BERKELEY LAB

TrkX

The ExaTrkX Project

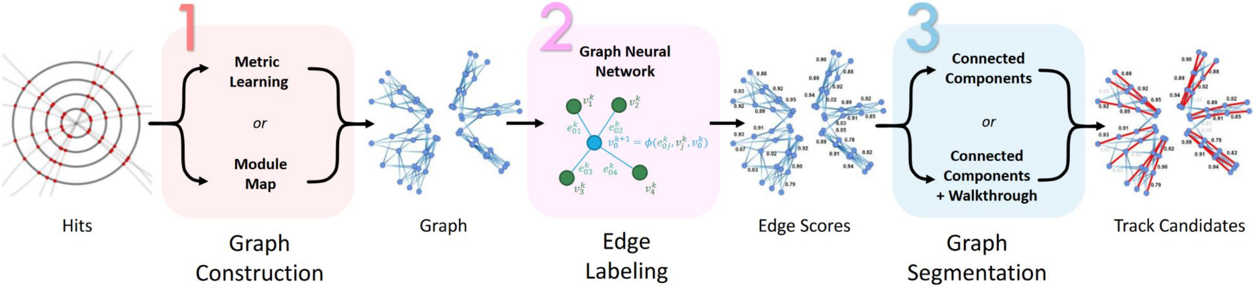
Xiangyang Ju

Nov 30, 2023

AI4EIC 2023 Annual Workshop

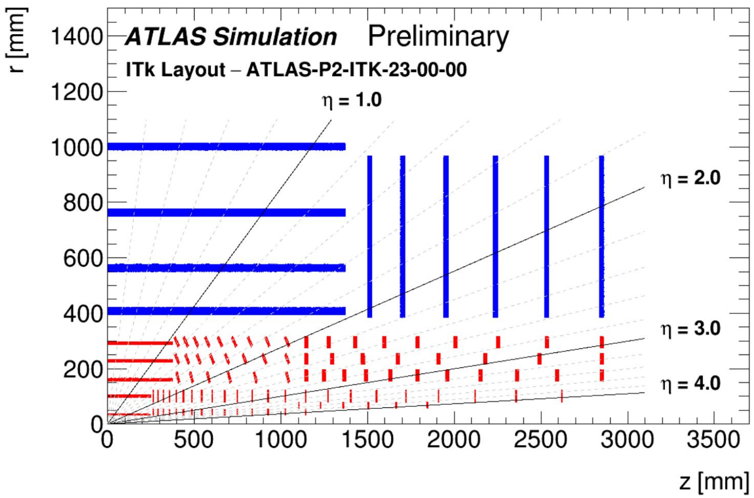
ExaTrkX Pipeline

Collaborating with L2IT group for ATLAS ITk



- ExaTrkX is a ML solution to track finding for High Luminosity LHC
- The pipeline consists 3 discrete steps: graph construction, edge labeling, graph segmentation
- Graph construction: module map and **Metric Learning**
- Edge labeling: Graph Neural Network
- Graph Segmentation: **Connected Components** & Walkthrough

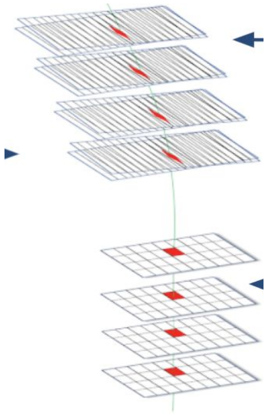
Apply ExaTrkX to ITk



Challenges:

- Large number of hits, ~300k (3x more than those in the Track ML dataset)
- About 50% noise hits per event in simulated tt events with mu = 200
- Shared hits → Ambiguity
- *Electrons → Energy loss, delta rays*
- Low resolution of hit positions for strip detectors → poor GNN performance

Heterogeneity in GNN tracking



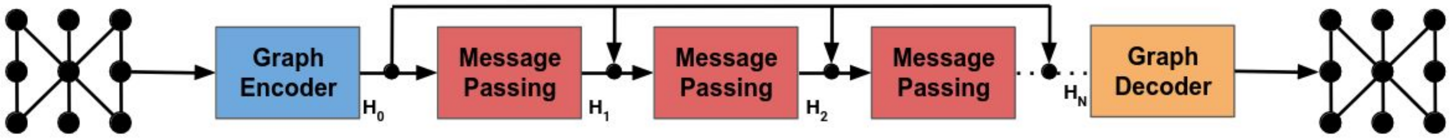
Heterogeneous data:

- Pixel detector: one spacepoint = one cluster, [r, φ, z]
- Strip detector: one spacepoint = two clusters, [r, φ, z] + cluster one + cluster two

In CTD 2022 results, the two cluster information for the strip SP **was not used.**

Heterogeneous GNN:

- In the Graph Encoder, use different MLPs to encode Strip and Pixel spacepoints differently
- Or / And in the message passing, encode messages differently for Pixel and Strip spacepoints



