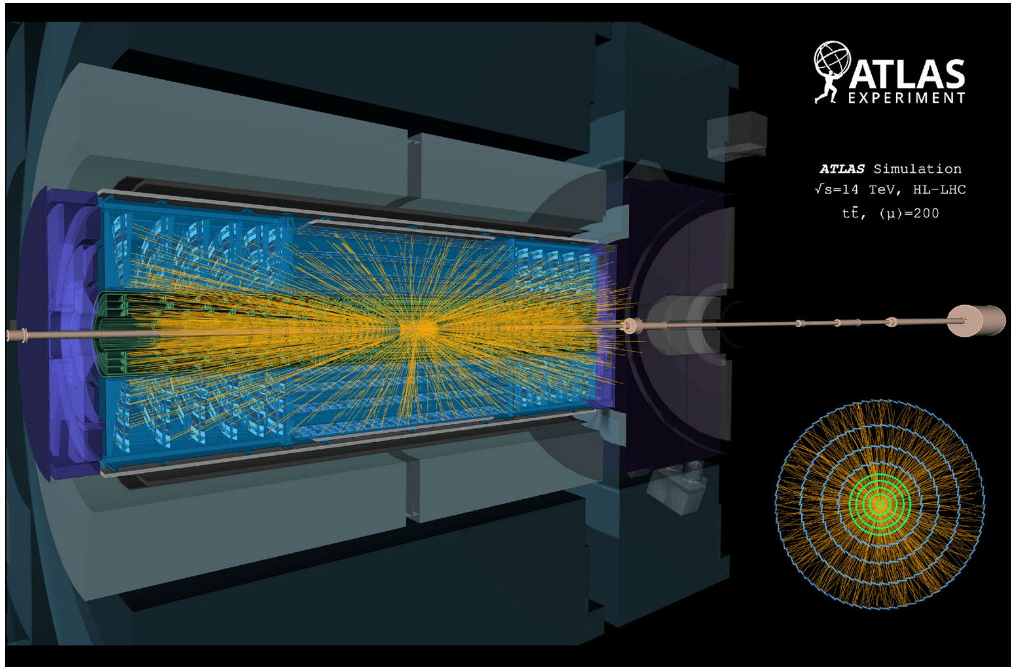
Improving Computational Performance of ATLAS GNN Track Reconstruction Pipeline

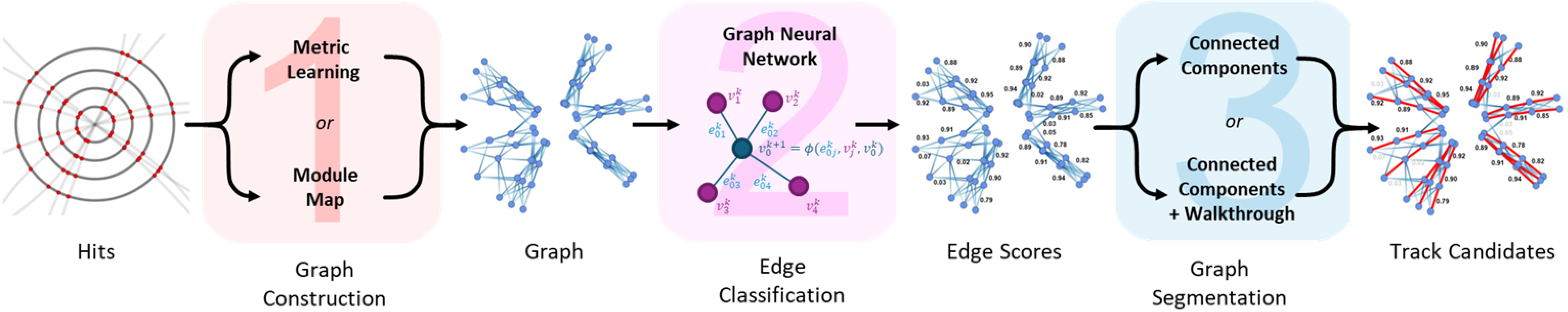
Alina Lazar, Jared Burleson, Jackson Burzynski, Sylvain Caillou, Paolo Calafiura, Jay Chan, Christophe Collard, Xiangyang Ju, Daniel Murnane, Levi Condren, Ryan Liu, Mark Neubauer, Minh-Tuan Pham, Jan Stark, Heberth Torres and Alexis Vallier
on behalf of the ATLAS Computing Activity

