



GPU PROFILING 기법을 통한 DEEP LEARNING 성능 최적화 기법 소개

Gwangsoo Hong, Solution Architect, ghong@nvidia.com



AGENDA

Introduction to Nsight Systems

Profiling from CLI

NVTX (NVIDIA Tools Extension)

Deep learning Optimization

Getting Started Resources

An abstract graphic featuring a network of glowing green nodes connected by thin, intersecting lines. The nodes are scattered across the frame, with some appearing as bright green dots and others as slightly larger, blurred circles. The lines create a complex web of connections, suggesting a data network or a system architecture. The background is a deep, dark blue or black, which makes the green elements stand out.

INTRODUCTION TO NSIGHT SYSTEMS

HOW TO SPEED-UP NETWORK

- ▶ Use the latest GPU and more GPU?
- ▶ Wait NVIDIA to release the new GPU or newer library?
- ▶ **Optimize Neural Network Computing or Something**
 - ▶ Need to understand your network operation

We are talking about this

PROFILING NEURAL NETWORK

Profiling with cProfiler + Snakeviz



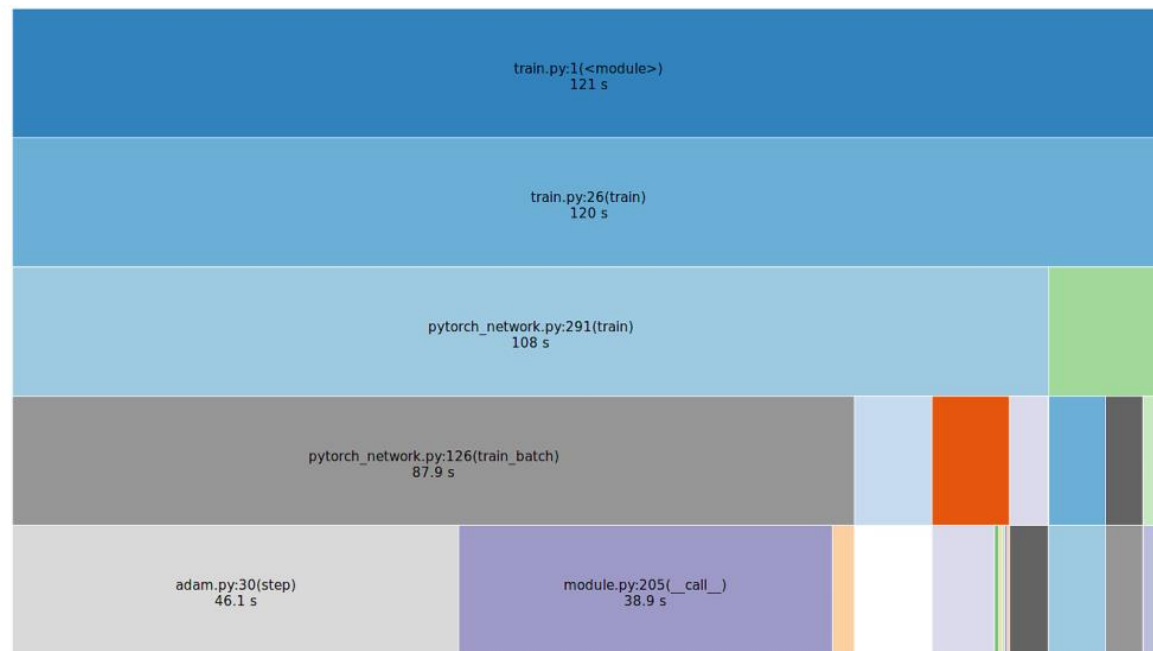
Following

Replying to @sleepinyourhat @PyTorch

I've found this helpful when profiling pytorch code: jiffyclub.github.io/snakeviz/
It takes cProfile outputs and gives much nicer viz

7:01 AM - 27 May 2017

5 Retweets 46 Likes

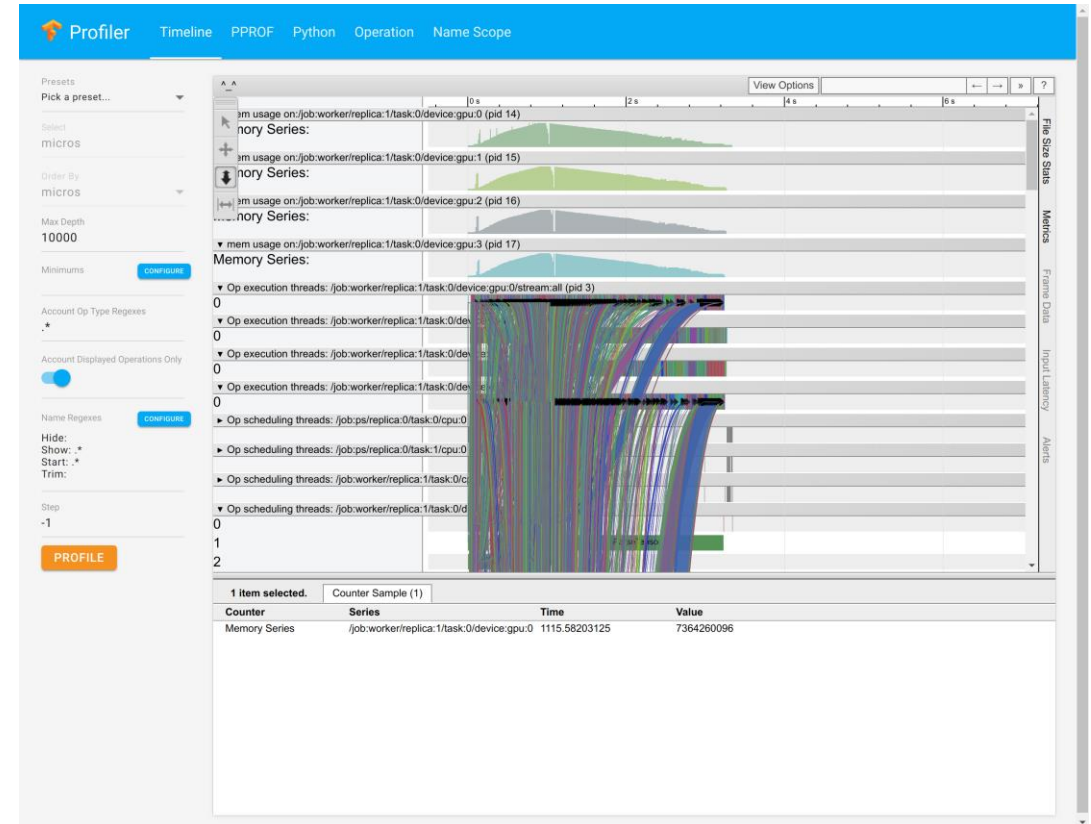


```
$python -m cProfile -o 100_percent_gpu_utilization.prof train.py  
$snakeviz 100_percent_gpu_utilization.prof
```

TENSORFLOW PROFILER

<https://github.com/tensorflow/profiler-ui>

- ▶ Show timeline and can trace network
- ▶ Still difficult to understand



NVIDIA PROFILING TOOLS

Nsight Systems



Nsight Compute



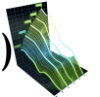
Nsight Visual Studio Edition

Nsight Eclipse Edition



NVIDIA Profiler (nvprof)

NVIDIA Visual Profiler (nvvp)



CUPTI (CUDA Profiling Tools Interface)

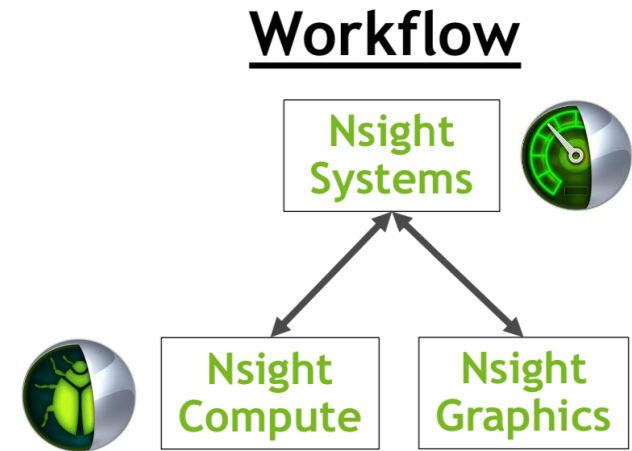
NSIGHT PRODUCT FAMILY

- ▶ Standalone Performance Tools

- ▶ **Nsight Systems** system-wide application algorithm tuning
- ▶ **Nsight Compute** Debug/optimize specific CUDA kernel
- ▶ **Nsight Graphics** Debug/optimize specific graphics

- ▶ IDE plugins

- ▶ **Nsight Eclipse Edicion/Visual Studio** editor, debugger, some perf analysis

