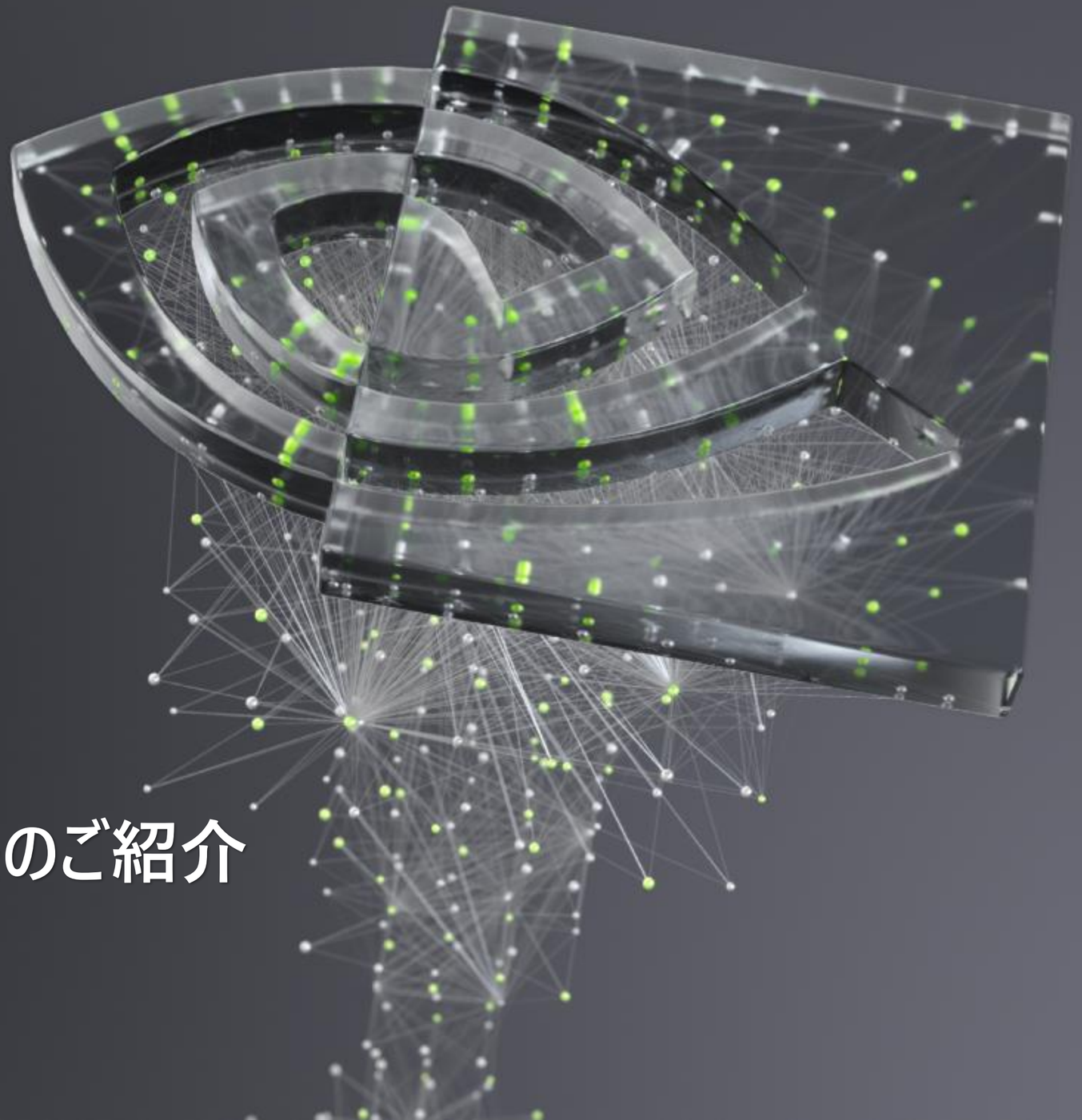


NVIDIA プロファイラを用いた PYTORCH 学習最適化手法のご紹介

2021.08.30

Mana Murakami, Solution Architect , NVIDIA

AGENDA

1. プロファイリングの重要性について
2. DLProf & Nsight Systems
3. まとめ

よくあるご質問

プロファイリングの重要性について

- GPU を学習に使用したら速くなったが、これ以上速くなるか分からない
- GPU を学習にしようしているが、GPU がどの程度使われているのかよく分からない
- そもそも最適化のステップが分からない