Track Reconstruction

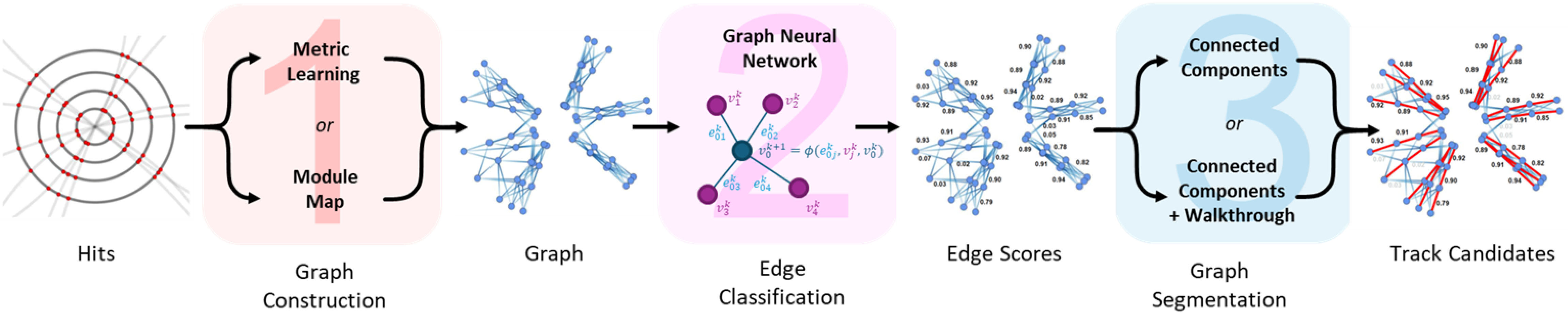
- In a collision event, generated particles leave hits in the detector. Track reconstruction recreates particle trajectories from detector hits.
- An expensive process, especially at high pile-up (μ = 200). HEP community seeks to develop hardware-accelerated, ML-based tracking algorithms.
- We build a machine learning pipeline based on Graph Neural Network (GNN) for track finding under HL-LHC conditions for the ATLAS ITk detector.



