ExaTrkX as a Service

Yongbing Feng(1), Shie-Chieh Hsu(2), Xiangyang Ju(3), Alina Lazar(4)

(1) Fermi National Accelerator Laboratory, (2) University of Washington, A3D3, (3) Lawrence Berkeley National Laboratory, (4) Youngstown State University

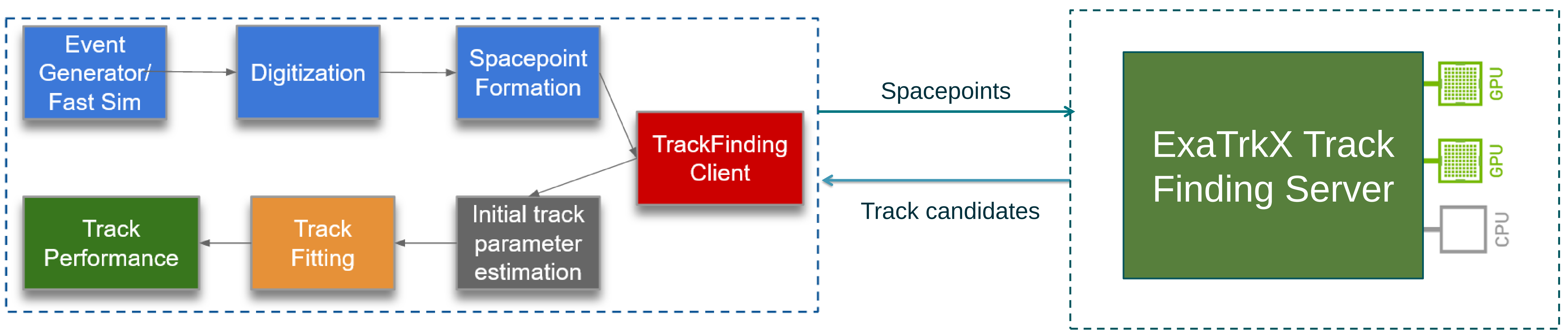
Motivations

Why Inference as-a-Service for production?

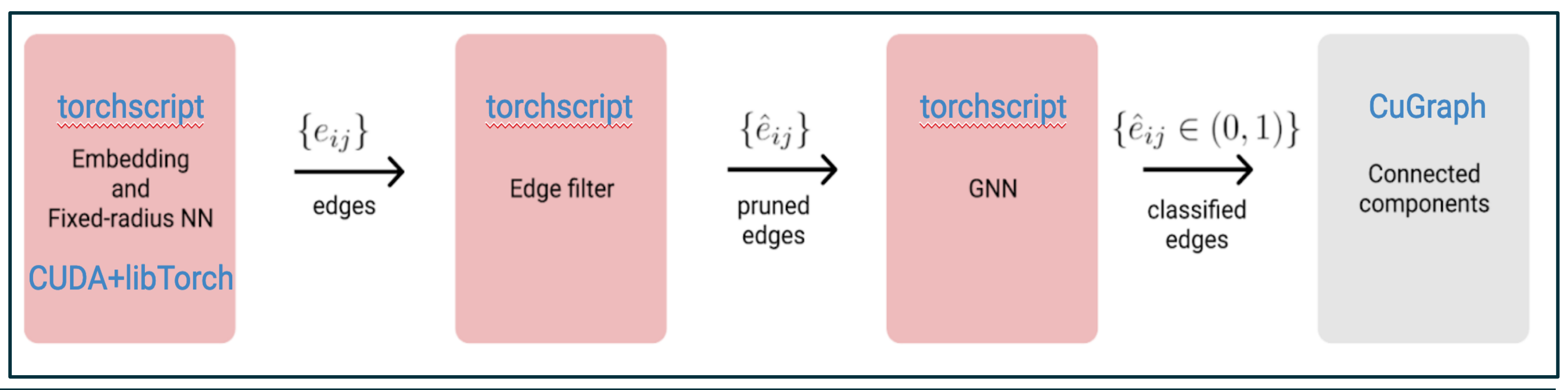
- Factorize out ML framework
  - Easy support for different ML frameworks & models
- Factorize out algorithm scheduling.
  - ML models can be deployed on different coprocessors simultaneously and easily
- Portable solution to supporting different coprocessors
  - No need for client to rewrite code for specific languages
- Event batching → more efficient and sufficient utilization of coprocessors
- Allow access to remote AI accelerators, like GPUs

Track reconstruction (Client)

ExaTrkX (Server)



Track finding takes majority of the tracking reconstruction time. The ExaTrkX pipeline provides a promising acceleration by using the Graph Neural Network. **The goal is to setup the ExaTrkX pipeline as a Service**



