P2r indexing scheme discussion

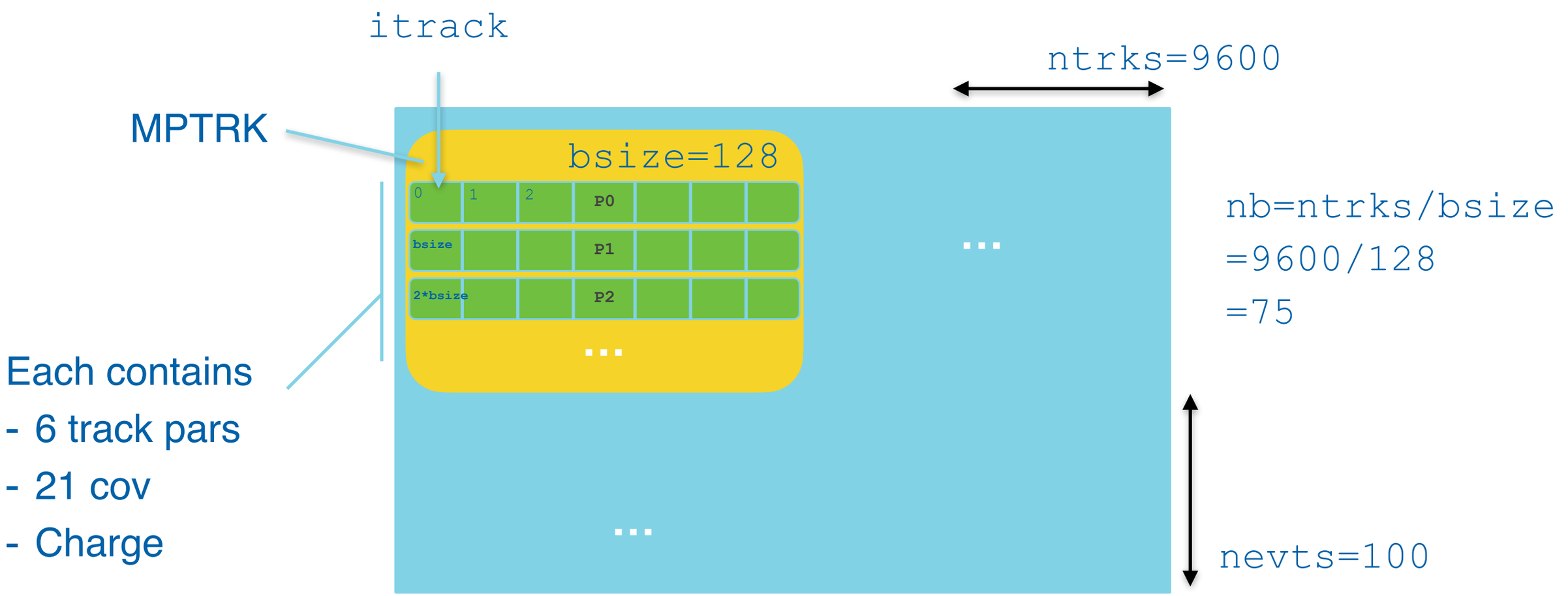
Martin Kwok, Matti Kortelainen (FNAL)

