Accelerating the Inference of Machine Learning-based Track Finding Pipeline

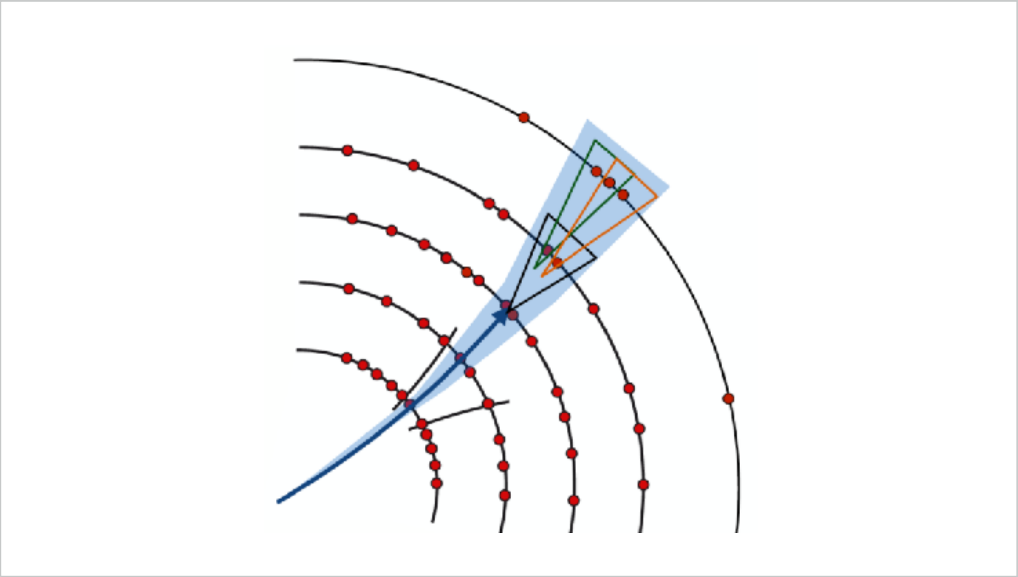
Alina Lazar
on behalf of the
Exa.TrkX
collaboration

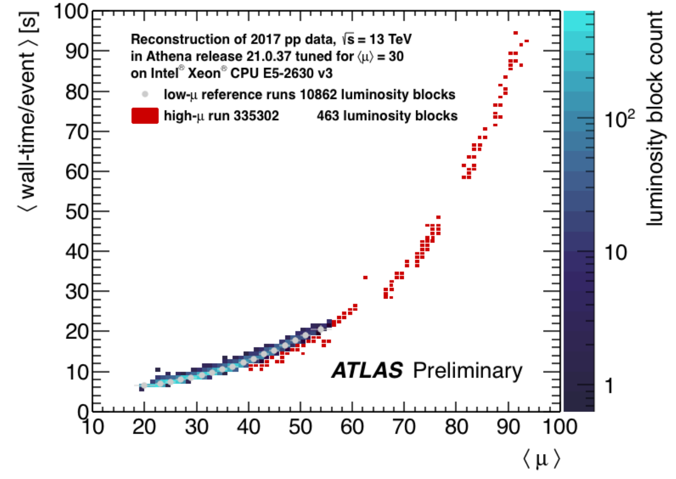
ACAT 2021
12/02/2021

BERKELEY LAB

Track Reconstruction

- Current track reconstruction techniques (variations of Kalman Filters) approximately scale **quadratically** with the number of events/collisions/particles, since solving a large combinatorial problem
- Graph Neural Networks (GNNs) offer the possibility of solving combinatorial problems in ***less-than-quadratic*** time
- Multithreading is essential

