Profiling Deep Learning Networks

Poonam Chitale, David Zier NVIDIA

DEEP LEARNING OPTIMIZATION

Performance Analysis at System and DNN Level & Visualization

System Level Tuning

- System Tuning
  - Thread Synchronization, Multi GPU and node communication
  - Memory management & Kernel profiling
- Leveraging/Optimizing Hardware
- Input Pipeline Optimization
- Many others....

DNN Level Tuning

- Algorithm Techniques & Data Representations
- Pruning
- Calibration
- Quantization
- Many others....

Visualization

TYPICAL CHALLENGES

INPUT PIPELINE

Optimal data processing, batching, copying, managing pipeline

DNN ALGORITHMS

Maximum parallel computation,fast matrix operations, precision

SYSTEM TUNING

System wide tracing, thread synchronization, memory transfers Kernel tuning

CALIBRATION

Layer fusion, calibrating, optimized inference

DATA CENTER CLUSTER LEVEL

Maximum availability and utilization in Data Center

DL PROFILING NEEDS OF DIFFERENT PERSONAS

Researchers

Fast development of best performant models for research, challenge and domains

Data Scientists & Applied Researchers

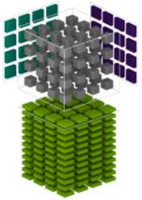
Reduce Training time, focus on data, develop and apply the best models for the applications

Sysadmins & DevOps

Optimized utilization and uptime, monitor GPU workloads, leverage hardware

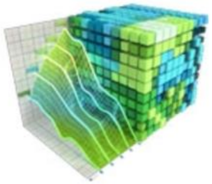
DL PROFILING: TOOLS & TECHNOLOGIES

Researchers



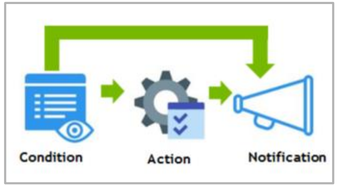
Use Advanced APIs
Developer Tools

Data Scientists & Applied Researchers



Use Tensorboard type of visualization
Developer Tools

Sysadmins & DevOps



Use Data Center Monitoring Tools

etc…

Tools: NSIGHT Tools, NVVP, NVPROF, DCGM, NVML
APIs: NVTX, CUPTI

INPUT DATA PIPELINE OPTIMIZATION

Highly dependent on application use cases

Training Data Preparation

Preprocessing and augmentation can become complex, learnings from a medical imaging segmentation use case:

- Cropping multiple batches from one single volume.
- Unzipping files and saving to local disk at first epoch.
- Storing foreground voxel coordinates to local disk space at first epoch.
- Caching etc…

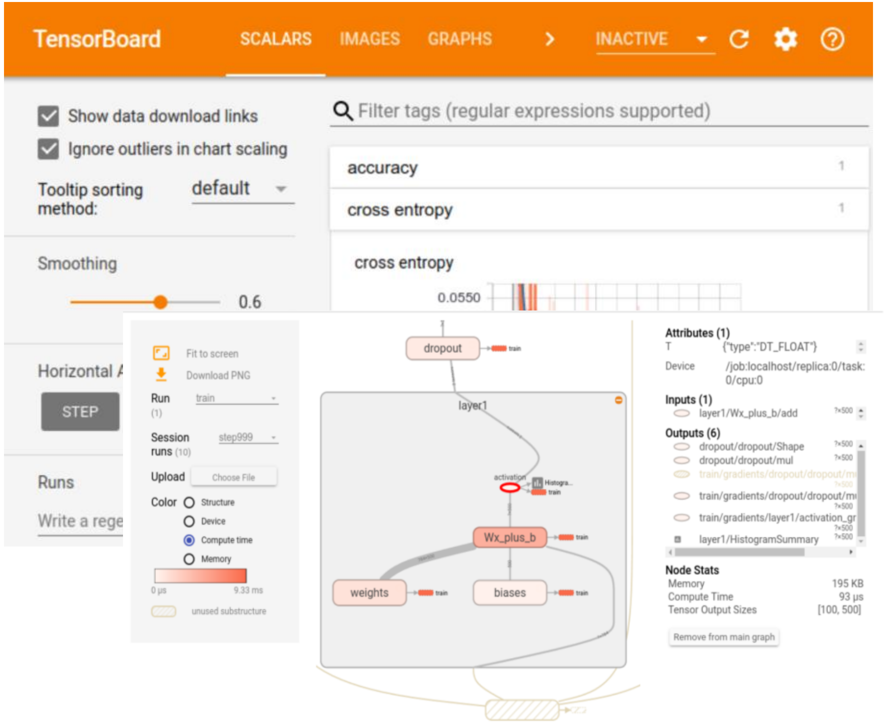
NVIDIA DALI: DAta loading LIbrary:

A GPU-accelerated data augmentation and image loading library for optimizing data pipelines of deep learning frameworks.

TENSORBOARD

Data Visualization

- Tensorboard is the most popular visualization tools used by data scientists and applied researchers using Tensorflow.
- Useful to understand network graph topology, training etc
- PyTorch users seem to use TensorboardX (also Visdom)
- MXBoard is a similar tool for mxnet



NVIDIA NSIGHT TOOLS

NSIGHT PRODUCT FAMILY

Standalone Performance Tools, IDE Plugins

Standalone Performance Tools

Nsight Systems System wide tracing, application algorithm tuning

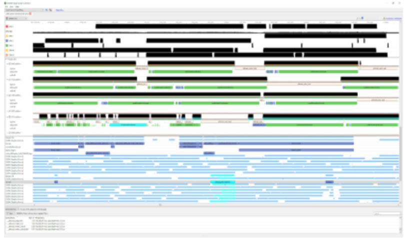
Nsight Compute Debug/Optimize specific CUDA kernels

Nsight Graphics Debug/Optimize specific graphics API and Shaders

IDE Plugins

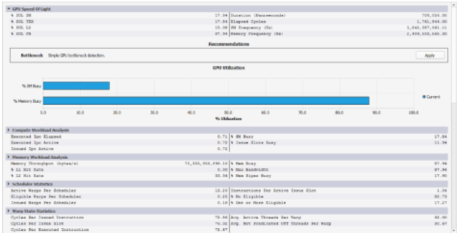
Nsight Visual Studio/Eclipse Edition editor, debugger, performance analysis

NSIGHT PRODUCT FAMILY



Nsight Systems

System-wide application algorithm tuning

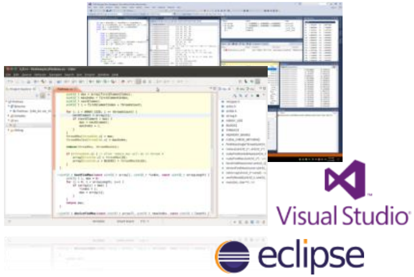


Nsight Compute

CUDA API Debugging & Kernel Profiling

Nsight Graphics

Graphics Debugging & Profiling



IDE Plugins

Nsight Eclipse Edition/Visual Studio (Editor, Debugger)

NSIGHT SYSTEMS

System-wide Performance Analysis

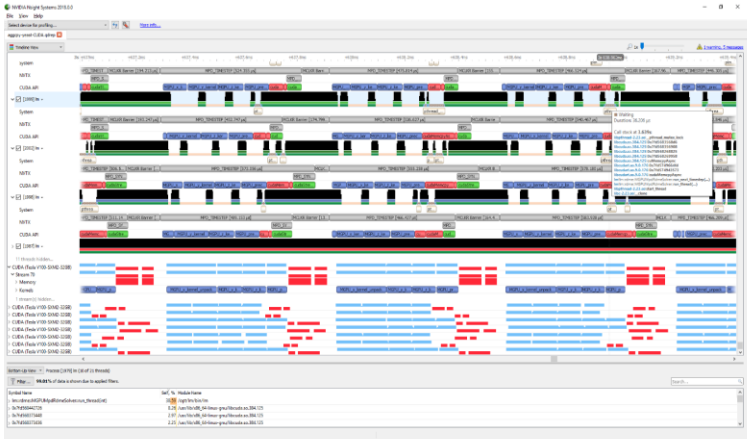
Observe Application Behavior: CPU threads, GPU traces, Memory Bandwidth and more