p2z meeting

30 Mar, 2021

Indexing scheme in p2z

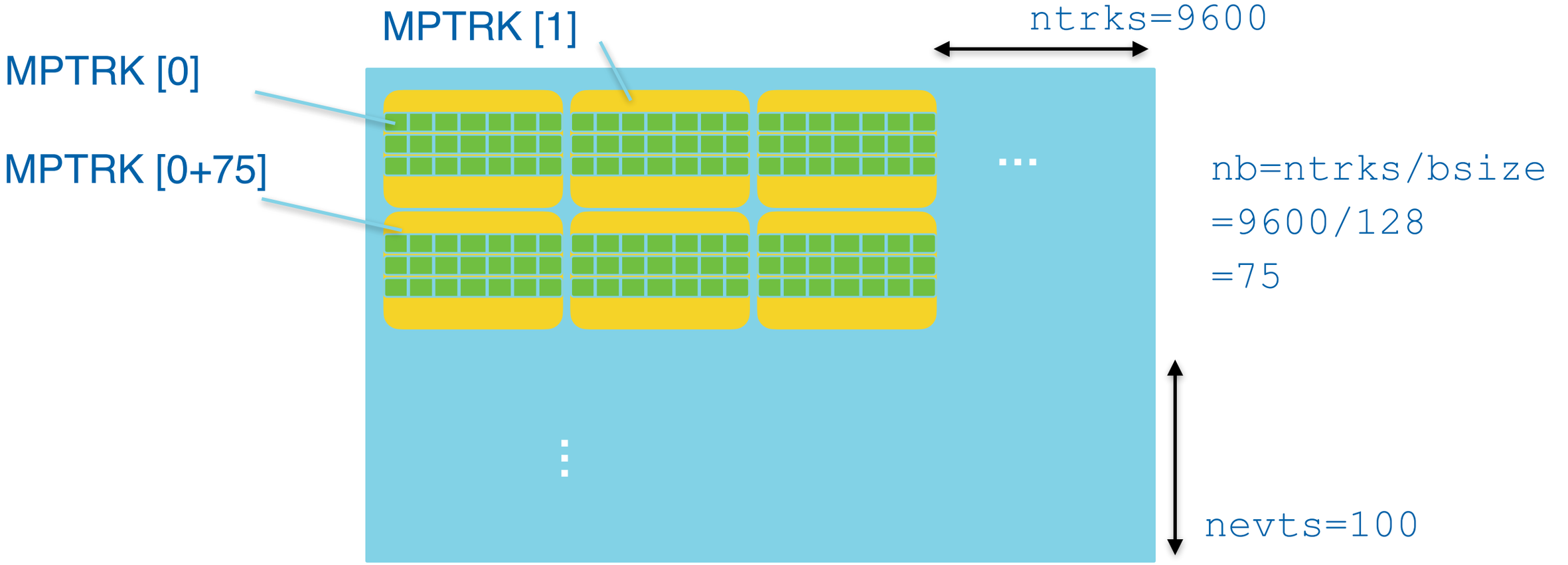
- Total work of `ntrks` x `nevts`, tracks in an event are grouped into batch of `bsize`
- Batch of tracks are put into the same data structure (MPTRK)
  - Neighboring tracks are consecutive in memory
- Similar arrangement for hits, with each track having `nlayers` of hits
- All tracks are arranged in an array of MPTRK



Fermilab

Indexing scheme in p2z

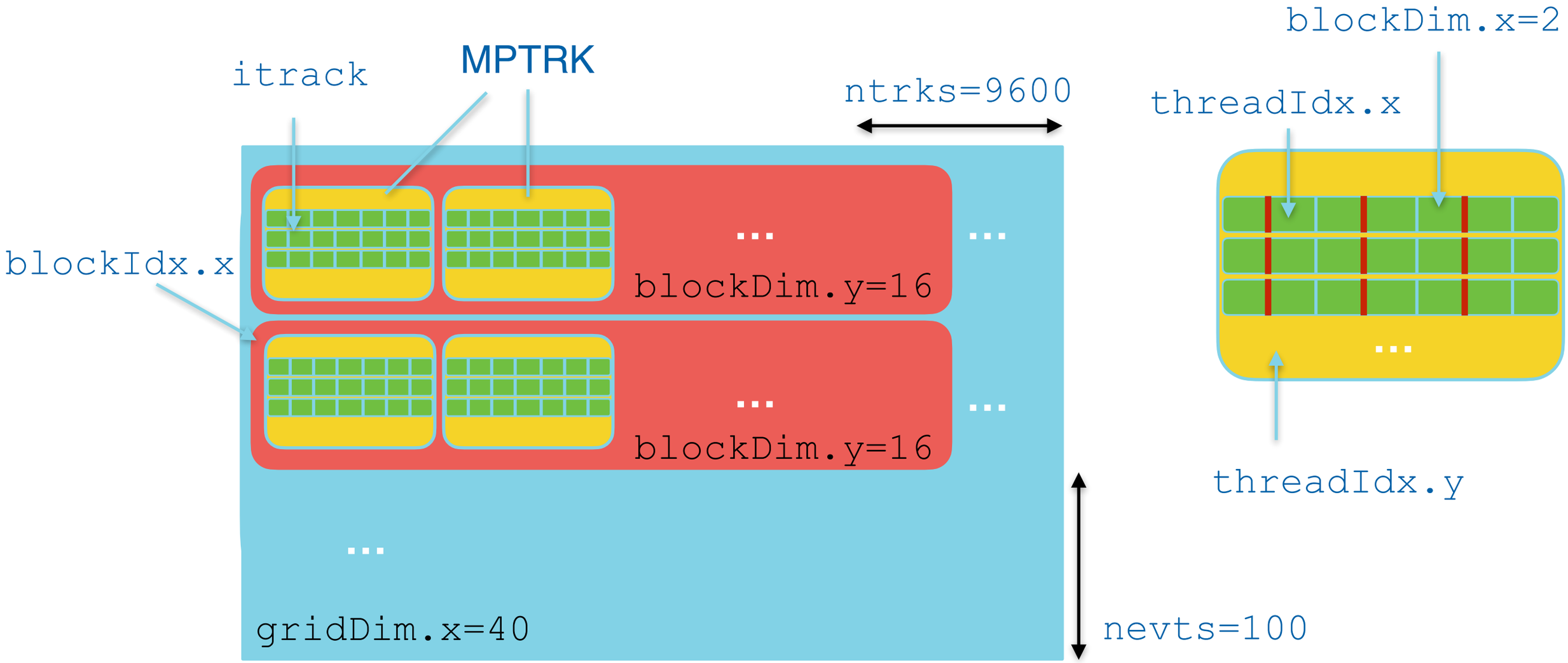
- Total work of `ntrks` x `nevts`, tracks in an event are grouped into batch of `bsize`
- Batch of tracks are put into the same data structure (MPTRK)
  - Neighboring tracks are consecutive in memory
- Similar arrangement for hits, with each track having `nlayers` of hits
- All tracks are arranged in an array of MPTRK
  - Batches in the same event are consecutive in the array index