Explore Heterogenous Data

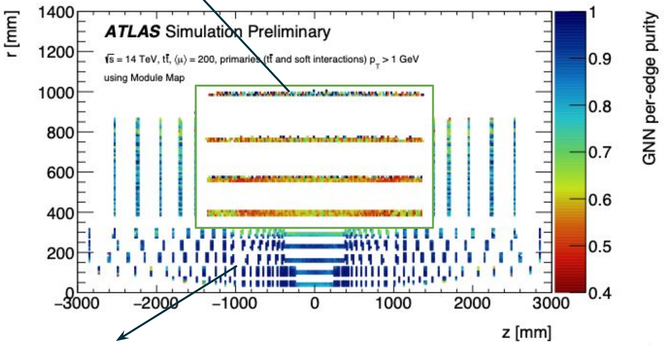
Key idea: Add cluster features to spacepoint features

- For Strip spacepoints in barrel region, add the two associated cluster information
- For Pixel spacepoints and Strip spacepoints in endcap region, repeat its features to reach the same length

The GNN model is re-trained with the "extended node features". We call the trained model as the "*Extended GNN*"

$(r_{cluster1}, \varphi_{cluster1}, z_{cluster1}, \eta_{cluster1}, r_{reco}, \varphi_{reco}, z_{reco}, \eta_{reco}, r_{cluster2}, \varphi_{cluster2}, z_{cluster2}, \eta_{cluster2})$

In the Extended GNN, we use different Message Passing Modules for each message passing step



$(r_{reco}, \varphi_{reco}, z_{reco}, \eta_{reco}, r_{reco}, \varphi_{reco}, z_{reco}, \eta_{reco}, r_{reco}, \varphi_{reco}, z_{reco}, \eta_{reco})$

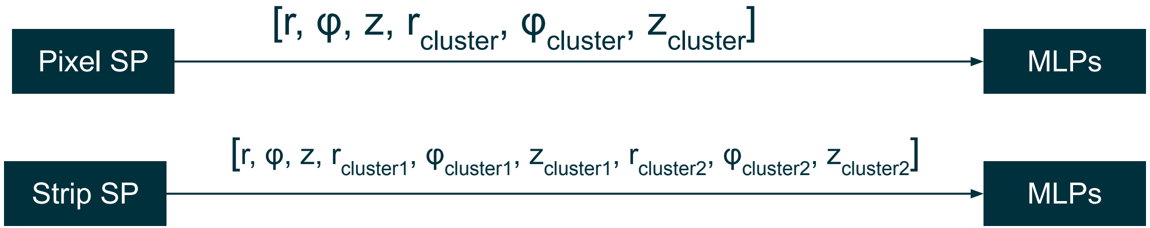
Explore Heterogeneous GNN

Experimental setup

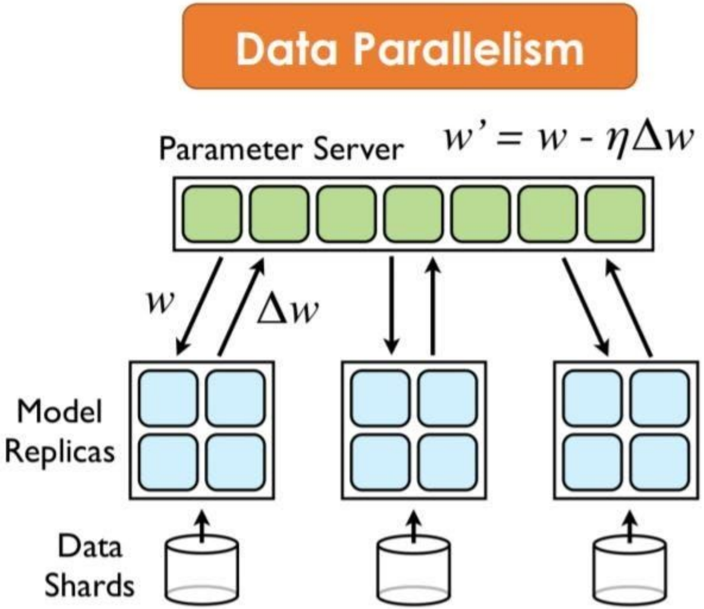
- Graphs constructed from the metric learning
- Use the "*extended spacepoint features*" without eta
- But ***do not*** pad pixel spacepoints with its features to reach the same length

Heterogeneous GNN

- Use a heterogenous Graph encoder



Distributed ML Training



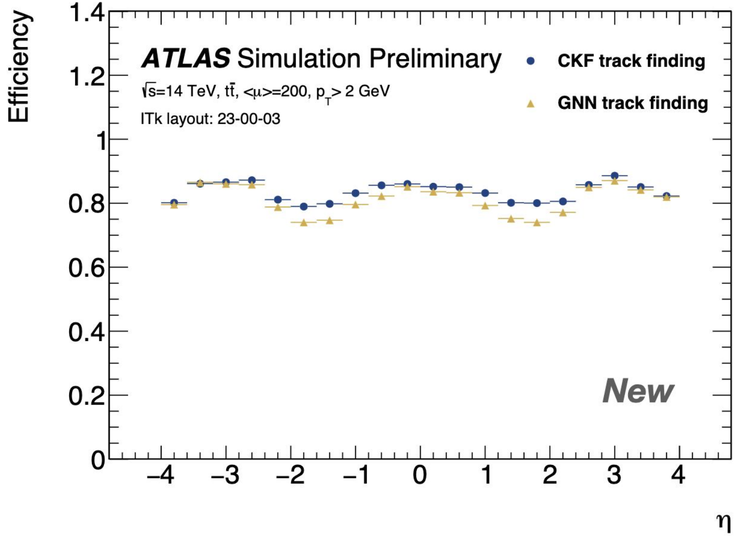
Thanks to the [Pytorch Lightning framework](), it becomes easy to perform "data parallelism" distributed training. It supports