Locate Optimization Opportunities: CUDA & OpenGL APIs, Unified Memory transfers, User Annotations using NVTX

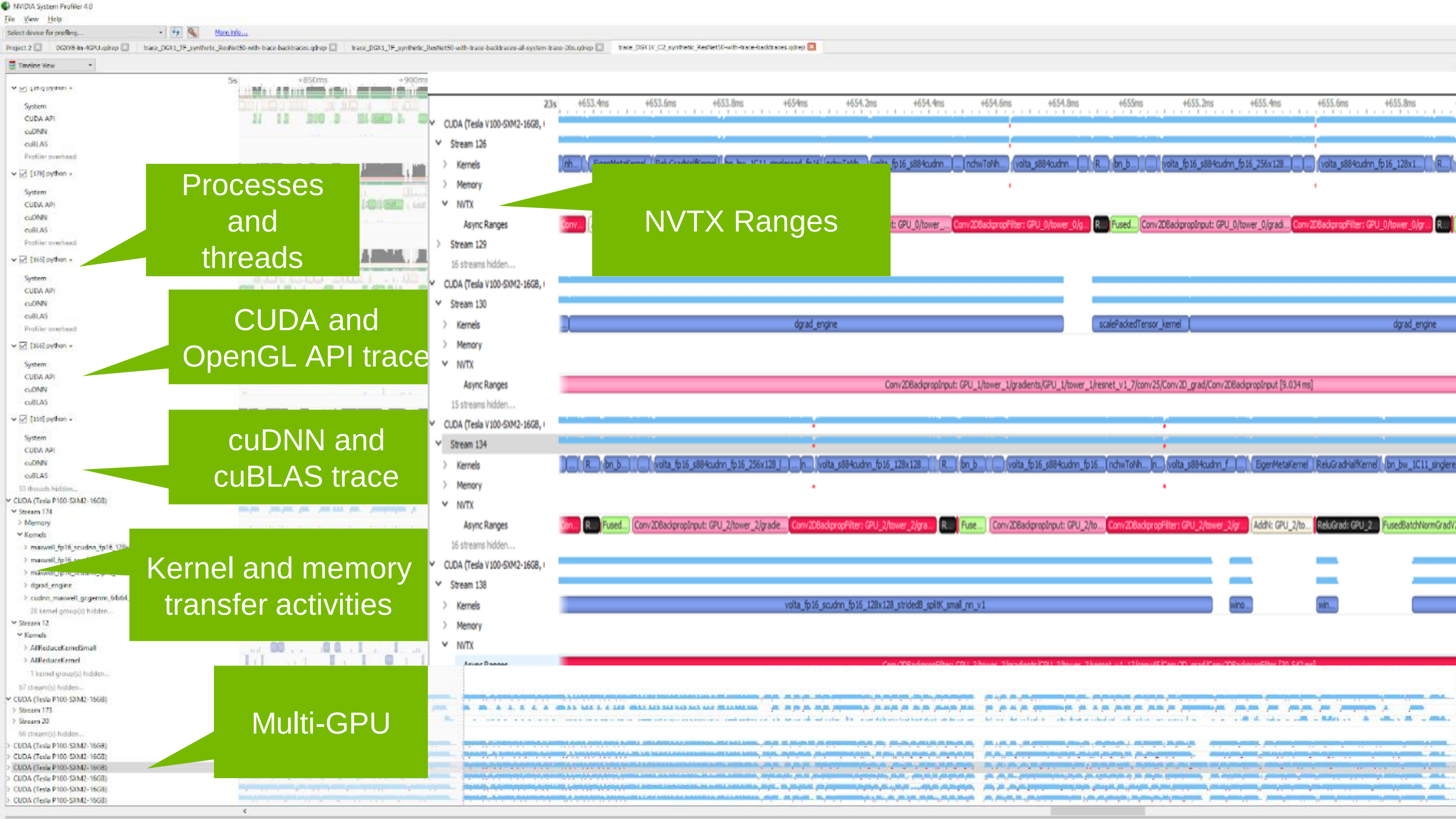
Ready for Big Data: Fast GUI capable of visualizing in excess of 10 million events on laptops, Container support, Minimum user privileges

NVIDIA® Nsight™

NVIDIA CUDA®



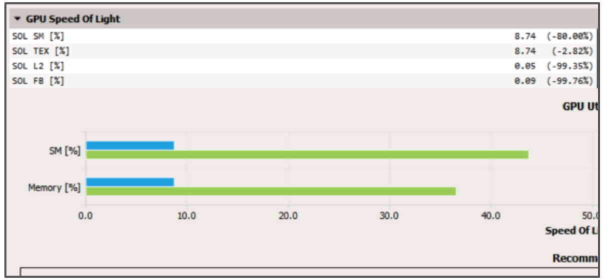
https://developer.nvidia.com/nsight-systems


NVIDIA System Profiler 4.0
Processes and threads
NVTX Ranges
CUDA and OpenGL API trace
cuDNN and cuBLAS trace
Kernel and memory transfer activities
Multi-GPU

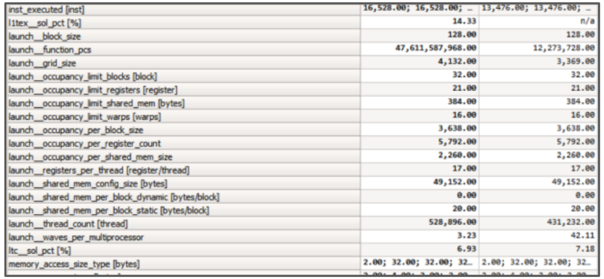
NVIDIA NSIGHT COMPUTE

Next Generation Kernel Profiler

- Interactive CUDA API debugging and kernel profiling
- Fast Data Collection
- Improved Workflow and Fully Customizable (Baselining, Programmable UI/Rules)
- Command Line, Standalone, IDE Integration
- Platform Support
  - OS: Linux (x86, ARM), Windows
  - GPUs: Pascal, Volta, Turing

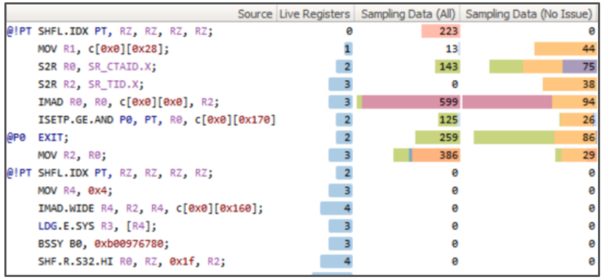


Kernel Profile Comparisons with Baseline



inst_executed [inst]	16,528.00; 16,528.00; …	13,476.00; 13,476.00; …
l1tex__sol_pct [%]	14.33	n/a
launch__block_size	128.00	128.00
launch__function_pcs	47,611,587,968.00	12,273,728.00
launch__grid_size	4,132.00	3,369.00
launch__occupancy_limit_blocks [block]	32.00	32.00
launch__occupancy_limit_registers [register]	21.00	21.00
launch__occupancy_limit_shared_mem [bytes]	384.00	384.00
launch__occupancy_limit_warps [warps]	16.00	16.00
launch__occupancy_per_block_size	3,638.00	3,638.00
launch__occupancy_per_register_count	5,792.00	5,792.00
launch__occupancy_per_shared_mem_size	2,260.00	2,260.00
launch__registers_per_thread [register/thread]	17.00	17.00
launch__shared_mem_config_size [bytes]	49,152.00	49,152.00
launch__shared_mem_per_block_dynamic [bytes/block]	0.00	0.00
launch__shared_mem_per_block_static [bytes/block]	20.00	20.00
launch__thread_count [thread]	528,896.00	431,232.00
launch__waves_per_multiprocessor	3.23	42.11
ltc__sol_pct [%]	6.93	7.18
memory_access_size_type [bytes]	2.00; 32.00; 32.00; 32…	2.00; 32.00; 32.00; 32…

