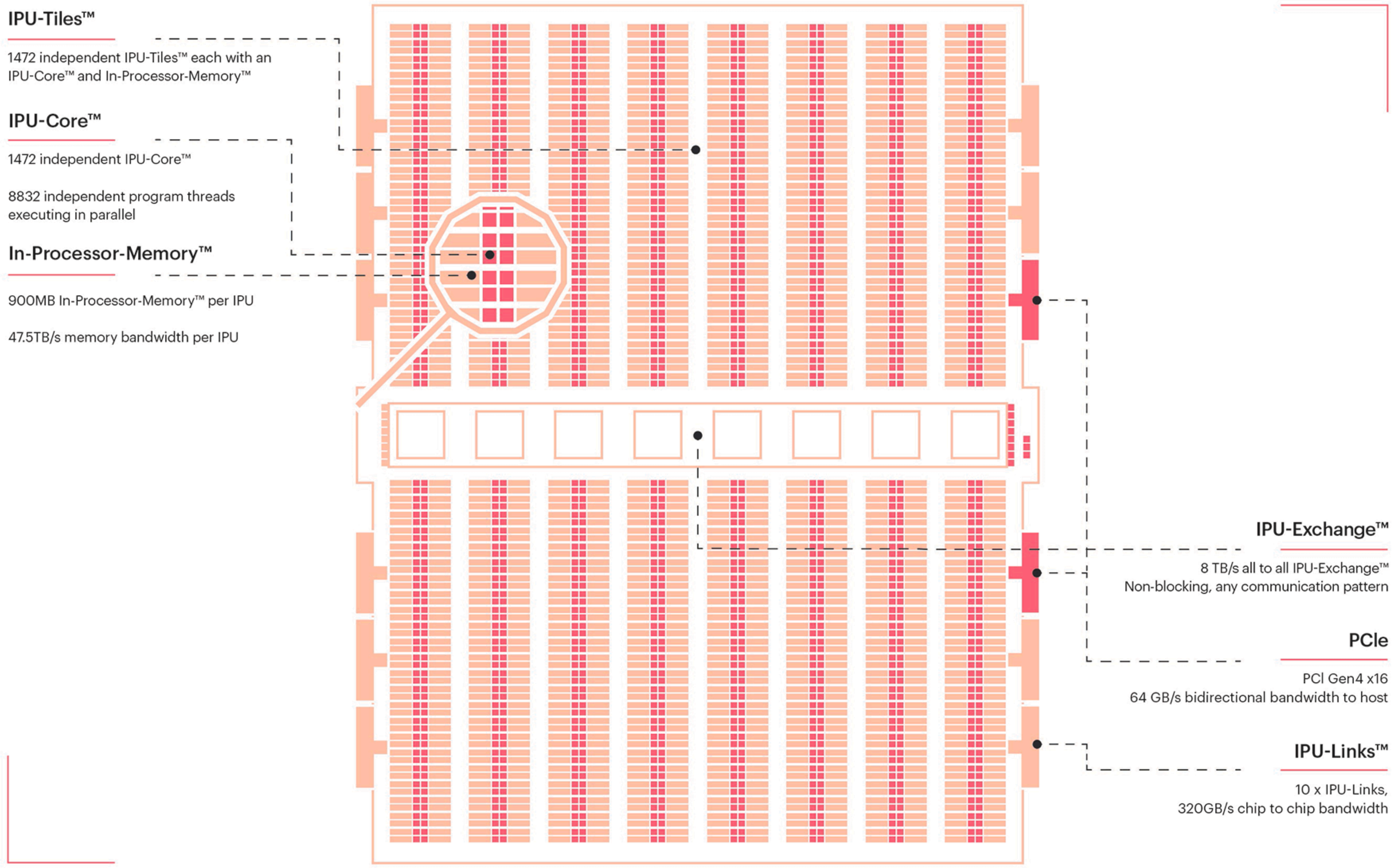
| As a Service | ms/evt |
|---|---|
| Embedding | 1.7 |
| Building | 7.3 |
| Filtering | 26.7 |
| GNN | 21.3 |
| Total | 64.4 |

- Benchmarked in the 0-PU dataset to start with.
- Time not including the labeling part (domain algorithm code; takes some efforts to prepare a custom backend for it)
- Similar inference time between CPU-GPU directly connected and CPU-Server with aaS:
    - Also checked the server-side metrics: the fraction of time to handle IOs are small. Most of the time are on computations.
- Working in integrating the workflow into the official ATCS/Athena software and testing the performances

