OpenVINO™ バックエンドを使用して NVIDIA* Triton Inference Server* でモデルをサービングする方法

この記事は、Medium に公開されている「How to Serve Models on NVIDIA Triton Inference Server* with OpenVINO Backend」の日本語参考訳です。原文は更新される可能性があります。原文と翻訳文の内容が異なる場合は原文を優先してください。

Triton Inference Server* はオープンソースのソフトウェアで、モデルサービングによりマシンラーニング・モデルを最適化およびデプロイするために使用されます。OpenVINO™ はオープンソースのツールキットで、インテル® アーキテクチャー上でディープラーニング・モデルを最適化およびデプロイするように設計されています。Triton Inference Server* を NVIDIA* GPU ですでに活用している組織は、Triton Inference Server* と OpenVINO™ バックエンドを統合することにより、シームレスにインテルのハードウェアに移行して、OpenVINO™ の恩恵を受けることができます。OpenVINO™ モデルサーバー(英語)に完全に移行する必要はありません。

Triton Inference Server* と OpenVINO™ バックエンドを統合することで、組織はインテルのハードウェアのハードウェア・アクセラレーション機能を最大限に活用して、高レベルの推論パフォーマンスを実現できます。これは、Bfloat16(BF16)精度をサポートする、インテル® アドバンスト・マトリクス・エクステンション(インテル® AMX)を搭載した第 4 世代および第 5 世代のインテル® Xeon® スケーラブル・プロセッサーで特に顕著です。

BF16 圧縮形式では、パフォーマンスと効率が向上します。16 ビット形式でデータを表現することにより、インテル® Xeon® プロセッサーは、従来の 32 ビット浮動小数点演算に比べて大幅に高いスループットで、行列乗算などの負荷の高い計算を実行できます。これにより、並列処理が容易になり、メモリー帯域幅の要件が軽減され、電力効率が向上して、リソース使用率が最適化されます。

OpenVINO™ バックエンドを使用することにより、組織はインテルのハードウェアの機能と OpenVINO™ の最適化を活用して、新たな可能性を切り拓くことができます。これらはすべて、Triton Inference Server* のモデル・サービング・プラットフォームにより処理されます。

OpenVINO™ バックエンドを使用して Triton Inference Server* でモデルをサービング

この記事では、モデルのダウンロードと準備から、クライアントからサーバーへの推論リクエストの送信まで、OpenVINO™ バックエンドを使用して Triton Inference Server* でモデルをデプロイする方法を説明します。取り上げるモデルは、ONNX*、PyTorch*(.bin および .xml の OpenVINO™ 形式)、TensorFlow* です。

次のチュートリアルは、Triton Inference Server* GitHub* の既存のチュートリアル、NVIDIA* ONNX* チュートリアル(英語)、NVIDIA* PyTorch* チュートリアル(英語)、NVIDIA* TensorFlow* チュートリアル(英語)をベースとしていますが、OpenVINO™ バックエンドとインテル® CPU を使用するように変更されています。NVIDIA* Triton Inference Server* のドキュメントは、こちら(英語)を参照してください。

テスト環境

Ubuntu* 20.04.6 LTS

Docker* バージョン 26.1.0、ビルド 9714adc

Triton Inference Server* 24.04(バージョンが異なる場合は、コマンドの 24.04 を置き換えてください)

要件

Docker(英語)をインストールする

wget をインストールする:

```
sudo apt install wget
```

ONNX* モデルのデプロイ

1. モデル・リポジトリーを構築して、ONNX* モデルをダウンロードします。

```
mkdir -p model_repository/densenet_onnx/1


wget -O model_repository/densenet_onnx/1/model.onnx \
https://contentmamluswest001.blob.core.windows.net/content/14b2744cf8d6418c87ffddc3f3127242/9502630827244d60a1214f250e3bbca7/08aed7327d694b8dbaee2c97b8d0fcba/densenet121-1.2.onnx
```

2. config.pbtxt という名前の新しいファイルを作成します。

```
name: "densenet_onnx"
backend: "openvino"
default_model_filename: "model.onnx"
```

3. config.pbtxt ファイルをモデル・リポジトリーに配置します。フォルダー構造を次に示します。

```
model_repository
|
+-- densenet_onnx
    |
    +-- config.pbtxt
    +-- 1
        |
        +-- model.onnx
```

注: Triton Inference Server* は、このフォルダー構造に基づいて構成ファイルとモデルファイルを読み取ります。指定された構造に従ってください。必要なモデルファイル以外のフォルダーやファイルをモデル・リポジトリーに配置しないでください。

4. Triton Inference Server* を実行します。

```
docker run --rm -p 8000:8000 -p 8001:8001 -p 8002:8002 -v
/path/to/model_repository:/models nvcr.io/nvidia/tritonserver:24.04-py3
tritonserver --model-repository=/models
```