Metric Data



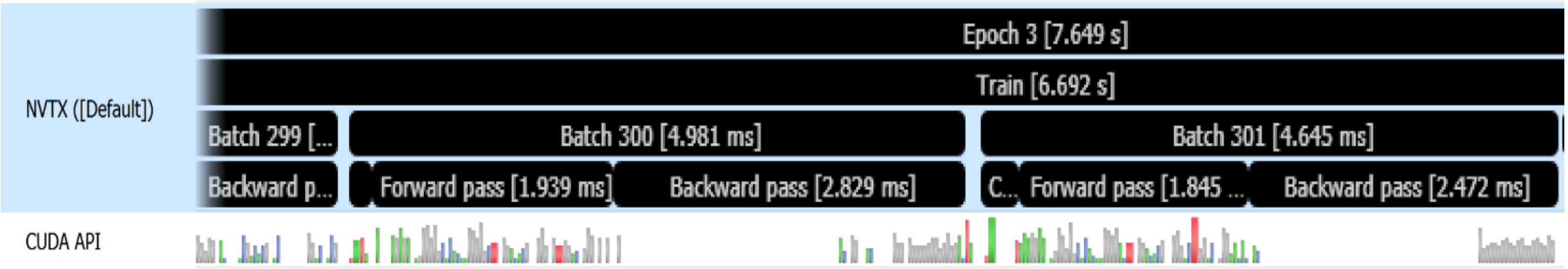
Source	Live Registers	Sampling Data (All)	Sampling Data (No Issue)
@!PT SHFL.IDX PT, RZ, RZ, RZ, RZ;	0	223	0
MOV R1, c[0x0][0x28];	1	13	44
S2R R0, SR_CTAID.X;	2	143	75
S2R R2, SR_TID.X;	3	0	38
IMAD R0, R0, c[0x0][0x0], R2;	3	599	94
ISETP.GE.AND P0, PT, R0, c[0x0][0x170]	2	125	26
@P0 EXIT;	2	259	86
MOV R2, R0;	3	386	29
@!PT SHFL.IDX PT, RZ, RZ, RZ, RZ;	2	0	0
MOV R4, 0x4;	3	0	0
IMAD.WIDE R4, R2, R4, c[0x0][0x160];	4	0	0
LDG.E.SYS R3, [R4];	3	0	0
BSSY B0, 0xb00976780;	3	0	0
SHF.R.S32.HI R0, RZ, 0x1f, R2;	4	0	0

Source Correlation

APIs & Libraries : NVTX and CUPTI

NVIDIA TOOLS EXTENSION LIBRARY (NVTX)

- NVTX is a platform agnostic, tools agnostic API
- Allows developers to annotate(mark) source code, events, code ranges etc
- NVIDIA optimized Tensorflow, PyTorch, MXnet have NVTX annotations built in!
- Enables a better more effective user experience with Nsight Tools, NVVP, NVPROF



https://docs.nvidia.com/cuda/profiler-users-guide/index.html#nvtx

PREVIEW: TENSORFLOW WITH NVTX ANNOTATION

Coming soon ….

- Library developed specifically for annotating Tensorflow to help visualize network better in Nsight Systems

- Workflow:
  - Import nvtx_tf library
  - Annotate python code
  - Run tensorflow
  - Get data through a profiler such as Nsight Systems

```python
import tensorflow as tf

import nvtx.plugins.tf as nvtx_tf

# Option 1: use decorators
@nvtx_tf.layers.trace(message="Dense Block", domain_name="Forward",
                      grad_domain_name="Gradient")
def dense_block(net):
    net = tf.layers.dense(net, 1, activation=tf.nn.relu, name='dense_1')
    net = tf.layers.dense(net, 64, activation=tf.nn.relu, name='dense_2')
    net = tf.layers.dense(net, 128, activation=tf.nn.relu, name='dense_3')
    net = tf.layers.dense(net, 256, activation=tf.nn.relu, name='dense_4')
    net = tf.layers.dense(net, 512, activation=tf.nn.relu, name='dense_5')
    return net


def build_model(inputs):

    x = dense_block(inputs)

    # Option 2: wrap parts of your graph with nvtx layers
    x, nvtx_context = nvtx_tf.layers.start(x, message="logits-softmax")
    x = tf.layers.dense(x, 512, activation=None, name='logits')
    x = tf.nn.softmax(x, name='probs')
    x = nvtx_tf.layers.end(x, nvtx_context)

    return x
```

Coming soon as a library

PREVIEW: PyTorch WITH NVTX ANNOTATION

Coming soon ....

Library for effectively using NVTX marker for PyTorch

- Custom NVTX marker as a python dictionary with module name, function name, arguments (tensor shapes & type, scalar type & value).

Workflow:

  - Import library
  - Annotate python code
  - Run with profiler

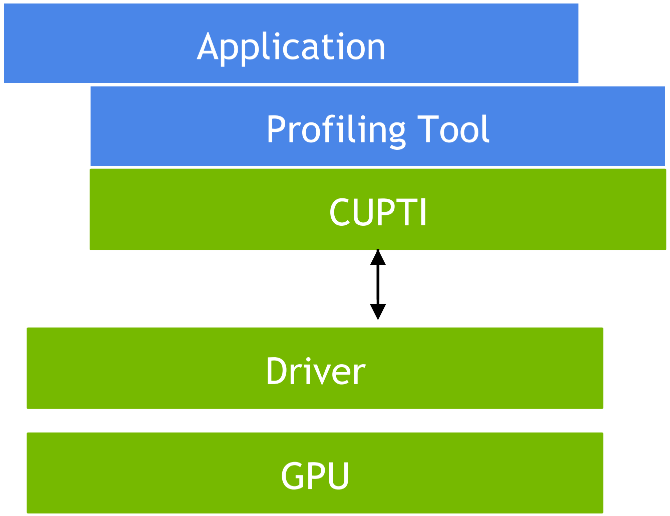
```
import torch.cuda.profiler as profiler
import nvtx_py
nvtx_py.nvtx.init()
with
torch.autograd.profiler.emit_nvtx():
  for epoch in range(100):
  for iteration in range(100):
  ….
```

CUDA PROFILING TOOLS INTERFACE (CUPTI)