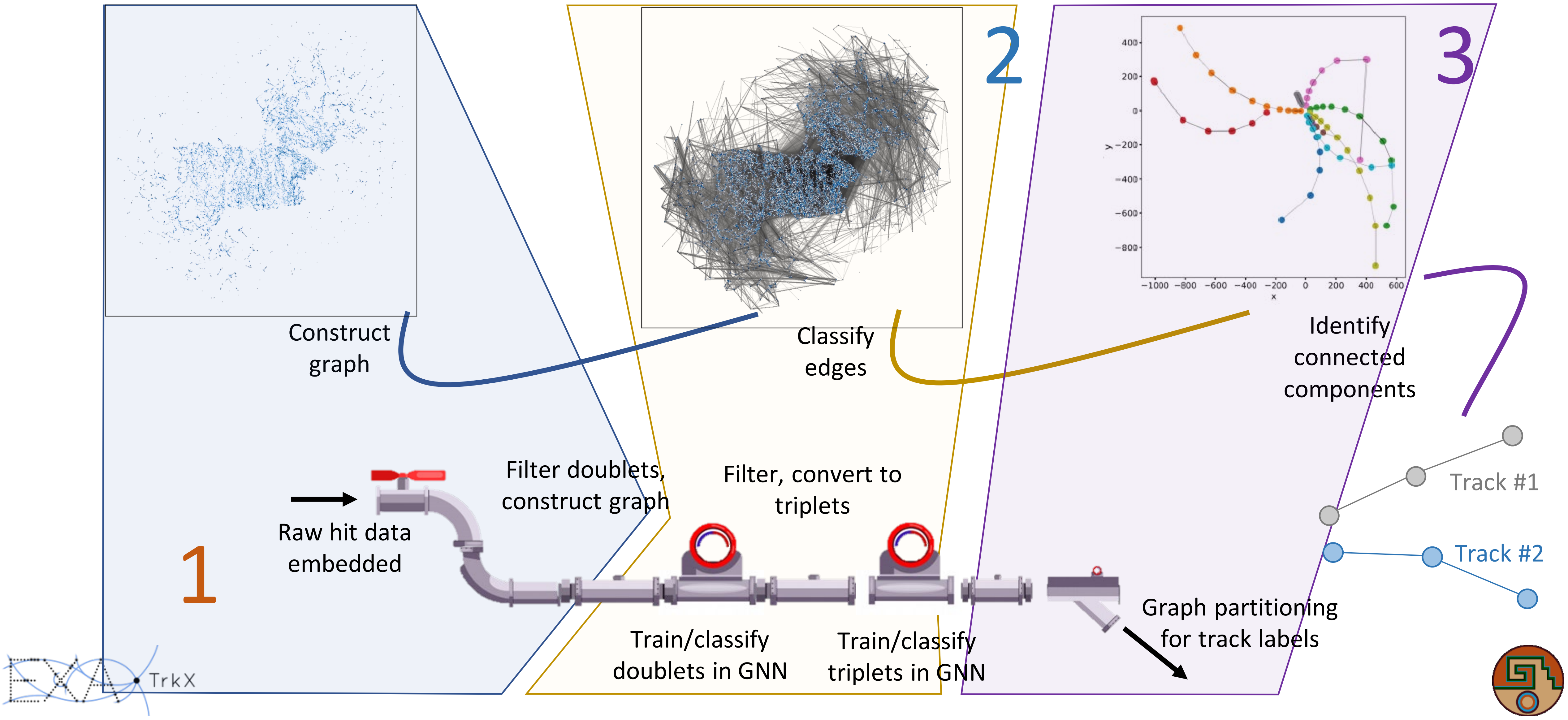




TrkX

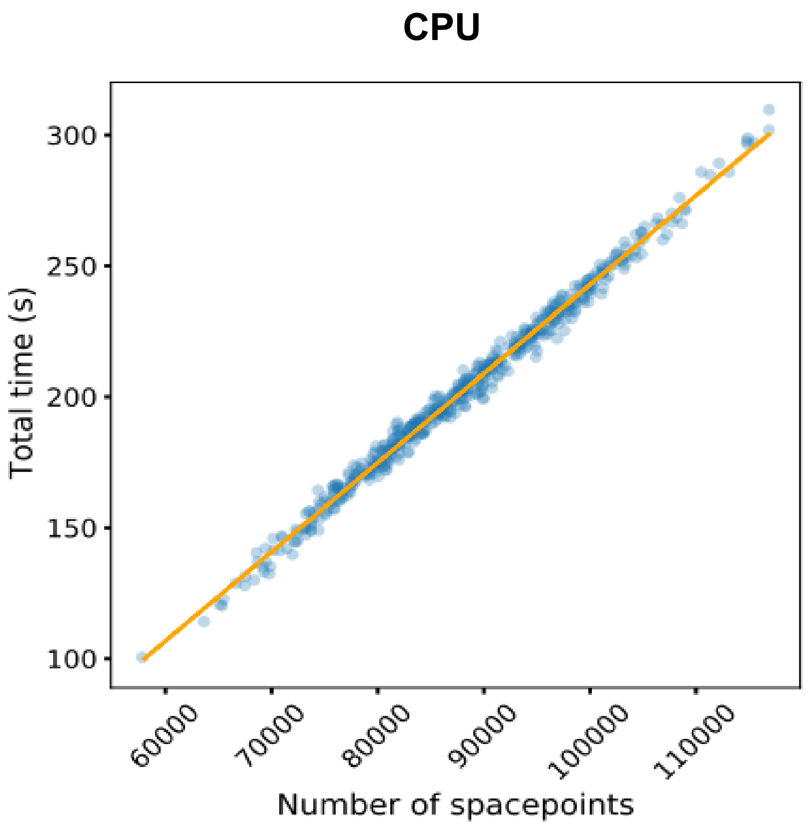
The Exa.TrkX Track Reconstruction Pipeline

