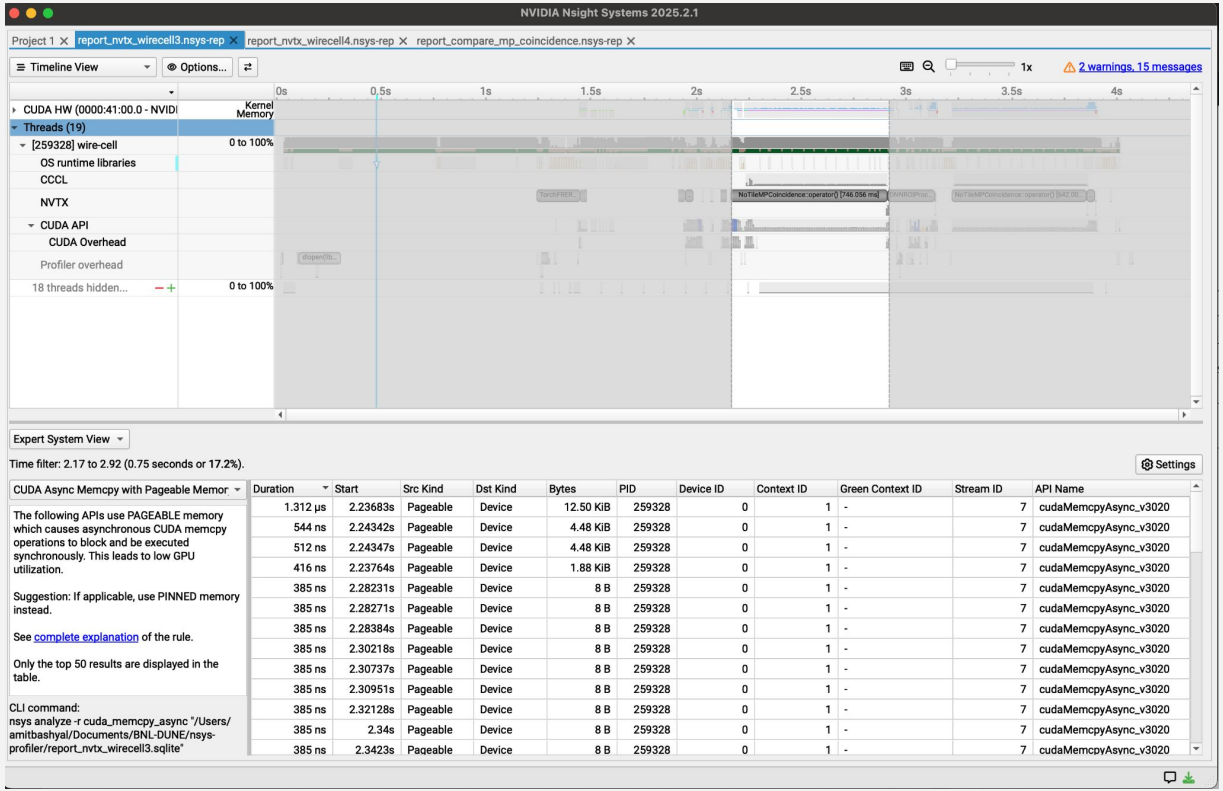
DUNE SPNG Updates
October 17, 2025

NvTools Extension and NVIDIA Insight Tools

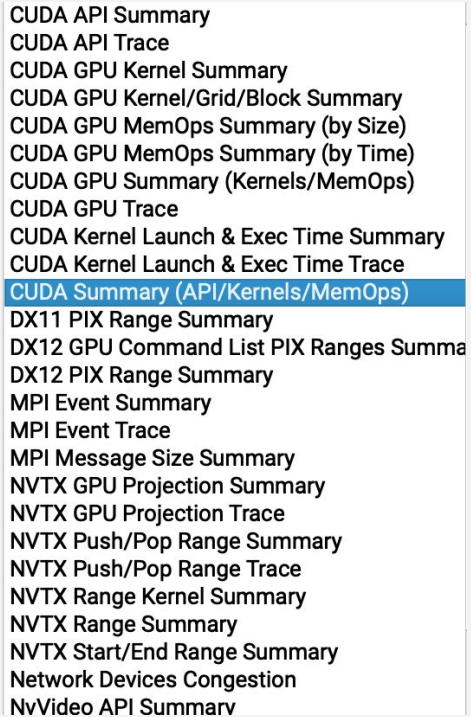
- NVIDIA insight tools provide profiling and debugging tools
  - GPU and CPU usage
  - Memory usage (CPU and GPU)
  - GPU Trace (Timeline of throughput, thread, duration of processes etc)
  - CPU Trace
  - Third Party API (MPI)
- Nvidia Tool extensions allows to mark part of the code to get meaningful insights and time ranges
- Works with NVIDIA tools (nvprof, nsys etc)
- Profiling Outputs in proprietary nsys-rep format, sqlite, hdf5, arrow, parquet
  - Nsys-rep format → Can use NVIDIA Nsight System (that supports UI)
    - Download in your laptop, copy the profile output and analyze
  - Sqlite format → Write your own python sqlite reader code
    - Flexibility to compare different output variables

Use NVIDIA NSight Systems to analyze the profile report



Content of the Profile Output

- Profile reports on extensive (API and I/O) categories



Report is extensive but probably we could be interested in a small set….