Build your own GPU performance tuning tools

- C APIs to enable creation of profiling and tracing tools that target CUDA applications
- Supports multiple APIs - CUDA API trace, GPU activity trace, GPU performance counters and metrics, PC sampling, Profiling (Unified Memory, OpenACC)
- Available as a dynamic library on all CUDA supported platforms

https://docs.nvidia.com/cupti/Cupti/index.html



Tensor Cores for Deep Learning

ALGORITHM OPTIMIZATION

Mixed Precision implementation using Tensor Cores on Turing and Volta GPUs

Tensor Cores

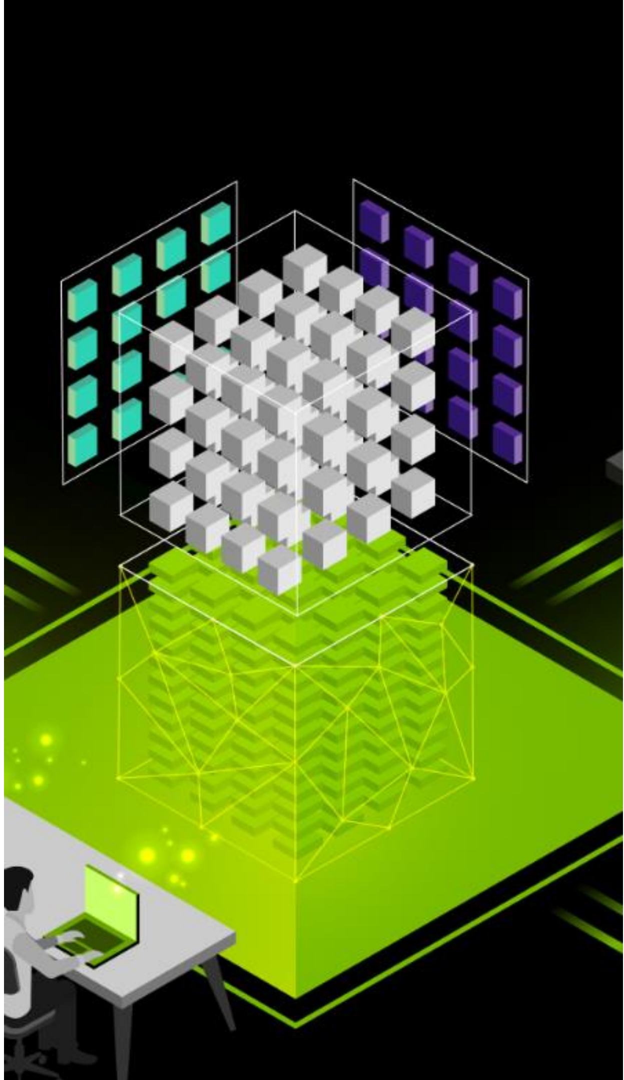
- A revolutionary technology that accelerates AI performance by enabling efficient mixed-precision implementation
- Accelerate large matrix multiply and accumulate operations in a single operation

Mixed Precision Technique
Combined use of different numerical precisions in a computational method; focus is on FP16 and FP32 combination.

Benefits

- Decreases the required amount of memory enabling training of larger models or training with larger mini-batches

Shortens the training or inference time by lowering the required resources by using lower-precision arithmetic



PREVIEW: NVIDIA DEEP LEARNING PROFILER

Deep Learning Profiler

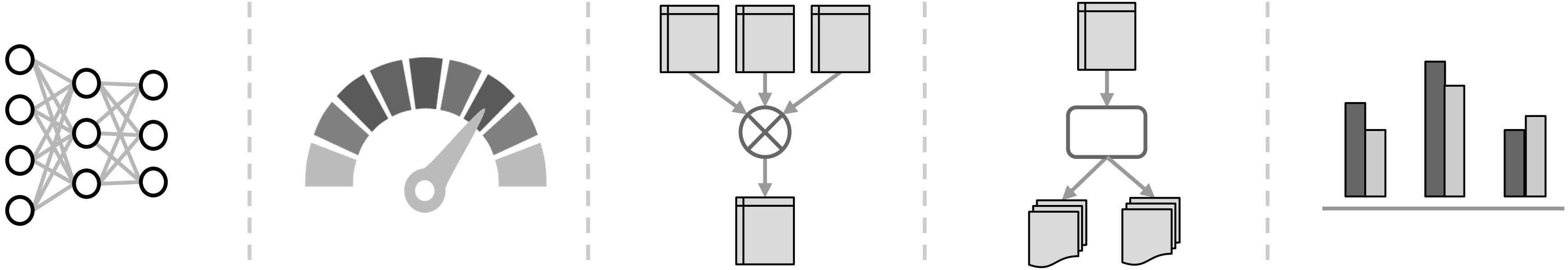
Core Purpose

Who: A data scientist/deep learning researcher
What: Able to

- Easily profile a DNN
- Understand GPU usage in terms of the model
- Present results in familiar tools, such as TensorBoard
- Leverage existing NVIDIA tools

Deep Learning Profiler Workflow

INPUT → PROFILE → CORRELATE → OUTPUT → ANALYZE



INPUT	PROFILE	CORRELATE	OUTPUT	ANALYZE
Graphdef file generate in Tensorflow	Use NSight tools to gather kernel and timing profile data	Correlate profile data with Tensorflow model	Generate TensorBoard event files and detailed reports	Analyze in TensorBoard or other 3rd party tools

Architecture

Automates workflow

Nsight Systems

- Gather timeline information
- Determines Tensor Core usage from name of kernels

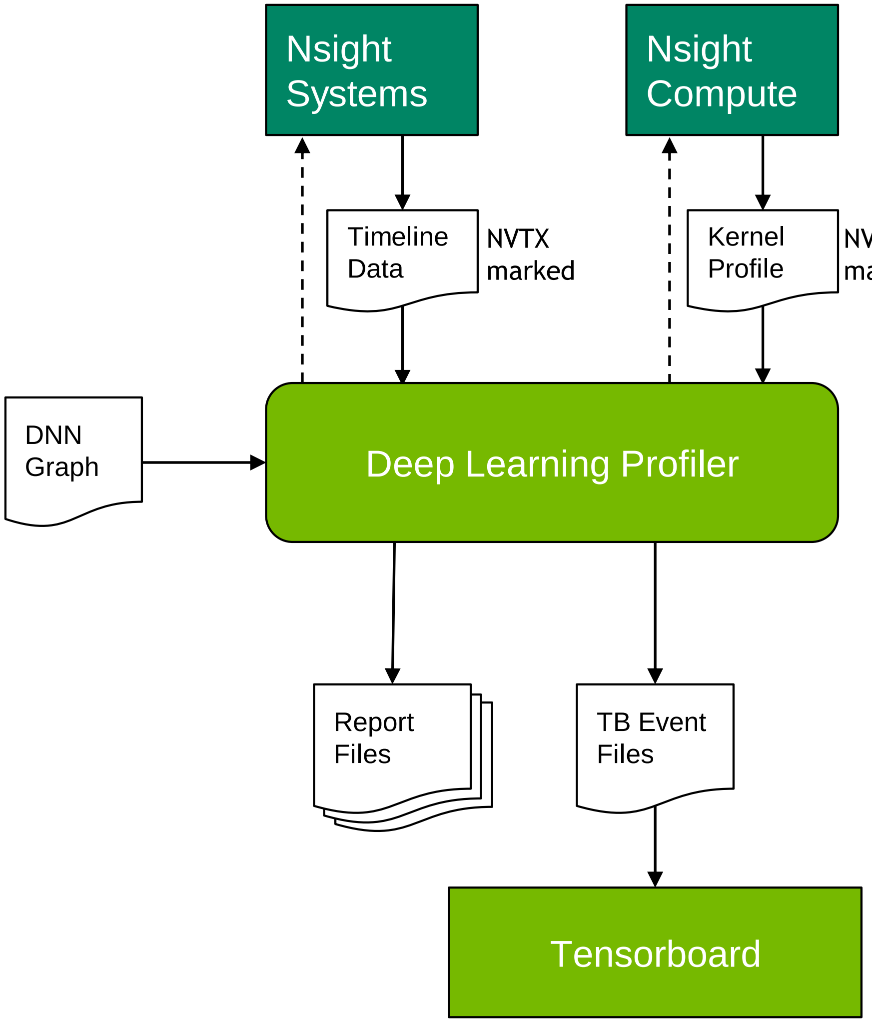
Nsight Compute

- Detailed kernel level profiling
- Determines Tensor Core usage from GPU program counters