よくあるご質問

プロファイリングの重要性について

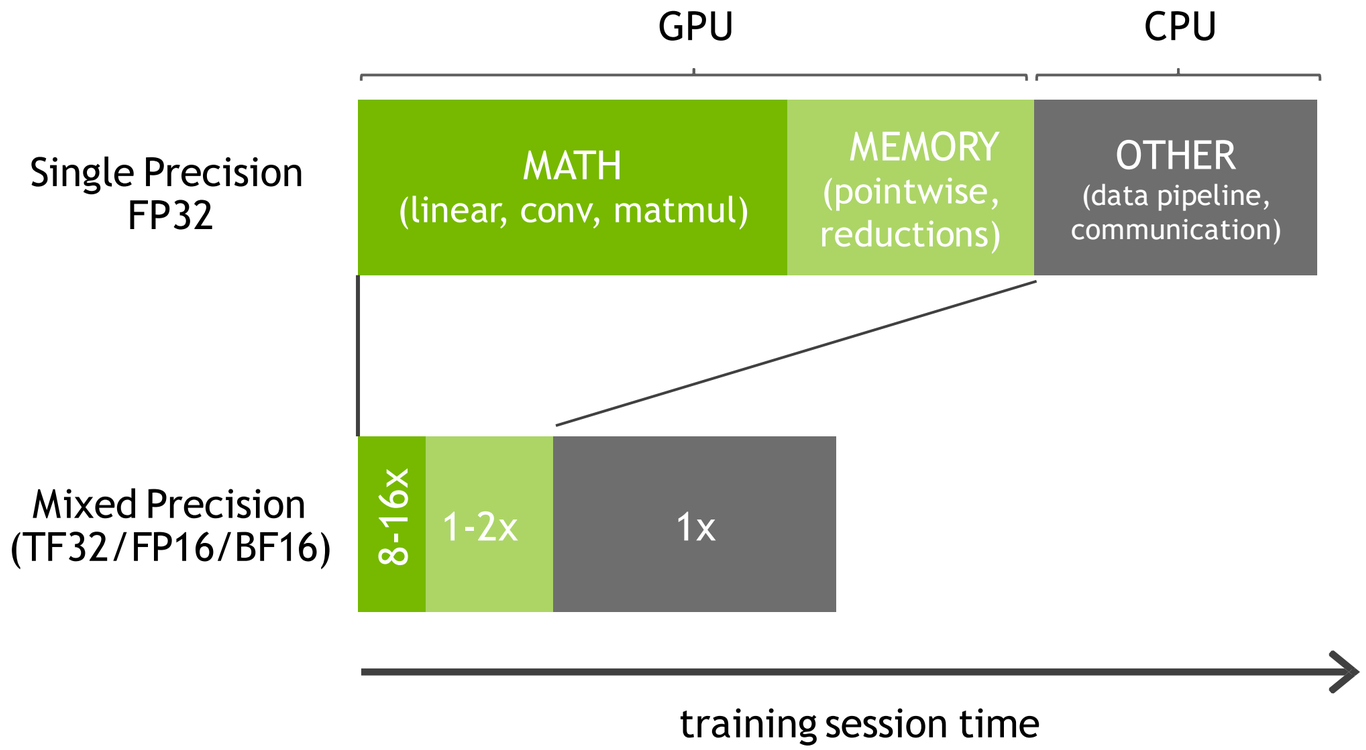
- GPU を学習に使用したら速くなったが、これ以上速くなるか分からない
- GPU を学習に
- そもそも最適

ボトルネック解析の為の便利なツールが
いくつか存在します

パフォーマンス最適化の限界

プロファイリングの重要性について

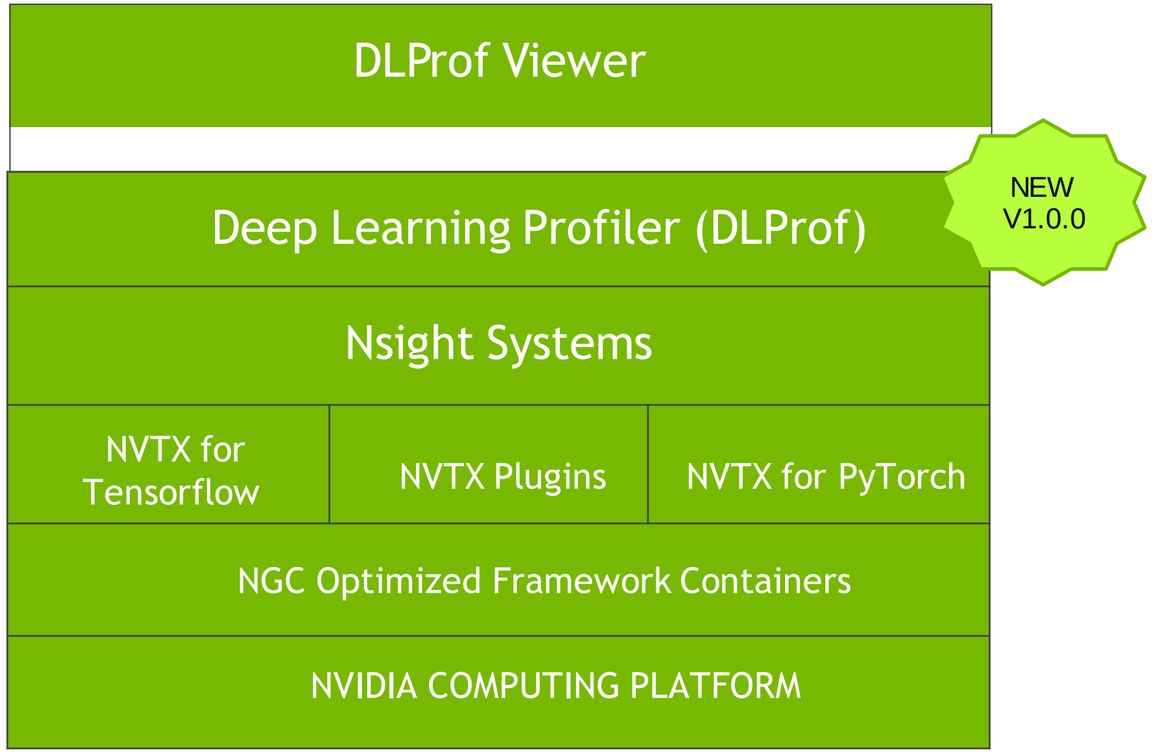
アムダールの法則: トレーニングセッションの一部 (GPU で動作) を高速化すると、残りの部分 (CPU で動作) が性能ボトルネックになる



NVIDIA プロファイリング スタック

用途毎に使い分け可能な階層型プロファイル スタック

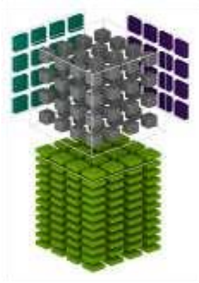
- **Nsight Systems** と **Nsight Compute** は CUPTI (Profiling Tools Interface) ベースの GPU アプリケーションの為のプロファイラ
- **NVTX (NVIDIA Tools Extension Library)** はソースコードにアノテーションをする為の CUDA ライブラリ
- **DLProf** は内部で **Nsight Systems** を実行してプロファイルデータを収集し、データサイエンティストが分かりやすい形に整形して可視化