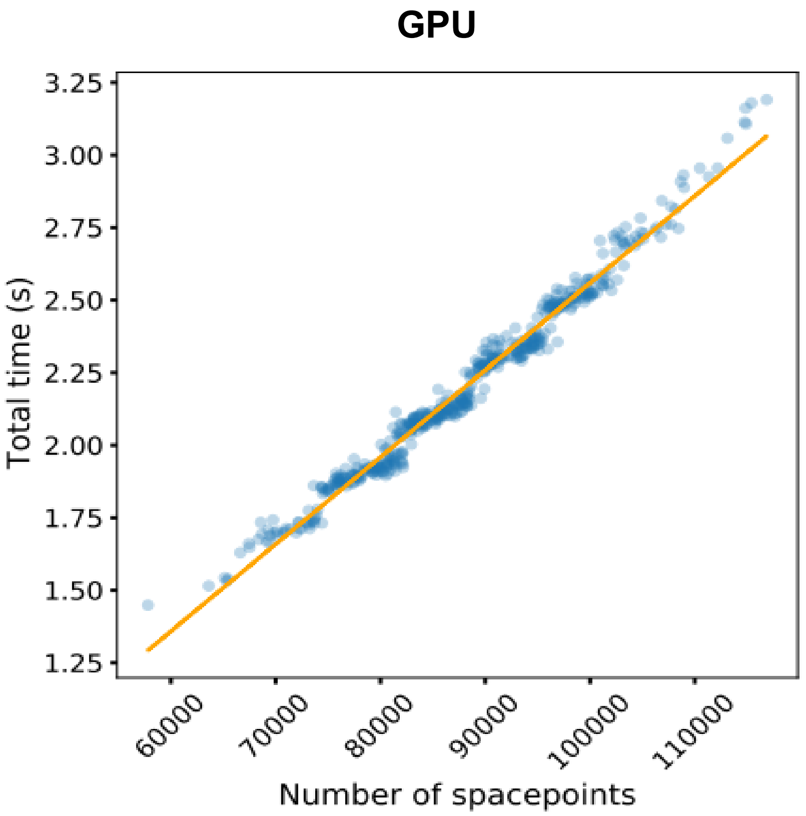
Code available @ https://github.com/HSF-reco-and-software-triggers/Tracking-ML-Exa.TrkX



Exatrkx Track Reconstruction - Scaling

Computing performance scales linearly with the number of input space points.





Ju, X., Murnane, D., Calafiura, P., Choma, N., Conlon, S., Farrell, S., Xu, Y., Spiropulu, M., Vlimant, J.R., Aurisano, A. and Hewes, J., 2021. Performance of a geometric deep learning pipeline for HL-LHC particle tracking. *The European Physical Journal C*, *81*(10), pp.1-14. https://link.springer.com/article/10.1140/epjc/s10052-021-09675-8

TrkX

Baseline (python) Implementation of the Exa.TrkX Inference Pipeline

GPU	Elapsed Time (s)
Data Loading	0.0022 ± 0.0003
Embedding	0.02 ± 0.003
Build Edge	11.52 ± 2.65
Filtering	0.67 ± 0.15
GNN	0.17 ± 0.03
Labeling	2.16 ± 0.3
Total Sync	**14.57 ± 3.14**

- Runs on both GPUs and CPUs.
- Embedding and filtering models are trained using the **PyTorch** deep learning framework.
- The build edges, graph construction is done using the **radius_graph** from torch_cluster.
- GNN uses a **TensorFlow** model.
- DBSCAN from scikit-learn was used to cluster edges into tracks.
- NVIDIA Volta **V100**s with 32GB GPU memory
- Record average timing over **500 events** from the TrackML Challenge.
- Peak Memory Usage:

  GPU **16.7 GB**, CPU 11 GB

Timing Optimization

Python Inference Pipeline

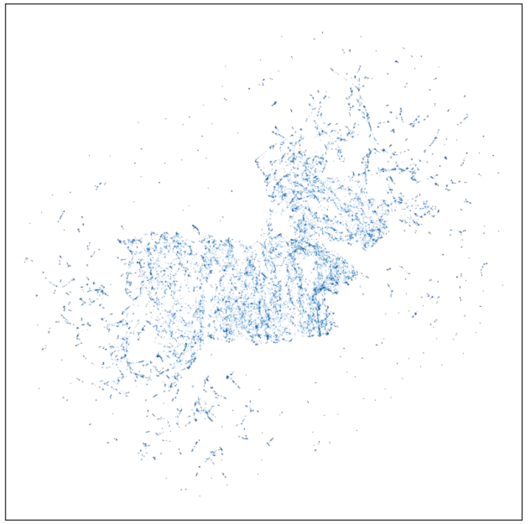
Track Labeling and Mixed Precision

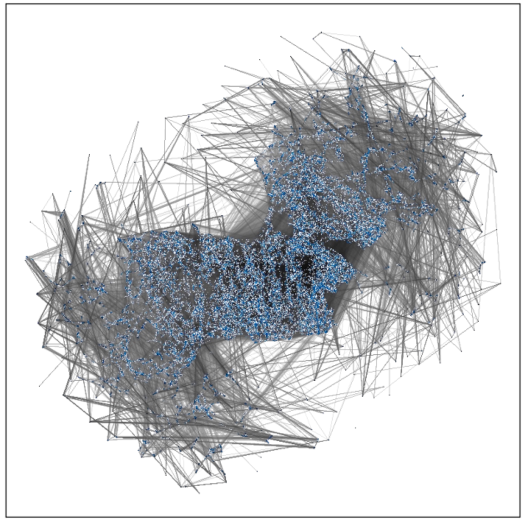
- **Graph Building** – `radius_graph` from `torch_cluster` was replaced with faiss's k-nearest neighbor search for the GPU.
- **Track Labeling** – DBSCAN was replaced by the graph weakly connected components algorithm. We used the RAPIDS cuGgraph on GPU and scikit-network on CPU for the Python implementation.
- **Mixed Precision for Pytorch** – Instances of `torch.cuda.amp.autocast` enable autocasting for chosen regions. Autocasting automatically chooses the best precision for GPU operations to improve performance while maintaining accuracy.

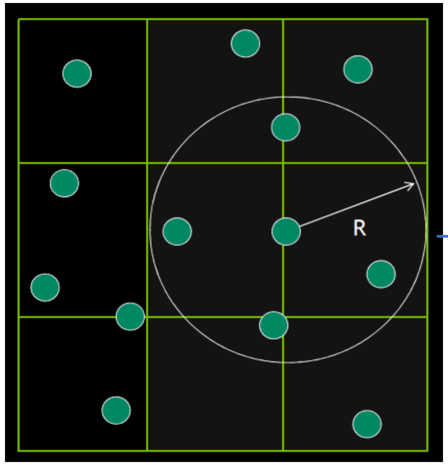
Fast Graph Construction

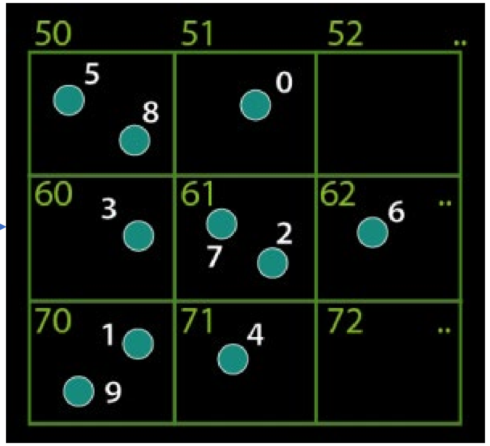
FAST FIXED-RADIUS NEAREST NEIGHBORS: INTERACTIVE MILLION-PARTICLE FLUIDS, Hoetzlein (NVIDIA), 2014

- Started with Faiss (KNN library)
- KNN produces sorted neighbors - this is unnecessary
- Only need **Fixed Radius** neighbors
- Cell-by-cell grid search is ~100x faster than Faiss
- We customized library (https://github.com/lxxue/FRNN/tree/larged) on **Fixed Radius Nearest Neighbor** search algorithm