Use NVTX markers to correlate kernels with DNN graph nodes

Any number of reports can be generated

- TB Event Files, CSV, JSON
- Analyze with tool of your choice



Deep Learning Profiler

Command Line Example

Example command to profile MobileNet V2 and generate a graphdef

```
$ /usr/bin/python tf_cnn_benchmarks.py --num_gpus=1 --batch_size=8 --model=mobilenet --device=gpu --
gpu_indices=1 --data_name=imagenet --data_dir=/data/train-val-tfrecord-480 --num_batches=1 --use_fp16
--fp16_enable_auto_loss_scale --graph_file=/results/mobilenet_graph.pb
```

Example Deep Learning Profiler command

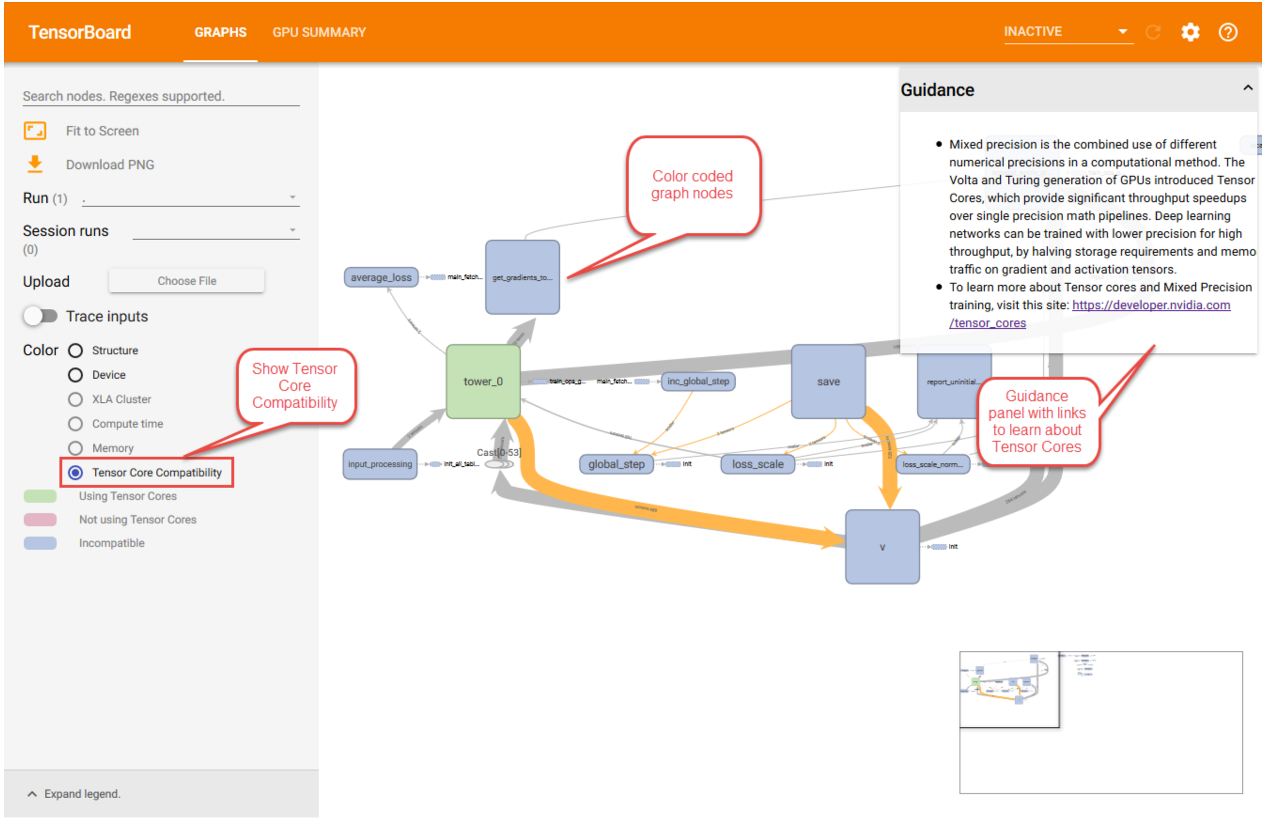
```
$ dlprof --in_graphdef=/results/mobilenet_graph.pb /usr/bin/python tf_cnn_benchmarks.py --num_gpus=1
--batch_size=8 --model=mobilenet --device=gpu --gpu_indices=1 --data_name=imagenet --
data_dir=/data/train-val-tfrecord-480 --num_batches=1 --use_fp16 --fp16_enable_auto_loss_scale
```

Launching TensorBoard

```
$ tensorboard --logdir ./event_files
```

Tensorboard Modifications

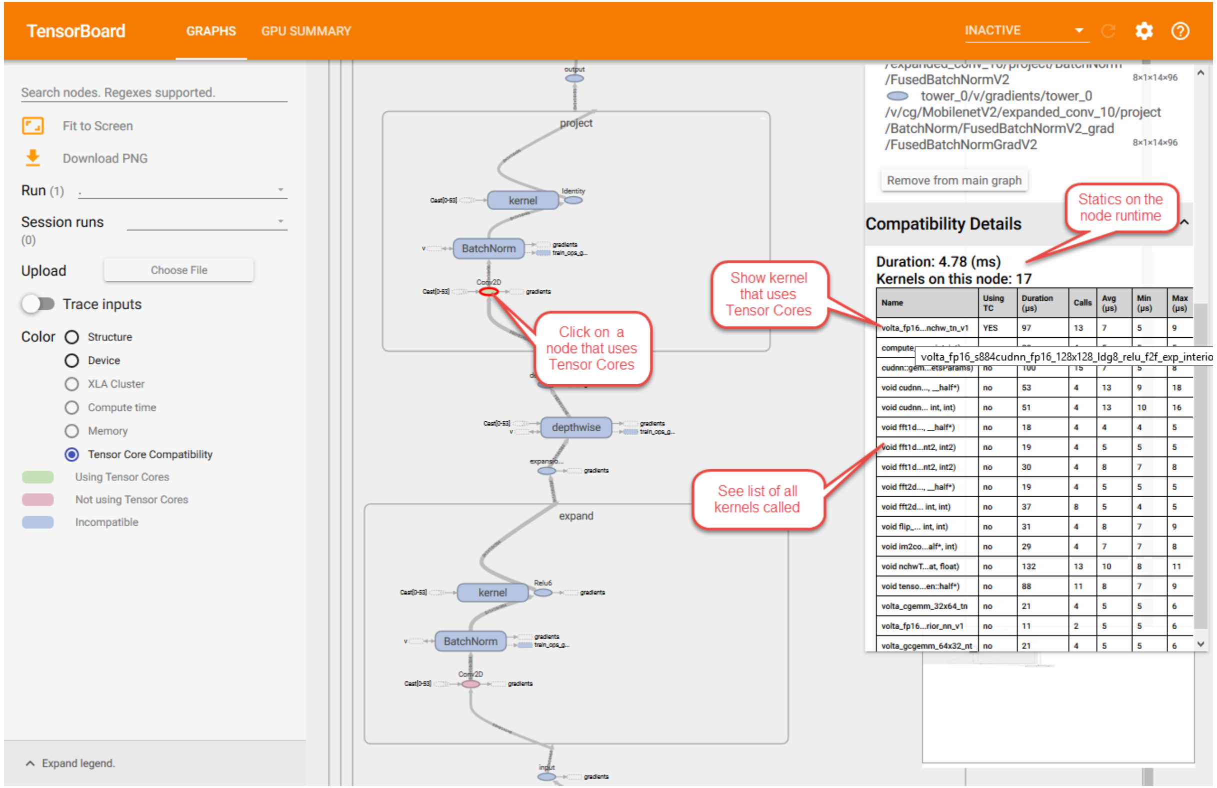
Start TensorBoard with NVIDIA modifications



NVIDIA.