Fermilab

Launching the kernels (p2z arrangement)

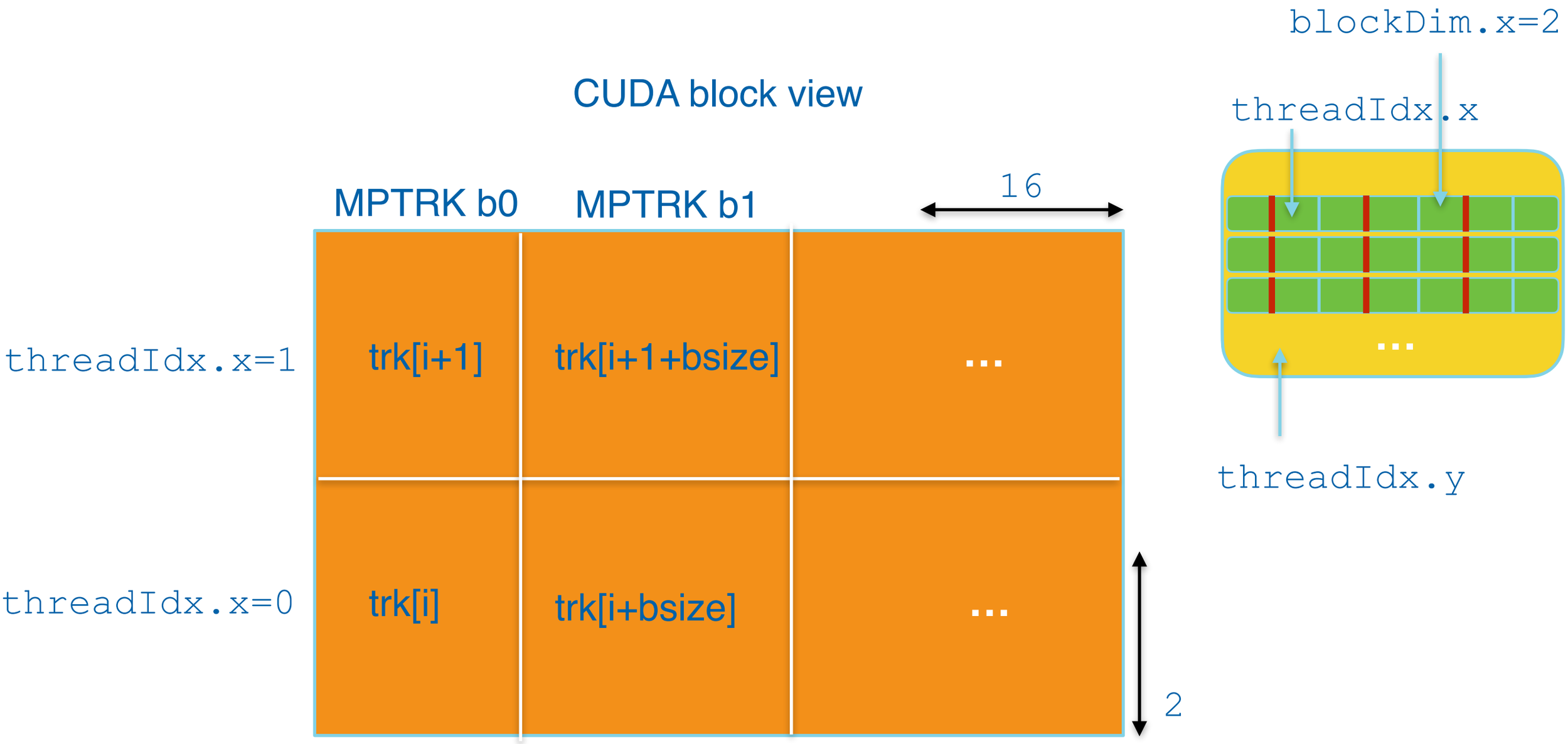
- 1D blocks, 2D threads
- MPTRK (containing `bsize` of tracks, `size` x `nlayers` of hits)
  - thread.y runs over number of batches (in groups of 16)
    thread.x runs over tracks inside a batch (in groups of 2)
- One block per event, 40 blocks per grid