# SONIC Development: GraphCore IPU Tests

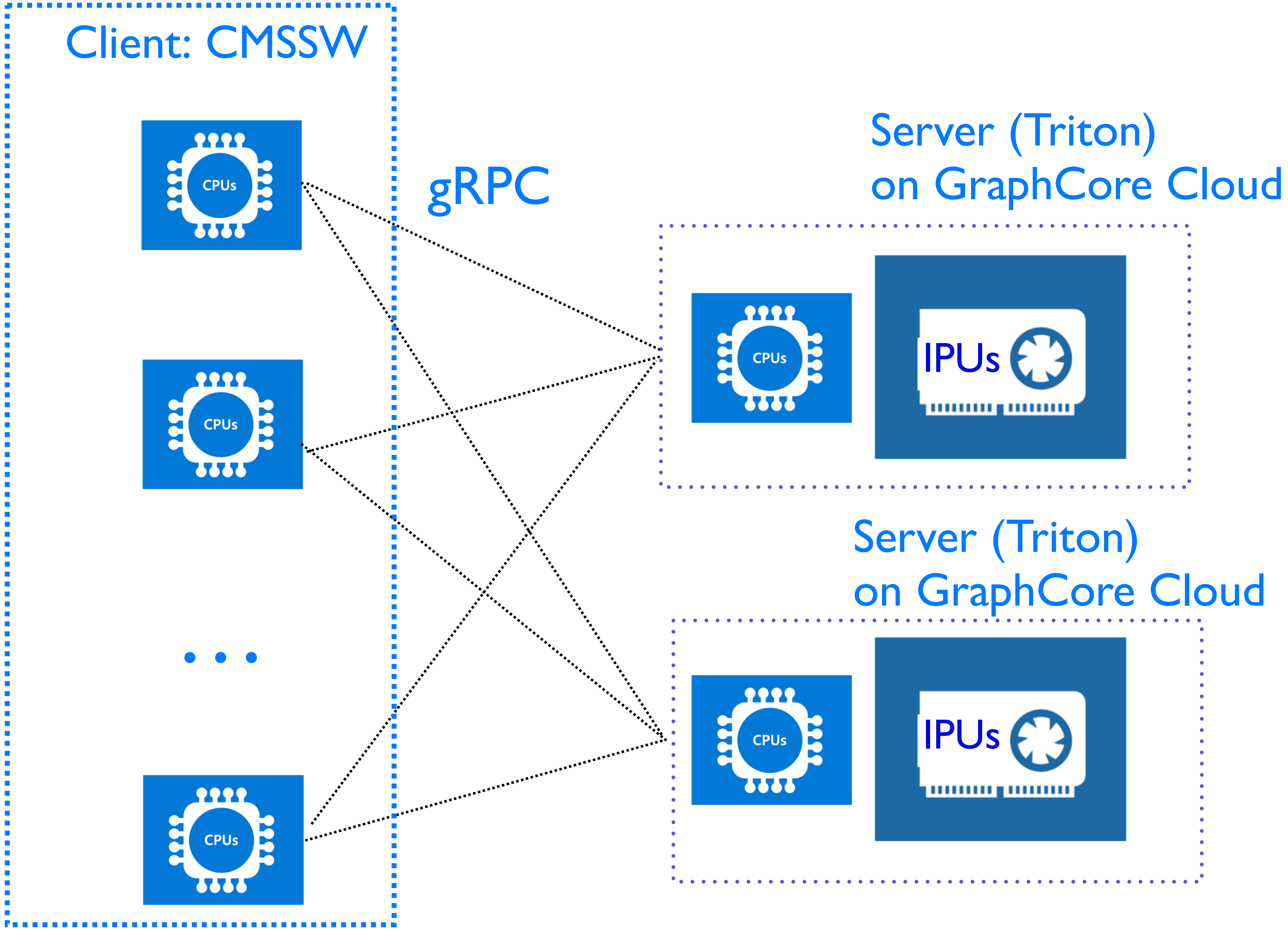
- As-a-Service allows easy deployment of inference on different times of coprocessors:
  - ✤ Prepared the CMS production workflow, with several intensive ML inferences tasks offloaded to coprocessors with SONIC
- GraphCore has developed Intelligence Processing Units (IPUs) AI chip, enabling very fast ML training and inferences
- GraphCore team is developing the Triton Custom Backends to support running TensorFlow models as-a-Service on the IPUs:
  - ✤ Tensorflow models supported with aaS
  - ✤ Pytorch(-Geometric) model supports under developments