CUDA API Summary

Time	Total Time	Num Calls	Avg	Med	Min	Max	StdDev	Name
71.8%	217.432 ms	57343	3.791 µs	2.293 µs	1.832 µs	10.166 ms	97.834 µs	cudaLaunchKernel
15.8%	47.873 ms	13286	3.603 µs	3.716 µs	861 ns	20.471 µs	435 ns	cudaStreamSynchronize
11.5%	34.927 ms	13520	2.583 µs	2.453 µs	1.612 µs	67.472 µs	1.137 µs	cudaMemcpyAsync
0.4%	1.171 ms	1	1.171 ms	1.171 ms	1.171 ms	1.171 ms	0 ns	cudaFree
0.3%	1.058 ms	2	529.107 µs	529.107 µs	12.509 µs	1.046 ms	730.580 µs	cudaHostAlloc
0.1%	291.711 µs	5	58.342 µs	58.969 µs	2.264 µs	98.819 µs	38.317 µs	cudaMalloc
0.1%	163.377 µs	838	195 ns	181 ns	70 ns	811 ns	97 ns	cuGetProcAddress_v2
0.0%	9.205 µs	18	511 ns	251 ns	170 ns	2.224 µs	645 ns	cudaEventCreateWithFlags
0.0%	5.228 µs	4	1.307 µs	1.146 µs	832 ns	2.103 µs	604 ns	cuInit
0.0%	1.583 µs	2	791 ns	791 ns	431 ns	1.152 µs	509 ns	cudaStreamIsCapturing_v10000
0.0%	531 ns	2	265 ns	265 ns	140 ns	391 ns	177 ns	cudaGetDriverEntryPoint_v11030
0.0%	281 ns	2	140 ns	140 ns	70 ns	211 ns	99 ns	cuModuleGetLoadingMode

Summary of the Cuda APIs being used

CUDA GPU Kernel Summary

Time	Total Time	Instances	Avg	Med	Min	Max	StdDev	Name
15.9%	14.850 ms	1052	14.115 µs	14.112 µs	14.079 µs	14.176 µs	17 ns	void at_cuda_detail::cub::DeviceRadixSortSingleTileKernel<at_cuda_detail::cub::DeviceRa
13.9%	13.003 ms	6403	2.030 µs	2.080 µs	1.664 µs	3.136 µs	231 ns	void at::native::index_elementwise_kernel<(int)128, (int)4, void at::native::gpu_index_kern
8.6%	8.064 ms	6000	1.344 µs	1.311 µs	895 ns	1.792 µs	268 ns	void at_cuda_detail::cub::DeviceReduceSingleTileKernel<at_cuda_detail::cub::DeviceRedu
8.5%	7.931 ms	3000	2.643 µs	2.432 µs	2.368 µs	3.264 µs	292 ns	void at::native::elementwise_kernel<(int)128, (int)4, void at::native::gpu_kernel_impl<void
7.6%	7.147 ms	5106	1.399 µs	1.280 µs	1.215 µs	1.824 µs	218 ns	void at_cuda_detail::cub::DeviceSelectSweepKernel<at_cuda_detail::cub::detail::device_s
7.2%	6.690 ms	7500	891 ns	896 ns	863 ns	929 ns	18 ns	void at_cuda_detail::cub::DeviceCompactInitKernel<at_cuda_detail::cub::ScanTileState<in
5.7%	5.324 ms	5716	931 ns	928 ns	895 ns	1.057 µs	31 ns	void at::native::vectorized_elementwise_kernel<(int)2, at::native::FillFunctor<long>, std::ar
3.8%	3.528 ms	2665	1.323 µs	1.312 µs	1.215 µs	1.537 µs	55 ns	void at::native::elementwise_kernel<(int)128, (int)2, void at::native::gpu_kernel_impl_noca
3.5%	3.253 ms	2050	1.586 µs	1.696 µs	1.344 µs	2.816 µs	188 ns	void at::native::elementwise_kernel<(int)128, (int)4, void at::native::gpu_kernel_impl<at::na
2.9%	2.757 ms	2248	1.226 µs	1.216 µs	1.183 µs	1.441 µs	38 ns	void at::native::vectorized_elementwise_kernel<(int)2, at::native::CUDAFunctor_add<doub
2.8%	2.643 ms	2108	1.253 µs	1.248 µs	1.215 µs	1.409 µs	28 ns	void at::native::vectorized_elementwise_kernel<(int)2, at::native::BinaryFunctor<double, d
2.7%	2.522 ms	2108	1.196 µs	1.215 µs	1.056 µs	1.408 µs	92 ns	void at::native::vectorized_elementwise_kernel<(int)4, void at::native::compare_scalar_ke
2.6%	2.388 ms	1127	2.120 µs	2.112 µs	1.760 µs	2.209 µs	90 ns	void at::native::index_elementwise_kernel<(int)128, (int)4, void at::native::gpu_index_kern