Compatibility Details

Select Compatible using Tensor Cores

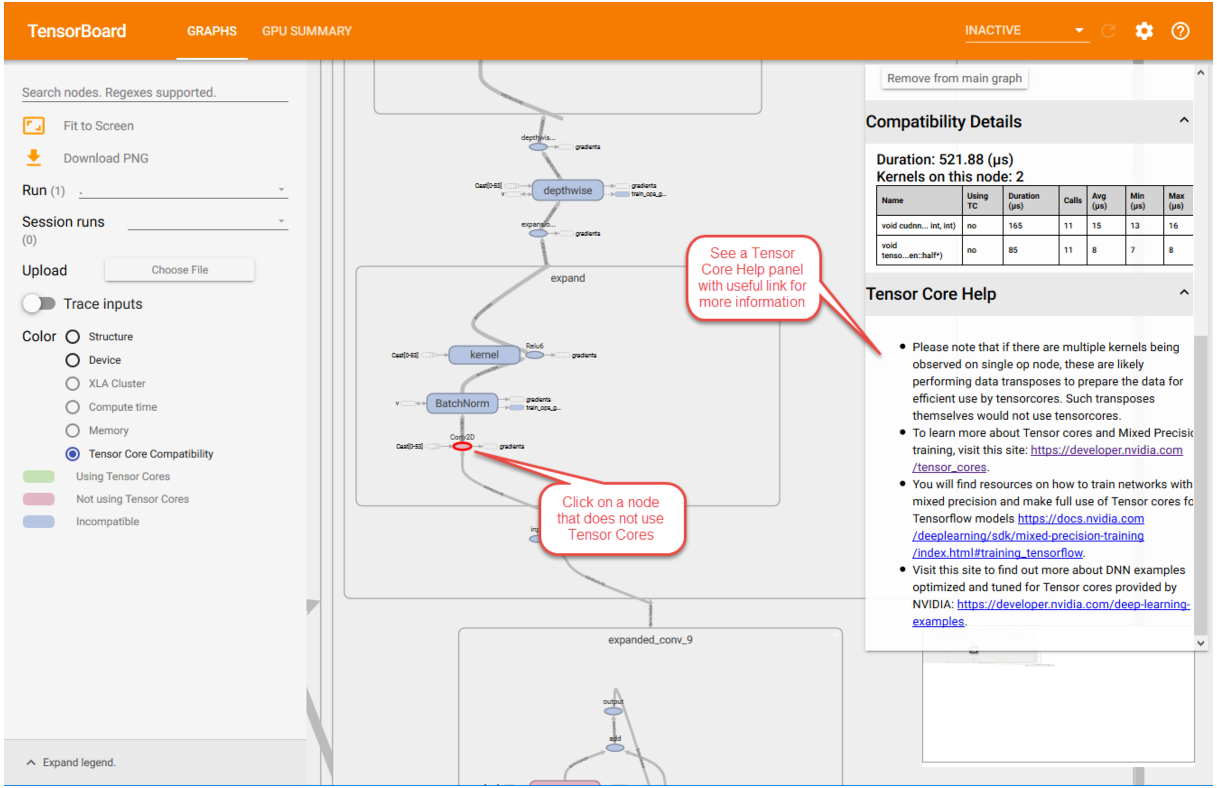


Name	Using TC	Duration (μs)	Calls	Avg (μs)	Min (μs)	Max (μs)
volta_fp16...nchw_tn_v1	YES	97	13	7	5	9
compute[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
cudnn::gem...etsParams)	no	100	15	7	5	8
void cudnn..., __half*)	no	53	4	13	9	18
void cudnn... int, int)	no	51	4	13	10	16
void fft1d..., __half*)	no	18	4	4	4	5
void fft1d...nt2, int2)	no	19	4	5	5	5
void fft1d...nt2, int2)	no	30	4	8	7	8
void fft2d..., __half*)	no	19	4	5	5	5
void fft2d... int, int)	no	37	8	5	4	5
void flip_... int, int)	no	31	4	8	7	9
void im2co...alf*, int)	no	29	4	7	7	8
void nchwT...at, float)	no	132	13	10	8	11
void tenso...en::half*)	no	88	11	8	7	9
volta_cgemm_32x64_tn	no	21	4	5	5	6
volta_fp16...rior_nn_v1	no	11	2	5	5	6
volta_gcgemm_64x32_nt	no	21	4	5	5	6

Compatibility Details

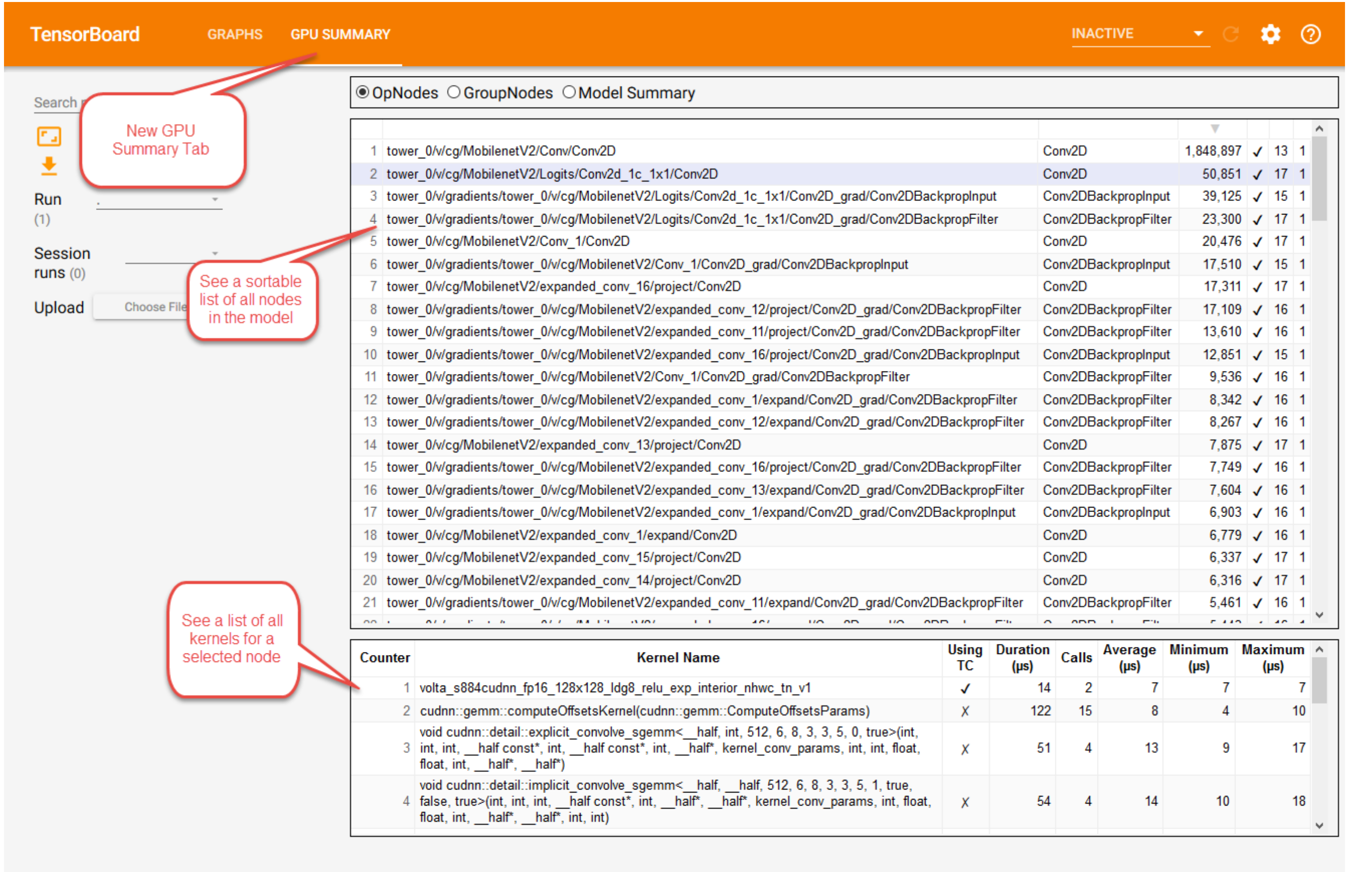
Select Compatible node not using Tensor Cores