Fermilab

Launching the kernels (p2z arrangement)

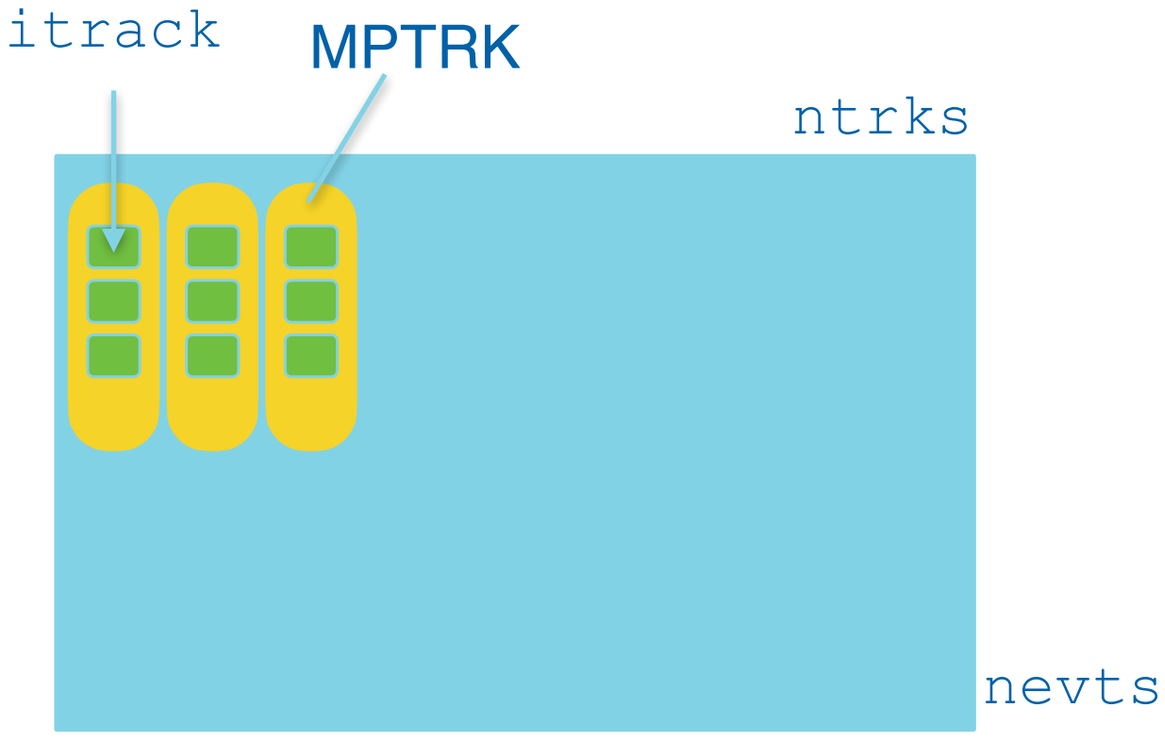
- Each CUDA block accesses 16 MPTRK at a time
  - Each warp of 32 threads therefore loads 16 MPTRK objects, before any computation of the warp
  - Need to keep these 16 MPTRK objects available in the block