CUDA Operations happening on the backend via torch

CUDA GPU Trace

Start	Duration	CorrId	GrdX	GrdY	GrdZ	BlkX	BlkY	BlkZ	Reg/Trd	StcSMem	DymSMem	Bytes	Throughput	SrcMemKd	DstMemK
3.04351s	352 ns	26612	-	-	-	-	-	-	-	-	-	80 B	216.74 MiB/s	Pageable	Device
3.04355s	1.664 µs	26625	1	1	1	128	1	1	38	0 B	0 B	-	-	-	-
3.04362s	1.440 µs	26642	1	1	1	128	1	1	16	0 B	0 B	-	-	-	-
3.05726s	1.600 µs	26655	1	1	1	128	1	1	16	0 B	0 B	-	-	-	-
3.05738s	1.280 µs	26668	1	1	1	128	1	1	16	0 B	0 B	-	-	-	-
3.05743s	1.280 µs	26678	1	1	1	128	1	1	40	0 B	0 B	-	-	-	-
3.06678s	1.504 µs	26688	1	1	1	128	1	1	38	0 B	0 B	-	-	-	-
3.06687s	1.984 µs	26698	1	1	1	128	1	1	27	0 B	0 B	-	-	-	-
3.07296s	1.440 µs	26714	1	1	1	128	1	1	16	0 B	0 B	-	-	-	-
3.0767s	1.696 µs	26728	256	2	1	512	1	1	16	0 B	0 B	-	-	-	-
3.07675s	1.472 µs	26738	1	1	1	128	1	1	16	0 B	0 B	-	-	-	-
3.07676s	1.344 µs	26748	1	1	1	128	1	1	16	0 B	0 B	-	-	-	-
3.07679s	1.888 µs	26759	1	2	1	128	1	1	26	0 B	0 B	-	-	-	-

Running NVIDIA Profiling Tool in Wire-cell SPNG

- SPNG uses torch libraries and APIs (Run in GPU, support CPU)
- The backend is NVIDIA
- So the profiler spits out profile report in terms of Core CUDA Calls

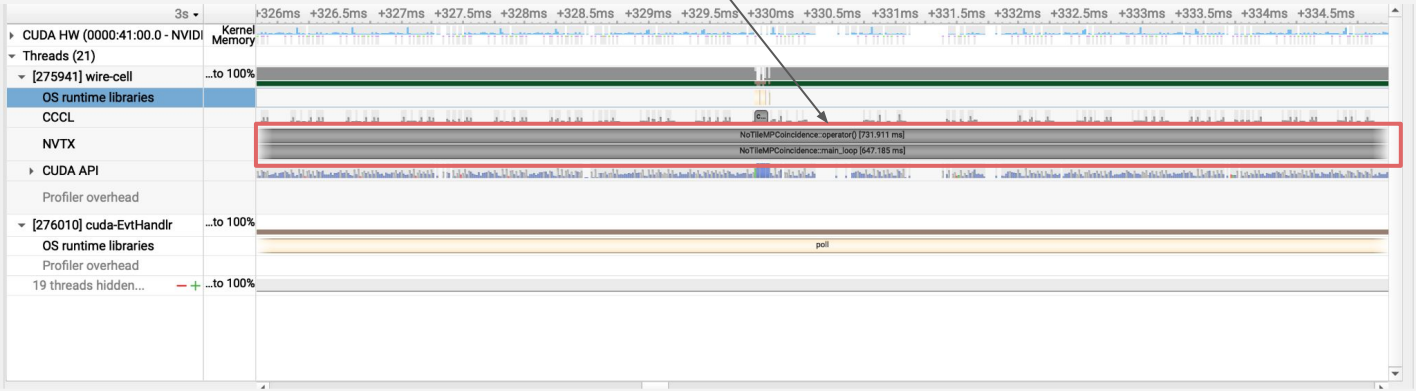
Duration (ns)	Start (ns)	PID	TID	API Name
5,995,345	3,988,078,688	275,941	275,941	cudaStreamSynchronize_v3020
5,983,467	4,718,549,610	275,941	275,941	cudaStreamSynchronize_v3020
5,975,265	3,978,597,109	275,941	275,941	cudaStreamSynchronize_v3020
5,966,472	4,709,060,730	275,941	275,941	cudaStreamSynchronize_v3020
5,175,169	4,728,859,837	275,941	275,941	cudaStreamSynchronize_v3020
3,698,413	4,701,803,544	275,941	275,941	cudaStreamSynchronize_v3020
3,492,032	3,971,549,839	275,941	275,941	cudaStreamSynchronize_v3020
1,644,178	2,112,931,089	275,941	275,941	cudaStreamSynchronize_v3020
1,120,038	3,036,051,392	275,941	275,941	cudaStreamSynchronize_v3020
791,352	3,002,198,717	275,941	275,941	cudaStreamSynchronize_v3020
750,241	3,986,505,887	275,941	275,941	cudaStreamSynchronize_v3020
748,869	3,996,009,970	275,941	275,941	cudaStreamSynchronize_v3020
746,455	4,736,167,600	275,941	275,941	cudaStreamSynchronize_v3020
746,284	4,716,974,797	275,941	275,941	cudaStreamSynchronize_v3020
746,254	4,726,470,477	275,941	275,941	cudaStreamSynchronize_v3020
742,078	3,977,021,585	275,941	275,941	cudaStreamSynchronize_v3020
730,090	4,707,473,989	275,941	275,941	cudaStreamSynchronize_v3020
702,459	3,985,603,277	275,941	275,941	cudaStreamSynchronize_v3020
701,377	4,716,067,930	275,941	275,941	cudaStreamSynchronize_v3020