TrkX

The complexity of finding fixed-radius near neighbors. Bentley, et al 1977

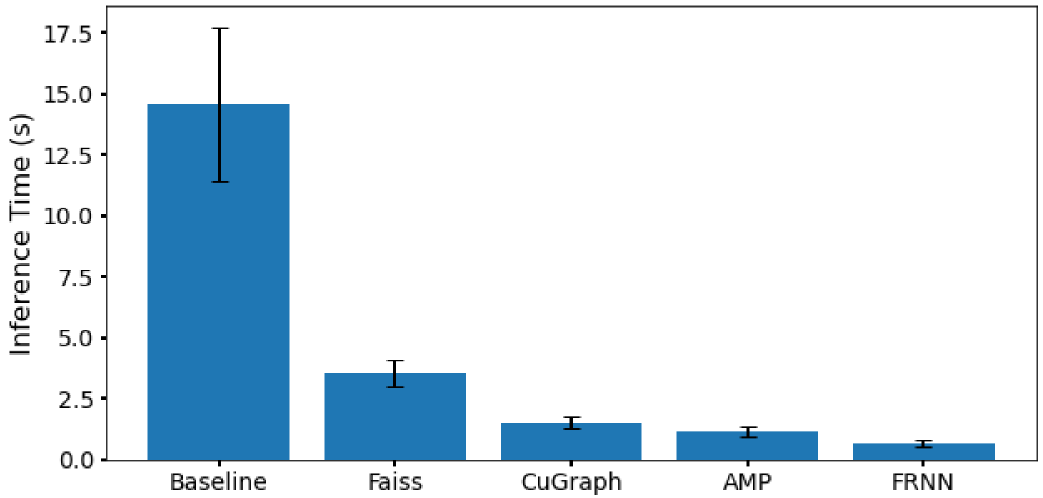
Accelerating NN Search on CUDA for Learning Point Clouds, Xue 2020

Inference Accelerator Technologies on GPUs

	Baseline Imp. (s)	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 ± 0.002	0.02 ± 0.002	0.0067 ± 0.0007	0.0067 ± 0.0007
Build Edge	11.52 ± 2.64	0.54 ± 0.07	0.53 ± 0.07	0.53 ± 0.07	0.04 ± 0.01
Filtering	0.67 ± 0.15	0.67 ± 0.15	0.67 ± 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	2.16 ± 0.3	2.14 ± 0.3	0.11 ± 0.01	0.09 ± 0.008	0.09 ± 0.008
Total Time	14.57 ± 3.14	3.56 ± 0.55	1.53 ± 0.26	1.17 ± 0.18	0.7 ± 0.13

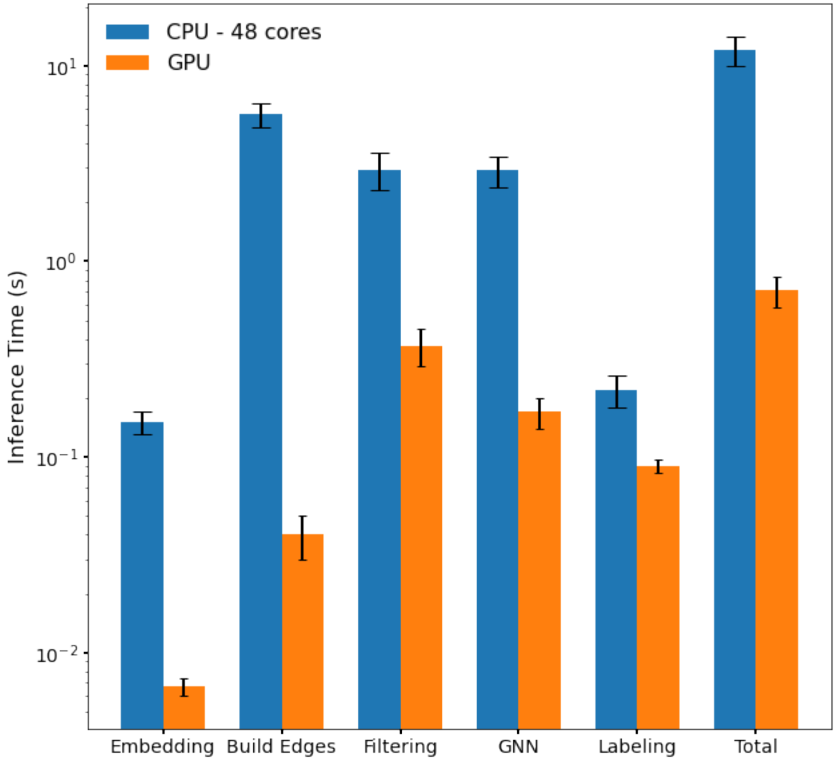
1. Baseline radius_graph → faiss KNN
2. DBSCAN clustering → cuGraph connected components
3. Full precision → mixed precision
4. Faiss knn → FRNN radius graph