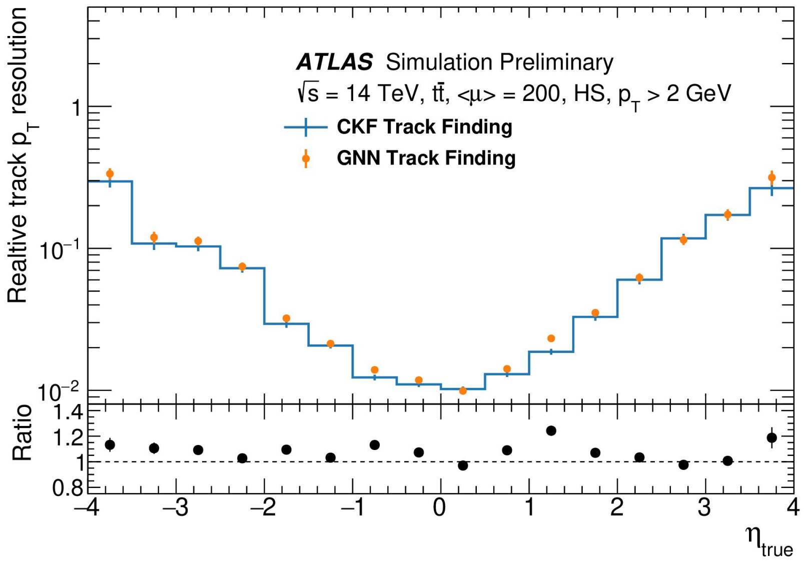
- different accelerators: CPUs, GPU, IPUs, and so on
- Distributed training on GPUs across different computing nodes (particularly useful for HPCs)
- Mixed precision (useful for memory hurry models)
- And so on…

Results on ATLAS ITk

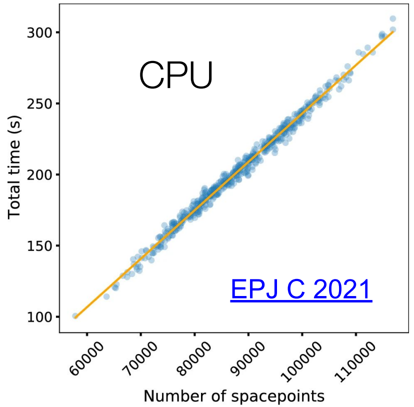
ATL-SOFT-PROC-2023-038
Public Plots, CHEP 2023 Talk,
CTD 2023 Talk

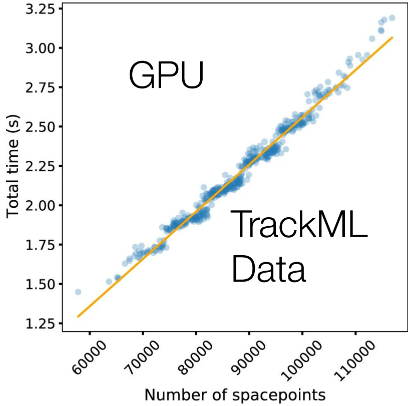
ExaTrkX + Global chi2





Accelerating the ExaTrkX Pipeline





Intel Xeon 8268s
NVIDIA V100

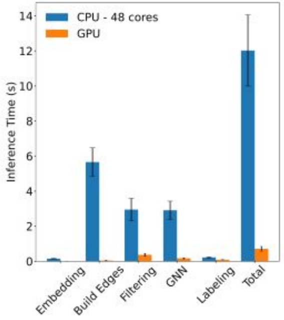


Figure 1. Inference Time on GPU and CPU (48 cores).

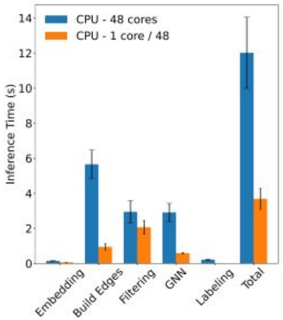


Figure 2. Inference Time on CPU, one core vs 48 cores.