Not so useful.
Hard to figure where these operations are happening.
Need to get insight in SPNG API level

NVIDIA Tools Extension (NvToolsExt) library

- Annotate the code and get insights

```
bool WireCell::SPNG::NoTileMPCoincidence::operator()(const input_pointer& in, output_pointer& out) {
    NVTX_SCOPED_RANGE("NoTileMPCoincidence::operator()");
    out = nullptr;
    if (!in) {
        log->debug("EOS ");
        return true;
    }
    log->debug("Running NoTileMPCoincidence");
```



NvTool to Annotate to understand insight in each code sections

Stats System View

Time filter: 3.04 to 3.78 (0.73 seconds or 14.2%).

- CUDA API Summary
- CUDA API Trace
- CUDA GPU Kernel Summary
- CUDA GPU Kernel/Grid/Block Summary
- CUDA GPU MemOps Summary (by Size)
- CUDA GPU MemOps Summary (by Time)
- CUDA GPU Summary (Kernels/MemOps)
- CUDA GPU Trace
- CUDA Kernel Launch & Exec Time Summary
- CUDA Kernel Launch & Exec Time Trace
- CUDA Summary (API/Kernels/MemOps)
- DX11 PIX Range Summary
- DX12 GPU Command List PIX Ranges Summa
- DX12 PIX Range Summary
- MPI Event Summary
- MPI Event Trace

Time	Total Time	Num Calls	Avg	Med	Min	Max	StdDev	Name
52.3%	1.000 s	2	500.085 ms	500.085 ms	500.077 ms	500.093 ms	11.379 µs	pthread_cond_timedwait
47.1%	901.385 ms	9	100.154 ms	100.147 ms	100.118 ms	100.238 ms	33.370 µs	poll
0.5%	8.864 ms	74	119.783 µs	122.344 µs	2.314 µs	735.779 µs	107.000 µs	ioctl
0.0%	920.068 µs	30	30.668 µs	28.593 µs	6.940 µs	71.498 µs	15.269 µs	pthread_rwlock_wrlock
0.0%	381.274 µs	22	17.330 µs	17.251 µs	3.315 µs	22.925 µs	3.631 µs	pthread_rwlock_rdlock
0.0%	37.907 µs	2	18.953 µs	18.953 µs	11.708 µs	26.199 µs	10.246 µs	pthread_mutex_lock
0.0%	28.883 µs	2	14.441 µs	14.441 µs	3.726 µs	25.157 µs	15.154 µs	mmap
0.0%	9.745 µs	2	4.872 µs	4.872 µs	1.022 µs	8.723 µs	5.445 µs	fwrite
0.0%	8.052 µs	2	4.026 µs	4.026 µs	3.465 µs	4.587 µs	793 ns	fflush
0.0%	6.580 µs	1	6.580 µs	6.580 µs	6.580 µs	6.580 µs	0 ns	munmap

Time	Total Time	Num Calls	Avg	Med	Min	Max	StdDev	Name
72.0%	215.882 ms	57343	3.764 µs	2.224 µs	1.822 µs	13.534 ms	104.819 µs	cudaLaunchKernel
16.1%	48.226 ms	13286	3.629 µs	3.756 µs	1.732 µs	16.555 µs	393 ns	cudaStreamSynchronize
11.1%	33.296 ms	13520	2.462 µs	2.334 µs	1.562 µs	53.581 µs	874 ns	cudaMemcpyAsync
0.4%	1.178 ms	1	1.178 ms	1.178 ms	1.178 ms	1.178 ms	0 ns	cudaFree
0.3%	924.414 µs	2	462.207 µs	462.207 µs	14.182 µs	910.232 µs	633.603 µs	cudaHostAlloc
0.1%	297.306 µs	5	59.461 µs	65.308 µs	2.193 µs	97.837 µs	36.290 µs	cudaMalloc
0.1%	153.659 µs	838	183 ns	170 ns	50 ns	651 ns	95 ns	cuGetProcAddress_v2
0.0%	8.964 µs	18	498 ns	260 ns	150 ns	2.033 µs	609 ns	cudaEventCreateWithFlags
0.0%	4.437 µs	4	1.109 µs	866 ns	741 ns	1.963 µs	572 ns	cuInit
0.0%	1.502 µs	2	751 ns	751 ns	631 ns	871 ns	169 ns	cudaStreamIsCapturing_v10000
0.0%	421 ns	2	210 ns	210 ns	131 ns	290 ns	112 ns	cudaGetDriverEntryPoint_v11030
0.0%	180 ns	2	90 ns	90 ns	70 ns	110 ns	28 ns	cuModuleGetLoadingMode