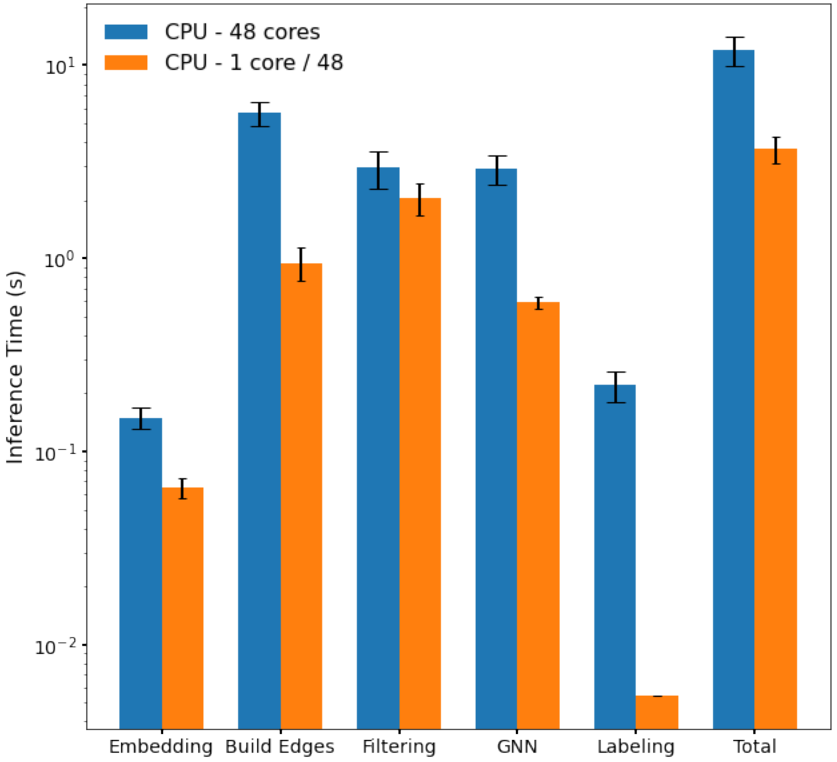
FRNN + mixed precision + cuGraph: 0.7 ± 0.13 sec →



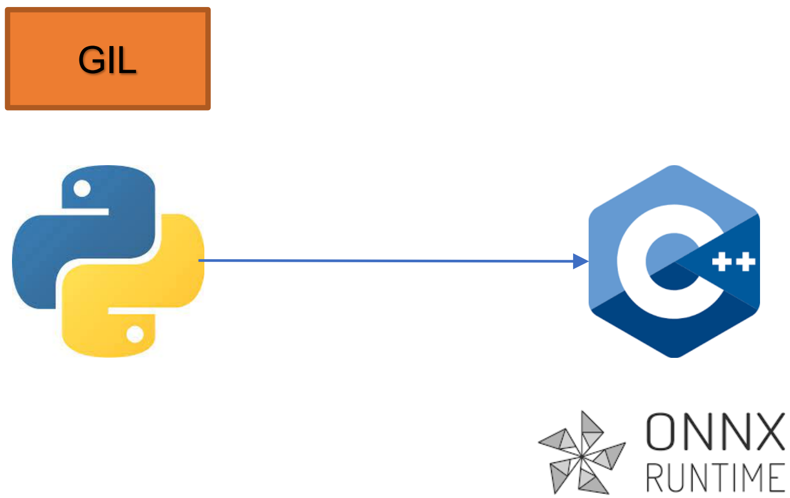
Inference Timing – CPU

- Dual Intel Xeon 8268s Cascade Lakes @2.9GHz with 48 cores/node and 178 GB/node.
- The pipeline steps automatically use multiprocessing when running on multiple cores.
- No optimizations have yet been made to the CPU implementation.
- The CPU-based timing results are still not competitive with the optimized GPU results.