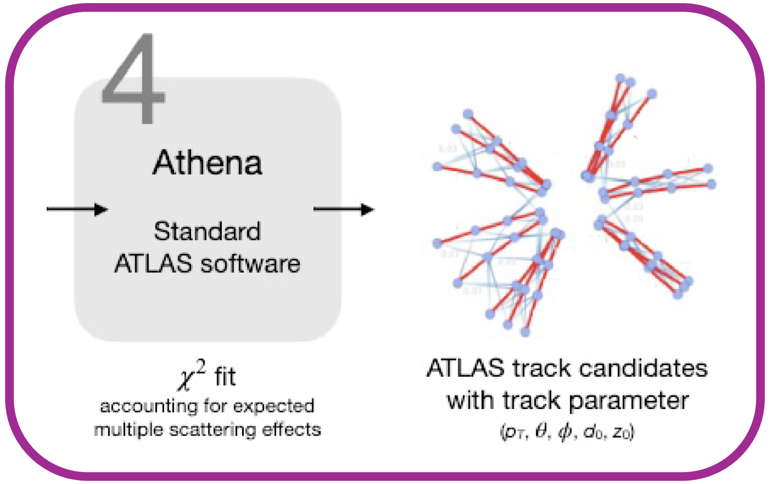
The GNN4ITk Reconstruction Pipeline



The GNN4ITk Reconstruction Pipeline



Track Fitting



To evaluate the Physics Performance

GNN4ITk for Track Reconstruction Goals

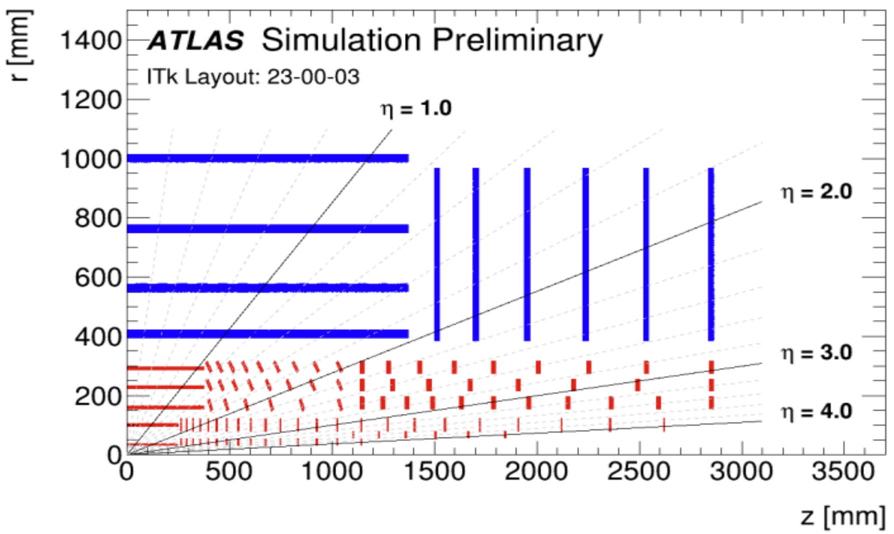
- The main goal is to optimize the GNN4ITk pipeline. To increase **throughput**, decrease **latency**, and reduce **memory usage** in the context of Offline & Online track reconstruction for the High-Luminosity Large Hadron Collider (HL-LHC).
- **Acceleration:** Implementing computing optimization techniques using **GPUs** (Graphics Processing Units) or FPGAs (Field-Programmable Gate Arrays) to accelerate the execution of the stages in the pipeline.
- **Memory Optimization:** Efficiently managing memory usage by optimizing data structures to minimize the memory footprint.
- **Algorithmic Efficiency:** Developing and refining algorithms that reduce computational complexity to maintain accuracy while lowering the number of computations.

Simulation Data - CTD 2023 Dataset

- Using ATLAS simulation event samples (10,000 events):

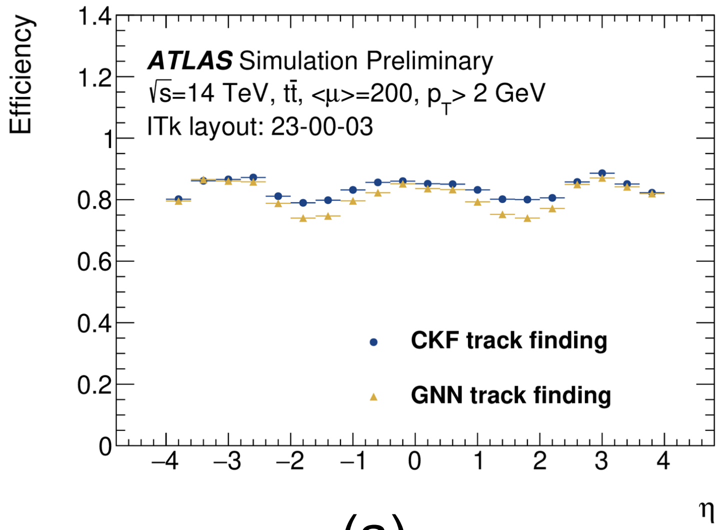
  pp collisions at $\sqrt{s} = 14\ \mathrm{TeV}$, $t\bar{t}$ process,