Compatibility details and panel providing guidance and links to help with mixed precision



OpNodes Summary Tab

GPU Summary tab showing all the Nodes, compatible and using Tensor Cores

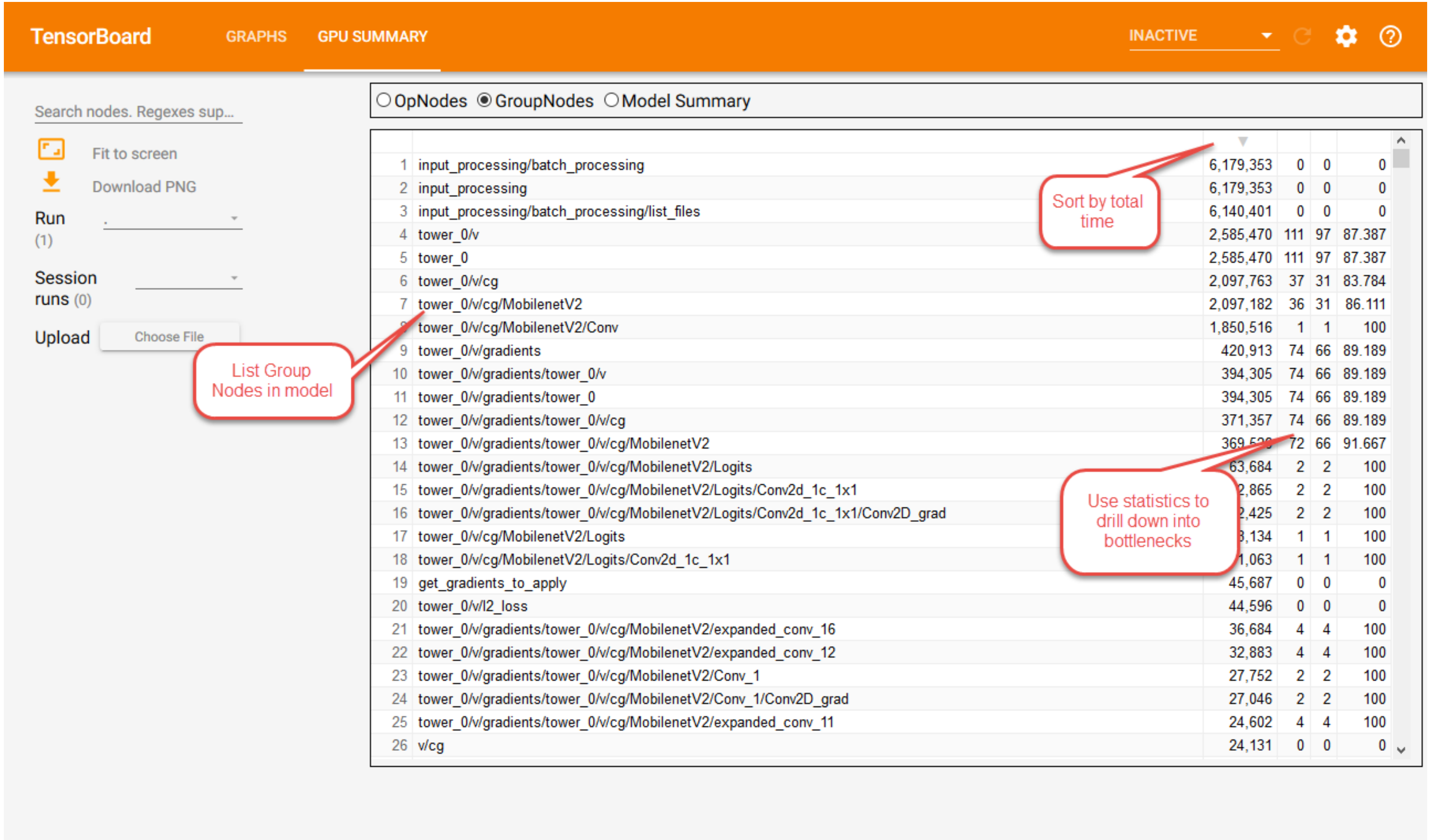


1	tower_0/v/cg/MobilenetV2/Conv/Conv2D	Conv2D	1,848,897	✓	13	1
2	tower_0/v/cg/MobilenetV2/Logits/Conv2d_1c_1x1/Conv2D	Conv2D	50,851	✓	17	1
3	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/Logits/Conv2d_1c_1x1/Conv2D_grad/Conv2DBackpropInput	Conv2DBackpropInput	39,125	✓	15	1
4	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/Logits/Conv2d_1c_1x1/Conv2D_grad/Conv2DBackpropFilter	Conv2DBackpropFilter	23,300	✓	17	1
5	tower_0/v/cg/MobilenetV2/Conv_1/Conv2D	Conv2D	20,476	✓	17	1
6	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/Conv_1/Conv2D_grad/Conv2DBackpropInput	Conv2DBackpropInput	17,510	✓	15	1
7	tower_0/v/cg/MobilenetV2/expanded_conv_16/project/Conv2D	Conv2D	17,311	✓	17	1
8	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/expanded_conv_12/project/Conv2D_grad/Conv2DBackpropFilter	Conv2DBackpropFilter	17,109	✓	16	1
9	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/expanded_conv_11/project/Conv2D_grad/Conv2DBackpropFilter	Conv2DBackpropFilter	13,610	✓	16	1
10	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/expanded_conv_16/project/Conv2D_grad/Conv2DBackpropInput	Conv2DBackpropInput	12,851	✓	15	1
11	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/Conv_1/Conv2D_grad/Conv2DBackpropFilter	Conv2DBackpropFilter	9,536	✓	16	1
12	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/expanded_conv_1/expand/Conv2D_grad/Conv2DBackpropFilter	Conv2DBackpropFilter	8,342	✓	16	1
13	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/expanded_conv_12/expand/Conv2D_grad/Conv2DBackpropFilter	Conv2DBackpropFilter	8,267	✓	16	1
14	tower_0/v/cg/MobilenetV2/expanded_conv_13/project/Conv2D	Conv2D	7,875	✓	17	1
15	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/expanded_conv_16/project/Conv2D_grad/Conv2DBackpropFilter	Conv2DBackpropFilter	7,749	✓	16	1
16	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/expanded_conv_13/expand/Conv2D_grad/Conv2DBackpropFilter	Conv2DBackpropFilter	7,604	✓	16	1
17	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/expanded_conv_1/expand/Conv2D_grad/Conv2DBackpropInput	Conv2DBackpropInput	6,903	✓	16	1
18	tower_0/v/cg/MobilenetV2/expanded_conv_1/expand/Conv2D	Conv2D	6,779	✓	16	1
19	tower_0/v/cg/MobilenetV2/expanded_conv_15/project/Conv2D	Conv2D	6,337	✓	17	1
20	tower_0/v/cg/MobilenetV2/expanded_conv_14/project/Conv2D	Conv2D	6,316	✓	17	1
21	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/expanded_conv_11/expand/Conv2D_grad/Conv2DBackpropFilter	Conv2DBackpropFilter	5,461	✓	16	1