OS (CPU) Runt Time summary and CUDA API Summary for NoTileMPConcidence::Operator

Configuring SPNG for nvTools

- Current spng branch of wire-cell toolkit
- Configuration is similar to how you configure SPNG except
  - Add *–with-nvtx* flag
  - Add path to nvTools library in *–with-libtorch-lib*
  - Add path to nvTools header in *–with-libtorch-include*
- Example from mine (may not work in yours)

```
./wcb configure --prefix=$PREFIX --boost-mt --boost-libs=$PREFIX/lib --boost-include=$PREFIX/include
--with-jsonnet-libs=gojsonnet --with-cuda-lib=/usr/lib/x86_64-linux-gnu,$PREFIX/targets/x86_64-linux/lib
--with-cuda-include=$PREFIX/targets/x86_64-linux/include --with-libtorch=$TDIR
--with-libtorch-include=$TDIR/include,$TDIR/include/torch/csrc/api/include,$PREFIX/targets/x86_64-linux/include
--with-libtorch-lib=$TDIR/lib,$PREFIX/targets/x86_64-linux/lib --with-root=$PREFIX --with-nvtx
```

For the code part to be annotated

- Add the header file
  - *WireCellSpng/TorchnvTools.h*
- Annotate the code to be profiler

```
bool WireCell::SPNG::NoTileMPCoincidence::operator()(const input_pointer& in, output_pointer& ou
    NVTX_SCOPED_RANGE("NoTileMPCoincidence::operator()");
    out = nullptr;
    if (!in) {
        log->debug("EOS ");
        return true;
    }
    log->debug("Running NoTileMPCoincidence");
```

- NVTX_SCOPED_RANGE profiles the *operator* function

```
NVTX_RANGE_PUSH("NoTileMPCoincidence::gather_channels");
convert_wires_to_channels(output_tensor_active, m_plane_channels_to_wires[m
output_tensor_active = output_tensor_active.reshape({nbatch, nsamples, -1})
convert_wires_to_channels(output_tensor_inactive, m_plane_channels_to_wires
output_tensor_inactive = output_tensor_inactive.reshape({nbatch, nsamples,
NVTX_RANGE_POP();
```

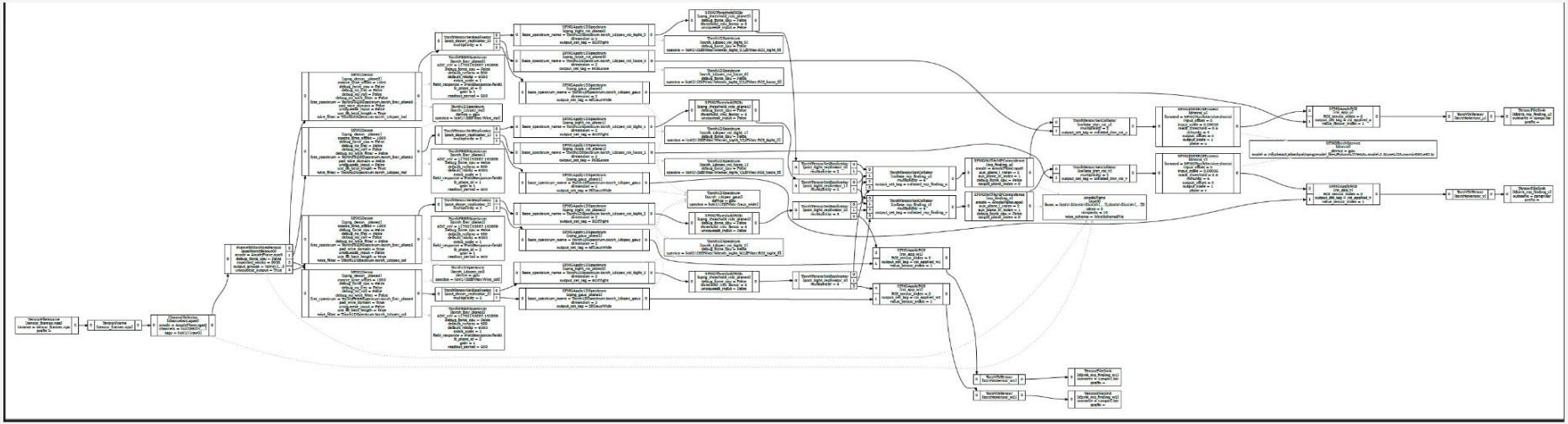
Annotate part of code using PUSH/POP combination
Can support nested annotation as well.