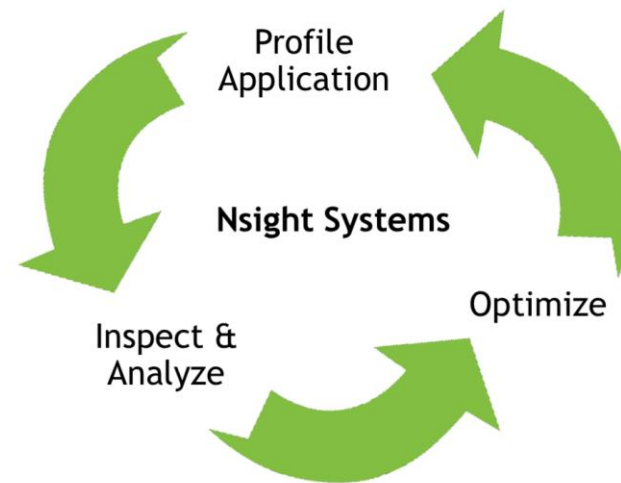


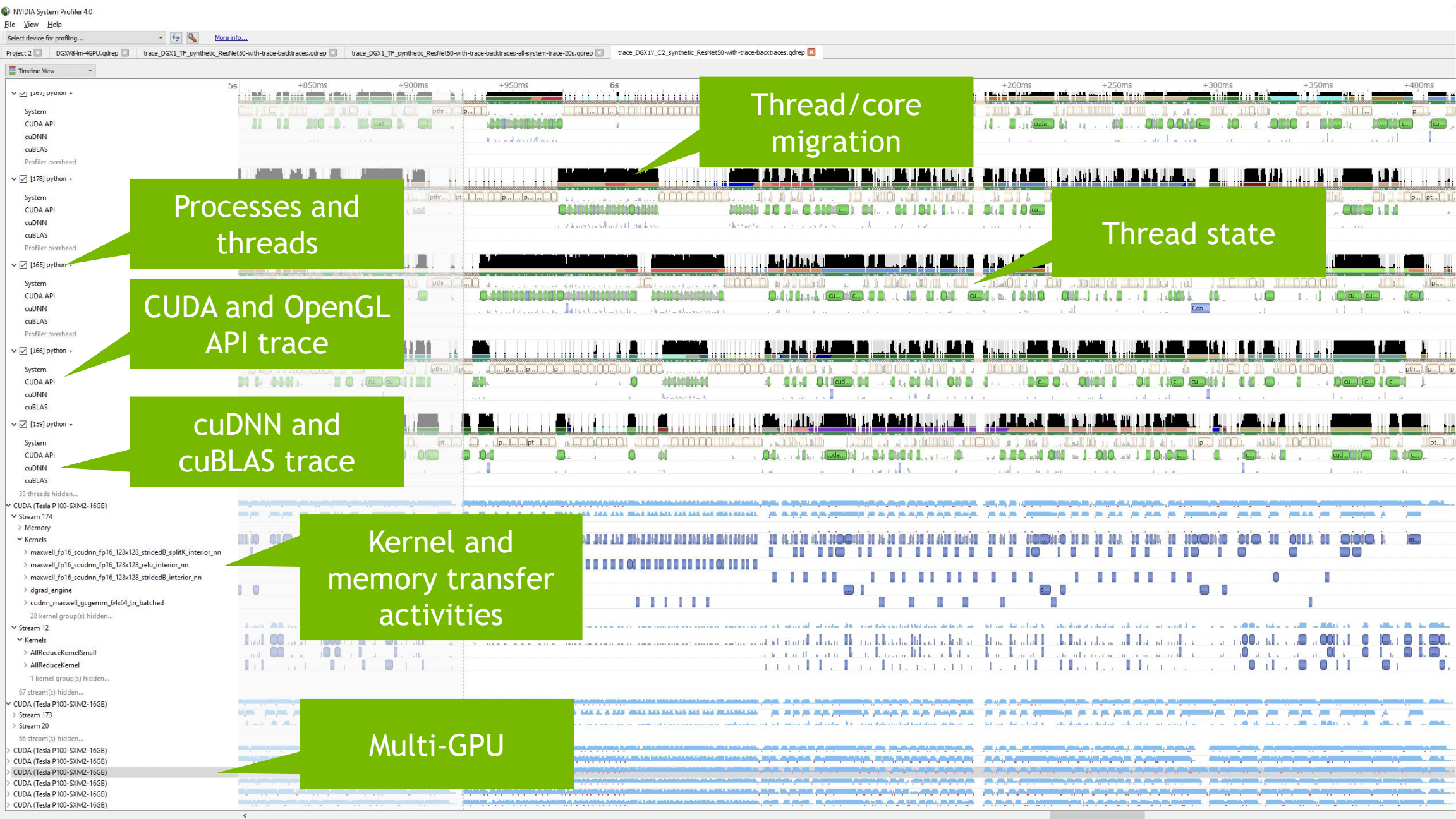


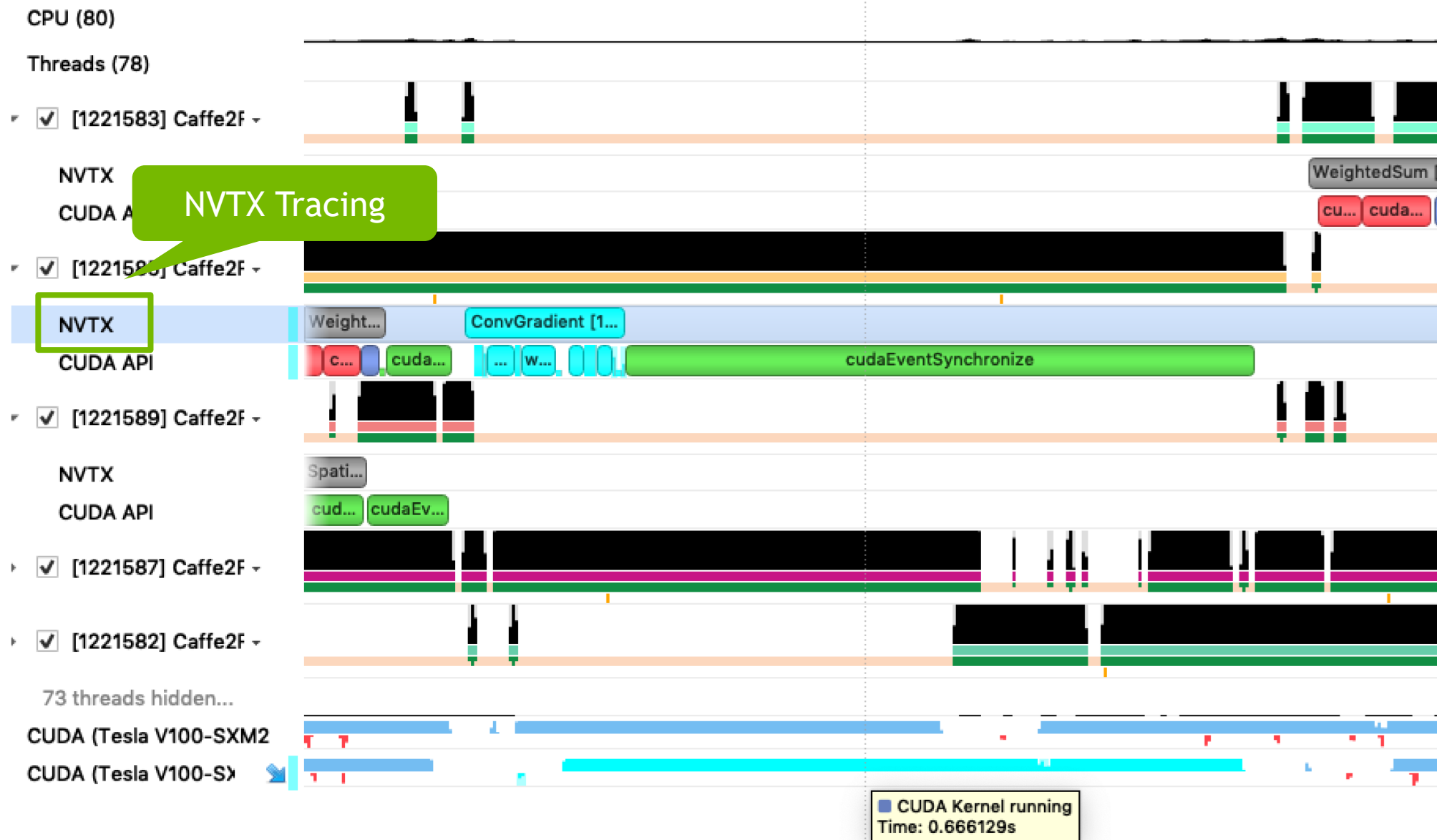
NSIGHT SYSTEMS

Overview

- ▶ Profile **System-wide** application
 - ▶ Multi-process tree, GPU workload trace, etc
- ▶ Investigate your workload across multiple CPUs and GPUs
 - ▶ CPU algorithms, utilization, and thread states
 - ▶ GPU streams kernels, memory transfers, etc
 - ▶ NVTX, CUDA & Library API, etc
- ▶ Ready for Big Data
 - ▶ docker, user privilege (linux), cli, etc

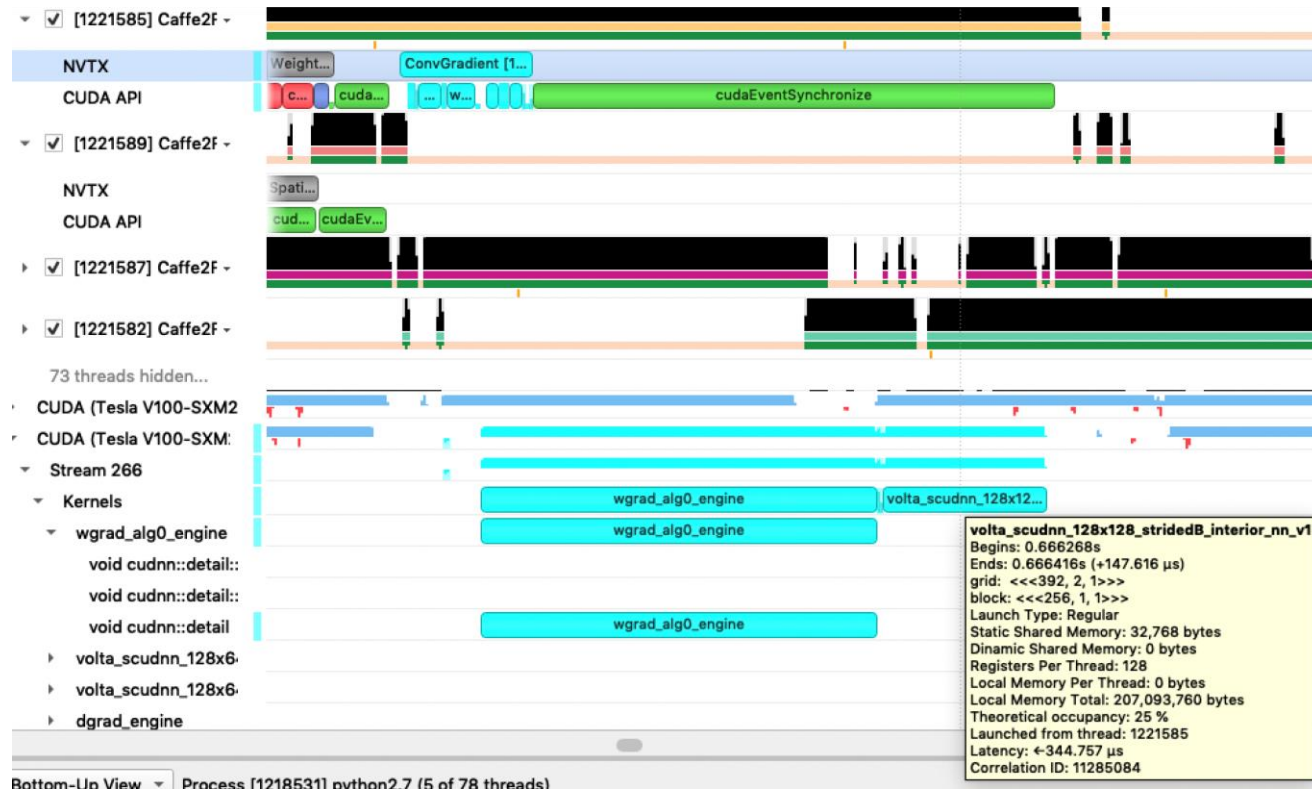






TRANSITIONING TO PROFILE A KERNEL

Dive into kernel analysis



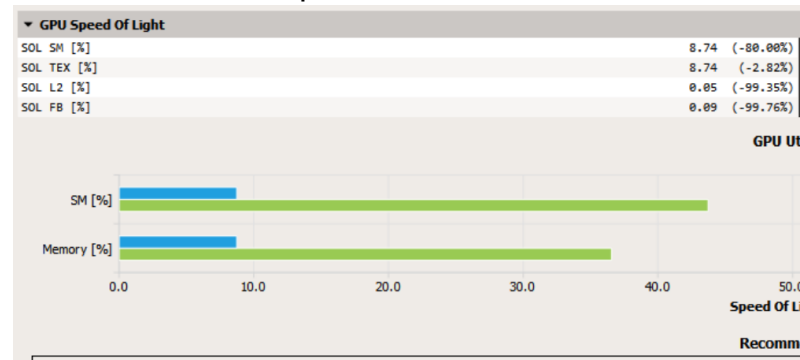


NVIDIA NSIGHT COMPUTE

Next Generation Kernel Profiler

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

Kernel Profile Comparisons with Baseline



Metric Data

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
l1c_sol_pct [%]	6.93	7.18
memory_access_size_type [bytes]	2.00; 32.00; 32.00; 32.00	2.00; 32.00; 32.00; 32.00

