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
Optimal data processing, batching, copying, managing pipeline

Maximum parallel computation, fast matrix operations, precision

System wide tracing, thread synchronization, memory transfers, Kernel tuning

Layer fusion, calibrating, optimized inference

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

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 is the most popular visualization tool 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

https://developer.nvidia.com/nsight-systems
Processes and threads
CUDA and OpenGL API trace
cuDNN and cuBLAS trace
Kernel and memory transfer activities
Multi-GPU

NVTX Ranges
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
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='prob')
    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

```python
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**
- Graphdef file
- generate in Tensorflow

**PROFILE**
- Use NSight tools to gather kernel and timing profile data

**CORRELATE**
- Correlate profile data with Tensorflow model

**OUTPUT**
- Generate TensorBoard event files and detailed reports

**ANALYZE**
- 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
- Timeline Data
- NVTX marked
- Kernel Profile
- NVTX marked

DsNN Graph

Report Files

TB Event Files

Tensorboard
Deep Learning Profiler

Command Line Example

Example command to profile MobileNet V2 and generate a graphdef

```bash
$ /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

```bash
$ 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

```bash
$ tensorboard --logdir ./event_files
```
Compatibility Details
Select Compatible using Tensor Cores
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

- New GPU Summary Tab
- See a sortable list of all nodes in the model
- See a list of all kernels for a selected node
Group Node Summary Tab

Roll up timing metrics and Tensor Core utilization per group node
Model Summary Table

Model Summary shows concise information on Tensor core usage

<table>
<thead>
<tr>
<th>Search nodes. Regexes sup...</th>
<th>Fit to screen</th>
<th>Download PNG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>(1)</td>
<td></td>
</tr>
<tr>
<td>Session runs (0)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Nodes in Graph</th>
<th>6158</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compatible Nodes</td>
<td>112</td>
<td></td>
</tr>
<tr>
<td>Compatible Nodes using Tensor Cores</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>Total GPU Time</td>
<td>15.10 (s)</td>
<td></td>
</tr>
<tr>
<td>% of time in TC compatible nodes</td>
<td>15.62%</td>
<td></td>
</tr>
</tbody>
</table>
THANK YOU