5. GitHub* から Triton クライアント・コード client.py をダウンロードして Triton クライアントを実行する場所に配置します。

```
wget https://raw.githubusercontent.com/triton-inference-
server/tutorials/main/Quick_Deploy/ONNX/client.py
```

6. client.py ファイルと同じ場所で Triton クライアントを実行し、依存ファイルをインストールして、サーバーにクエリーを送ります。

```
docker run -it --rm --net=host -v ${PWD}:/workspace/
nvcr.io/nvidia/tritonserver:24.04-py3-sdk bash

pip install torchvision

wget  -O img1.jpg "https://www.hakaimagazine.com/wp-
content/uploads/header-gulf-birds.jpg"

python3 client.py
```

7. 出力は次のとおりです。

```
['11.549026:92' '11.232335:14' '7.528014:95' '6.923391:17'
'6.576575:88']
```

PyTorch* モデルのデプロイ

1. PyTorch* モデルをダウンロードして準備します。

   PyTorch* モデル(.pt)を OpenVINO™ 形式に変換する必要があります。PyTorch* モデルをダウンロードし、OpenVINO™ モデル・コンバーターを使用して model.xml と model.bin を保存するように downloadAndConvert.py ファイルを作成します。

   ```
   import torchvision
   import torch
   import openvino as ov
   model = torchvision.models.resnet50(weights='DEFAULT')
   ov_model = ov.convert_model(model)
   ov.save_model(ov_model, 'model.xml')
   ```

   依存ファイルをインストールします。

   ```
   pip install openvino
   pip install torchvision
   ```

   downloadAndConvert.py を実行します。

   ```
   python3 downloadAndConvert.py
   ```

   PyTorch* モデルの変換の詳細は、PyTorch* モデルの変換(英語)を参照してください。

2. config.pbtxt という名前の新しいファイルを作成します。

   ```
   name: "resnet50 "
   backend: "openvino"
   max_batch_size : 0
   input [
     {
       name: "x"
       data_type: TYPE_FP32
       dims: [ 3, 224, 224 ]
       reshape { shape: [ 1, 3, 224, 224 ] }
     }
   ]
   output [
     {
       name: "x.45"
       data_type: TYPE_FP32
       dims: [ 1, 1000 ,1, 1]
       reshape { shape: [ 1, 1000 ] }
     }
   ]
   ```

3. config.pbtxt ファイルと model.xml および model.bin をモデル・リポジトリーに配置します。フォルダー構造を次に示します。

   ```
   model_repository
   |
   ```

```
+-- resnet50
    |
    +-- config.pbtxt
    +-- 1
        |
        +-- model.xml
        +-- model.bin
```

注: Triton Inference Server* は、このフォルダー構造に基づいて構成ファイルとモデルファイルを読み取ります。指定された構造に従ってください。必要なモデルファイル以外のフォルダーやファイルをモデル・リポジトリーに配置しないでください。

4. Triton Inference Server* を実行します。

```
docker run --rm -p 8000:8000 -p 8001:8001 -p 8002:8002 -v
/path/to/model_repository:/models nvcr.io/nvidia/tritonserver:24.04-py3
tritonserver --model-repository=/models
```

5. 別のターミナルで、GitHub* から Triton クライアント・コード client.py をダウンロードして Triton クライアントを実行する場所に配置します。

```
wget https://raw.githubusercontent.com/triton-inference-
server/tutorials/main/Quick_Deploy/PyTorch/client.py
```

使用するモデルは Triton チュートリアルのモデルと若干異なるため、バックエンドで想定される値と一致するように client.py ファイルのモデルの入力名と出力名を変更する必要があります。例えば、PyTorch* モデルで使用する入力名(input__0)を、OpenVINO™ バックエンドで使用する名前(x)に変更します。

古い値	新しい値
input__0	x
output__0	x.45

6. client.py ファイルと同じ場所で Triton クライアントを実行し、依存ファイルをインストールして、サーバーにクエリーを送ります。

```
docker run -it --net=host -v ${PWD}:/workspace/
nvcr.io/nvidia/tritonserver:24.04-py3-sdk bash

pip install torchvision

wget  -O img1.jpg "https://www.hakaimagazine.com/wp-
content/uploads/header-gulf-birds.jpg"

python3 client.py
```

7. 出力は次のとおりです。

```
[b'6.354599:14' b'4.292510:92' b'3.886345:90' b'3.333909:136'
b'3.096908:15']
```

注: OpenVINO™ は TorchServe にも統合されていて、OpenVINO™ IR 形式に変換することなく PyTorch* モデルをサービングできます。サンプルコードはこちら(英語)を参照してください。