  $\langle \mu \rangle = 200\ pp$ interaction pileup



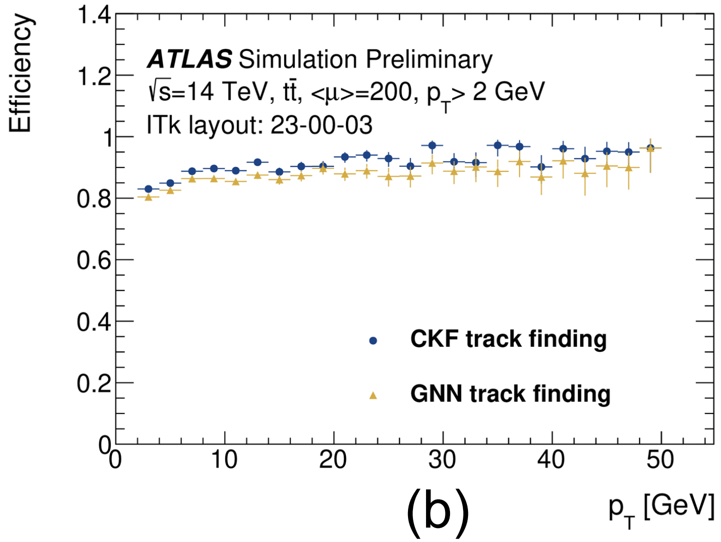
- Updated ITk layout 23-00-03 (reduced radius of innermost pixel layer, and distribution of passive material with greater detail and accuracy)
- Target particles (dominated by soft interactions):
  - ❖ $p_T > 1\ \mathrm{GeV}$, with at least 3 space points, no electron, with production radius < 26 cm
  - ❖ Only primary particles (including B hadron decays, without “secondary” Geant4 particles from material interactions)

Physics Performance



(a)

Efficiency = # of charged particles with at least one reconstructed track / # of generated charged particles.
Tracks found by the GNN are required to satisfy the following criteria: at least 8 silicon hits, transverse impact parameter $|d_0| < 20$ mm, longitudinal impact parameter $|z_0| < 25$ cm and $p_T > 1$ GeV.
The simulated charged particles matched to reconstructed tracks are required to satisfy $p_T > 2$ GeV to avoid turn-on effects (link).



(b)

Track reconstruction efficiency as a function of the generator-level pseudorapidity η (a) and p_T (b)

CFK is the current reconstruction method using combinatorial Kalman Filter algorithm

ATLAS Collaboration, IDTR-2023-06, October 2023 (link)

H. Torres of behalf of the ATLAS Collaboration, Proceeding of Connecting the Dots 2023 (link)

Computing Performance

CTD 2023 Dataset		
Steps	**Module Map (ms)**	**Metric Learning (ms)**
Graph Construction	69	505
GNN	323	108
Graph Segmentation	118	118
Total:	**510**	**731**

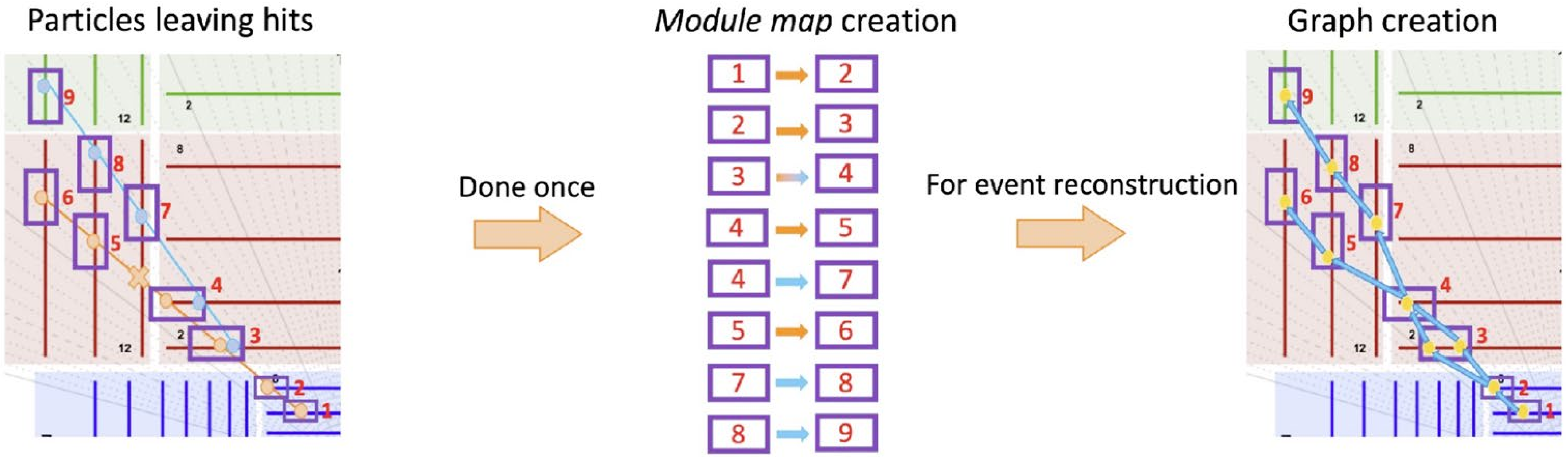
Per-event running times of each stage in the GNN4ITk pipeline, for both choices of graph construction technique. Stages 1 and 2 are evaluated on Nvidia A100 40GB GPUs. Stage 3 is evaluated on CPU (AMD EPYC 7763).