Source Correlation

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

PROFILING GPU APPLICATION

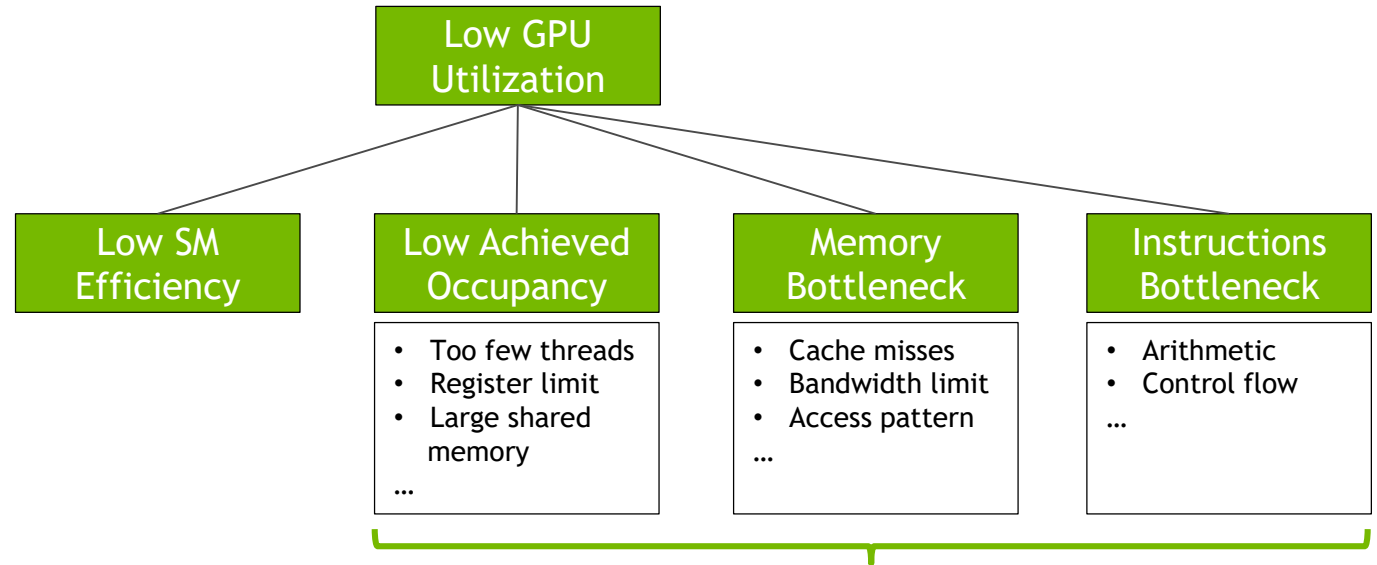
Focusing GPU Computing

How to measure

GPU Profiling

CPU/GPU
Tracing

Application
Tracing



NVIDIA (Visual) Profiler / Nsight Compute

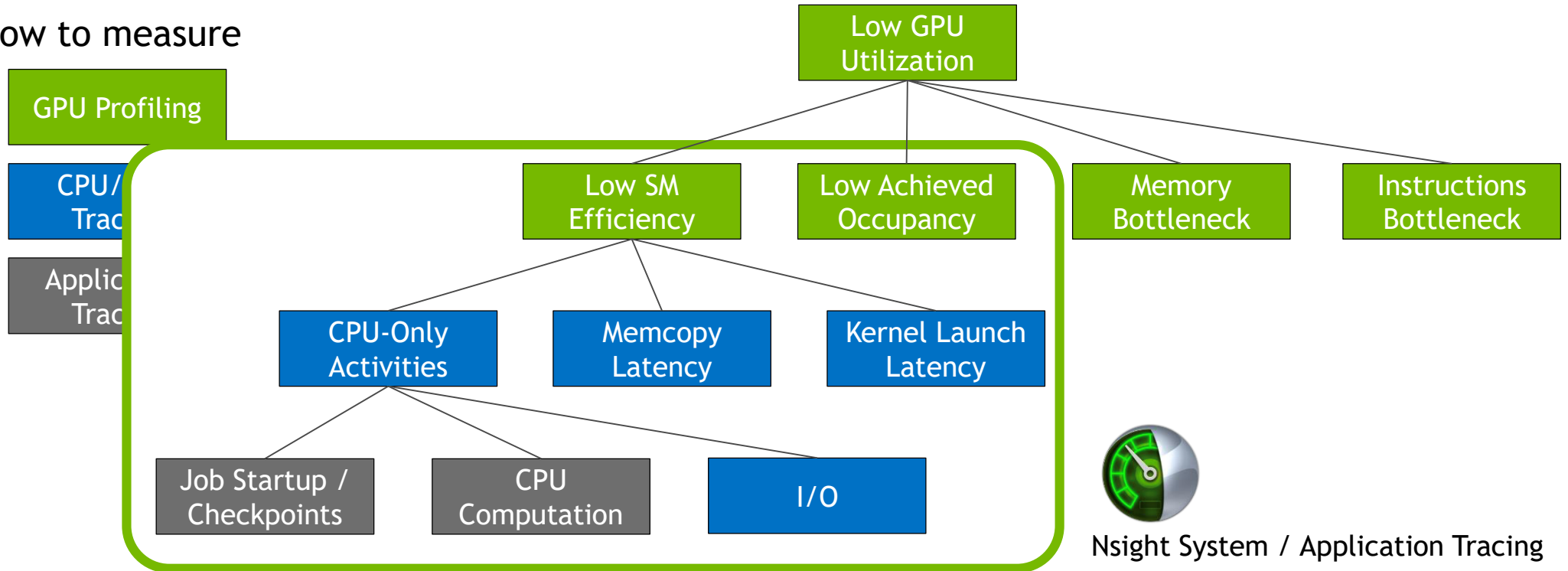


NVIDIA Supports them with cuDNN, cuBLAS, and so on

PROFILING GPU APPLICATION

Focusing System Operation

How to measure



The background is a dark blue gradient. It features a network of thin, light green lines that crisscross the frame. At various points where these lines intersect or terminate, there are small, bright green circular dots. Some of these dots are slightly larger and more prominent than others. The overall effect is that of a digital or scientific network visualization.

HOW TO USE

NSIGHT SYSTEMS PROFILE

Profile with CLI

APIs to be traced

```
$ nsys profile -t cuda,osrt,nvtx,cudnn,cublas \  
-o baseline.qdstrm -w true python main.py
```

Name of
output file

Show output
on console

Application
command

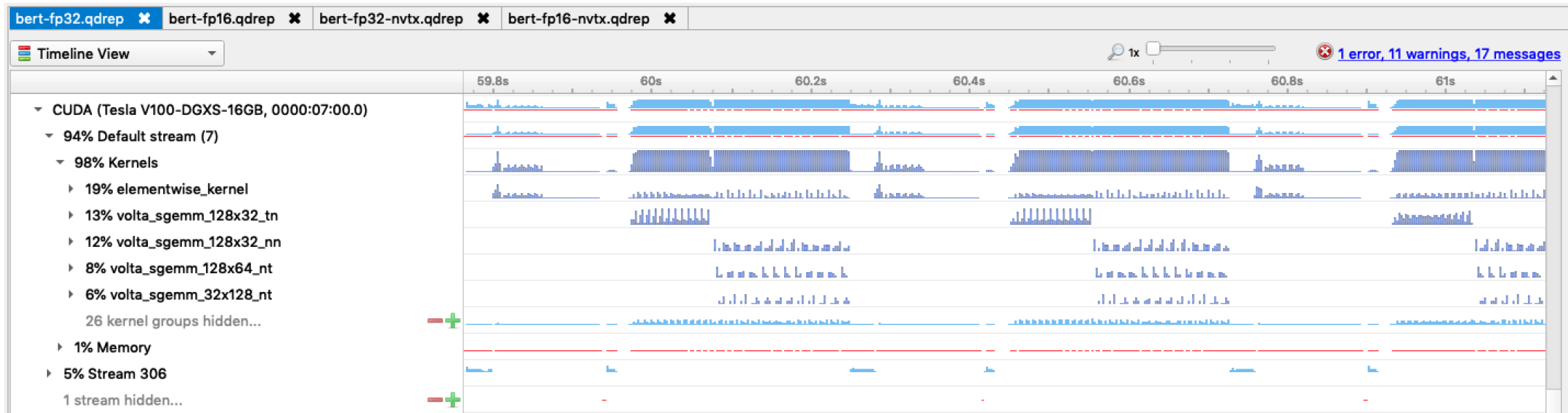
Automatic conversion of .qdstrm temp results file to .qdrep format if converter utility is available.

cuda - GPU kernel
osrt - OS runtime
nvtx - NVIDIA Tools Extension
cudnn - CUDA Deep NN library
cublas - CUDA BLAS library

https://docs.nvidia.com/nsight-systems/#nsight_systems/2019.3.6-x86/06-cli-profiling.htm

NSIGHT SYSTEMS PROFILE

No NVTX



- Difficult to understand → no useful

The background is a dark blue gradient with a complex network of thin, glowing green lines. These lines connect various points, some of which are highlighted as bright green dots. The overall effect is a sense of a dynamic, interconnected system or data flow.

