TensorFlow

blocked URL

Opis

Tensorflow je python knjižnica namijenjena razvoju aplikacija temeljenih na dubokom učenju koja se oslanja na ubrzanje grafičkim procesorima. Jedna od glavnih značajki ove knjižnice je postojanje API-a za brži razvoj modela strojnog učenja Keras, koja u sebi sadrži module i funkcije za svaki dio pipelinea u tipičnoj ML aplikaciji (preprocessing podataka, definicija modela, načina optimizacije i validacije)

Verzije

verzija	modul	Supek	Padobran
2.10.1	scientific/tensorflow/2.10.1-ngc	✓	
2.12.0	scientific/tensorflow/2.12.0		✓
2.15.0	scientific/tensorflow/2.15.0		✓

Korištenje aplikacije na Supeku

Python aplikacije i knjižnice na Supeku su dostavljene u obliku kontejnera i zahtijevaju korištenje wrappera kao što je opisano ispod.

Više informacija o python aplikacijama i kontejnerima na Supeku možete dobiti na sljedećim poveznicama:

- Python, pip i conda
- Apptainer

Dokumentacija

- Službena stranica - https://www.tensorflow.org
- Priručnik - https://www.tensorflow.org/versions
- Keras API za razvoj aplikacija - https://keras.io/api
- Repozitorij modela i baza podataka - https://www.tensorflow.org/resources/models-datasets

Supek

Ispod se nalaze primjeri aplikacija umjetnog benchmarka koji testira performanse na modelu Resnet50.

Primjeri su redom:

- **singlegpu.*** - skripte za pokretanje na jednom grafičkom procesoru
- **multigpu-singlenode.*** - skripte za pokretanje na više grafičkih procesora na jednom čvoru
- **multigpu-multinode.*** - skripte za pokretanje na više grafičkih procesora na jednom čvoru

Jedan grafički procesor

singlegpu.sh

```
     #!/bin/bash

#PBS -q gpu
#PBS -l select=1:ncpus=8:ngpus=1:mem=10GB

# pozovi modul
module load scientific/tensorflow/2.10.1-ngc

# pomakni se u direktorij gdje se nalazi skripta
cd ${PBS_O_WORKDIR:-""}

# potjeraj skriptu
run-singlenode.sh singlegpu.py
```

singlegpu.py

```
     #!/usr/bin/env python3

# source:
# - https://github.com/leondgarse/Keras_insightface/discussions/17

import sys
import time
import argparse
import numpy as np
import tensorflow as tf

def main():

    # vars
    batch_size = 256
    samples = 256 * 20
    epochs = 10

    # do not allocate all GPU memory
    gpus = tf.config.experimental.list_physical_devices('GPU')
    for gpu in gpus:
        tf.config.experimental.set_memory_growth(gpu, True)

    # use fp16 for faster inference
    tf.keras.mixed_precision.set_global_policy('mixed_float16')

    # strategy
    gpus = tf.config.experimental.list_physical_devices('GPU')
    devices = [ gpu.name[-5:] for gpu in gpus ]
    strategy = tf.distribute.OneDeviceStrategy(device=devices[0])

    # dataset
    data = np.random.uniform(size=[samples, 224, 224, 3])
    target = np.random.uniform(size=[samples, 1], low=0, high=999).astype("int64")
    dataset = tf.data.Dataset.from_tensor_slices((data, target))
    dataset = dataset.batch(batch_size*strategy.num_replicas_in_sync)

    # define model
    with strategy.scope():
        model = tf.keras.applications.ResNet50(weights=None)
        loss = tf.keras.losses.SparseCategoricalCrossentropy()
        optimizer = tf.optimizers.SGD(0.01)
        model.compile(optimizer=optimizer, loss=loss)

    # fit
    callbacks = []
    model.fit(dataset,
              callbacks=callbacks,
              epochs=epochs,
              verbose=2)

if __name__ == "__main__":
    main()
```

Više grafičkih procesora na jednom čvoru

multigpu-singlenode.sh

```
     #!/bin/bash

#PBS -q gpu
#PBS -l select=1:ncpus=16:ngpus=2:mem=10GB

# pozovi modul
module load scientific/tensorflow/2.10.1-ngc

# pomakni se u direktorij gdje se nalazi skripta
cd ${PBS_O_WORKDIR:-""}

# potjeraj skriptu
run-singlenode.sh multigpu-singlenode.py
```

multigpu-singlenode.py

```
     #!/usr/bin/env python3

# source:
# - https://github.com/leondgarse/Keras_insightface/discussions/17

import sys
import time
import argparse
import numpy as np
import tensorflow as tf

def main():

    # vars
    batch_size = 256
    samples = 256 * 20
    epochs = 10

    # do not allocate all GPU memory
    gpus = tf.config.experimental.list_physical_devices('GPU')
    for gpu in gpus:
        tf.config.experimental.set_memory_growth(gpu, True)

    # use fp16 for faster inference
    tf.keras.mixed_precision.set_global_policy('mixed_float16')

    # strategy
    gpus = tf.config.experimental.list_physical_devices('GPU')
    devices = [ gpu.name[-5:] for gpu in gpus ]
    strategy = tf.distribute.MirroredStrategy(devices=devices)

    # dataset
    data = np.random.uniform(size=[samples, 224, 224, 3])
    target = np.random.uniform(size=[samples, 1], low=0, high=999).astype("int64")
    dataset = tf.data.Dataset.from_tensor_slices((data, target))
    dataset = dataset.batch(batch_size*strategy.num_replicas_in_sync)

    # define model
    with strategy.scope():
        model = tf.keras.applications.ResNet50(weights=None)
        loss = tf.keras.losses.SparseCategoricalCrossentropy()
        optimizer = tf.optimizers.SGD(0.01)
        model.compile(optimizer=optimizer, loss=loss)

    # fit
    callbacks = []
    model.fit(dataset,
              callbacks=callbacks,
              epochs=epochs,
              verbose=2)

if __name__ == "__main__":
    main()
```