Profiling the SPNG

- We can attempt to profile the SPNG code in a generic way to get some general sense
- Target I/O, memory hot/cold areas from the information we gather from the generalized profiling
- Generalized profiling still needs annotation to know the ball park of where I/O, memory, hot/cold areas fall
- **SPNG Workflow in General**

Input waveform → Deconvolution → Filters → Initial ROIs + MP2/MP3 formation→ DNNROI → Output waveform with ROIs

Actual Data Flow Graph (To Run DNNROI)

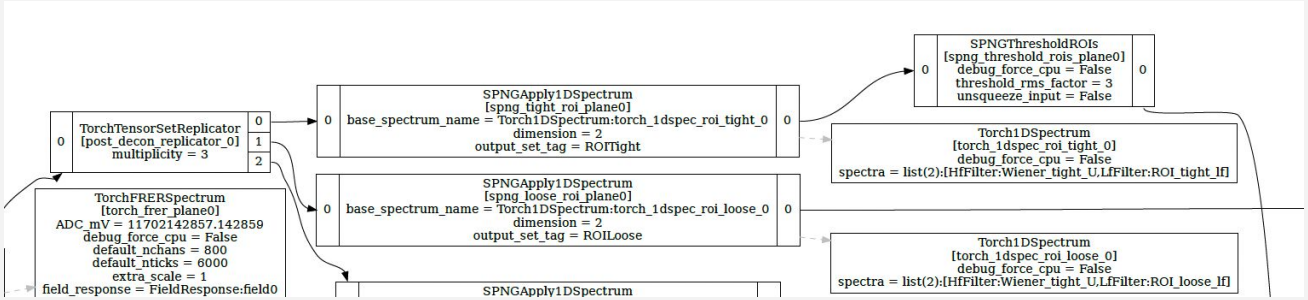


Tasks to produce ingredients to find ROIs in 1 plane:

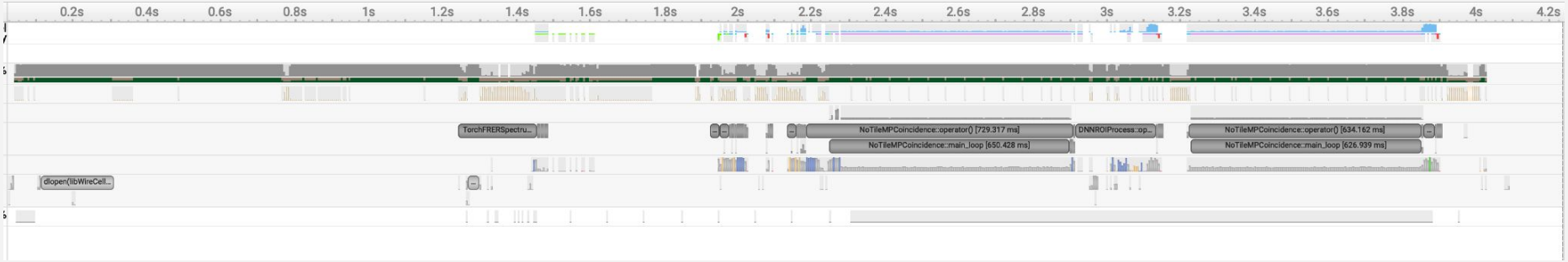
- Decon (for all Planes)
- Filters (For all planes, filters depending upon induction/collection type)
- MP Finding (Target Induction Plane, MP information in Other 2 planes)

Annotating the SPNG

- Annotate each FunctionNode nvtx tool
- Annotate any function or parts of the code that does heavy lifting (both in CPU or GPU)
- Exclude functions that do configurations
- Sometimes functions call other function. But we want to include that
  - For example DNNROI calls TorchService. Annotate both of them.
  - Nested Annotation is supported.