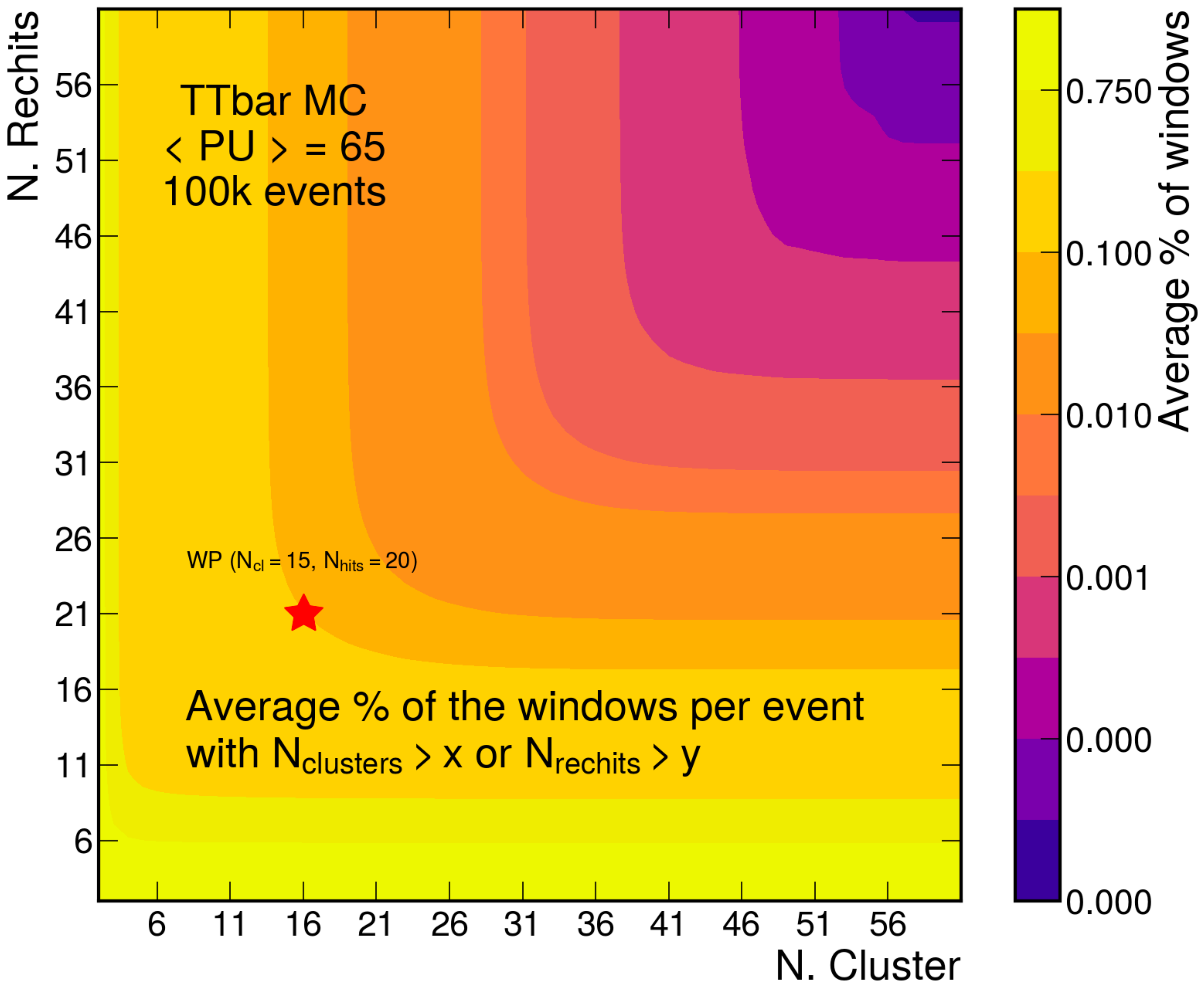


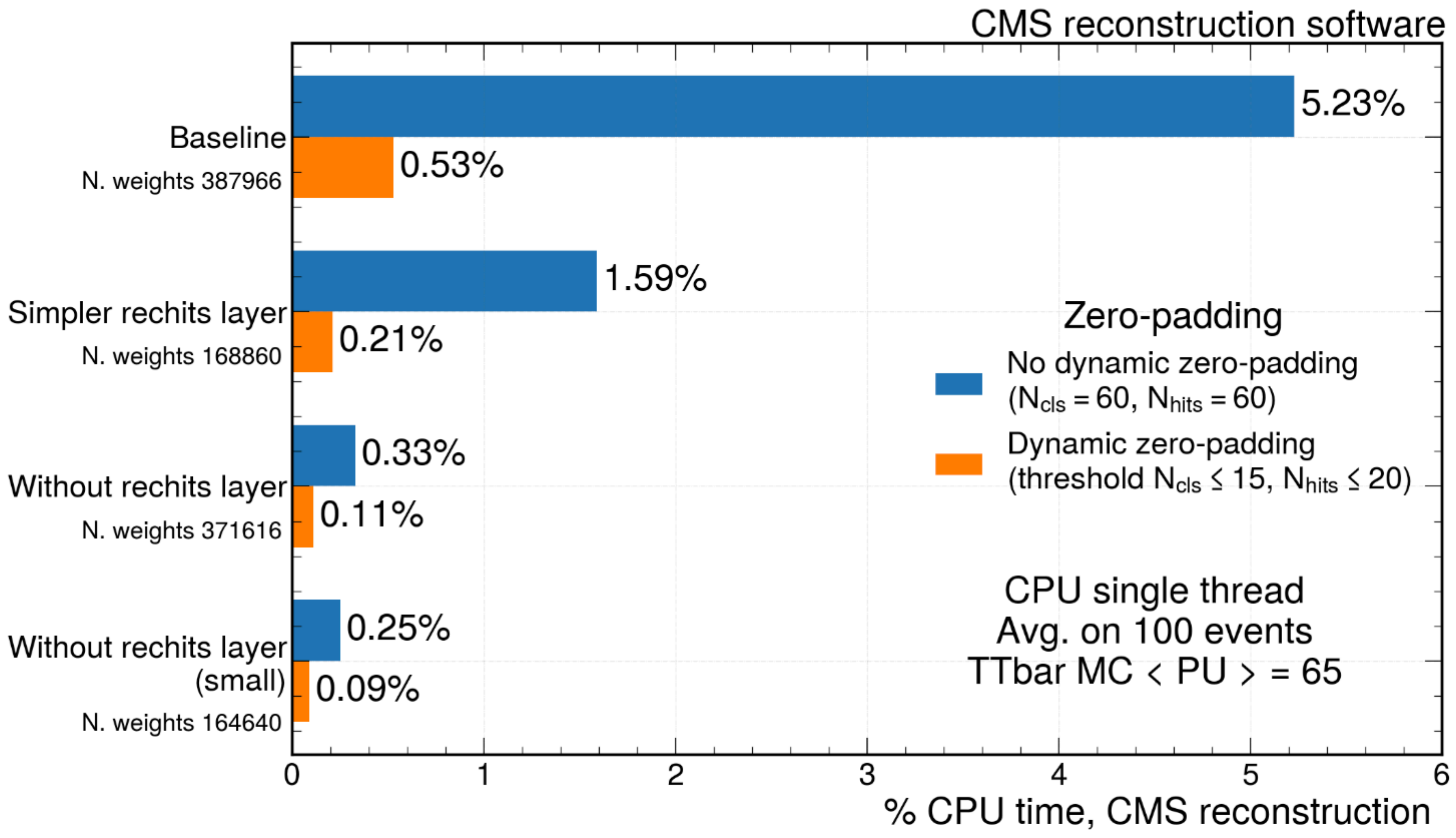
# SONIC Development: GraphCore IPU Tests

- Run the CMSSW MiniAOD production on the cluster, with DeepMET and DeepTau inference (Tensorflow models) aaS on IPU-POD16s:
  - Workflow runs well; outputs as expected and consistent with CPU/GPU results; 5% MiniAOD throughput gains as expected.
- For the ML model inferences, throughputs are roughly a factor of 3 higher compared with NVIDIA Tesla V100 for these models
  - DeepMET and DeepTau Models tends to be I/O bounded. Expect more improvements for more computing intensive models
- Can run large-scale production tests with IPUs once having PyTorch/ONNX models supported and having enough CPU clients to saturate the IPUs



# More: Ragged Batching Exploration




↳ Inference time in CMSSW of the DeepSC algorithm: comparison between fixed or dynamic zero-padding strategy

- ECAL electron and photon supercluster reconstruction with GraphNN:
  - Number of inputs varies a lot event-by-event; inference performance strongly depends on the number of inputs
  - Triton provides ragged-batching feature to vary the number of inputs; under investigation

# Summary

- With more data and more complicated algorithms, computing challenges expected for the (HL-)LHC

- Coprocessors, such as GPUs, is one solution to such computing challenges

- Coprocessors with as-a-Service can more efficiently utilize coprocessor resources and boost the performances

# Back Up