Fermilab

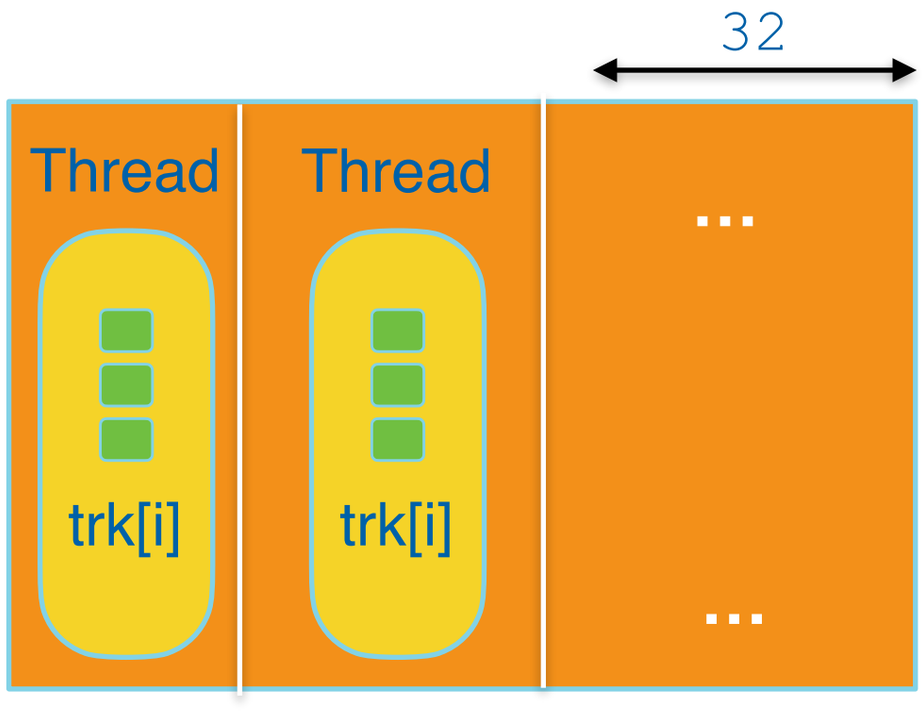
p2r CUDA v1

- p2r CUDA v1: Use bsize=1 and 1-D CUDA block to launch kernels
- Global thread index directly corresponds to each track
- Kernel launched with
  - Blocks per grid = (nevts * nTrks) / threads_per_block
  - Threads_per_block = const