NVTX (NVIDIA TOOLS EXTENSION)

NVTX ANNOTATIONS

```
def train(args, model, device, train_loader, optimizer, epoch):  
    model.train()  
    for batch_idx, (data, target) in enumerate(train_loader):  
  
        data, target = data.to(device), target.to(device)  
  
        optimizer.zero_grad()  
        output = model(data)  
        loss = F.nll_loss(output, target)
```

NVTX ANNOTATIONS

NVTX in PyTorch

```
import torch.cuda.nvtx as nvtx
def train(args, model, device, train_loader, optimizer, epoch):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        nvtx.range_push("Batch " + str(batch_idx))
        nvtx.range_push("Copy to device")
        data, target = data.to(device), target.to(device)
        nvtx.range_pop()
        nvtx.range_push("Forward pass")
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        nvtx.range_pop()
        nvtx.range_pop()
```

Batch %d



copy to device

Forward pass

NVTX ANNOTATIONS

NVTX using cupy package

```
from cupy.cuda import nvtx
def train(args, model, device, train_loader, optimizer, epoch):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        nvtx.RangePush("Batch " + str(batch_idx))
        nvtx.RangePush("Copy to device")
        data, target = data.to(device), target.to(device)
        nvtx.RangePop()
        nvtx.RangePush("Forward pass")
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        nvtx.RangePop()
        nvtx.RangePop()
```

Batch %d



copy to device

Forward pass

NSIGHT SYSTEM PROFILE

NVTX range marker tip

- ▶ NVTX for data loading (data augmentation)

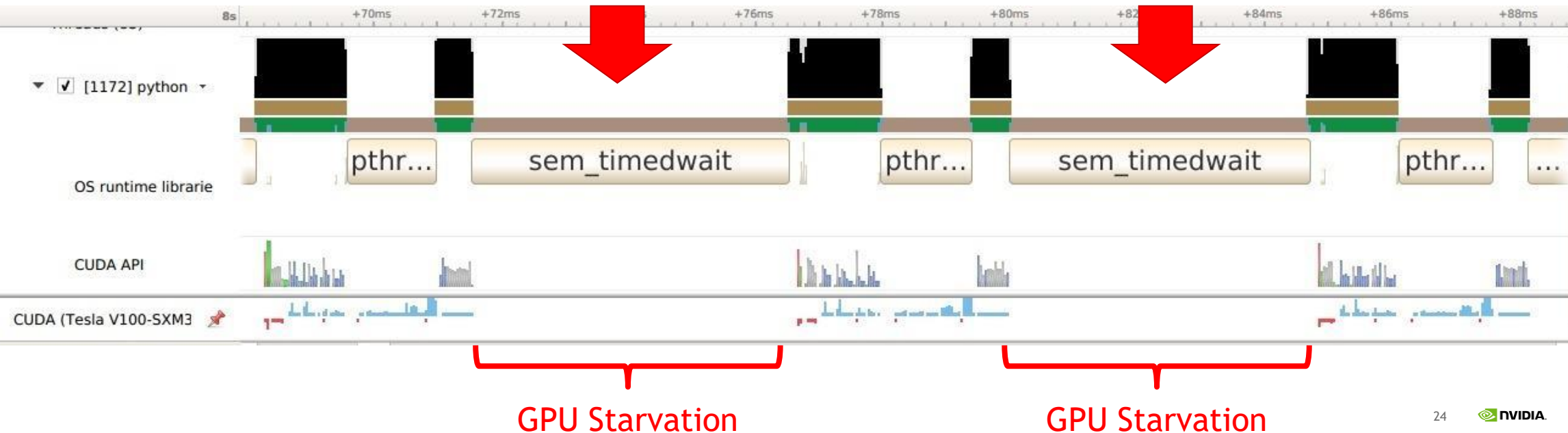
```
for batch_idx, (data,target) in enumerate(train_loader):
```



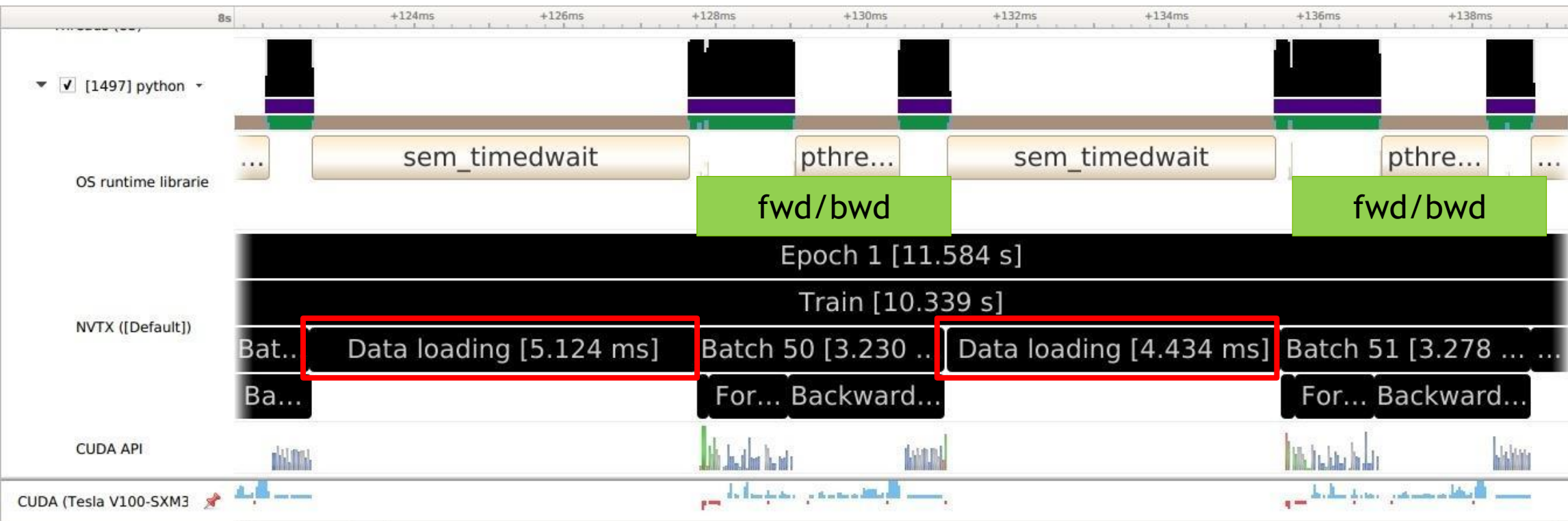
```
nvtx.range_push('Data loading')  
(data,target) = train_loader.next()  
nvtx.range_pop()
```

BASELINE PROFILE

- ▶ MNIST Training: 89 sec, <5% utilization
- ▶ CPU waits on a semaphore and starves the GPU!



BASELINE PROFILE (WITH NVTX)



- ▶ GPU is idle during **data loading**
- ▶ Data is loaded using a single thread. This starves the GPU!

OPTIMIZE SOURCE CODE

- ▶ Data loader was configured to use 1 worker thread

```
kwargs = {'num_workers': 1, 'pin_memory' True if use_cuda else {}}
```

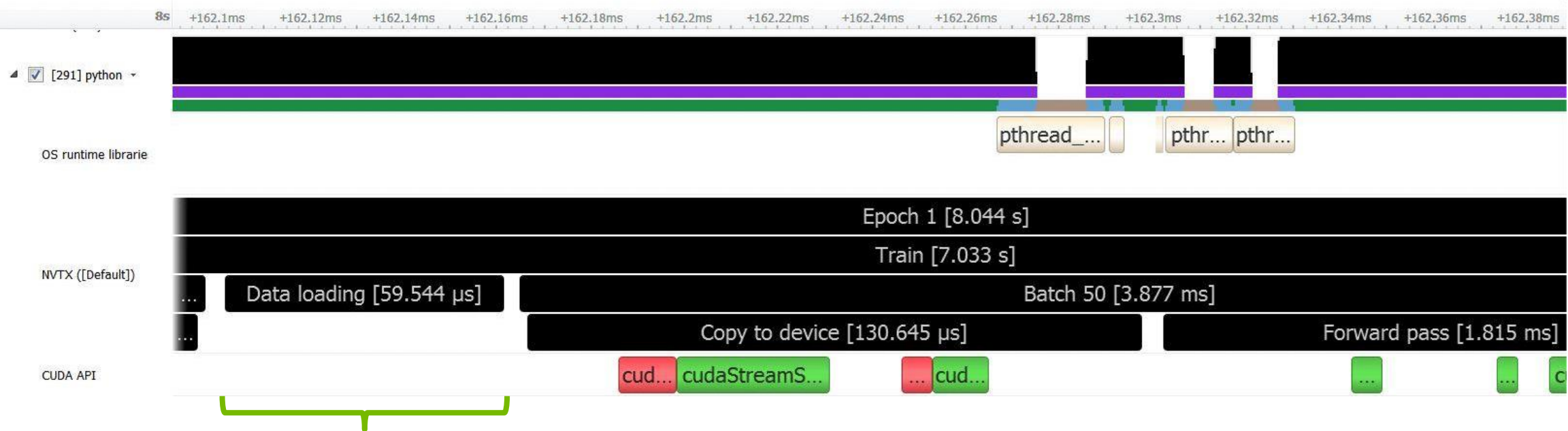


- ▶ Let's switch to using 8 worker threads:

```
kwargs = {'num_workers': 8, 'pin_memory' True if use_cuda else {}}
```

AFTER OPTIMIZATION

- ▶ Time for data loading reduced for each bath



60us

Reduced from 5.1ms to 60us for batch 50

The background is a dark, almost black, field with a complex network of thin, glowing green lines. These lines intersect at various points, creating a web-like structure. At several of these intersection points, there are small, bright green circular dots or nodes. The overall effect is reminiscent of a neural network diagram or a data visualization of complex connections.

DEEP LEARNING OPTIMIZATION

OPTIMIZATION STRATEGY FOR DL

- ▶ Algorithm optimization

- ▶ Tensor Cores

- ▶ Training: Automatic Mixed Precision

- ▶ Inference: TensorRT

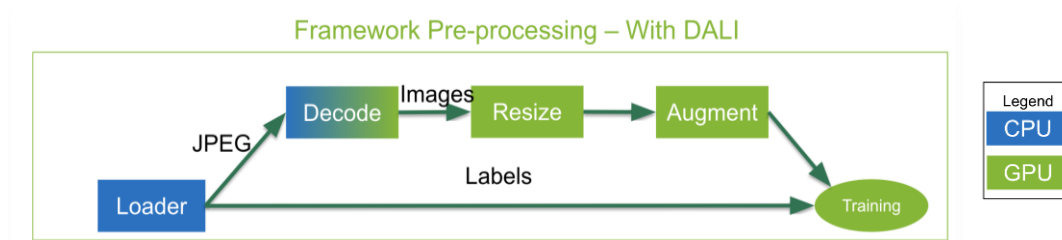
$$\mathbf{D} = \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix} + \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}$$

FP16 or FP32 FP16 FP16 FP16 or FP32

- ▶ Available in **Volta** and **Turing** architecture GPUs
 - ▶ 125 Tflops in FP16 vs. 15.7 Tflops in FP32 (**8x** speed-up)
 - ▶ Optimized **4x4x4** dot operation (GEMM)

- ▶ Data pipeline optimization

- ▶ DALI (Data loading Library)



The background is a dark blue gradient with a complex network of thin, glowing green lines. These lines connect various points, some of which are larger, brighter green dots. The overall effect is a sense of a dynamic, interconnected system, possibly representing a neural network or a data flow.