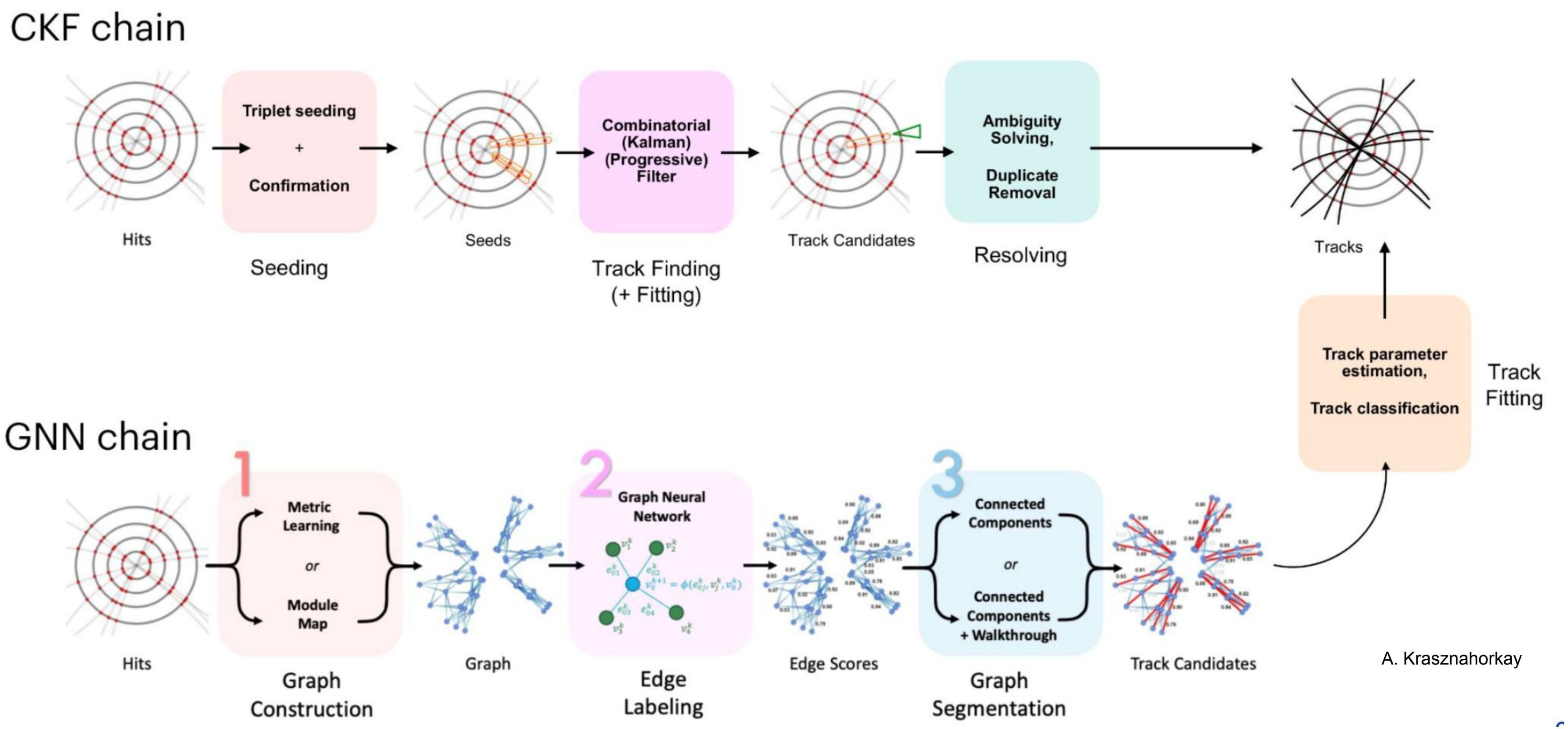
Lazar et al
ACAT 2021
CTD 2022

- ✓ *Achieved ~linear scaling vs # hits*
- Sped up GPU inference 20x
  - ✓ *< 1s wall-clock on GPUs*
  - Now dominated by Filtering MLP & GNN
- CPU inference 15x-200x slower
  - Parallelized, not yet optimized

Table 1. Wall time of the Python-based Inference pipeline for the baseline and optimized implementations. The time is calculated with 500 events on an Nvidia Volta 100 GPU with a memory of 16 GB. The reported times are the average time and the standard deviation of the time in the unit of seconds.

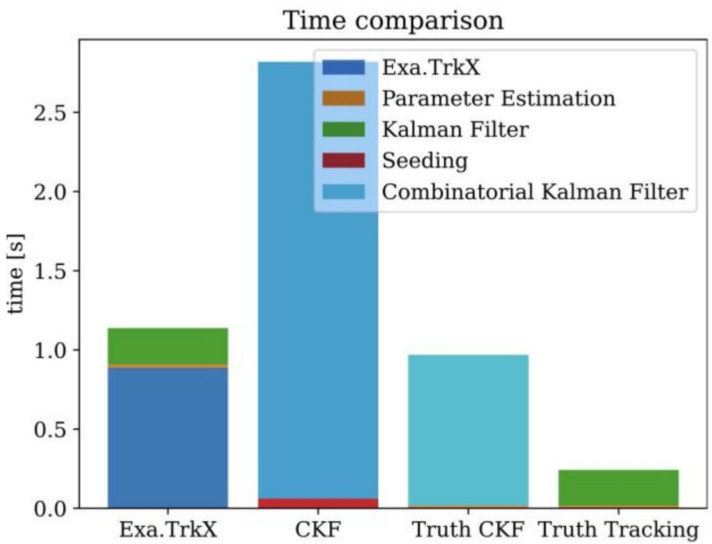
	Baseline	Faiss	cuGraph	AMP	FRNN
Data Loading	0.0022 ± 0.0003	0.0021 ± 0.0003	0.0023 ± 0.0003	0.0022 ± 0.0003	0.0022 ± 0.0003
Embedding	0.02 ± 0.003	0.02 ± 0.003	0.02 ± 0.003	0.0067 ± 0.0007	0.0067 ± 0.0007
Build Edges	12 ± 2.64	0.54 ± 0.07	0.53 ± 0.07	0.53 ± 0.07	0.04 ± 0.01
Filtering	0.7 ± 0.15	0.7 ± 0.15	0.7 ± 0.15	0.37 ± 0.08	0.37 ± 0.08
GNN	0.17 ± 0.03	0.17 ± 0.03	0.17 ± 0.03	0.17 ± 0.03	0.17 ± 0.03
Labeling	2.2 ± 0.3	2.1 ± 0.3	0.11 ± 0.01	0.09 ± 0.008	0.09 ± 0.008
Total time	$15 \pm 3.$	3.6 ± 0.6	1.6 ± 0.3	1.2 ± 0.2	0.7 ± 0.1

End-to-End tracking



A. Krasznahorkay

ExaTrkX + ACTS demonstrator



ICHEP 2022, Benjamin Huth,
ACTS, University of Regensburg

Integrating into ACTS allows us to compare ExaTrkX pipeline with the existing algorithms. Link to the code.