性能最適化の為の便利なツール

DL Prof と Nsight Systems

研究者と開発者



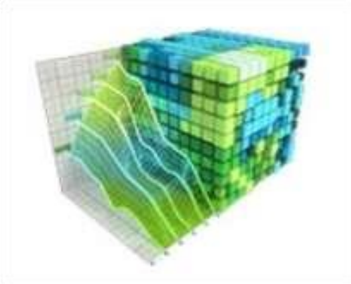
NVTX

Nsight Systems

Nsight Compute

アルゴリズム開発者

データサイエンティストと応用研究者



DLProf

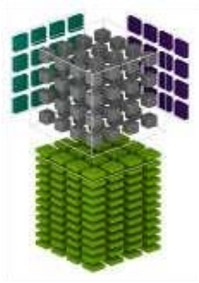
<Nsight Systems w/ NVTX>

特定のドメイン向けのモデル開発やアプリケーション開発者

性能最適化の為の便利なツール

DL Prof と Nsight Systems

研究者と開発者



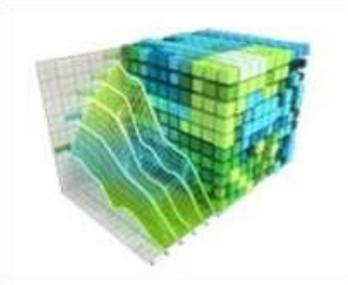
NVTX

Nsight Systems

Nsight Compute

アルゴリズムやフレームワーク開発者の為のプロファイルツール
オーバーヘッドも低く軽量でCUDA処理の流れを
細かく把握する事ができる

データサイエンティストと応用研究者



DLProf

<Nsight Systems w/ NVTX>

特定の
ア

解析結果をデータサイエンティストが理解しやすい形に
整形・可視化して学習コードの最適化を支援

DLProf

DLProf とは?

DLProf :ディープラーニングモデルの為のプロファイリングツール

TensorFlow, PyTorch, TensorRT をサポート

解析結果や最適化のアドバイスを表示

DLProf とは?

ダッシュボード

- **GPU 使用率チャート**
  - wall clock time のうち GPU がアクティブになっている割合の表示、複数 GPU の場合すべての GPU の平均利用率を示す
- **オペレーション GPU 時間チャート：**
  - すべてのオペレーションを「Tensor コアを使用した処理」「Tensor コア使用できたが使用しなかった処理」「Tensor コアを使用する事が出来ない処理」の 3つに分類してチャートを表示
- **CUDA カーネルの GPU 時間チャート:**
  - 全 CUDA カーネル実行時間を「カーネル内で Tensor コアを使用した時間」「カーネル内でメモリ処理を行っていた時間」「カーネル内のその他すべての処理」の 3つに分類してチャートを表示
- **Tensor コア使用率チャート**
  - Tensor コアを使用した処理の全 GPU 時間に対する割合をチャートで表示

DLProf とは?

ダッシュボード

- **性能 サマリー:**
  - 実行時に重要な主要指標を一覧として表示 (実行時間、Tensor コア使用率、GPU 使用率など)
- **イテレーション サマリー:**
  - 実行中に各イテレーションでかかった時間を示す棒グラフ。Tensor コアを使用した時間、Tensor コア以外でGPUを使用した時間、GPU を使用していない時間のインテレーション毎の内訳が表示される。
- **トップ 10 GPU オペレーション :**
  - 実行時間がかかっている上位10オペレーションをソートして表示。ボトルネックになっている箇所の特定に有効

DLProf のインストール

DLProf を使うには?

1. NGC 上で配布されている TensorFlow および PyTorch コンテナに同梱されている DLProf を使う (PyTorch と TensorFlow (1.x/2.x))
    - TensorFlow https://ngc.nvidia.com/catalog/containers/nvidia:tensorflow
    - PyTorch https://ngc.nvidia.com/catalog/containers/nvidia:pytorch

2. Python pip 経由のインストール (PyTorch と TF1.x のみ)
    - PyTorch の例: (py Index、DLProf および依存パッケージ、DLProf Viewer Plugin for TensorBoard のインストール)

```
$ pip install nvidia-pyindex
$ pip install nvidia-dlprof[pytorch]
$ pip install nvidia-tensorboard-plugin-dlprof
```

DLProf のインストール

DLProf を使うには?

1. NGC 上で配布されている TensorFlow および PyTorch コンテナに同梱されている DLProf を使う (PyTorch と TensorFlow (1.x/2.x))
    - TensorFlow https://ngc.nvidia.com/catalog/containers/nvidia:tensorflow
    - PyTorch https://ngc.nvidia.com/catalog/containers/nvidia:pytorch

NOTE:
NGCで配布されているDeep Learningコンテナを Singularityで動かす方法はAppendix.のブログを参照のこと

各コンテナに同梱されている DLProf のバージョンは以下のドキュメントで確認可能
https://docs.nvidia.com/deeplearning/frameworks/pytorch-release-notes/

2. Python pip 経由のインストール
    - PyTorch の例: (py Inc

```
$ pip install nvidia-pyindex
$ pip install nvidia-dlprof[pytorch]
$ pip install nvidia-tensorboard-plugin-dlprof
```

DLProf のインストール

DLProf を使うには?