TRAINING OPTIMIZATION CASE

NVTX TAGGING

BERT in PyTorch

```
loss = model(input_ids, segment_ids, input_mask, start_positions, end_positions)

...

if args.fp16:
    optimizer.backward(loss)
else:
    loss.backward()
if (step + 1) % args.gradient_accumulation_steps == 0:
    if args.fp16:
        # modify learning rate with special warm up BERT uses
        # if args.fp16 is False, BertAdam is used and handles this automatically
        lr_this_step = args.learning_rate * warmup_linear.get_lr(global_step, args.warmup_proportion)
        for param_group in optimizer.param_groups:
            param_group['lr'] = lr_this_step
    optimizer.step()
    optimizer.zero_grad()
    global_step += 1
```

NVTX TAGGING

BERT in PyTorch

```
nvtx.range_push("Batch " + str(step))
nvtx.range_push("Forward pass")
loss = model(input_ids, segment_ids, input_mask, start_positions, end_positions)
nvtx.range_pop()

...
nvtx.range_push("Backward pass")
if args.fp16:
    optimizer.backward(loss)
else:
    loss.backward()
if (step + 1) % args.gradient_accumulation_steps == 0:
if args.fp16:
    # modify learning rate with special warm up BERT uses
    # if args.fp16 is False, BertAdam is used and handles this automatically
    lr_this_step = args.learning_rate * warmup_linear.get_lr(global_step, args.warmup_proportion)
    for param_group in optimizer.param_groups:
        param_group['lr'] = lr_this_step
optimizer.step()
optimizer.zero_grad()
global_step += 1
nvtx.range_pop()
nvtx.range_pop()
```

forward pass

backward pass

Batch %d

ALGORITHM OPTIMIZATION - TRAINING

Single Precision (FP32) Training for BERT

Getting API List

~60% GEMM

98% Kernels
19% elementwise_kernel
13% volta_sgemm_128x32_tn
12% volta_sgemm_128x32_nn
8% volta_sgemm_128x64_nt
6% volta_sgemm_32x128_nt

Memory

hidden...



ALGORITHM OPTIMIZATION - TRAINING

Automatic Mixed Precision (FP32 + FP16) Training for BERT

- **2.1x** Speed up (~460ms vs. ~222ms)

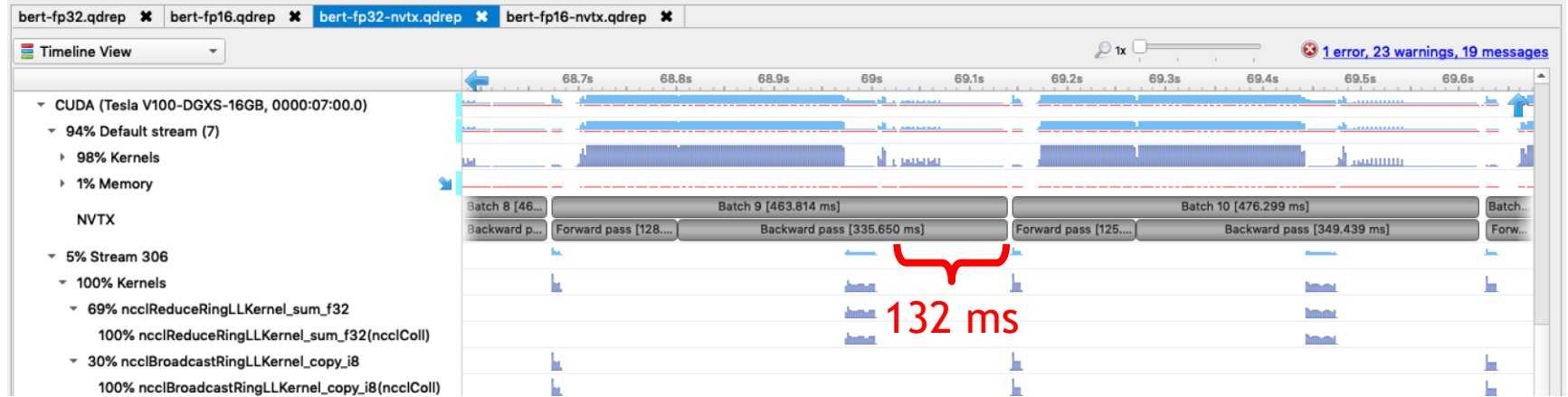


- **volta_fp16_s884gemm** indicates for using the tensor cores.

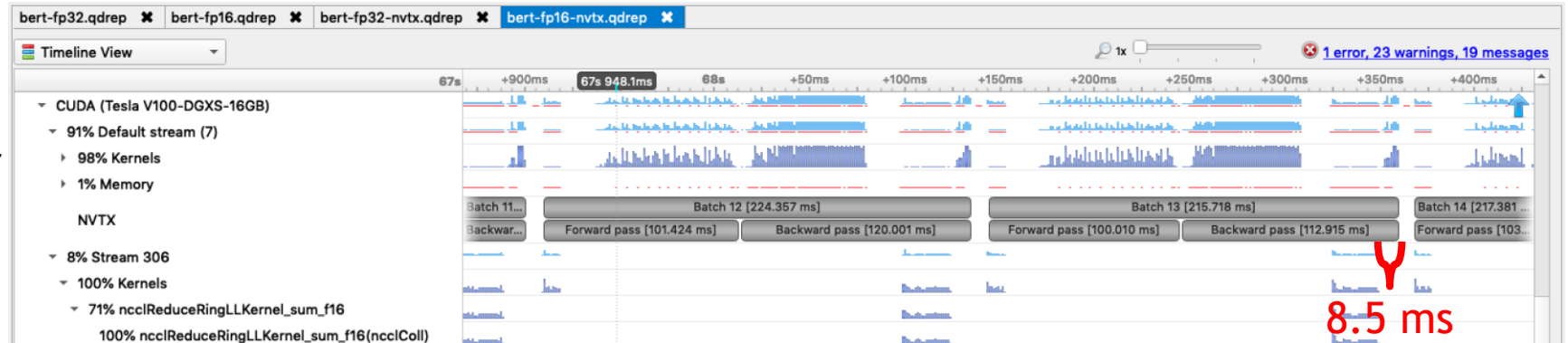
ALGORITHM OPTIMIZATION - TRAINING

APEX ADAM Optimizer Optimization

- ADAM optimizer
 - Low Utilization



- APEX ADAM optimizer
 - High utilization





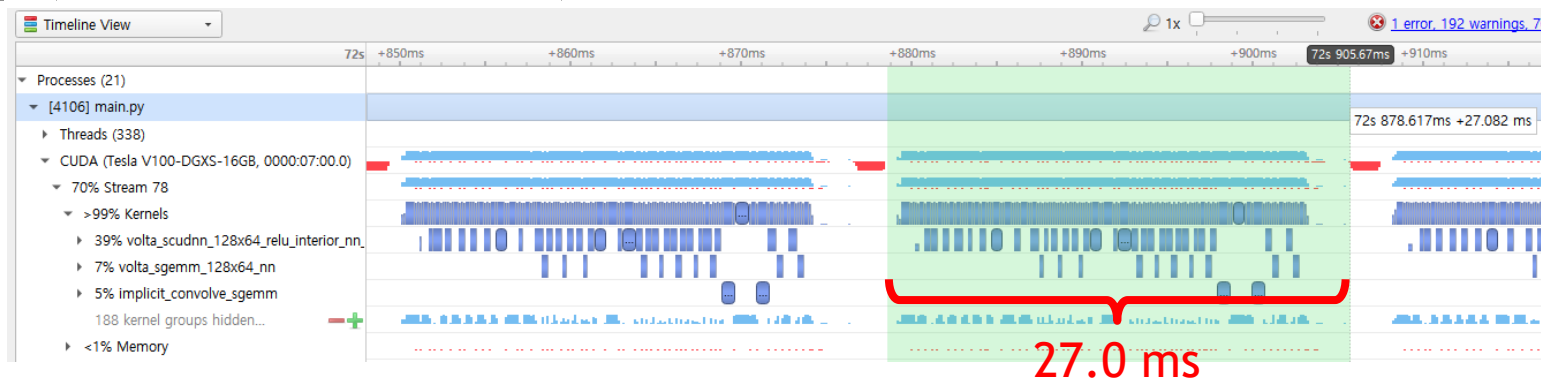
INFERENCE OPTIMIZATION CASE

ALGORITHM OPTIMIZATION - INFERENCE

TensorFlow Single Precision (FP32) vs. half Precision (FP16)

- 2.4x Speed up (~27.0ms vs. ~11.0ms)

Single Precision (FP32)



Half Precision (FP16)

Tensor Cores
APIs

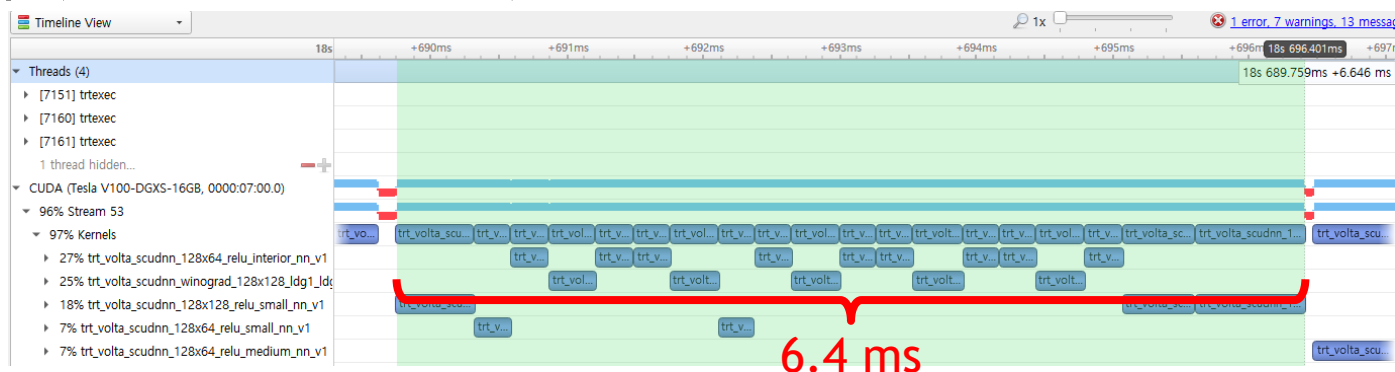


ALGORITHM OPTIMIZATION - INFERENCE

TensorRT Single Precision (FP32) vs. Half Precision (FP16)