A preliminary computing time comparison between conventional algorithms (CKF) and the ExaTrkX

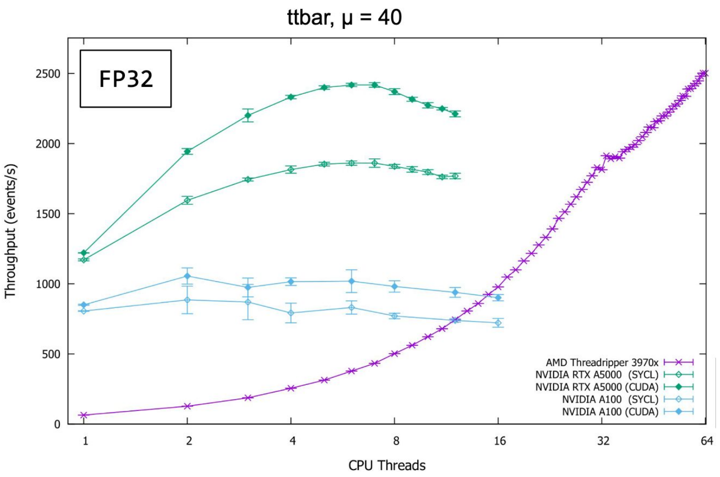
- ExaTrkX was run in GPUs, while CKF in CPUs

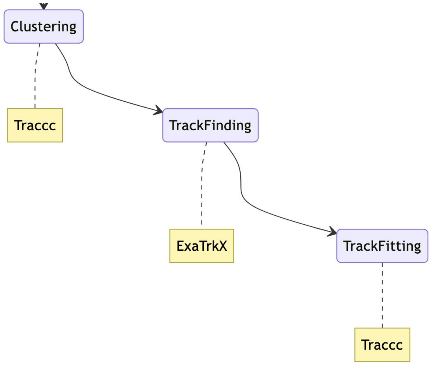
A GPU-version of ACTS is under development [traccc]. Would be interesting to compare ExaTrkX with the GPU version.

Traccc

Category	Algorithms	CPU	CUDA	SYCL	Futhark
Clusterization	CCL	✅	✅	✅	✅
	Measurement creation	✅	✅	✅	✅
Seeding	Spacepoint formation	✅	✅	✅	⚪
	Spacepoint binning	✅	✅	✅	⚪
	Seed finding	✅	✅	✅	⚪
	Track param estimation	✅	✅	✅	⚪
Track finding	Combinatorial KF	✅	✅	🟡	⚪
Track fitting	KF	✅	✅	✅	⚪

✅: exists, 🟡: work started, ⚪: work not started yet

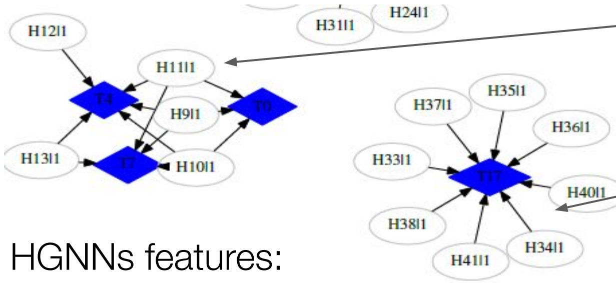




- WIP: ExaTrkX + Traccc to achieve a GPU-supported End-to-End tracking

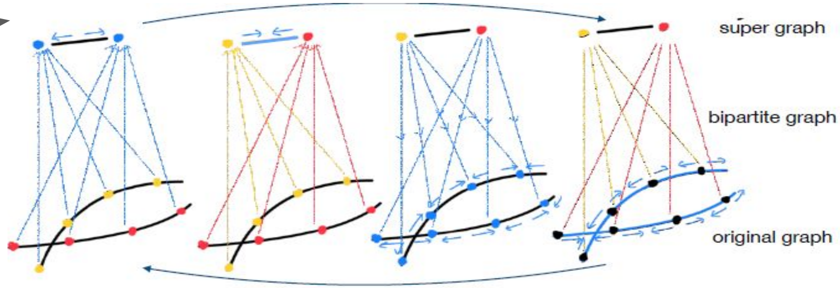
Hierarchical GNN

[Ryan Liu - ACAT 2022](#)



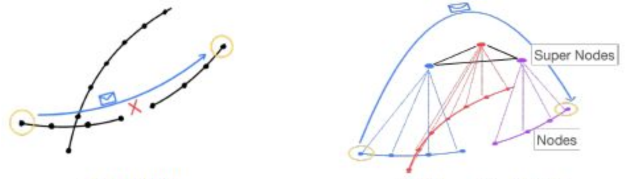
Tracks

Hits



HGNNs features:

- Direct representation of high-level concepts (“track”)
- Multi-scale information pooling
- Long-range interactions even across **missing edges**