p2r CUDA v1

CUDA block view

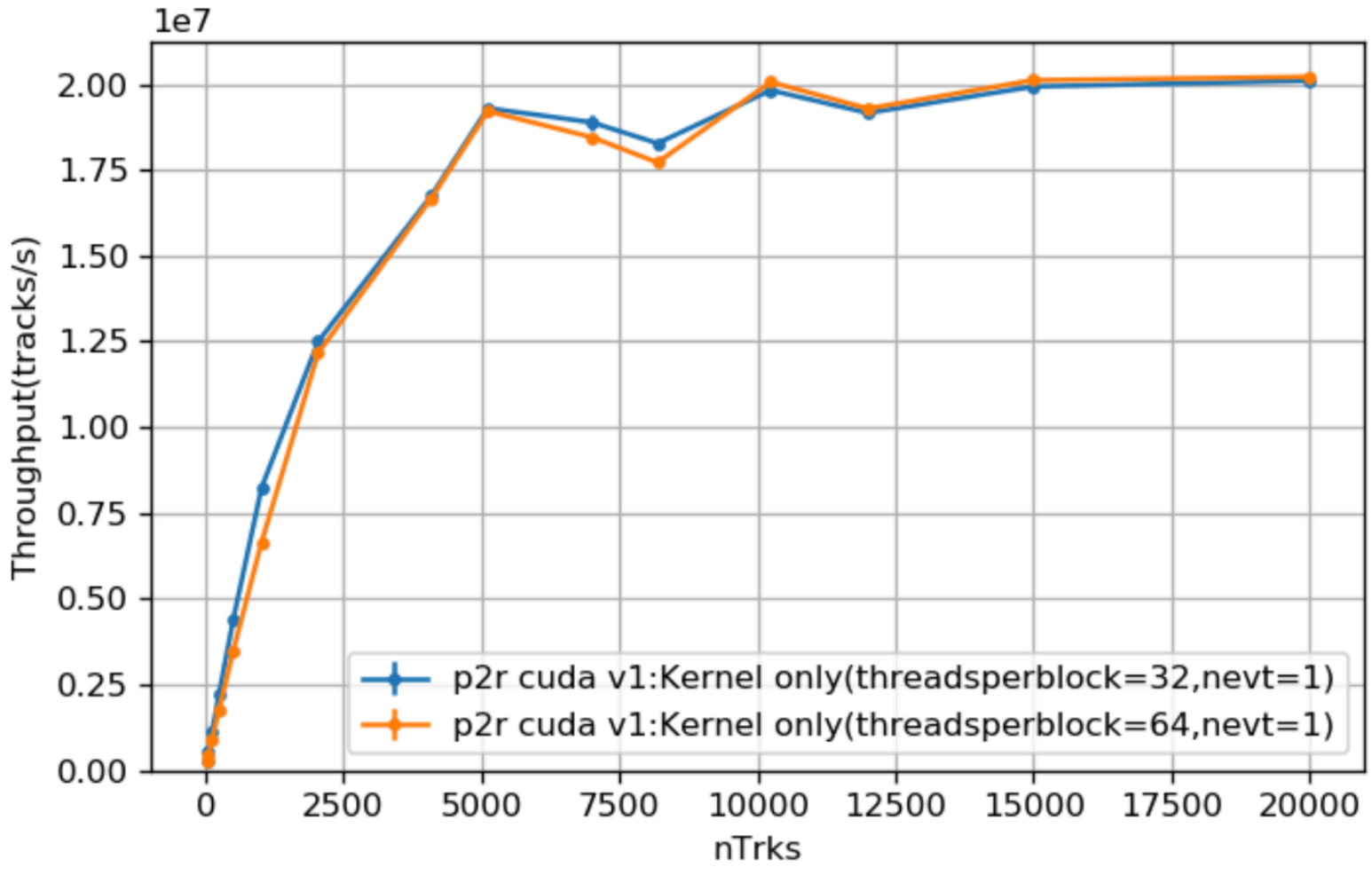


p2r CUDA v1

Fermilab

p2r CUDA v1 result

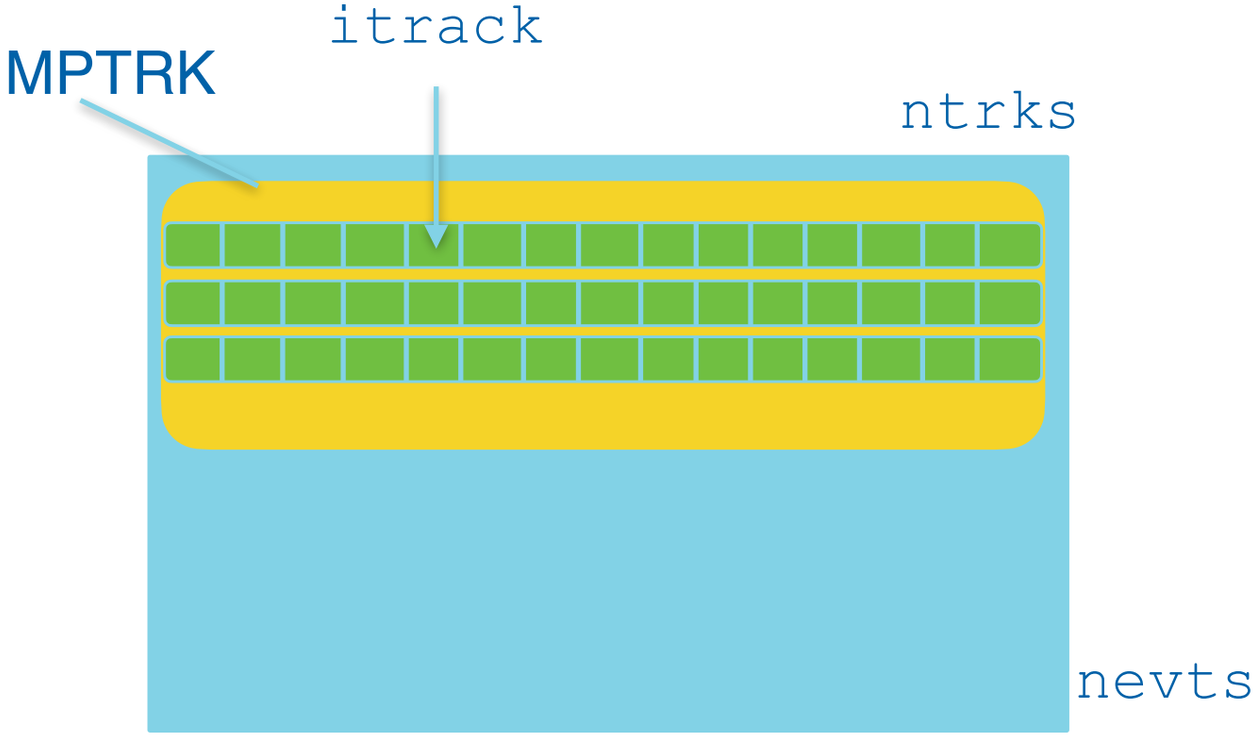
- Scan with nevts=1, increase nTrks until full GPU is saturated
- V100 has 80SM, 64 32-bit cores/ SM = 5120 32-bit cores
- Throughput saturates when full GPU is utilized
- Similar throughput with 64 threads per block