[ATL-PHYS-PUB-2024-018]

Computing Optimizations of the GNN4ITk Reconstruction Pipeline

Graph Construction
GNN Inference and Architecture Optimizations
Graph Segmentation
Inference As a Service (IaaS)

Graph Construction with Module Map

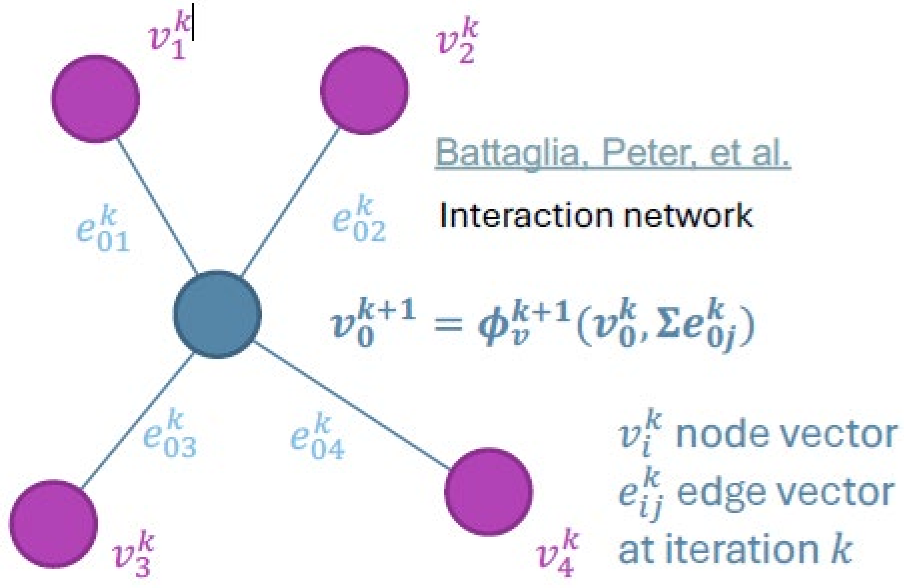


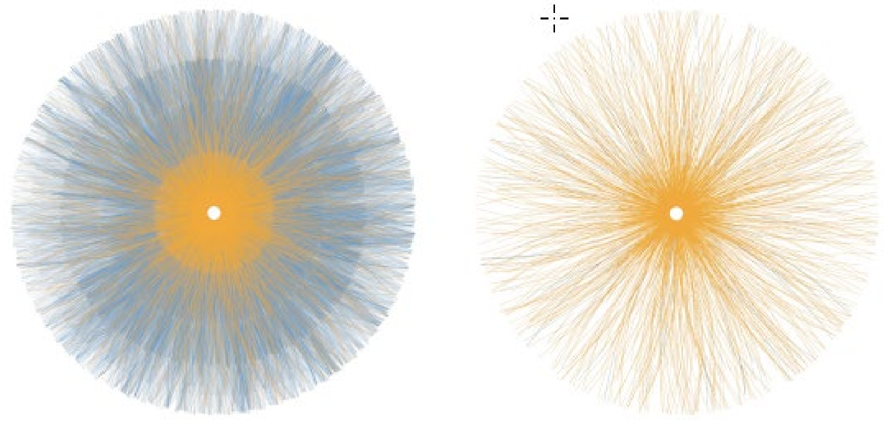
Current timing around 69 ms on an Nvidia A100 GPU (**140x speedup)**

Christophe Collard @CHEP2024

GNN for Edge Classification

1. Encode nodes and edge features.
2. Aggregate edge vectors, acting as messages between nodes.
3. Update node features with aggregated messages. Update edge features using updated node features.
4. Repeat n times steps 2 and 3.
5. Compute an edge score representing the probability of being a true edge.