Towards Realistic Python to C++ Conversion

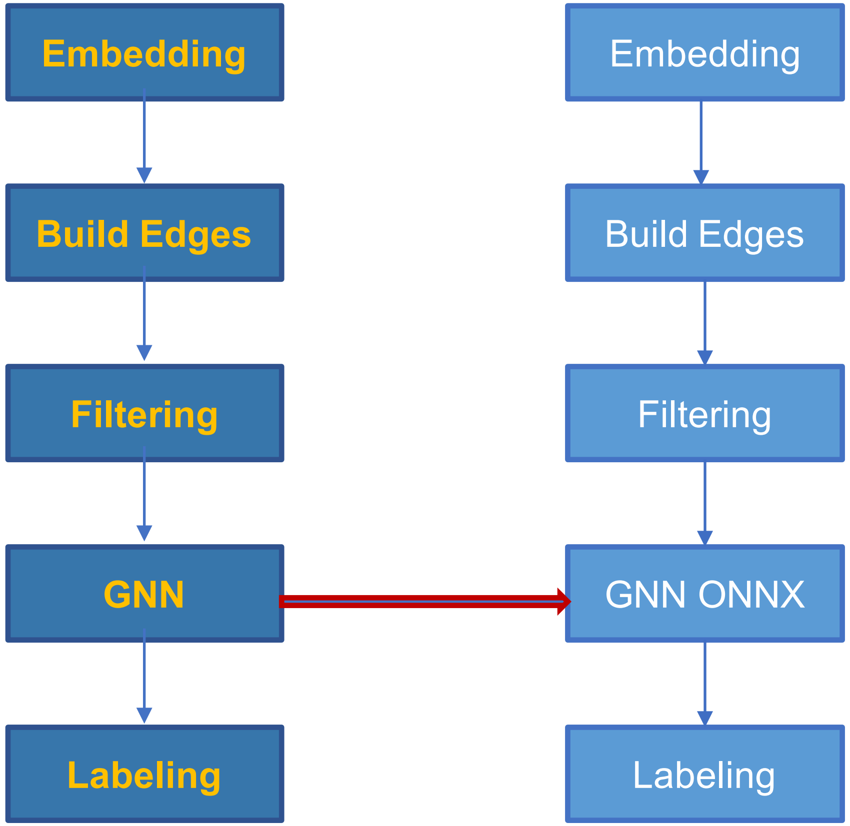


- Provide a mechanism to integrate the **Exa.trkX pipeline** with C++-based event reconstruction workflows.

- Deep learning inference runs predominantly on the **GPUs**.
- Python's threading model is limited by the Global Interpreter Lock (**GIL**), slowing down throughput.
- By converting the pipeline to **C++**, we can overcome Python threading drawbacks.

Python to C++ with ONNX Runtime

- Converted embedding and filtering to ONNX models, GNN to torchscript and to ONNX models, FRNN to C++ using libtorch, and cuGraph to libcugraph
- Technical challenges we had to solve:
  - Used a Docker container to get all the dependencies to work together.
  - Integrated libtorch, ONNX Runtime, libcugraph (Rapids AI)
- **Blocker:** The physics performance of GNN ONNX is compromised by bug in ONNX implementation of **scatter_add** onnx operator.



C++ Integration with ACTS

Integration of Exa.TrkX Inference with acts

Acts - A Common Tracking Software

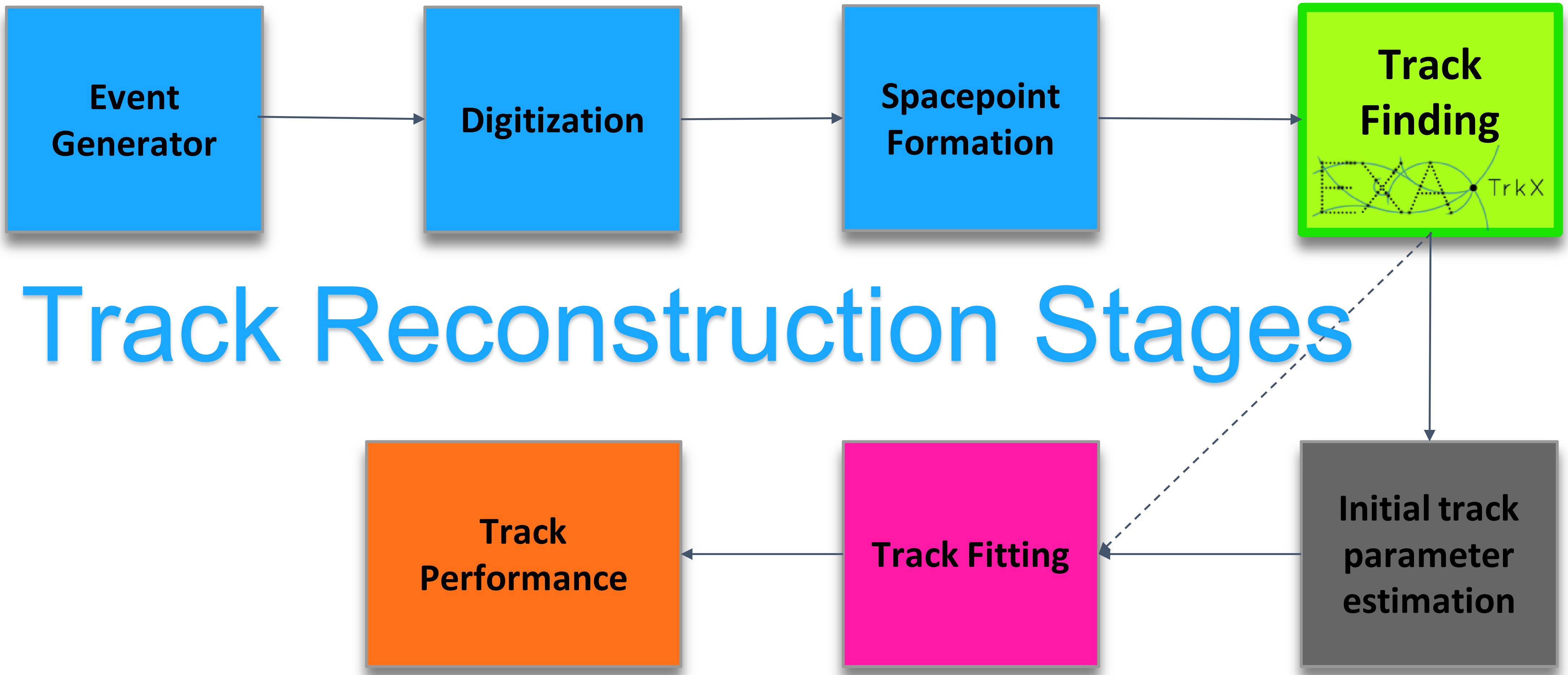
Experiment-independent toolkit for track reconstruction (for future detectors)