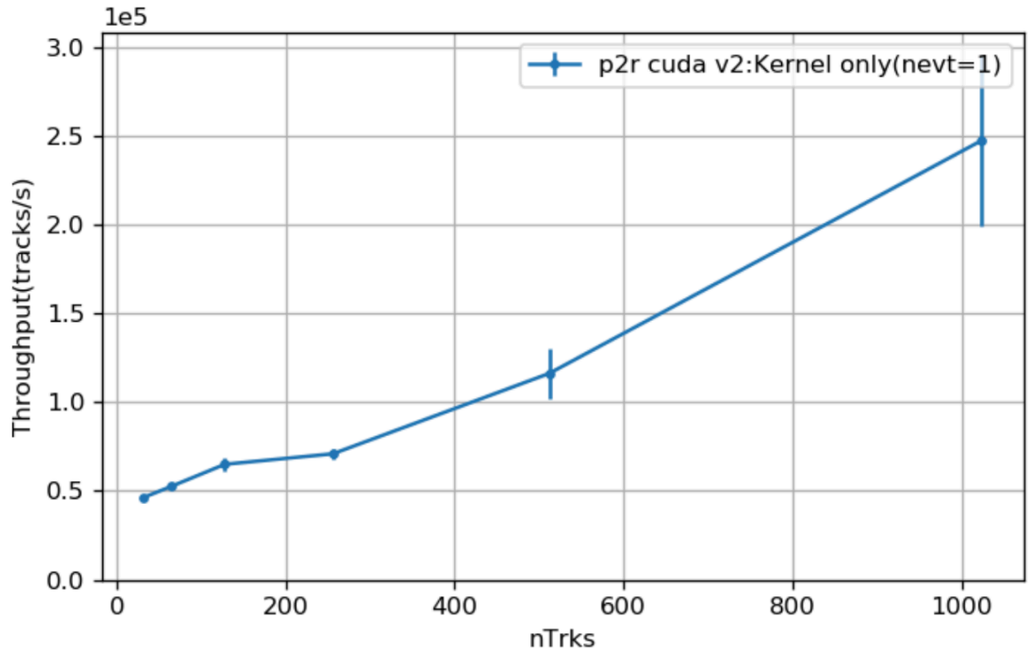


Fermilab

p2r CUDA v2

- p2r CUDA v2: use bsize=ntrks, to have maximum coalescence in memory (under development)
- 1D blocks of kernels launched with
  - `blocks_per_grid = nevts`,
  - `threads_per_block = ntrks`
- Throughput ~O(10^5), 2 orders of magnitude smaller than v1