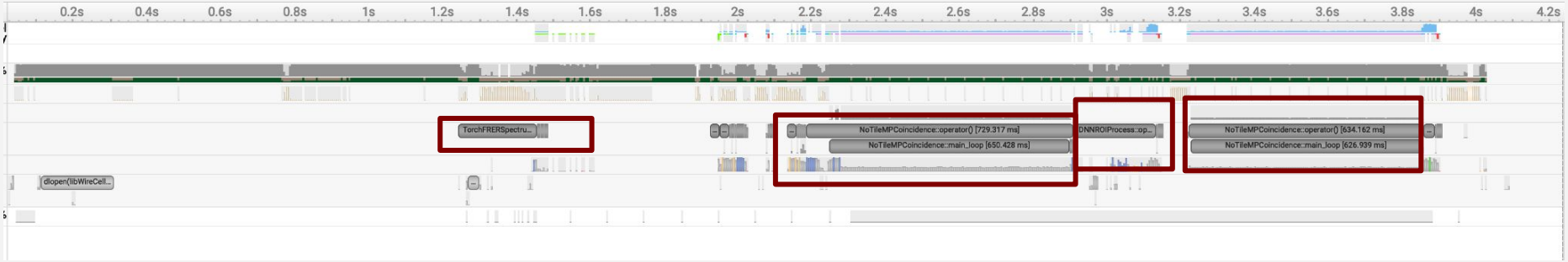


After Annotating with Nvtx Library



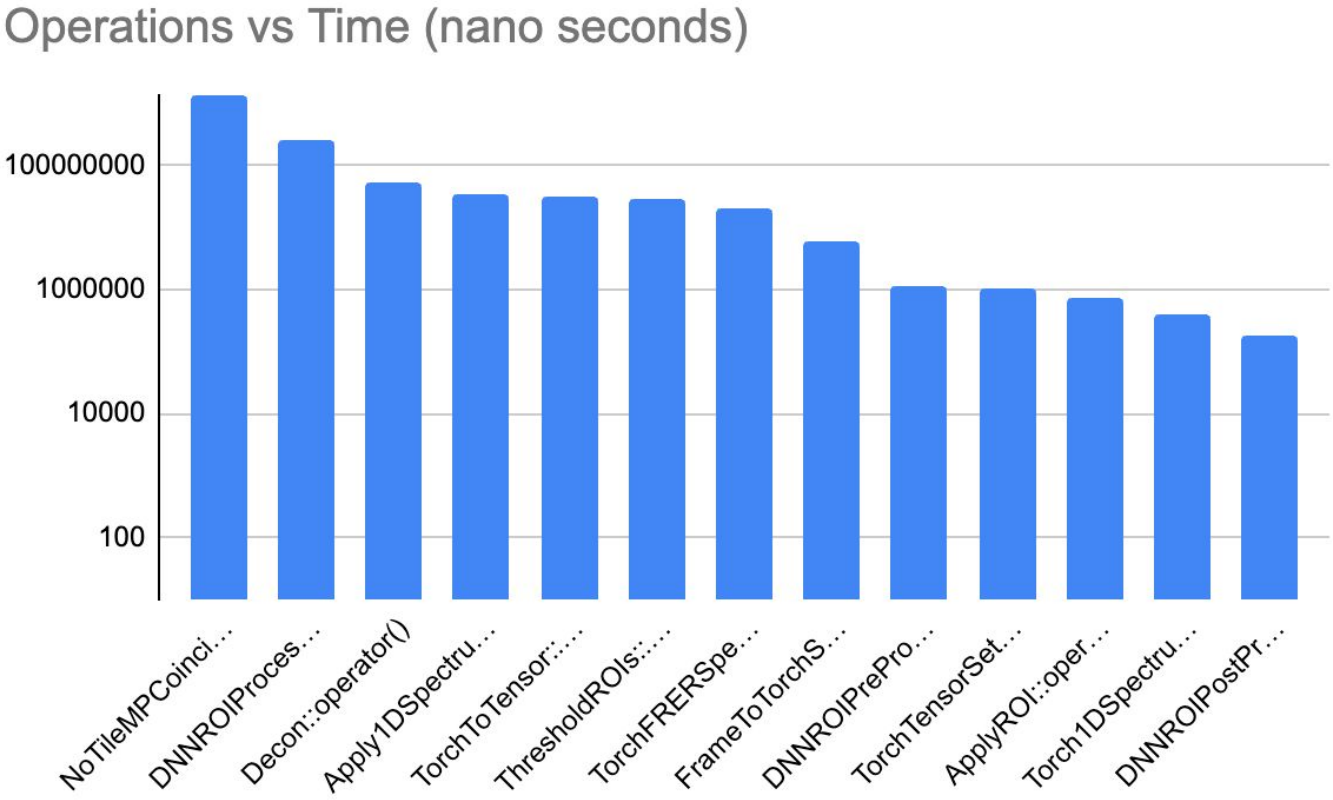
Runs of 4.2 seconds ~ Consistent with the wire-cell timer