Open-source platform for implementing new tracking techniques and hardware architectures

To be useful Exa.TrkX inference must be integrated with experiment tracking pipelines, and ACTS is an experiment-neutral one.

Ai, X., 2019. Acts: A common tracking software. *arXiv preprint arXiv:1910.03128*.
https://arxiv.org/pdf/1910.03128.pdf

Exa.TrkX Integration with acts



Conclusions

- Implemented Faiss KNN for graph construction, replaced DBSCAN with weakly connected components, mixed precision speeds up GPU running time, fixed radius NN for building the radius-based graph
- Event GPU-based **inference** runs in **sub-second** time
- Running inference on multiple CPU cores speeds up running the pipeline, but it still takes ~17x longer.
- We have an implementation of the inference pipeline running in **C++**, in a **multithreading** environment
- This allows the integration of the pipeline with other tracking frameworks such as the **ACTS framework**.
- The C++ pipeline currently runs on CUDA and GPUs

Future Plans

- **Optimize** the performance of the **ONNX Runtime**
- Run the half precision ONNX models in the C++ inference pipeline.
- Run the pipeline with **tensorRT** as the provider for the ONNX Runtime (requires special compilation of the ONNX Runtime)
    - Working with Nvidia experts and other HEP groups to accelerate **GNNs with tensorRT**
- Reduce the number of software library dependencies for Python and C++

Thank you!

Exa.TrkX Collaboration Members:

- Maria Spiropulu, Jean-Roch Vlimant (Caltech)
- Giuseppe Cerati, Lindsey Gray, Thomas Klijnsma, Jim Kowalkowski (FNAL)
- Paolo Calafiura (PI), Xiangyang Ju, Daniel Murnane (LBNL)
- Nick Choma, Sean Conlon, Steven Farrell, Yaoyuan Xu (LBNL)
- Ankit Agrawal, Alexandra Day, Claire Lee, Wei-keng Liao, (Northwestern)
- Gage DeZoort, Savannah Thais (Princeton)
- Pierre Cote De Soux, François Drielsma, Kasuhiro Terao, Tracy Usher (SLAC)
- Adam Aurisano, Jeremy Hewes (UCincinnati)
- Markus Atkinson, Mark Neubauer (UIUC)
- Aditi Chauhan, Alex Schuy, Shih-Chieh Hsu (UWashington)
- Alex Ballow, Alina Lazar (Youngstown State)

TrkX

Exa.TrkX Project

GitHub Repository for C++ implementation: exatrkx/exatrkx-acat2021: The exa.trkx pipeline used for the C++ inference studies presented at ACAT 2021 (github.com)

Docker Container: exatrkx-acat2021/docker at main · exatrkx/exatrkx-acat2021 (github.com)

More details on Exa.TrkX at this conference:

Poster #643, Graph Neural Network for Large Radius Tracking
Poster #774, A Comprehensive Comparison of GNN Architectures for Jet Tagging
Poster #730, Graph Neural Network for Object Reconstruction in Liquid Argon Time Projection Chambers

Thank you!

Background image of first and last slide source: Manuchi via Pixabay
https://pixabay.com/illustrations/background-abstract-line-2462435/