(a) Flat GNN (b) Hierarchical GNN

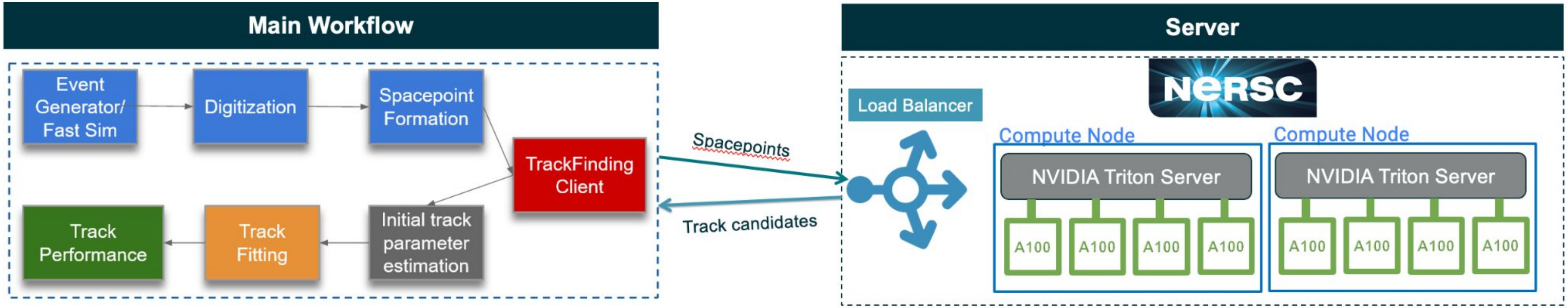
Percent Edge Removed	0%	10%	20%	30%	40%	50%
BC Efficiency	98.55%	98.39%	97.68%	96.63%	95.10%	92.79%
BC Fake Rate	1.23%	1.55%	2.13%	3.10%	4.75%	7.31%

Tested on small “TrackML-1GEV” events (~1K particles/10K spp)

Scaling to higher densities to be understood

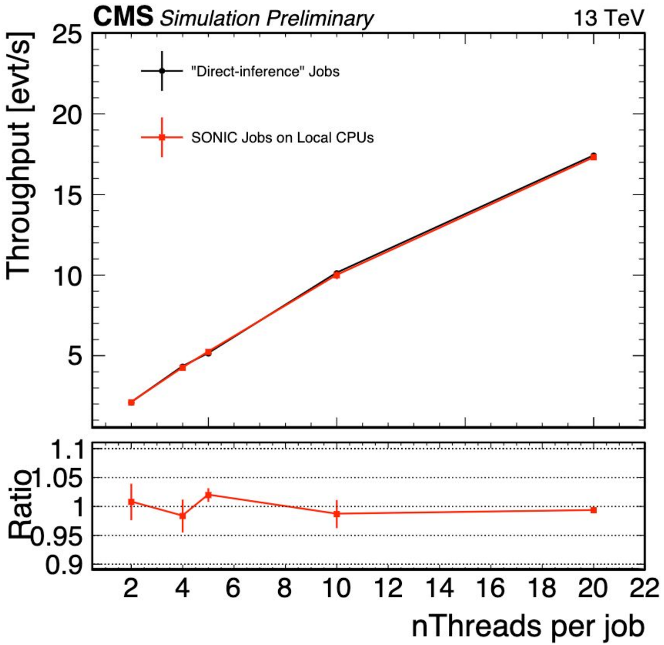
ExaTrkX as a Service

CTD 2022, CTD 2023



- It separates ML algorithms from the production framework.
  - No need to install dependencies in the production framework that will only be used by one algorithm
  - No need to change the production code when the algorithm is changed
    - ML models can run on different coprocessors in different ML frameworks
- Server can be local.

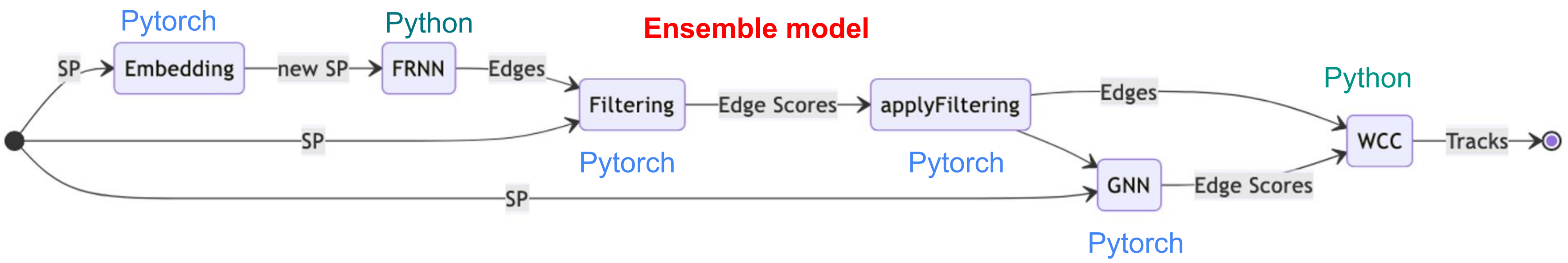
Overhead for local services



- Studies by CMS shows no overhead of running a "server" on the same machine as the "client"
- That means we can have the framework factorization for free, enabling a quick R&D turnaround time