Input graph (left) and classified graph (right). **Fake** = blue. **True** = orange

Computing Optimizations for the GNN of the Module Map Pipeline

Automatic Mixed Precision (AMP) – **1.5x**

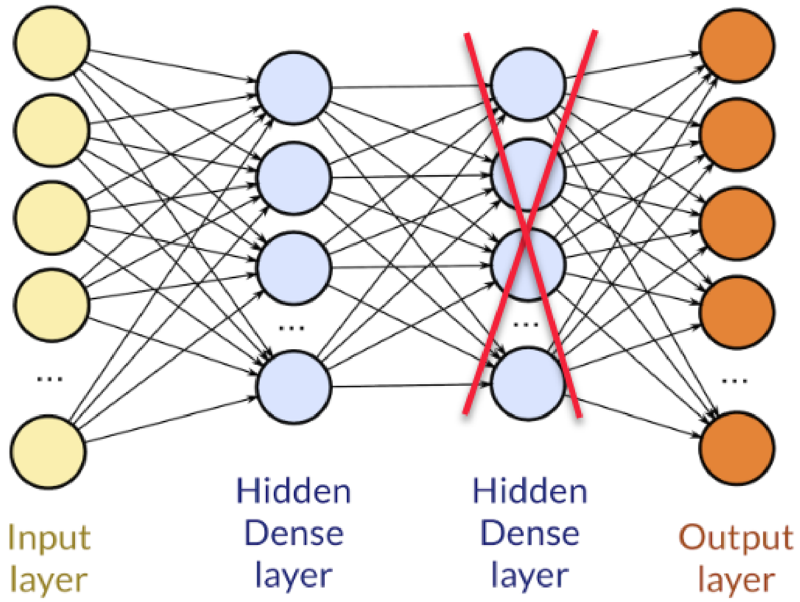
torch.compile() – **1.3x**

AMP + torch_compile()
Total – **3.2x**

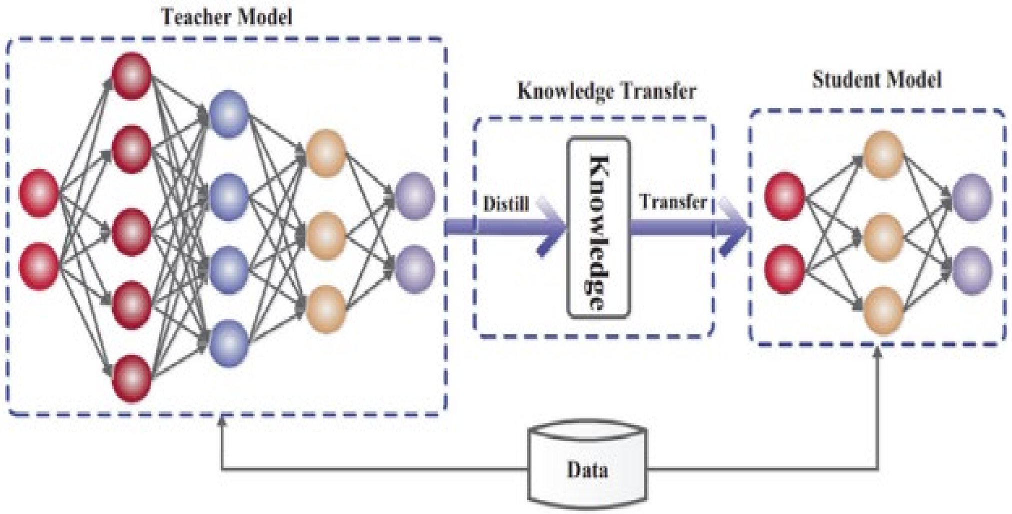
- A variety of improvements have brought the GNN timing to around 322 ms
- Average running time over 1,000 events, evaluated on a Nvidia A100 40 GB GPU, CUDA 12.1, the stable release of PyTorch 2.3 and PyG 2.5.
- Observed further **2x speedup** running on a Nvidia A100 80 GB GPU, PyTorch 2.5.0.dev20240902+cu121 and PyG 2.6.