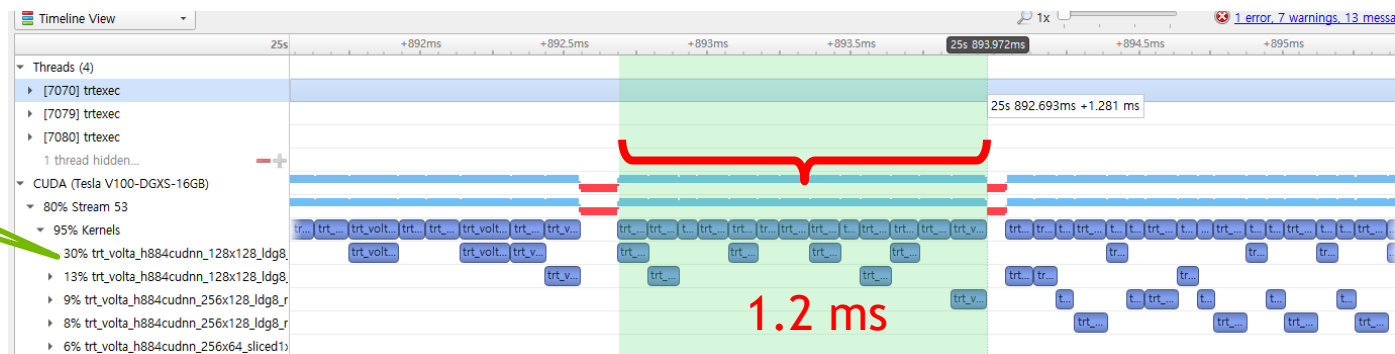
- 5.3x Speed up (~6.4ms vs. ~1.2ms)

Single Precision (FP32)



Tensor Cores
APIs

Half Precision (FP16)



An abstract network diagram with green nodes and lines on a dark background. The nodes are represented by small, glowing green circles of varying sizes, and the lines are thin, green, semi-transparent lines connecting the nodes in a complex, web-like pattern. The background is a dark, almost black, gradient with some subtle light effects.

DATA PIPELINE OPTIMIZATION CASE

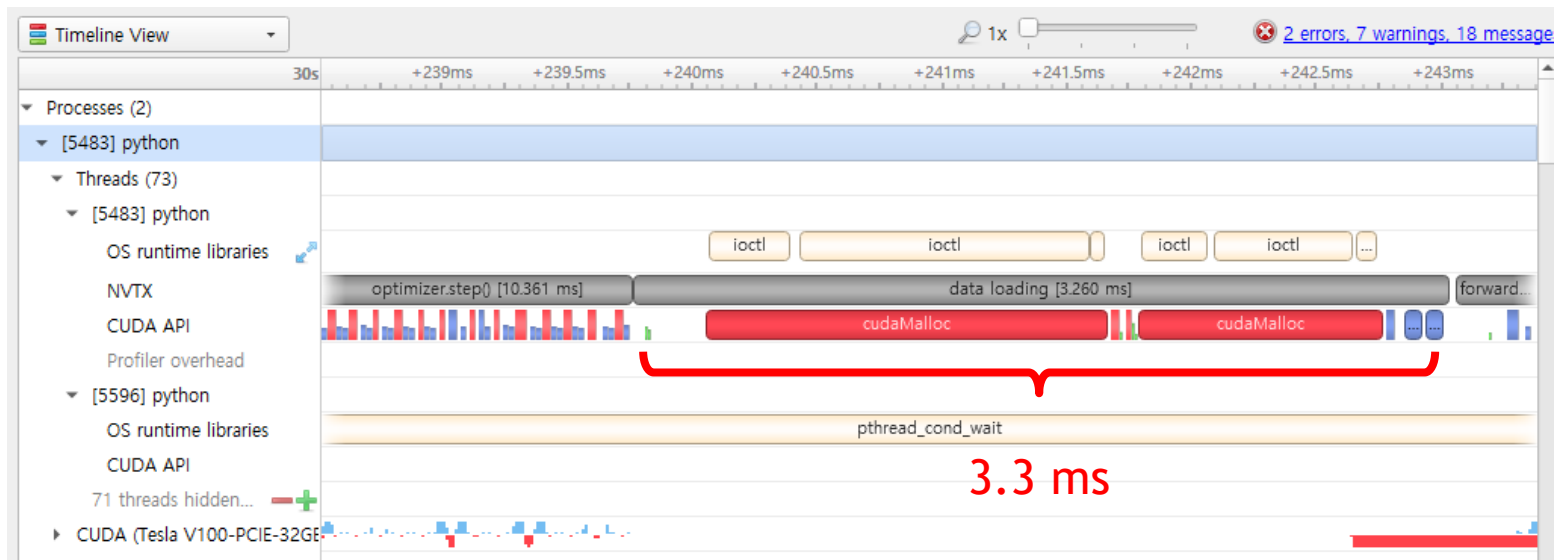
DATA PIPELINE OPTIMIZATION

Naïve data augmentation pipeline



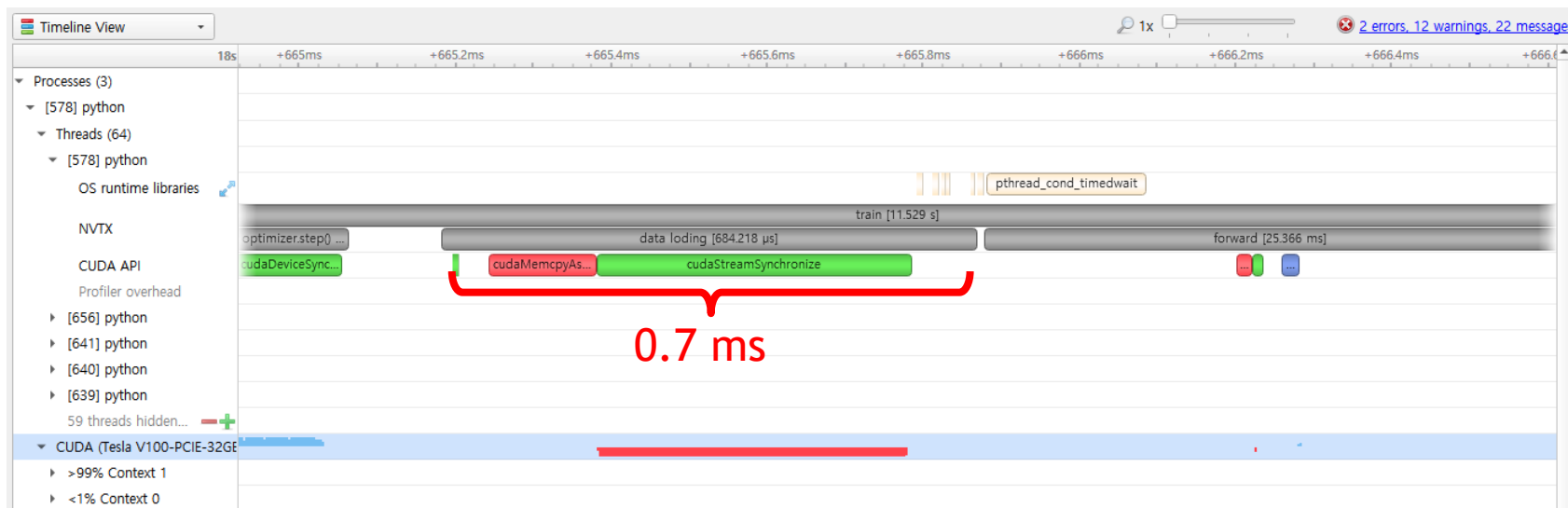
■ CPU

■ GPU



DATA PIPELINE OPTIMIZATION

DALI (Data Loading Library)



4.7x Speed up (3.3ms vs. ~0.7ms)

OPTIMIZATION STRATEGY FOR DL

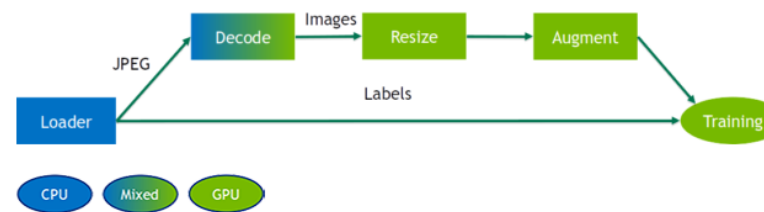
Reminding

- ▶ Algorithm optimization
 - ▶ Tensor Cores
 - ▶ Training: Automatic Mixed Precision
 - ▶ Inference: TensorRT
- ▶ Data pipeline optimization
 - ▶ DALI (Data loading Library)

$$D = \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix} + \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}$$

FP16 or FP32 FP16 FP16 FP16 or FP32

- ▶ Available in **Volta** and **Turing** architecture GPUs
- ▶ 125 Tflops in FP16 vs. 15.7 Tflops in FP32 (8x speed-up)
- ▶ Optimized **4x4x4** dot operation (GEMM)



DALI example pipeline

The background is a dark, almost black, field with a complex network of thin, glowing green lines. These lines intersect at various points, creating a web-like structure. At many of these intersection points, there are small, bright green circular dots or nodes. Some of these dots are slightly larger and more prominent than others. The overall effect is one of a dynamic, interconnected system, possibly representing a network or a data structure. The text 'GETTING STARTED RESOURCES' is positioned in the lower half of the image, centered horizontally.