Implementation

Step name in the pipeline	Triton server backend
Embedding	Pytorch
Building (FRNN)	Python
Filtering	Pytorch
GNN	Pytorch
Track labeling (CC)	Python
ExaTrkX Model	Ensemble

- Implemented a standalone client code that sends and receives data from the server
- Offloading each step to the server will inevitably introduce overhead
  - Data preparation for offloading
  - Data transfer
- To minimize the overhead, we ensemble individual models and build an ensemble model that performs the end-to-end track finding

Experimental setup

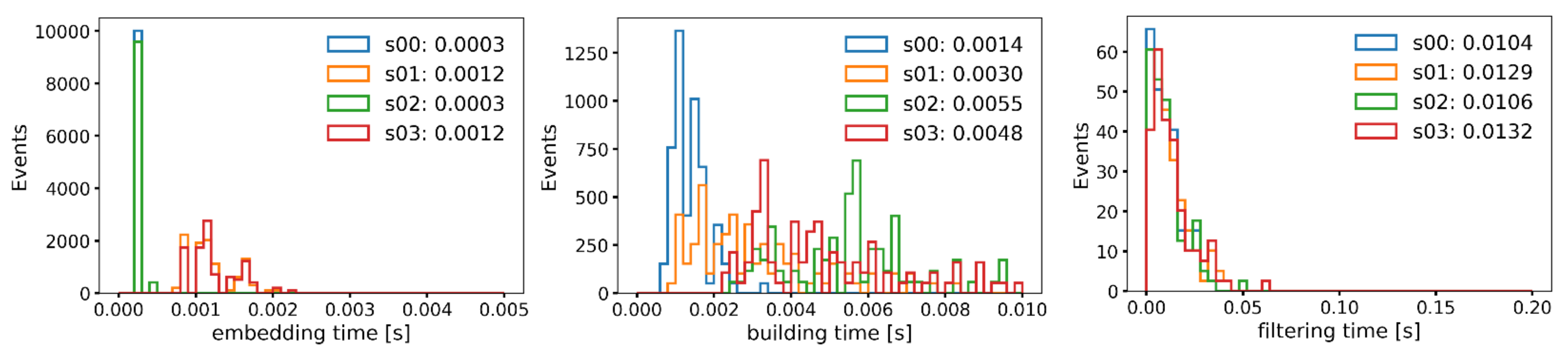
- The timing measurements use 100 heavy neutral lepton events without pileup generated by the ACTS framework
- Events are reprocessed and saved as csv files
- We measure the average time it takes to process the 100 events
- Platform: NERSC V100 GPUs
- Set up the Server to serve the whole pipeline or partial pipeline

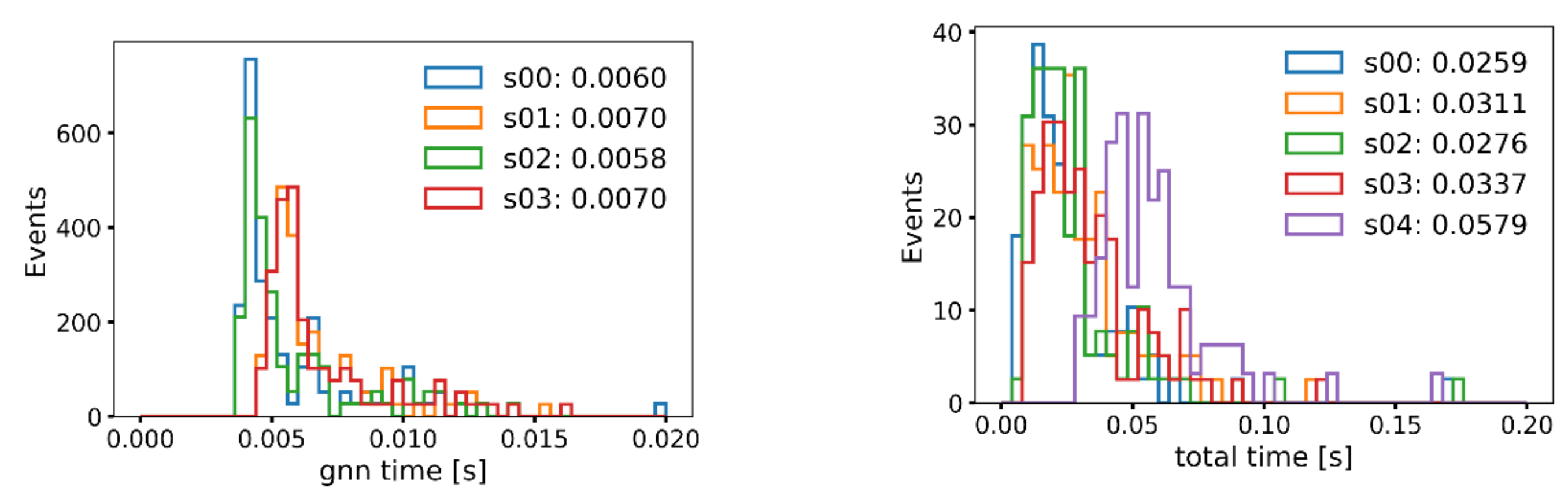
Five service options:

S = 0: no server (direct inference)
S = 1: serving Pytorch models
S = 2: serving FRNN
S = 3: serving all models separately
S = 4: serving the ensemble model

Those measurements were performed on the same machine

Results



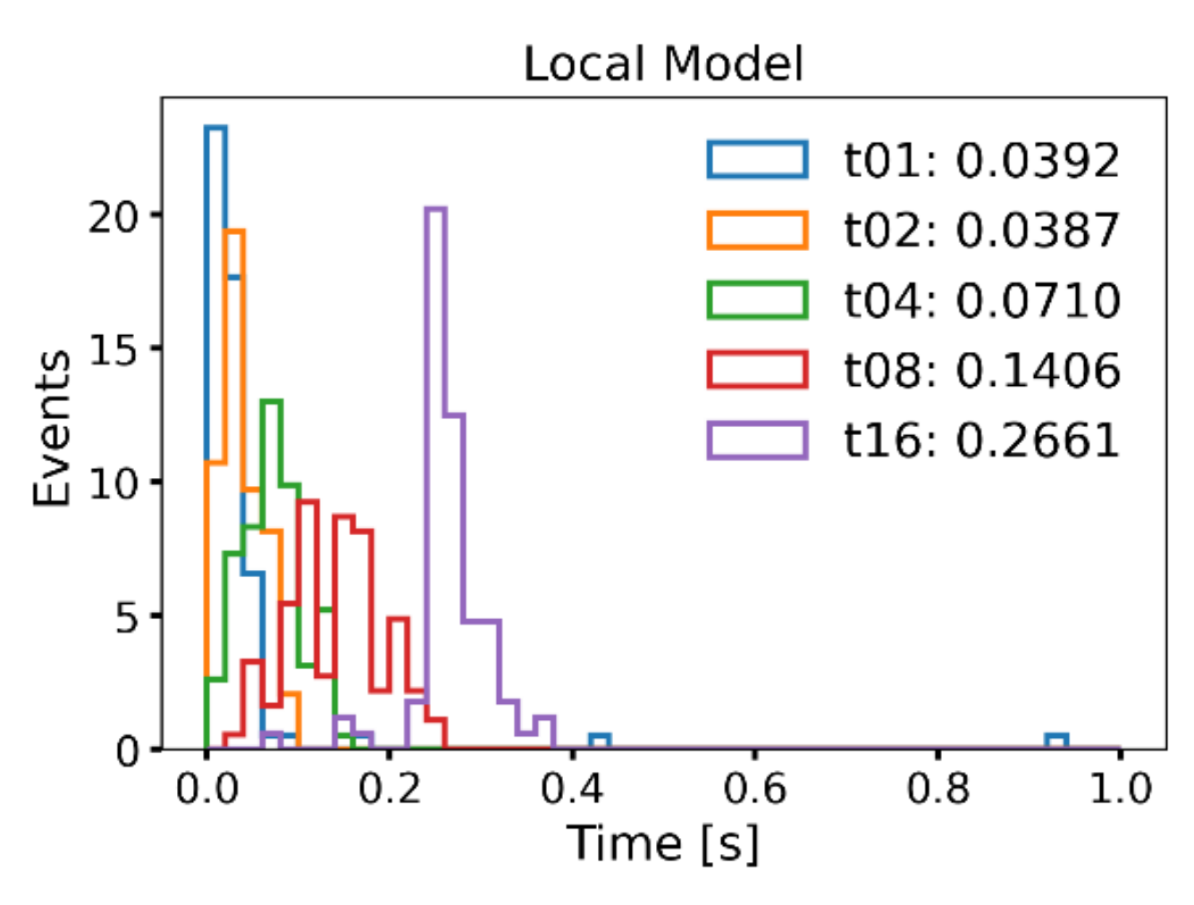


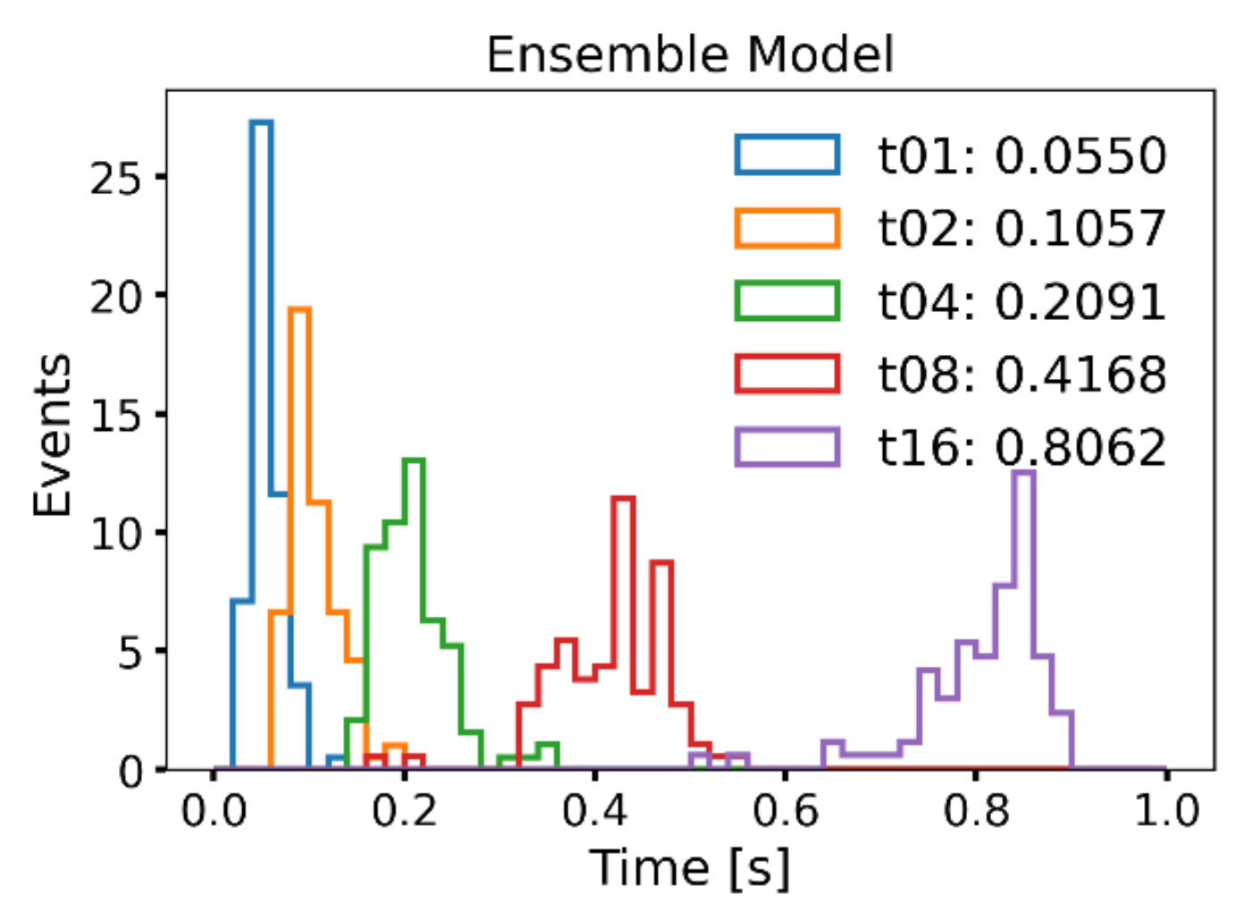
- Embedding runs on the server for options s01 and s03. We observe ~0.9 ms overhead, which would depend on the input dataset size
- FRNN are implemented slightly differently between “no server” and with server
  - Zero copy for “no server” case
- Filtering seems to yield similar performance without or without server, however, we see different performance for GNN
- Ensemble model seems not performing as well as models served separated. Under investigation.

Concurrent event processing

From the client side, we use multithreading to process events for local models or to send requests for remote ensemble models

We observe that for both local models and remote ensemble models, the total inference time increases when concurrent requests/executions are enabled





Conclusions and Outlook

- We implemented the ExaTrkX track finding as a Service and first measured the overhead for offloading crucial steps to the server, respectively and collectively
- We observed that communications between the client and server is a small overhead, even thought relatively large, for a small dataset
- Continue studies in the future
  - Measuring the performance with more realistic data and models
  - Testing the scalability of the service
  - Measuring the network latency
  - Estimating resource requirements for online data processing

Reference: *Eur. Phys. J. C* **81,** 876 (2021)

Acknowledgement:
This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231 using NERSC award ERCAP0021226
Hsu is supported by NSF award No. 2117997.