Computing Optimizations for the GNN Pipeline ~ Work in Progress

GNN Model Reduction



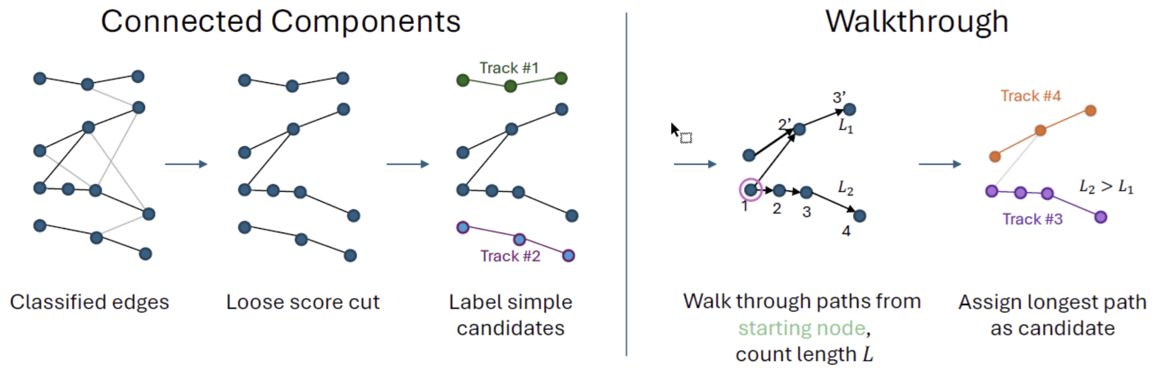
Knowledge Distillation



Track Building

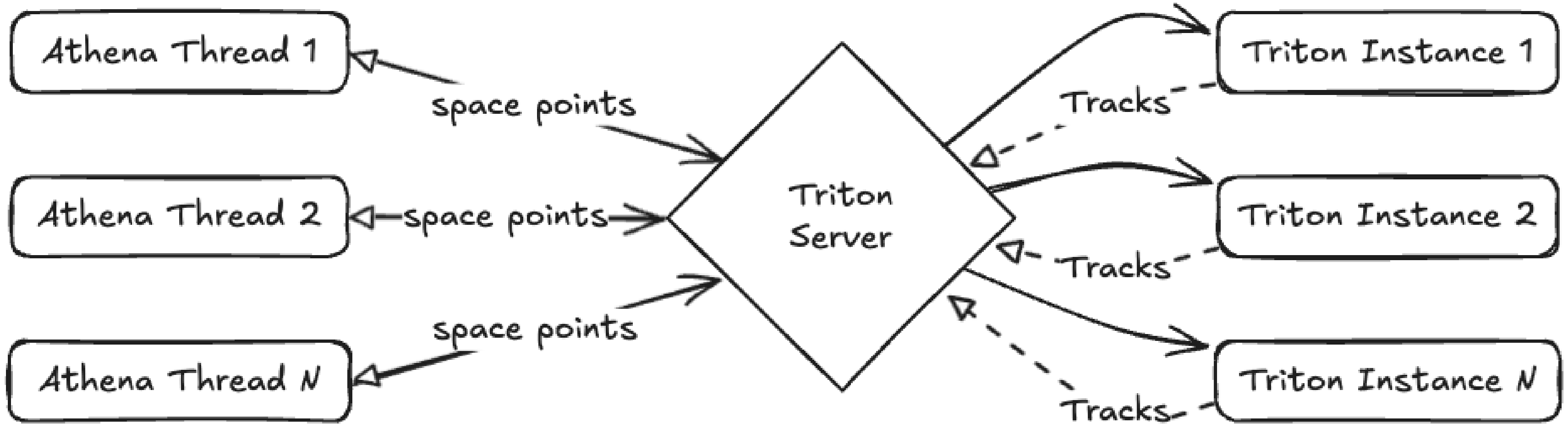
- 2-step sequence: connected components (CC) and walkthrough:
  1. Use CC to isolate subgraphs with no branching.
  2. On subgraphs with branching, use a walkthrough to separate track candidates.
- Each track candidate is a list of hits → extract track parameters by a track fit and match to generator-level particles for physics performance evaluation.