Ensemble Backend

- GNN-Based Tracking is a complex workflow, consisting of 5 discrete sub-algorithms
- Ensemble scheduling uses greedy algorithms to schedule each algorithms
  - **Pros**: directly use existing Triton inference backends
  - **Cons**: little control with the data flow and algorithm scheduling, increasing the IO operations and latency

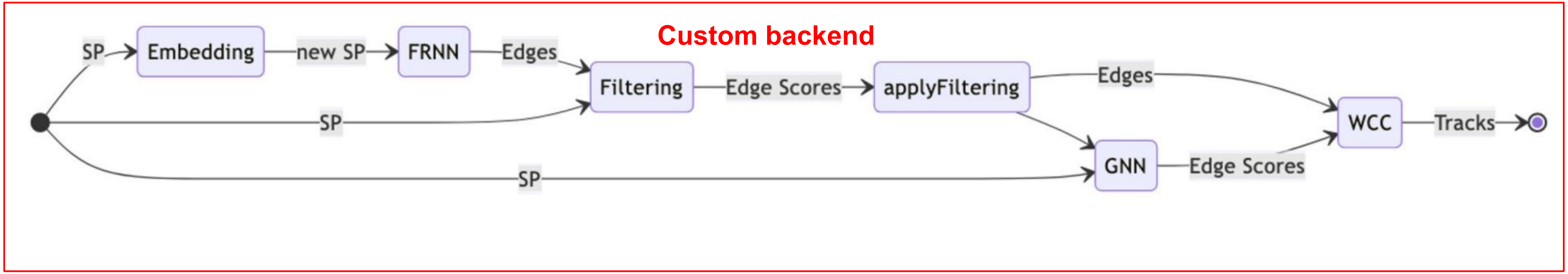


Customized Backend

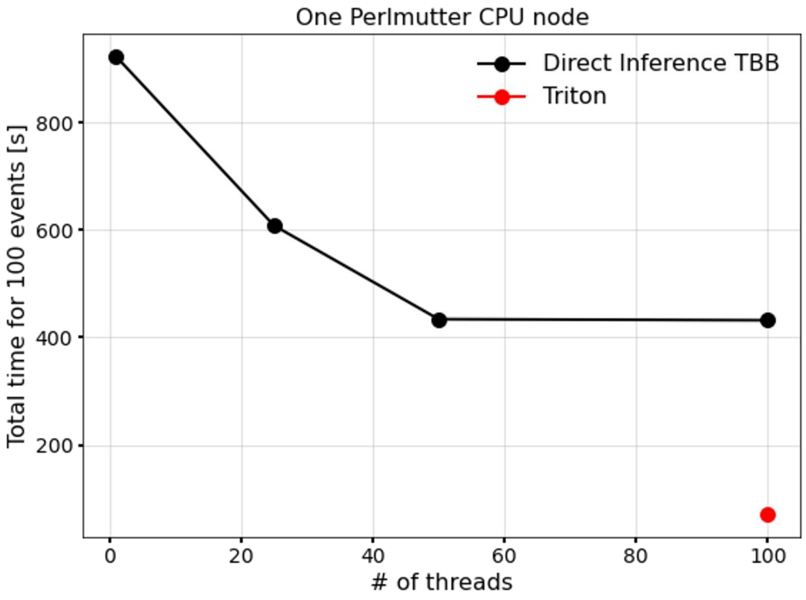
Customized backend provides means to receive requests from and send outputs to the client.

Pros : low overhead, full control of data flow and devices;

Cons : need to write user's own inference code We build customized backends for the GPU-only ExaTrkX inference service and the CPU-only (fallback).

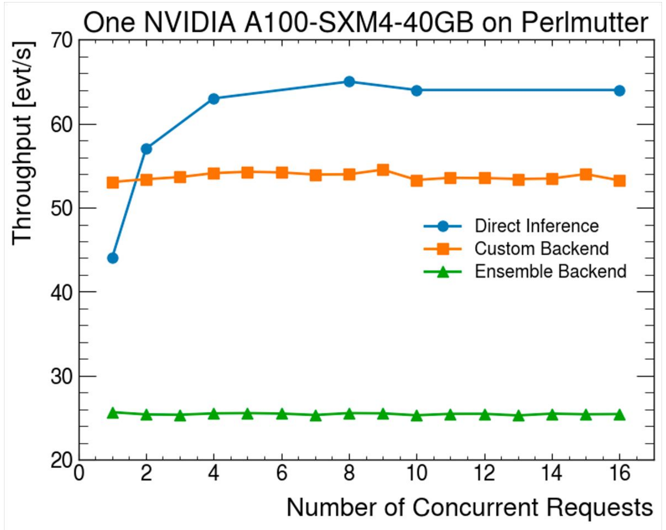


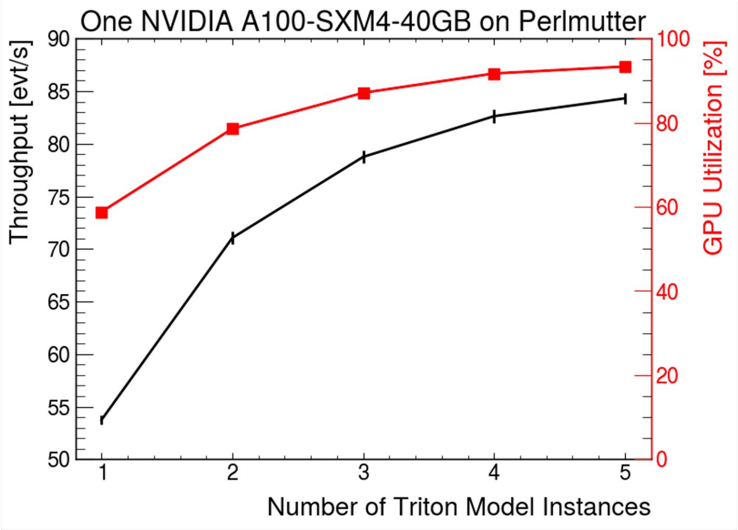
CPU-based GNN Tracking Service



Triton Server knows how to better utilize CPU resources than a simple TBB scheduling