GETTING STARTED RESOURCES

LEARN MORE

- Nsight System
 - <https://developer.nvidia.com/nsight-systems>
- Official Documentation (Nsight System developer guide)
 - <https://docs.nvidia.com/nsight-systems/>
- Nsight System Blog
 - <https://devblogs.nvidia.com/nsight-systems-exposes-gpu-optimization/>

LEARN MORE DURING AI CONFERENCE

- ▶ 17:20 ~ 18:00 Track2

- ▶ 효율적인 Deep Learning 서비스 구축을 위한 핵심 애플리케이션 - NVIDIA TensorRT Inference Server
by NVIDIA 정소영 상무

RELATED SESSIONS

Automatic Mixed Precision (AMP) for training optimization

▶ 13:00 - 13:40 Track1

- ▶ Tensor Core를 이용한 딥러닝 학습 가속을 쉽게 하는 방법
(Getting more DL Training Acceleration using Tensor Cores and AMP) by NVIDIA 한재근 과장

TensorRT for inference optimization

▶ 13:50 ~ 14:30 Track2

- ▶ Deep Learning inference 가속화를 위한 NVIDIA의 기술 소개 by NVIDIA 이종환 과장

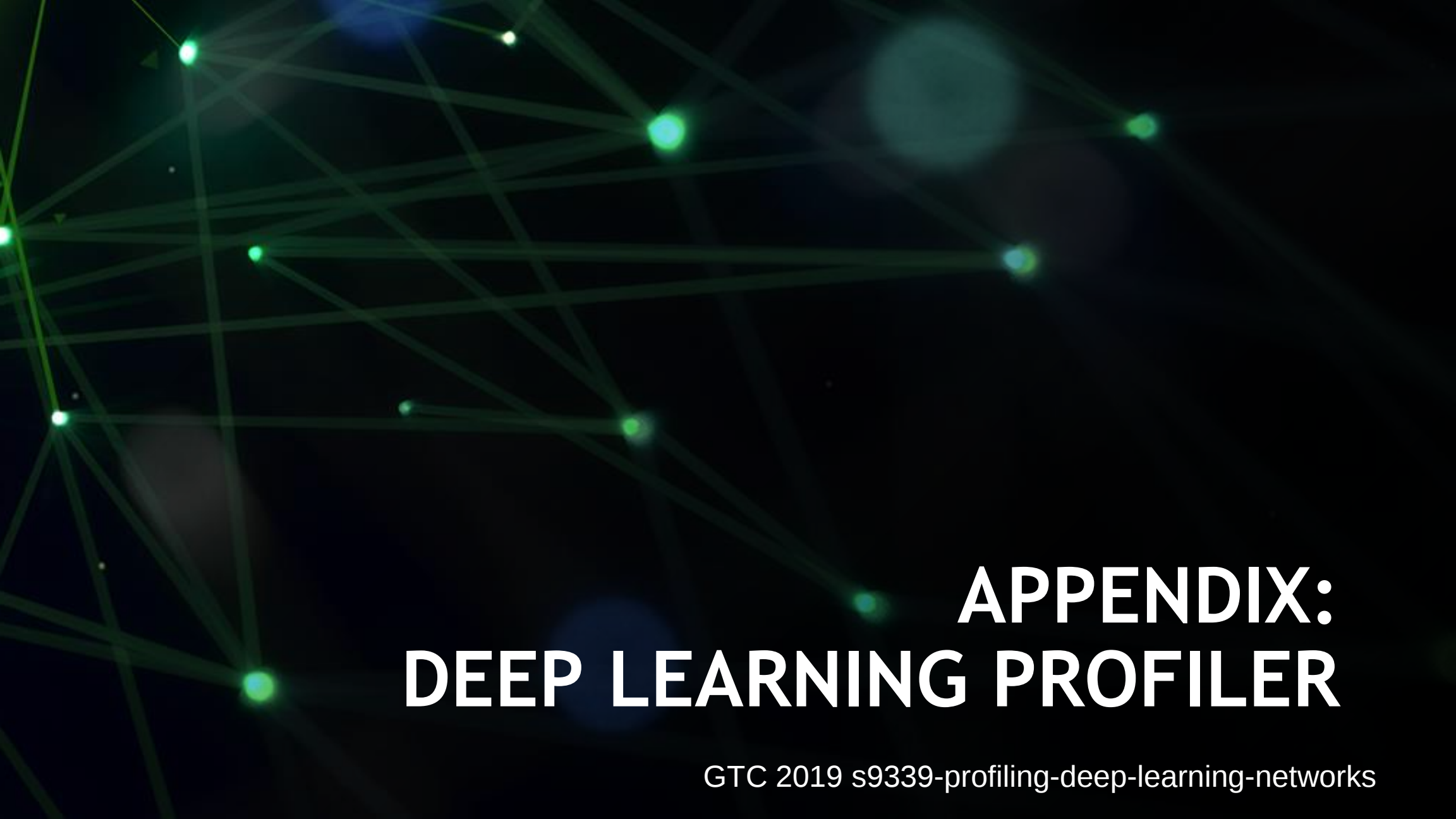
▶ 14:40 ~ 15:20 Track2

- ▶ TensorRT를 이용한 OCR Model Inference 성능 최적화 by KAKAO 이현수

DALI for data pipeline optimization

▶ 15:40 ~ 16:20 Track1

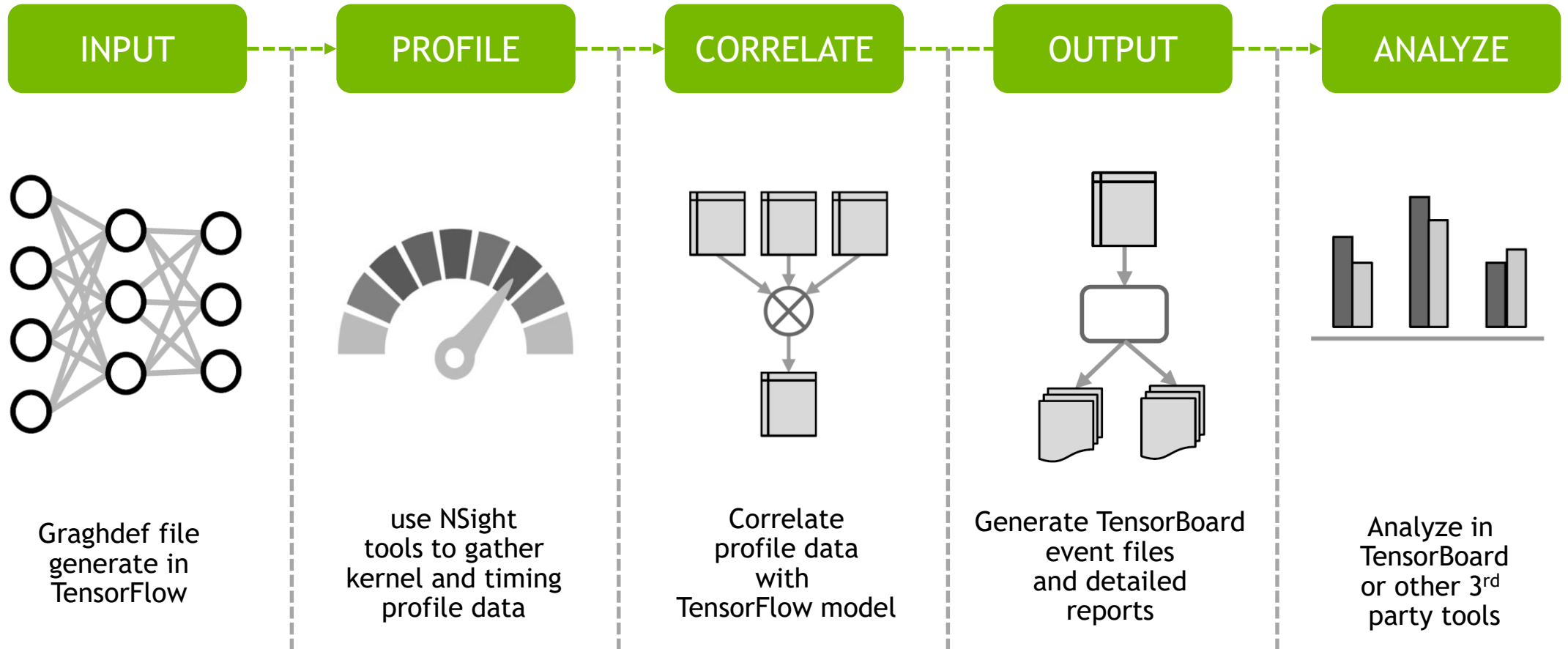
- ▶ GPU를 활용한 Image Augmentation 가속화 방안 - DALI by NVIDIA 한재근 과장

The background of the slide features an abstract network diagram. It consists of numerous small, bright green circular nodes scattered across a dark, almost black, background. These nodes are interconnected by a dense web of thin, light green lines, creating a complex, web-like structure that suggests a neural network or a data graph. The lines vary in opacity and thickness, giving a sense of depth and connectivity.

APPENDIX: DEEP LEARNING PROFILER

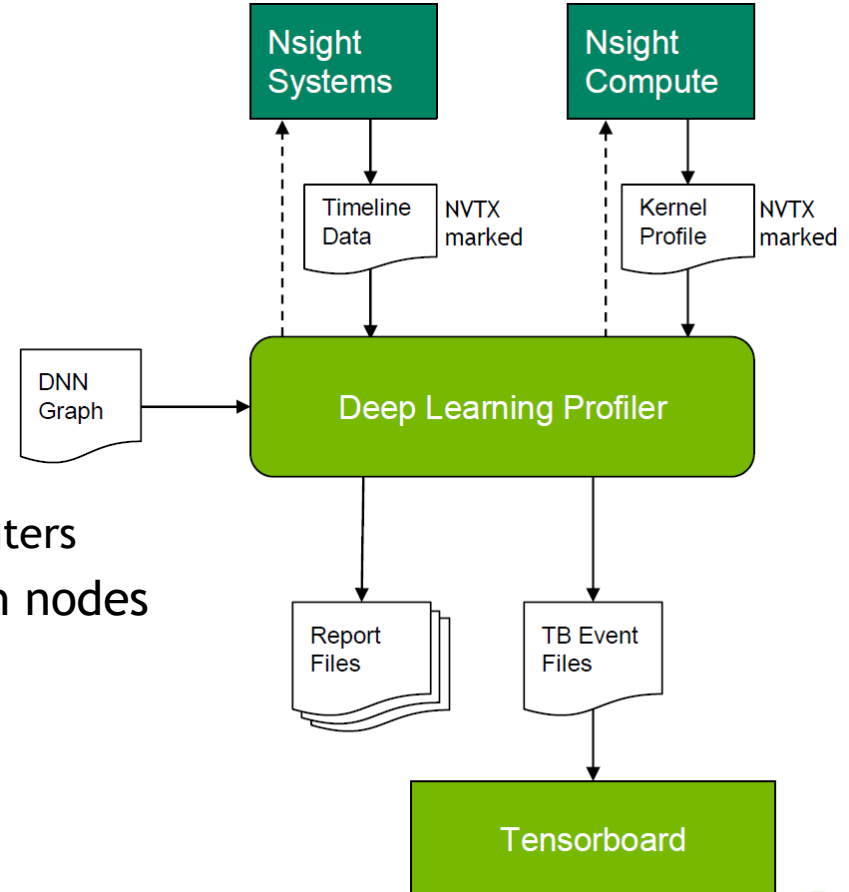
GTC 2019 s9339-profiling-deep-learning-networks

DEEP LEARNING PROFILER WORKFLOW



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 LARNING 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