1. NGC 上で配布されている TensorFlow および PyTorch コンテナに同梱されている DLProf を使う (PyTorch と TensorFlow (1.x/2.x))
   - TensorFlow https://ngc.nvidia.com/catalog/containers/nvidia:tensorflow
   - PyTorch https://ngc.nvidia.com/catalog/containers/nvidia:pytorch

2. Python pip 経由のインストール (PyTorch と TF1.x のみ)
   - PyTorch の例: (py Index、DLProf および依存パッケージ、DLProf Viewer Plugin for TensorBoard のインストール)

```
$ pip install nvidia-pyindex
$ pip install nvidia-dlprof[pytorch]
$ pip install nvidia-tensorboard-plugin-dlprof
```

NOTE:
CUDA toolkit および driver と依存関係がある為、構築環境の CUDA バージョンと互換性があるバージョンを入れる必要がある
(参考)
https://docs.nvidia.com/deeplearning/frameworks/dlprof-release-notes/index.html

PyTorch スクリプトのプロファイル手順

DLProf を使うには?

1. プロファイル対象の PyTorch コードに以下を追加

```
import nvidia_dlprof_pytorch_nvtx as nvtx
nvtx.init(enable_function_stack=True)
```

```
with torch.autograd.profiler.emit_nvtx():
     for iter in range(iters):
          #forward
          #backward
```

2. DLProf の実行

```
$ dlprof --mode=pytorch python main.py
```

3. DLProfViewer による結果の可視化

```
$ dlprofviewer -b 0.0.0.0 -p 8000 dlprof_dldb.sqlite
```

References:https://docs.nvidia.com/deeplearning/frameworks/dlprof-user-guide/index.html#quickstart_topic

PyTorch スクリプトのプロファイル手順

DLProf を使うには?

1. プロファイル対象の PyTorch コードに以下を追加

```
import nvidia_dlprof_pytorch_nvtx as nvtx
nvtx.init(enable_function_stack=True)
```

3行追加するだけ

```
with torch.autograd.profiler.emit_nvtx():
     for iter in range(iters):
          #forward
          #backward
```

2. DLProf の実行

```
$ dlprof --mode=pytorch python main.py
```

3. DLProfViewer による結果の可視化

```
$ dlprofviewer -b 0.0.0.0 -p 8000 dlprof_dldb.sqlite
```

References:https://docs.nvidia.com/deeplearning/frameworks/dlprof-user-guide/index.html#quickstart_topic

PyTorch スクリプトのプロファイル手順

DLProf を使うには?

1. プロファイル対象の PyTorch コードに以下を追加

```
import nvidia_dlprof_pytorch_nvtx as nvtx
nvtx.init(enable_function_stack=True)
```

```
with torch.autograd.profiler.emit_nvtx():
     for iter in range(iters):
          #forward
          #backward
```

DLProf の解析は時間がかかる為、イテレーション数を少なくするのが良い (10~20 mini-batchくらい)
--delay オプションを付けて warmup 部をスキップしてプロファイルする事も可能

2. DLProf の実行

```
$ dlprof --mode=pytorch python main.py
```

3. DLProfViewer による結果の可視化

```
$ dlprofviewer -b 0.0.0.0 -p 8000 dlprof_dldb.sqlite
```

References:https://docs.nvidia.com/deeplearning/frameworks/dlprof-user-guide/index.html#quickstart_topic

PyTorch スクリプトのプロファイル手順

DLProf を使うには?

1. プロファイル対象の PyTorch コードに以下を追加

```
import nvidia_dlprof_pytorch_nvtx as nvtx
nvtx.init(enable_function_stack=True)
```

```
with torch.autograd.profiler.emit_nvtx():
     for iter in range(iters):
          #forward
          #backward
```

2. DLProf の実行

```
$ dlprof --mode=pytorch pytl
```

特にファイル名を指定せずに実行した場合、「dlprof_dldb.sqlite」と「nsys_profile.sqlite」が出力される
Dlprofviewerには「dlprof_dldb.sqlite」を指定