After Annotating with Nvtx Library

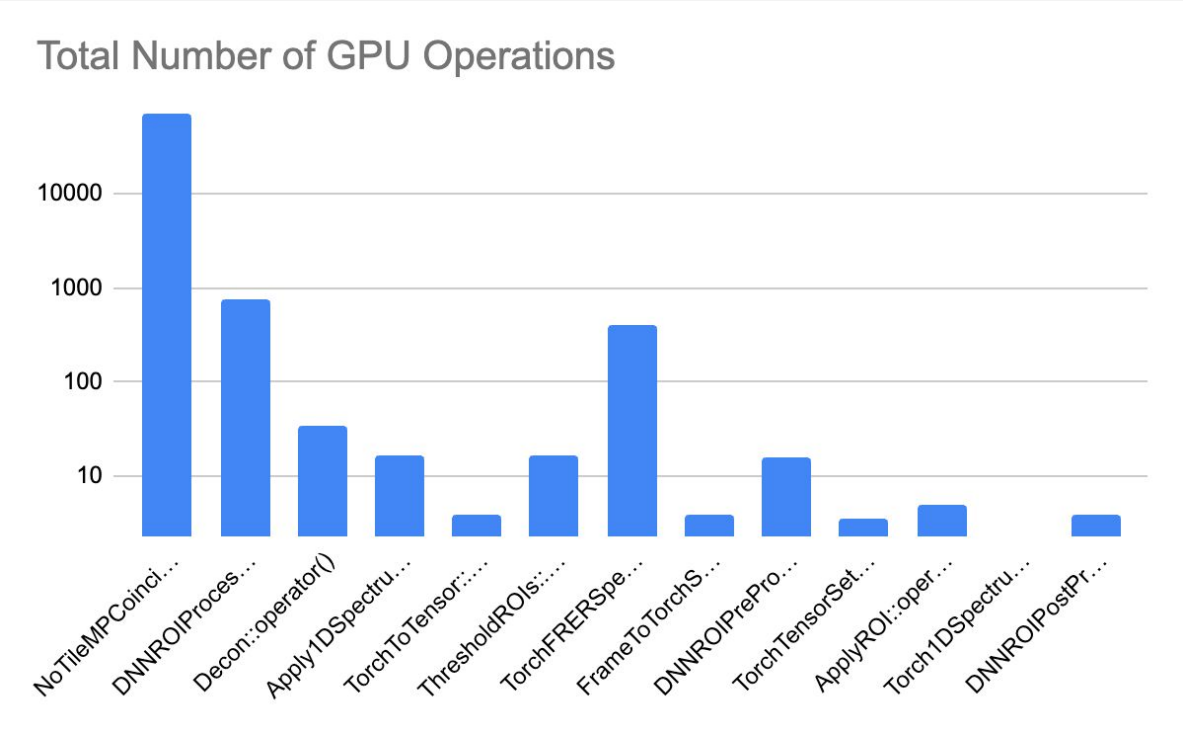


- Runs of 4.2 seconds ~ Consistent with the wire-cell timer
- SPNG spends most of its time on (descending order)
  - NoTileMPCoincidence
  - TorchFRERSpectrum
  - DNNROI

Time spent by Wire-cell in each Component



Average Number of GPU Operations



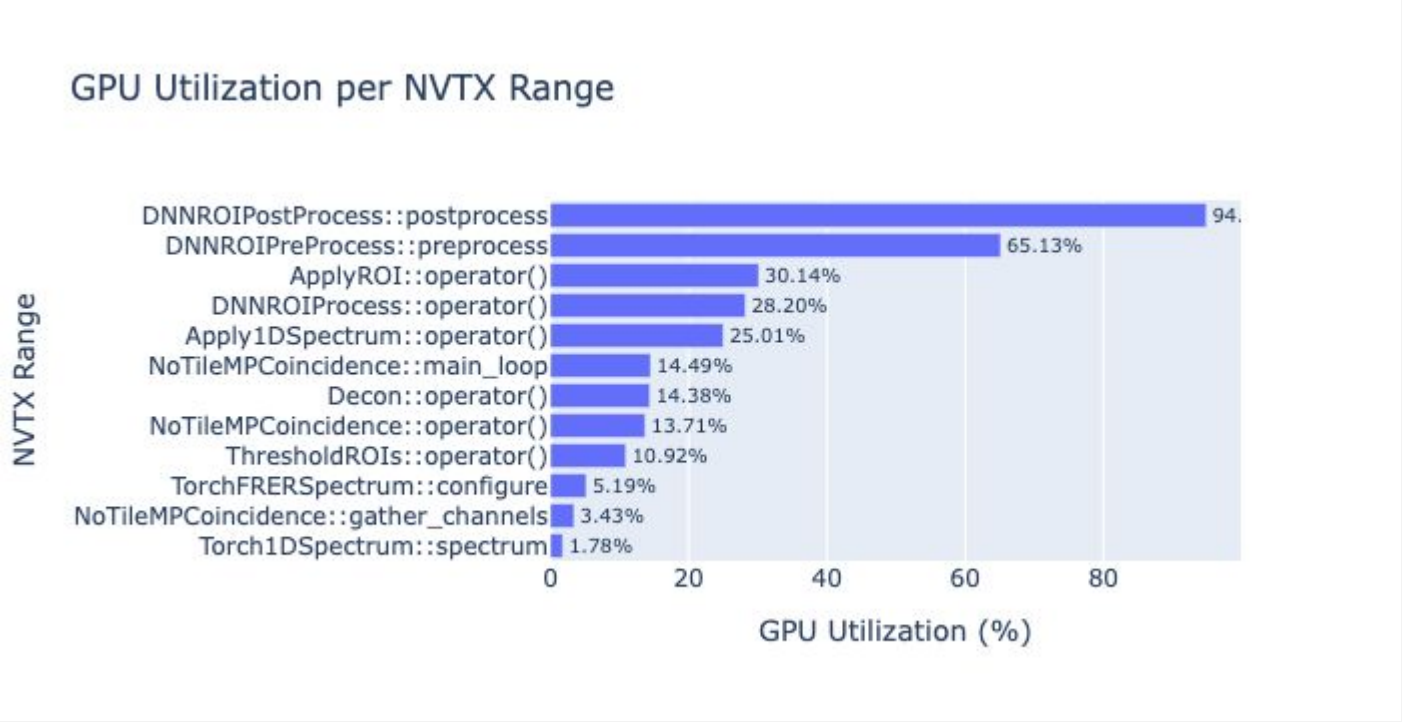
Includes both GPU operations and I/O

Very Highly Dominated by NoTileMPCoincidence
More on this later….

GPU Utilization

- Fraction of operations handled by the GPU (Total = GPU + CPU)
- Overall GPU Utilization is ~12%
- In those 12%:
  - 86% of the time, GPU does actual work (operations
  - 14% of the time it does I/O
-

GPU Utilization in Annotated parts of the code



Note that NoTileMPCoincidence dominates the SPNG GPU Operations.

NoTileMPCoincidence

- More on this next week
-