Više grafičkih procesora na više čvorova

Pri definiranju traženih resursa, potrebno je osigurati jednak broj grafičkih procesora po čvoru.

multigpu-multinode.sh

```
    #!/bin/bash

#PBS -q gpu
#PBS -l select=2:ncpus=8:ngpus=2:mem=10GB
#PBS -l place=scatter

# pozovi modul
module load scientific/tensorflow/2.10.1-ngc

# pomakni se u direktorij gdje se nalazi skripta
cd ${PBS_O_WORKDIR:-""}

# potjeraj skriptu
run-multinode.sh multigpu-multinode.py
```

multigpu-multinode.py

```
     #!/usr/bin/env python3

# source:
# - https://github.com/leondgarse/Keras_insightface/discussions/17

import os
import sys
import time
import socket
import argparse
import numpy as np
import tensorflow as tf

def main():

    # vars
    batch_size = 256
    samples = 256*20
    epochs = 10

    # do not allocate all GPU memory
    gpus = tf.config.experimental.list_physical_devices('GPU')
    for gpu in gpus:
        tf.config.experimental.set_memory_growth(gpu, True)

    # use fp16 for faster inference
    tf.keras.mixed_precision.set_global_policy('mixed_float16')

    # strategy
    communication_options = tf.distribute.experimental.CommunicationOptions(
        implementation=tf.distribute.experimental.CommunicationImplementation.NCCL)
    strategy = tf.distribute.MultiWorkerMirroredStrategy(
        communication_options=communication_options)

    # dataset
    data = np.random.uniform(size=[samples, 224, 224, 3])
    target = np.random.uniform(size=[samples, 1], low=0, high=999).astype("int64")
    dataset = tf.data.Dataset.from_tensor_slices((data, target))
    dataset = dataset.batch(batch_size*strategy.num_replicas_in_sync)

    # define model
    with strategy.scope():
        model = tf.keras.applications.ResNet50(weights=None)
        loss = tf.keras.losses.SparseCategoricalCrossentropy()
        optimizer = tf.optimizers.SGD(0.01)
        model.compile(optimizer=optimizer, loss=loss)

    # fit
    callbacks = []
    verbose = 2 if os.environ['PMI_RANK'] == '0' else 0
    model.fit(dataset,
              callbacks=callbacks,
              epochs=epochs,
              verbose=verbose)

if __name__ == "__main__":
    main()
```

Padobran

Ispod se nalaze primjeri aplikacija umjetnog benchmarka koji testira performanse na modelu Resnet50.

Primjeri su redom:

- **singlenode.*** - skripte za pokretanje na jednom čvoru

Jedan čvor

singlenode.sh

```
    #PBS -q cpu
#PBS -l ncpus=32
#PBS -l mem=50GB

# dopremi modul
module load scientific/tensorflow/2.12.0

# postavi broj cpu jezgri
export OMP_NUM_THREADS=${NCPUS}
export TF_NUM_INTEROP_THREADS=${NCPUS}
export TF_NUM_INTRAOP_THREADS=${NCPUS}

# pomakni se u direktorij i pokreni
cd ${PBS_O_WORKDIR}
python singlenode.py
```

singlenode.py

```
    import sys
import time
import argparse
import numpy as np
import tensorflow as tf

def main():

    # vars
    batch_size = 16
    samples = 16*10
    epochs = 3

    # dataset
    data = np.random.uniform(size=[samples, 224, 224, 3])
    target = np.random.uniform(size=[samples, 1], low=0, high=999).astype("int64")
    dataset = tf.data.Dataset.from_tensor_slices((data, target))
    dataset = dataset.batch(batch_size)

    # define model
    model = tf.keras.applications.ResNet50(weights=None)
    loss = tf.keras.losses.SparseCategoricalCrossentropy()
    optimizer = tf.optimizers.SGD(0.01)
    model.compile(optimizer=optimizer, loss=loss)

    # fit
    callbacks = []
    model.fit(dataset,
              callbacks=callbacks,
              epochs=epochs,
              verbose=1)

if __name__ == "__main__":
    main()
```

Napomene

Apptainer i run-singlenode.sh

Ova knjižnica je dostavljena u obliku kontejnera, zbog opterećenja koje pip/conda virtualna okruženja stvaraju na Lustre dijeljenim datotečnim sustavima.

Za ispravno izvršavanje python aplikacija, potrebno ih je koristiti wrappere **run-singlenode.sh** ili **run-multinode.sh** u skriptama sustava PBS:

```
        ...
run-singlenode.sh moja_python_skripta.py
...
```

Korištenje više grafičkih procesora

Tensorflow **ne raspodjeljuje** automatski aplikaciju na više grafičkih procesora.

Pri razvoju aplikacije, bitno je koristiti odgovarajuće funkcionalnosti poput primjera iznad:

MirroredStrategy primjer

```
        ...
strategy = tf.distribute.MirroredStrategy()
with strategy.scope():
        model = ... definirati model ...
        model.compile()
```