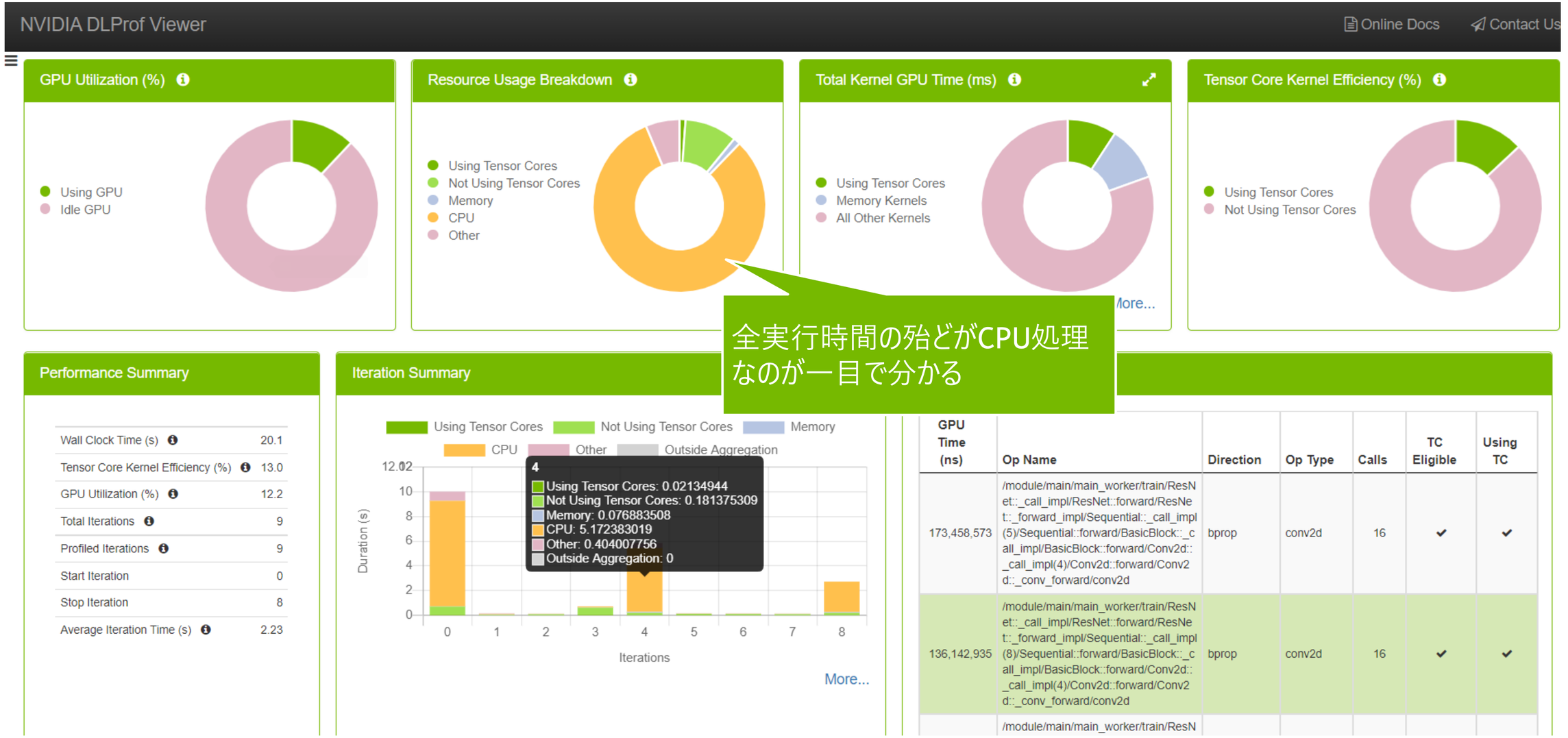
3. DLProfViewer による結果の可視化

```
$ dlprofviewer -b 0.0.0.0 -p 8000 dlprof_dldb.sqlite
```

References:https://docs.nvidia.com/deeplearning/frameworks/dlprof-user-guide/index.html#quickstart_topic

例: DLProf + DLProfViewer によるプロファイル結果

GPU 最適化前 (AMP未使用/バッチサイズ小)



例: DLProf + DLProfViewer によるプロファイル結果

GPU 最適化前 (AMP未使用/バッチサイズ小)

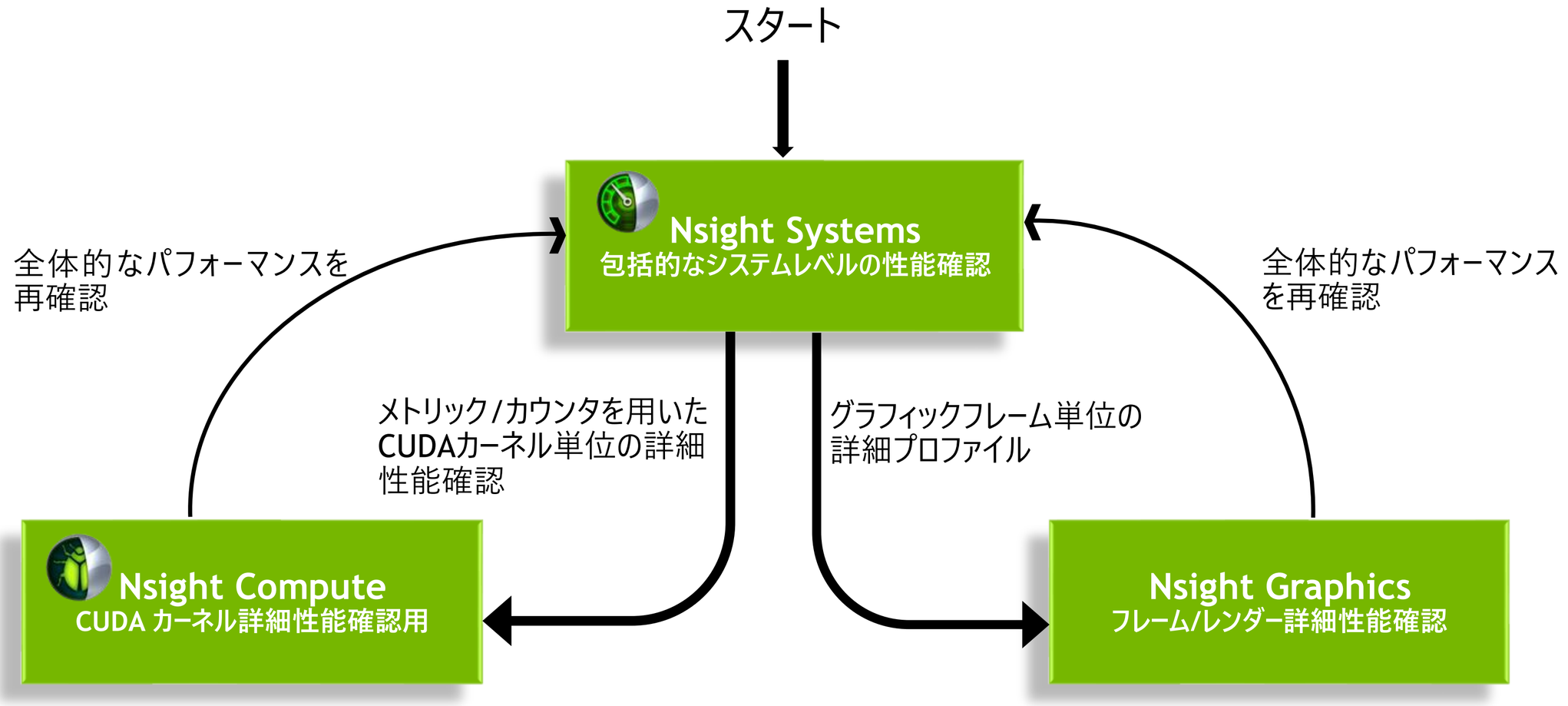
```
[DLProf-03:53:19] Aggregating profile data
[DLProf-03:53:19] Creating dlprof database at ./dlprof_dldb.sqlite
[DLProf-03:53:19] Writing profile data to dlprof database
[DLProf-03:53:22] Writing aggregated data to dlprof database
[DLProf-03:53:25] Writing expert_systems report to (stdout)
Expert Systems Feedback: 4 issues detected. Note that expert systems is still experimental as are all recommended changes

Problem detected:
  40 ops were eligible to use tensor cores but none are using FP16
Recommended change:
  Try enabling AMP (Automatic Mixed Precision). For more information: https://developer.nvidia.com/automatic-mixed-precision

Problem detected:
  45.6% of the aggregated run was spent in the dataloader while not simultaneously runnin
Recommended change:
  Focus on reducing time spent in the training data input process. This could be time spe                                    r.
Consider using NVIDIA DALI, a library that is a high performance alternative to built-in                                    r.nvidia.com/
DALI

Problem detected:
  Slow debug APIs were enabled. When not debugging, these APIs can slow down execution of your model
Recommended change:
  Do not use the record_function decorator or context manager unless debugging.


Problem detected:
  GPU Memory is underutilized: Only 31% of GPU Memory is used
Recommended change:
  Try increasing batch size by 2x to increase data throughput
```