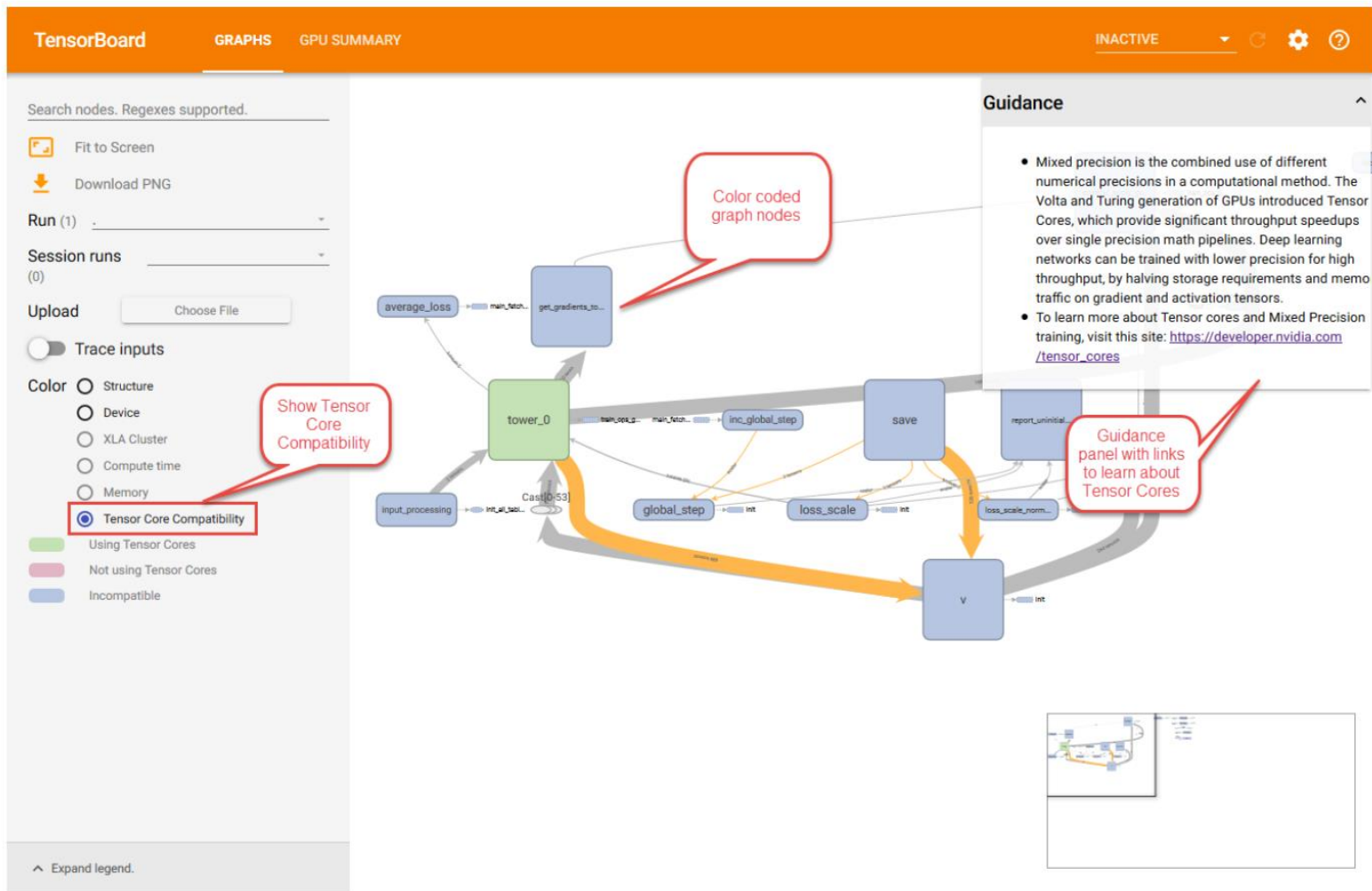
```
$ dlprof --in_graphdef=/results/mobilenet_graph.pb /usr/bin/python tf_cnn_benchmarks.py --num_gpus=1  
--batch_size=8 --model=mobilenet --device=gpu --gpu_indices=1 --data_name=imagenet --  
data_dir=/data/train-val-tfrecord-480 --num_batches=1 --use_fp16 --fp16_enable_auto_loss_scale
```

- ▶ Launching TensorBoard

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

TENSORBOARD MODIFICATIONS

Start TensorBoard with NVIDIA modifications



COMPATIBILITY DETAILS

Select Compatible using Tensor Cores

The screenshot shows the TensorBoard interface with the 'GPU SUMMARY' tab selected. The main graph displays various operations like 'BatchNorm', 'depthwise', and 'expand'. A red circle highlights a '0-to-2D' node, with a callout indicating to 'Click on a node that uses Tensor Cores'. Another callout points to the 'kernel' node, stating 'Show kernel that uses Tensor Cores'. A third callout points to the 'Compatibility Details' panel, saying 'See list of all kernels called'.

Compatibility Details

Duration: 4.78 (ms)
Kernels on this node: 17

Name	Using TC	Duration (µs)	Calls	Avg (µs)	Min (µs)	Max (µs)
volta_fp16_nchw_in_v1	YES	97	13	7	5	9
compute...						
volta_fp16_s884cudnn_fp16_128x128_ldg8_relu_f2f_exp_interio						
cudnn:gemm_ets_paramsj	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	10	4	5	5	5
void fft1d..._nt2, int2)	no	30	4	8	7	8
void fft2d..._half*)	no	19	4	5	5	5
void rft2d..._int, int)	no	37	8	5	4	5
void flip..._int, int)	no	31	4	8	7	9
void im2coo...alf*, int)	no	29	4	7	7	8
void nchwT...at, float)	no	132	13	10	8	11
void tenso...enc_half*)	no	88	11	8	7	9
volta_cgmemm_32x64_tn	no	21	4	5	5	6
volta_fp16_rlor_ne_v1	no	11	2	5	5	6
volta_cgmemm_64x32_nt	no	21	4	5	5	6

Statics on the node runtime

Expand legend.

COMPATIBILITY DETAILS

Select Compatible using Tensor Cores

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

The screenshot shows the TensorBoard interface with an orange header bar containing 'TensorBoard', 'GRAPHS', 'GPU SUMMARY', and 'INACTIVE'. On the left, there are controls for searching nodes, downloading the graph, and selecting session runs. The main area displays a computational graph with nodes like 'depthwise', 'expand', 'kernel', 'BatchNorm', and 'expanded_conv_9'. A red circle highlights a node in the graph, with a callout box saying 'Click on a node that does not use Tensor Cores'. On the right, two panels are visible: 'Compatibility Details' and 'Tensor Core Help'. The 'Compatibility Details' panel shows the duration (521.88 µs) and kernels on this node (2), followed by a table of kernel details. The 'Tensor Core Help' panel contains a list of links and information about Tensor Cores and Mixed Precision training.

TensorBoard GRAPH GPU SUMMARY INACTIVE

Search nodes. Regexes supported.

Fit to Screen

Download PNG

Run (1)

Session runs (0)

Upload Choose File

Trace inputs

Color

- Structure
- Device
- XLA Cluster
- Compute time
- Memory
- Tensor Core Compatibility

Using Tensor Cores

Not using Tensor Cores

Incompatible

Expand legend.

Remove from main graph

Compatibility Details

Duration: 521.88 (µs)
Kernels on this node: 2

Name	Using TC	Duration (µs)	Calls	Avg (µs)	Min (µs)	Max (µs)
void cudnn... int, int)	no	165	11	15	13	16
void tenso...en:half*)	no	85	11	8	7	8

Tensor Core Help

- Please note that if there are multiple kernels being observed on single op node, these are likely performing data transposes to prepare the data for efficient use by tensorcores. Such transposes themselves would not use tensorcores.
- To learn more about Tensor cores and Mixed Precision training, visit this site: https://developer.nvidia.com/tensor_cores.
- You will find resources on how to train networks with mixed precision and make full use of Tensor cores for Tensorflow models https://docs.nvidia.com/deeplearning/sdk/mixed-precision-training/index.html#training_tensorflow.
- Visit this site to find out more about DNN examples optimized and tuned for Tensor cores provided by NVIDIA: <https://developer.nvidia.com/deep-learning-examples>.

OPNODES SUMMARY TAB

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

TensorBoard

GRAPHS GPU SUMMARY

INACTIVE

Search

New GPU Summary Tab

Run (1)

Session runs (0)

Upload Choose File

See a sortable list of all nodes in the model

See a list of all kernels for a selected node

OpNodes GroupNodes Model Summary

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, __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, __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

The screenshot shows the TensorBoard interface with the GPU Summary tab selected. The top navigation bar includes 'TensorBoard', 'GRAPHS', 'GPU SUMMARY', and 'INACTIVE'. Below the navigation bar, there are tabs for 'OpNodes', 'GroupNodes' (selected), and 'Model Summary'. A search bar is present on the left. The main content area displays a table of group nodes with columns for node name, total time, and Tensor Core utilization. Annotations highlight the 'Sort by total time' button and the 'Use statistics to drill down into bottlenecks' button.

Search nodes. Regexes sup...

Fit to screen
Download PNG

Run (1)
Session runs (0)
Upload Choose File

OpNodes GroupNodes Model Summary

1	input_processing/batch_processing	6,179,353	0	0
2	input_processing	6,179,353	0	0
3	input_processing/batch_processing/list_files	6,140,401	0	0
4	tower_0/v	2,585,470	111	97
5	tower_0	2,585,470	111	97
6	tower_0/v/cg	2,097,763	37	31
7	tower_0/v/cg/MobilenetV2	2,097,182	36	31
8	tower_0/v/cg/MobilenetV2/Conv	1,850,516	1	1
9	tower_0/v/gradients	420,913	74	66
10	tower_0/v/gradients/tower_0/v	394,305	74	66
11	tower_0/v/gradients/tower_0	394,305	74	66
12	tower_0/v/gradients/tower_0/v/cg	371,357	74	66
13	tower_0/v/gradients/tower_0/v/cg/MobilenetV2	369,520	72	66
14	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/Logits	63,684	2	2
15	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/Logits/Conv2d_1c_1x1	2,865	2	2
16	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/Logits/Conv2d_1c_1x1/Conv2D_grad	2,425	2	2
17	tower_0/v/cg/MobilenetV2/Logits	3,134	1	1
18	tower_0/v/cg/MobilenetV2/Logits/Conv2d_1c_1x1	1,063	1	1
19	get_gradients_to_apply	45,687	0	0
20	tower_0/v/l2_loss	44,596	0	0
21	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/expanded_conv_16	36,684	4	4
22	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/expanded_conv_12	32,883	4	4
23	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/Conv_1	27,752	2	2
24	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/Conv_1/Conv2D_grad	27,046	2	2
25	tower_0/v/gradients/tower_0/v/cg/MobilenetV2/expanded_conv_11	24,602	4	4
26	v/cg	24,131	0	0

Sort by total time

Use statistics to drill down into bottlenecks

List Group Nodes in model

MODEL SUMMARY TABLE

Model Summary shows concise information on Tensor core usage

TensorBoard GRAPHS GPU SUMMARY

Search nodes. Regexes sup...

 Fit to screen

 Download PNG

Run (1)

Session runs (0)

Upload

Choose File

☐ OpNodes ☐ GroupNodes ☒ Model Summary

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%



nVIDIA®