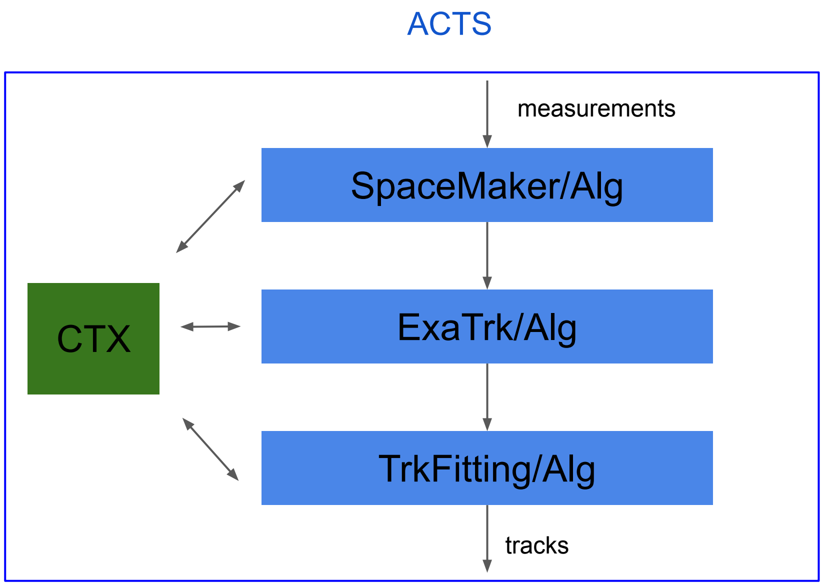
GPU-based GNN Tracking Service





- Increasing Triton model instances increases the GPU utilization and throughput
- Customized backend is better than Ensemble model for complex workflow like the GNN-based Tracking
- Direct inferences require higher concurrency to reach maximum throughput

ACTS with ExaTrkX for track finding

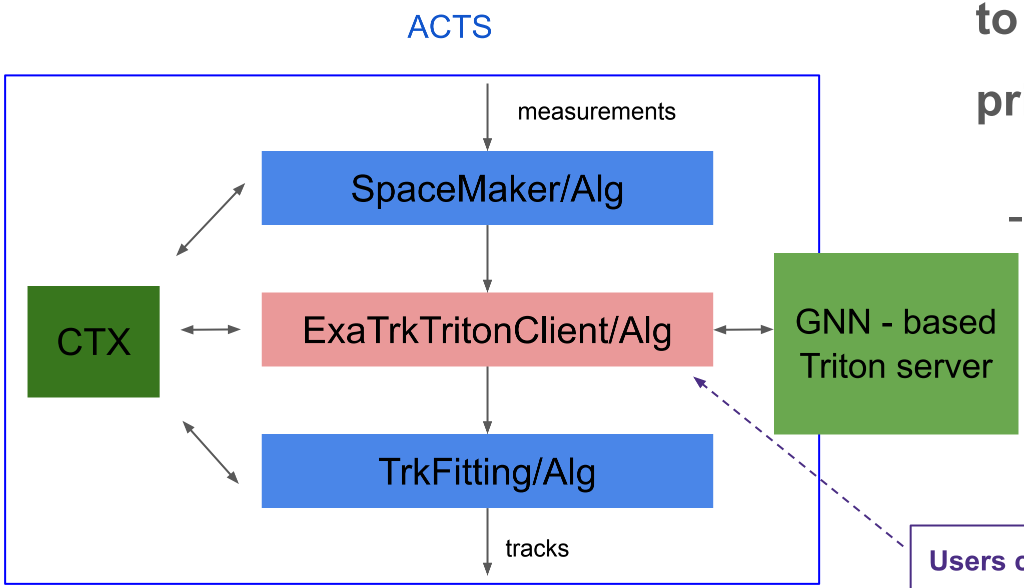


ExaTrk TorchScript implementation done by Benjamin Huth

TrackFinding (ExaTrk) can run locally with CPU/GPU

ACTS TrkFitting still run only on CPU

ACTS with ExaTrkX aaS



Share most of the direct inference code to do preprocessing (scaling, covert into primitive type vector…etc)

- More fair comparison when doing timing studies

Users can swap between direct or triton inference easily

Conclusion

- ExaTrkX pipeline is stepping towards production-level particle tracking for offline tracking
  - Our current focus is on making the paper public,
  - Next is on computational performance
- From R&D to production, I recommend Triton. It will significantly reduce the integration time and keeps full flexibility of updating the pipeline.
- For the distributed training of ML models, use existing frameworks.