“Problem detected:”と”Recommended Change:”が表示され、問題点が分かる

NVIDIA NSIGHT SYSTEMS

NSIGHT ツールワークフロー

新しくなった CUDA プロファイルツール群



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

Nsight Systems

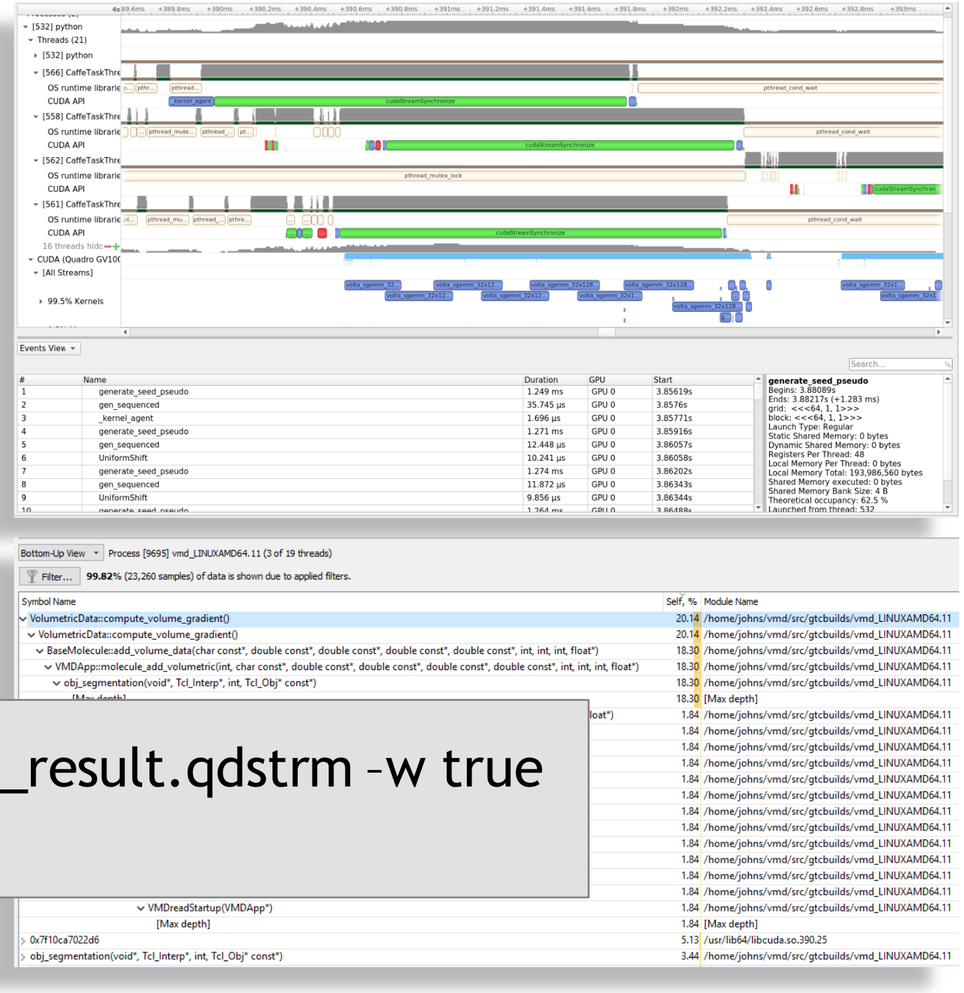
新しくなった CUDA プロファイルツール群

主な機能:

- システム全体のアルゴリズム最適化
  - マルチプロセスのアプリケーション解析のサポート
- アプリケーション内のボトルネックを探しに有効
  - 非常に高速なGUIタイムライン上で何百万ものイベントを視覚化
- コマンドライン、IDE(統合型開発環境)の両方に対応