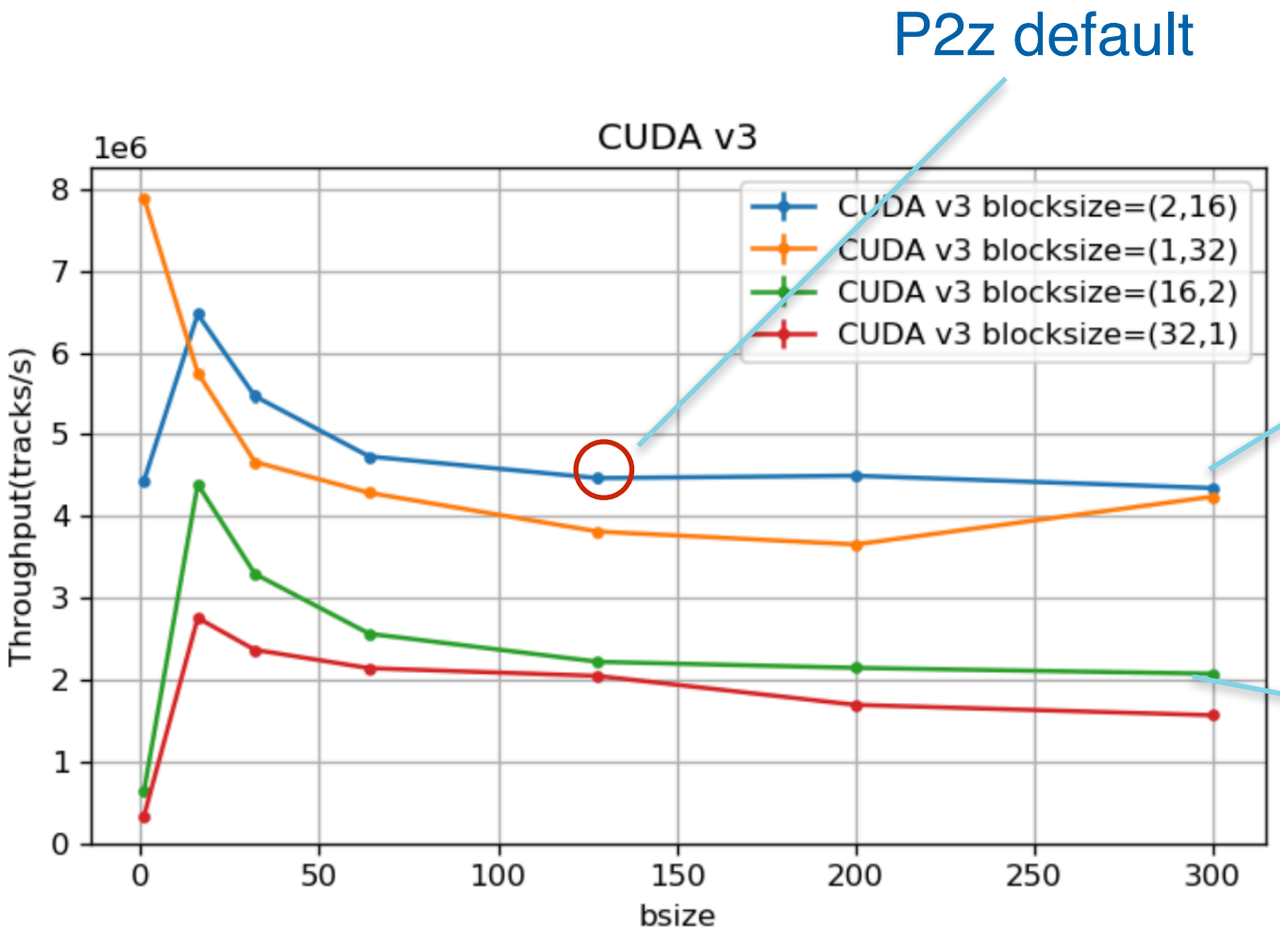


p2r CUDA v2

Fermilab

p2r CUDA v3

- p2r CUDA v3:
  Follow p2z indexing scheme and kernel launch scheme (V2) with single stream
- Try to scan the throughput dependence on block sizes dimension
  - Best throughput with blocksize=(1,32) and bsize=1 (i.e. 32 MPTRK, 1 track each)
  - Throughput ~O(10^6), 1 order of magnitude smaller than v1



P2z default

16 MPTRK,
2 tracks each
in a CUDA block

2 MPTRK,
16 tracks each
in a CUDA block

Fermilab

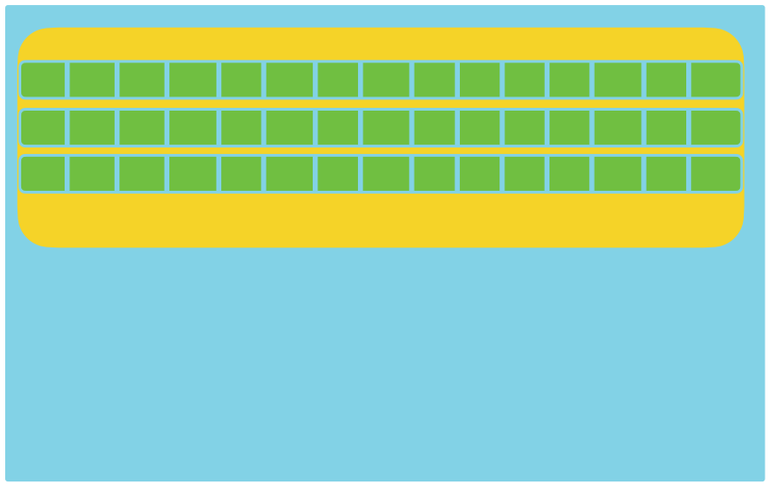
Summary

- Implemented 3 versions of p2r with different indexing/launching schemes
- Very different throughput measured: v1>v3>v2

Throughput: ~(2×10^7)

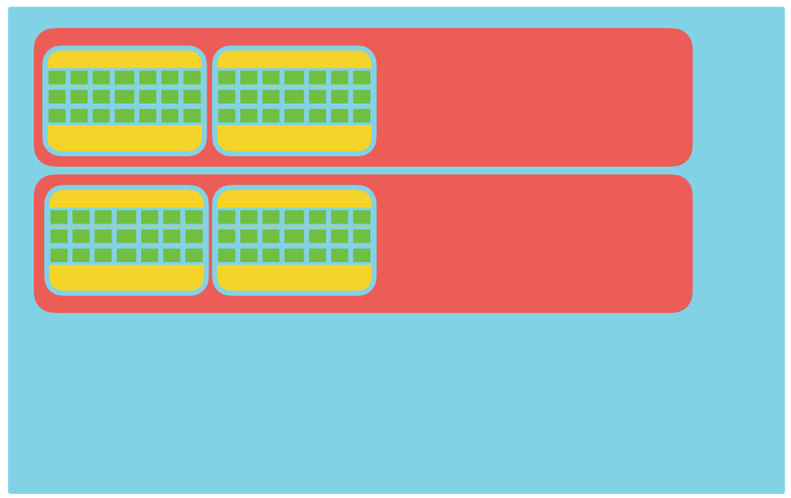
p2r CUDA v1

Throughput: ~(2×10^5)



p2r CUDA v2

Throughput: ~(5×10^6)



p2r CUDA v3