Counter	Kernel Name	Using TC	Duration (µs)	Calls	Average (µs)	Minimum (µs)	Maximum (µs)
1	volta_s884cudnn_fp16_128x128_ldg8_relu_exp_interior_nhwc_tn_v1	✓	14	2	7	7	7
2	cudnn::gemm::computeOffsetsKernel(cudnn::gemm::ComputeOffsetsParams)	X	122	15	8	4	10
3	void cudnn::detail::explicit_convolve_sgemm<__half, int, 512, 6, 8, 3, 3, 5, 0, true>(int, int, int, __half const*, int, __half const*, int, __half*, kernel_conv_params, int, int, float, float, int, __half*, __half*)	X	51	4	13	9	17
4	void cudnn::detail::implicit_convolve_sgemm<__half, __half, 512, 6, 8, 3, 3, 5, 1, true, false, true>(int, int, int, __half const*, int, __half*, __half*, kernel_conv_params, int, float, float, int, __half*, __half*, int, int)	X	54	4	14	10	18

Group Node Summary Tab

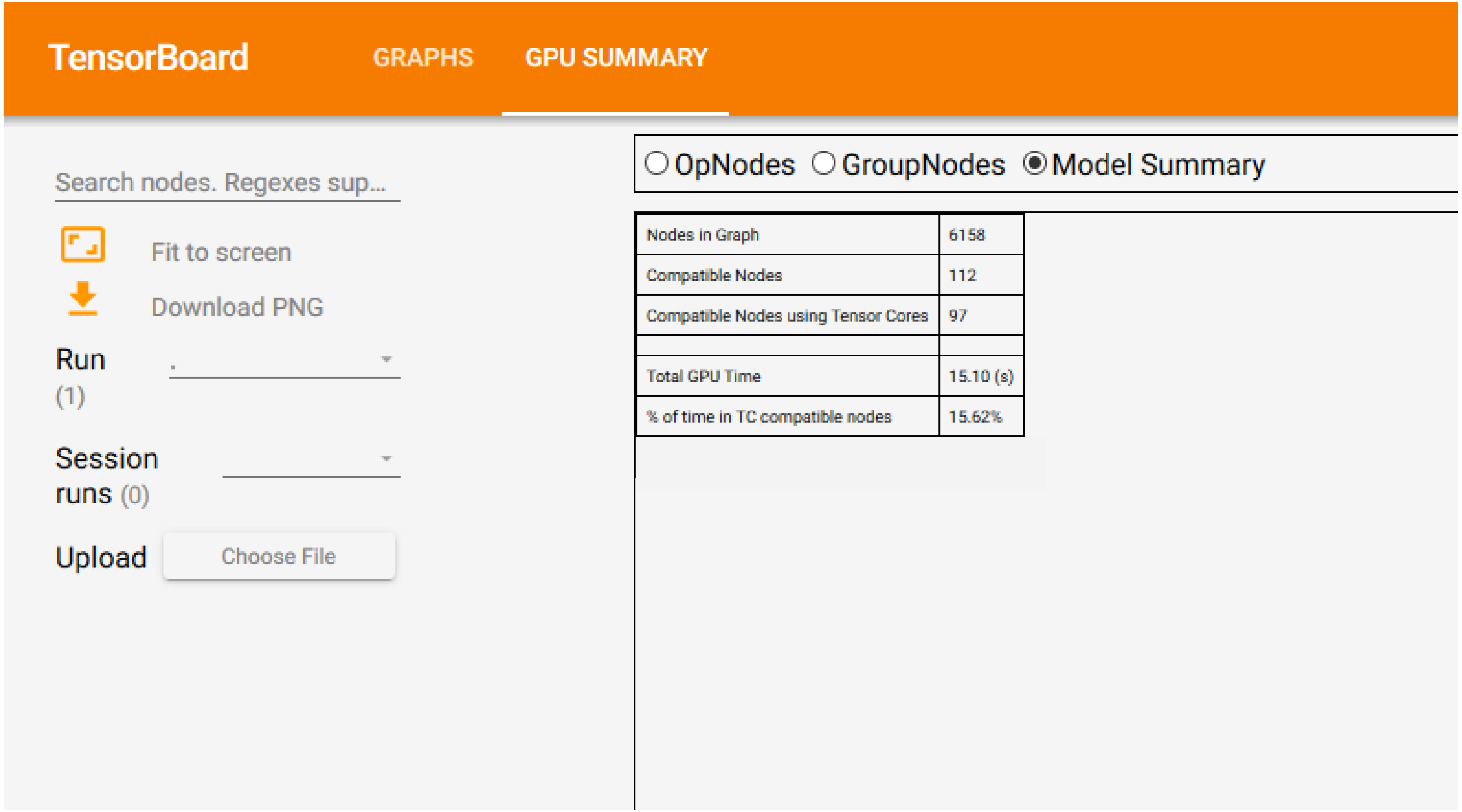
Roll up timing metrics and Tensor Core utilization per group node



1	input_processing/batch_processing	6,179,353	0	0	0
2	input_processing	6,179,353	0	0	0
3	input_processing/batch_processing/list_files	6,140,401	0	0	0
4	tower_0/v	2,585,470	111	97	87.387
5	tower_0	2,585,470	111	97	87.387
6	tower_0/v/cg	2,097,763	37	31	83.784
7	tower_0/v/cg/MobilenetV2	2,097,182	36	31	86.111
8	tower_0/v/cg/MobilenetV2/Conv	1,850,516	1	1	100
9	tower_0/v/gradients	420,913	74	66	89.189
10	tower_0/v/gradients/tower_0/v	394,305	74	66	89.189
11	tower_0/v/gradients/tower_0	394,305	74	66	89.189
12	tower_0/v/gradients/tower_0/v/cg	371,357	74	66	89.189
13	tower_0/v/gradients/tower_0/v/cg/MobilenetV2	369,5[illegible]	72	66	91.667
14	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/Logits	63,684	2	2	100
15	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/Logits/Conv2d_1c_1x1	2,865	2	2	100
16	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/Logits/Conv2d_1c_1x1/Conv2D_grad	2,425	2	2	100
17	tower_0/v/cg/MobilenetV2/Logits	3,134	1	1	100
18	tower_0/v/cg/MobilenetV2/Logits/Conv2d_1c_1x1	1,063	1	1	100
19	get_gradients_to_apply	45,687	0	0	0
20	tower_0/v/l2_loss	44,596	0	0	0
21	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/expanded_conv_16	36,684	4	4	100
22	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/expanded_conv_12	32,883	4	4	100
23	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/Conv_1	27,752	2	2	100
24	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/Conv_1/Conv2D_grad	27,046	2	2	100
25	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/expanded_conv_11	24,602	4	4	100
26	v/cg	24,131	0	0	0

Model Summary Table

Model Summary shows concise information on Tensor core usage



Nodes in Graph	6158
Compatible Nodes	112
Compatible Nodes using Tensor Cores	97
Total GPU Time	15.10 (s)
% of time in TC compatible nodes	15.62%

THANK YOU