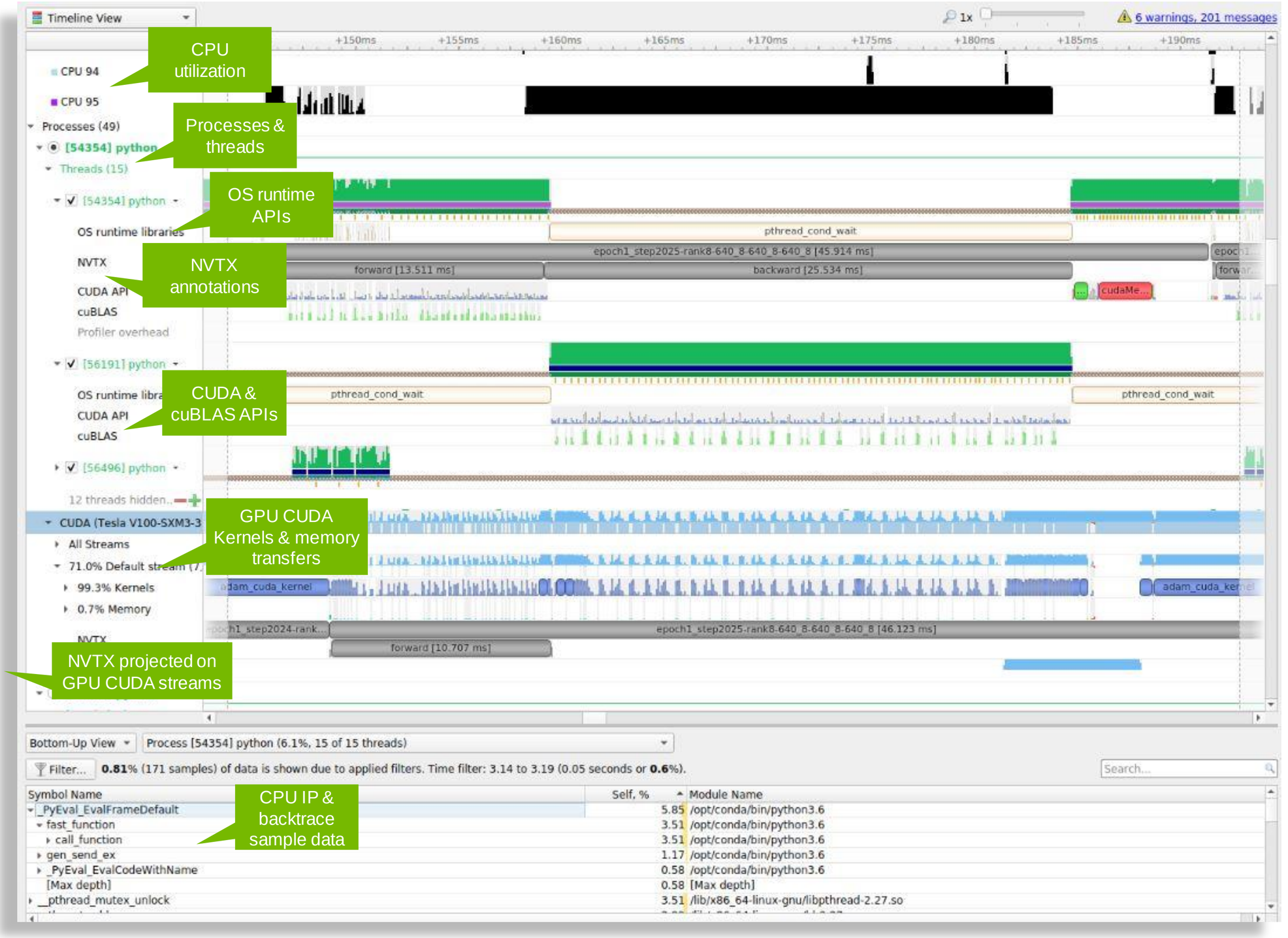
OS: Linux (x86, Power, Arm SBSA, Tegra), Windows, MacOSX (host)

GPUs: Pascal+




```
$ nsys profile -t cuda,osrt,nvtx,cudnn,cublas  -o inference_result.qdstrm -w true
python inference.py
```

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


Timeline View
1x
6 warnings, 201 messages
+150ms
+155ms
+160ms
+165ms
+170ms
+175ms
+180ms
+185ms
+190ms
CPU 94
CPU 95
Processes (49)
[54354] python
Threads (15)
[54354] python
OS runtime libraries
NVTX
CUDA API
cuBLAS
Profiler overhead
[56191] python
OS runtime libra
CUDA API
cuBLAS
[56496] python
12 threads hidden...
CUDA (Tesla V100-SXM3-3
All Streams
71.0% Default stream (7
99.3% Kernels
0.7% Memory
NVTX
pthread_cond_wait
epoch1_step2025-rank8-640_8-640_8-640_8 [45.914 ms]
forward [13.511 ms]
backward [25.534 ms]
cudaMe...
pthread_cond_wait
pthread_cond_wait
adam_cuda_kernel
adam_cuda_kernel
epoch1_step2024-rank...
epoch1_step2025-rank8-640_8-640_8-640_8 [46.123 ms]
forward [10.707 ms]
CPU utilization
Processes & threads
OS runtime APIs
NVTX annotations
CUDA & cuBLAS APIs
GPU CUDA Kernels & memory transfers
NVTX projected on GPU CUDA streams
Bottom-Up View
Process [54354] python (6.1%, 15 of 15 threads)
Filter...
0.81% (171 samples) of data is shown due to applied filters. Time filter: 3.14 to 3.19 (0.05 seconds or 0.6%).
Search...
Symbol Name
Self, %
Module Name
_PyEval_EvalFrameDefault 5.85 /opt/conda/bin/python3.6
fast_function 3.51 /opt/conda/bin/python3.6
call_function 3.51 /opt/conda/bin/python3.6
gen_send_ex 1.17 /opt/conda/bin/python3.6
_PyEval_EvalCodeWithName 0.58 /opt/conda/bin/python3.6
[Max depth] 0.58 [Max depth]
__pthread_mutex_unlock 3.51 /lib/x86_64-linux-gnu/libpthread-2.27.so
CPU IP & backtrace sample data