Stages	Physics Efficiency	Running Time (ms)
CTD23 Walkthrough	0.750	42,000
FastWalkthrough	0.754	118
ConnectedComponents (CC)	0.740	6.0
CC + Junction Removal	0.757	40



Daniel Murnane @CHEP2024

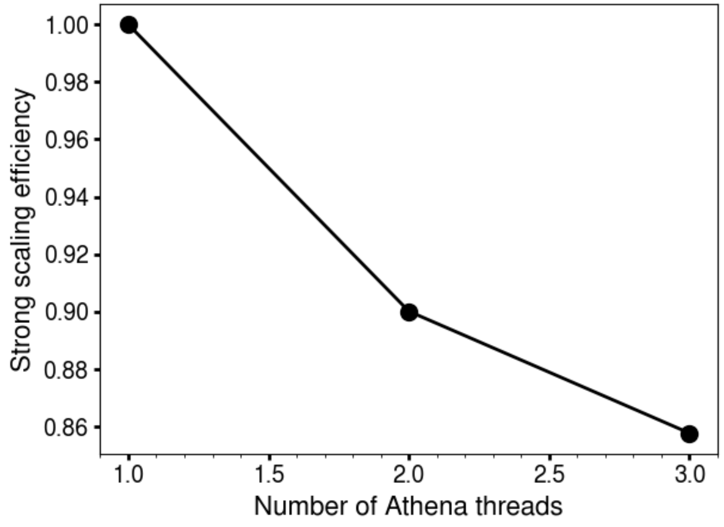
Optimizing GPU Utilization with IaaS



Can run 3 NVIDIA Triton instances on a 80GB A100 before running out of memory

→ AthenaTriton provides a **2.4 throughput increase**

Yuan-Tang Chou @CHEP2024



Computing Optimizations Summary

- Graph Construction with Module Map - 140x speedup
- GNN:
  - Automatic Mixed Precision (AMP) → 1.5x speedup
  - torch.compile() → 1.3x speedup
  - AMP + torch.compile() → 3.2x speedup
  - New version of torch.compile() from PyTorch 2.5
  - Nvidia A100 80 GB GPU
  - Total - 5x speedup
- Track Building:
  - CC+Fast WalkThrough → 350x speedup
  - CC+Junction Removal → another 3x speedup
- AthenaTriton provides a 2.4x throughput increase

Conclusions

- We have demonstrated that GNN4ITk is a **viable** algorithm for **HL-LHC tracking**
- **Physics performance comparable with CKF, run @ $\gtrsim$10 evts/s on A100**
- Focus on **decreasing latency**, and reducing the memory usage
- Leveraged acceleration techniques for the GPU, such as **AMP and torch_compile**
- Reducing the size and computation complexity of the GNN models while preserving their **physics performance**
- Optimizing the other steps of our pipeline, including the **Metric Learning** models

Related Contributions to this Conference

- High Performance Graph Segmentation for ATLAS GNN Track Reconstruction by **Daniel T. Murnane**
- EggNet: An Evolving Graph-based Graph Attention Network for End-to-end Particle Track Recontruction by **Jay Chan**
- Energy-efficient graph-based algorithm for tracking at the HL-LHC by **Heberth Torres**
- New approaches for fast and efficient graph construction on CPU, GPU and heterogeneous architectures for the ATLAS event reconstruction by **Christophe Collard** (poster)
- AthenaTriton: A Tool for running Machine Learning Inference as a Service in Athena by **Yuan-Tang Chou**
- Machine Learning Inference in Athena with ONNXRuntime **by Xiangyang Ju**
- Performance of the ATLAS GNN4ITk Particle Track Reconstruction GPU pipeline by **Aleksandra Poreba** (poster)
- Online track reconstruction with graph neural networks on FPGAs for the ATLAS experiment by **Sebastian Dittmeier**

The End

Thank you!

Back-up

GNN Architecture Optimizations