TensorFlow* モデルのデプロイ

1. TensorFlow* モデルをダウンロードして準備します。

   TensorFlow* モデルを SavedModel 形式でエクスポートします。

   ```
   docker run -it --gpus all -v ${PWD}:/workspace
   nvcr.io/nvidia/tensorflow:24.04-tf2-py3
   python3 export.py
   ```

   モデルを OpenVINO™ 形式に変換する必要があります。OpenVINO™ モデル・コンバーターを使用して model.xml と model.bin を保存するように convert.py ファイルを作成します。

   ```
   import openvino as ov
   ov_model = ov.convert_model(' path_to_saved_model_dir’)
   ov.save_model(ov_model, 'model.xml')
   ```

   依存ファイルをインストールします。

   ```
   pip install openvino
   ```

   convert.py を実行します。

   ```
   python3 convert.py
   ```

   TensorFlow* モデルの変換の詳細は、TensorFlow* モデルの変換(英語)を参照してください。

2. config.pbtxt という名前の新しいファイルを作成します。

   ```
   name: "resnet50"
   backend: "openvino"
   max_batch_size : 0
   input [
     {
       name: "input_1"
       data_type: TYPE_FP32
       dims: [-1, 224, 224, 3 ]
     }
   ]
   output [
     {
       name: "predictions"
       data_type: TYPE_FP32
       dims: [-1, 1000]
     }
   ]
   ```

3. config.pbtxt ファイルと model.xml および model.bin をモデル・リポジトリーに配置します。フォルダー構造を次に示します。

```
model_repository
|
+-- resnet50
    |
    +-- config.pbtxt
    +-- 1
        |
        +-- model.xml
        +-- model.bin
```

注: Triton Inference Server* は、このフォルダー構造に基づいて構成ファイルとモデルファイルを読み取ります。指定された構造に従ってください。必要なモデルファイル以外のフォルダーやファイルをモデル・リポジトリーに配置しないでください。

4. Triton Inference Server* を実行します。

```
docker run --rm -p 8000:8000 -p 8001:8001 -p 8002:8002 -v
/path/to/model_repository:/models nvcr.io/nvidia/tritonserver:24.04-py3
tritonserver --model-repository=/models
```

5. 別のターミナルで、GitHub* から Triton クライアント・コード client.py をダウンロードして Triton クライアントを実行する場所に配置します。

```
wget https://raw.githubusercontent.com/triton-inference-
server/tutorials/main/Quick_Deploy/TensorFlow/client.py
```

6. client.py ファイルと同じ場所で Triton クライアントを実行し、依存ファイルをインストールして、サーバーにクエリーを送ります。

```
docker run -it --net=host -v ${PWD}:/workspace/
nvcr.io/nvidia/tritonserver:24.04-py3-sdk bash

pip install --upgrade tensorflow

pip install image

wget  -O img1.jpg "https://www.hakaimagazine.com/wp-
content/uploads/header-gulf-birds.jpg"

python3 client.py
```

7. 出力は次のとおりです。

```
[b'0.301167:90' b'0.169790:14' b'0.161309:92' b'0.093105:94'
 b'0.058743:136' b'0.050185:11' b'0.033802:91' b'0.011760:88'
 b'0.008309:989' b'0.004927:95' b'0.004905:13' b'0.004095:317'
 b'0.004006:96' b'0.003694:12' b'0.003526:42' b'0.003390:313'
 ...
 b'0.000001:751' b'0.000001:685' b'0.000001:408' b'0.000001:116'
 b'0.000001:627' b'0.000001:933' b'0.000000:661' b'0.000000:148']
```

まとめ

Triton Inference Server* と OpenVINO™ バックエンドを組み合わせると、ハードウェア・アクセラレーション、最適化、モデルサービング機能を含むマシンラーニング・モデルのデプロイとサービングのための強力なソリューションが提供されます。

サーバーの ‘config.pbtxt’のモデル・パラメーターの最適化と調整の詳細は、GitHub* の triton-inference-server/openvino_backend: Triton 向け OpenVINO™ バックエンド(英語)を参照してください。

Triton Inference Server* をまだ使用していない場合は、代わりに OpenVINO™ モデルサーバー(英語)を使用することを推奨します。

OpenVINO™ ツールキットとは

AI を加速する無償のツールである OpenVINO™ ツールキットは、インテルが無償で提供しているインテル製の CPU や GPU、VPU、FPGA などのパフォーマンスを最大限に活用して、コンピューター・ビジョン、画像関係をはじめ、自然言語処理や音声処理など、幅広いディープラーニング・モデルで推論を最適化し高速化する推論エンジン/ツールスイートです。

OpenVINO™ ツールキット・ページでは、ツールの概要、利用方法、導入事例、トレーニング、ツール・ダウンロードまでさまざまな情報を提供しています。ぜひ特設サイトにアクセスしてみてください。

https://www.intel.co.jp/content/www/jp/ja/internet-of-things/openvino-toolkit.html