開発環境に NSIGHT SYSTEMS がインストールされていない場合

NSIGHT SYTEMS

Setting Up and Using Nsight Systems Inside Containers

CUDA 11.4: install

```
$ apt-get update –y
$ apt-get install -y cuda-nsight-systems-11-4 nsight-systems-2021.2.4
```

CUDA 11.3: install

```
$ apt-get update –y
$ apt-get install -y cuda-nsight-systems-11-3 nsight-systems-2021.1.3
```

CUDA 11.2: install

```
$ apt-get update –y
$ apt-get install -y cuda-nsight-systems-11-2 nsight-systems-2020.4.3
```

Mapping an Nsight Systems Host Installation into a Container

```
$ docker run --rm -it --network=host --gpus=all -v /opt/nvidia/nsight-systems/2021.1.3:/opt/nvidia/nsight-systems/2021.1.3          \
  nvcr.io/nvidia/pytorch:21.08-py3 bash
```

NSIGHT SYSTEMS を使うには?

NSIGHT SYTEMS

Example

```
$ nsys profile -t nvtx,cuda,osrt,cublas        \
               --stats=true                    \
               -f true                         \
               -o pusch_result                 \
               python main.py
```

← APIs to be traced
← Outputs profiling information similar to nvprof
← Overwrite the output
← Output filename

cuda – GPU kernel
osrt – OS runtime
nvtx – NVIDIA Tools Extension
cublas – CUDA BLAS library

https://docs.nvidia.com/nsight-systems/2020.3/profiling/index.html#cli-options

NSIGHT SYSTEMS を使うには?

NSIGHT SYTEMS

Example

```
$ nsys profile -t nvtx,cuda,osrt,cublas        \
               --stats=true                    \
               -f true                         \
               -o pusch_result                 \
               python main.py
```

⟸ APIs to be traced
⟸ Outputs profiling information similar to nvprof
⟸ Overwrite the output

Other Userful Options

- --delay (-y) : Collection start delay in seconds
- --duration(-d): Collection duration in seconds.
- --capture-range(-c): none/cudaProfilerApi/nvtx

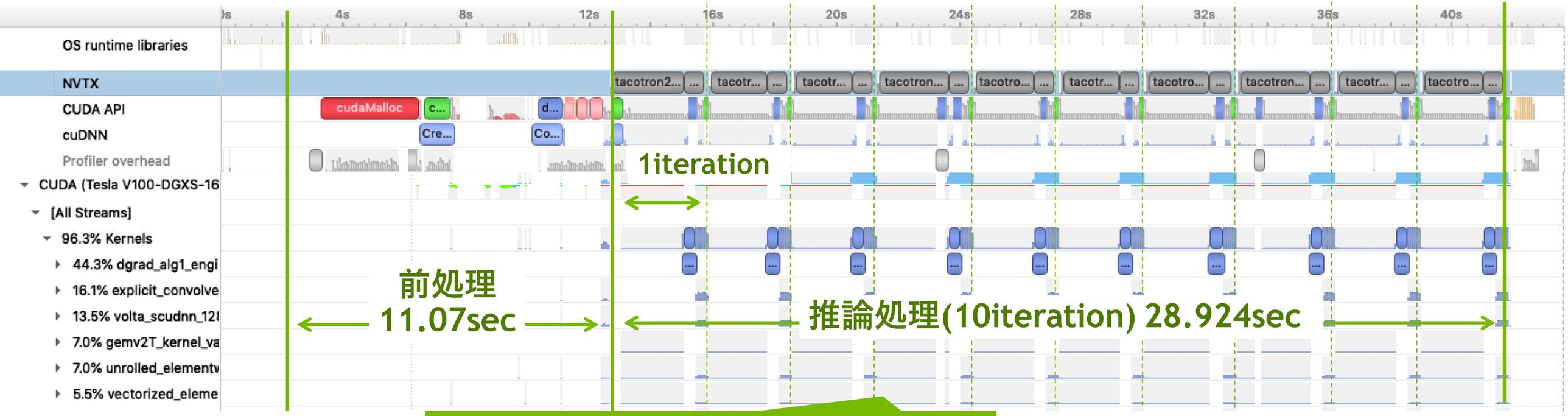
etc..

cuda – GPU kernel
osrt – OS runtime
nvtx – NVIDIA Tools Extension
cublas – CUDA BLAS library

https://docs.nvidia.com/nsight-systems/2020.3/profiling/index.html#cli-options

例: Nsight Systems + NVTX

Nsight Systems プロファイル結果(NVTX あり)



アノテーションする事で
タイムライン上で処理を把握しやすくなる！

Appendix. 技術ブログ・関連セッション

NVIDIA プロファイラを用いた Pytorch 学習最適化手法のご紹介

Deep Learning Examples

- https://github.com/NVIDIA/DeepLearningExamples/

How to Run NGC Deep Learning Containers with Singularity

- https://developer.nvidia.com/blog/how-to-run-ngc-deep-learning-containers-with-singularity/

Profiling and Optimizing Deep Neural Networks with DLProf and PyProf (TensorFlow)

- https://developer.nvidia.com/blog/profiling-and-optimizing-deep-neural-networks-with-dlprof-and-pyprof/

Deep Learning Performance Optimization with Profiling Tools

- https://www.nvidia.com/en-us/on-demand/session/gtcspring21-s31228/

Profiling and Optimizing Deep Neural Networks with DLProf and PyProf

- https://www.nvidia.com/en-us/on-demand/session/gtcspring21-s31341/

PyTorch Performance Tuning Guide

- https://www.nvidia.com/en-us/on-demand/session/gtcspring21-s31831/


THANK YOU!
NVIDIA