AIPod導入時のハイパフォーマンスジョブの例

NetApp Solutions

NetApp
May 10, 2024

目次

AIPod導入時のハイパフォーマンスジョブの例

シングルノードの AI ワークロードを実行

Kubernetes クラスタでシングルノードの AI ジョブと ML ジョブを実行するには、導入ジャンプホストから次のタスクを実行します。Trident を使用すると、数ペタバイトのデータが含まれる可能性のあるデータボリュームをすばやく簡単に作成し、 Kubernetes のワークロードからアクセスできます。Kubernetes ポッド内からこのようなデータボリュームにアクセスできるようにするには、ポッドの定義で PVC を指定します。

このセクションでは、 Kubernetes クラスタで実行しようとしている特定の AI および ML ワークロードを（ Docker コンテナ形式で）コンテナ化済みであることを前提としています。

1. 次のコマンド例は、 ImageNet データセットを使用する TensorFlow ベンチマークワークロード用の Kubernetes ジョブを作成する方法を示しています。ImageNet データセットの詳細については、を参照してください "ImageNet の Web サイト"。

   このジョブ例では、 8 個の GPU を要求するため、 8 個以上の GPU を搭載した 1 つの GPU ワーカーノードで実行することができます。このジョブ例は、 8 個以上の GPU を搭載したワーカーノードが存在しない、または現在別のワークロードを使用しているクラスタで送信できます。その場合、そのようなワーカーノードが使用可能になるまで、ジョブは保留状態のままになります。

   また、ストレージ帯域幅を最大限にするために、必要なトレーニングデータを含むボリュームが、このジョブで作成されるポッド内に 2 回マウントされます。ポッドには別のボリュームもマウントされています。この 2 つ目のボリュームには、結果と指標を格納します。これらのボリュームは、 PVC の名前を使用してジョブ定義内で参照されます。Kubernetes ジョブの詳細については、を参照してください "Kubernetes の公式ドキュメント"。

   「 M emory 」の値が「 emory 」である「 emptyDir 」ボリュームは、この例のジョブで作成されるポッド内の「 /dev/shm 」にマウントされます。Docker コンテナランタイムによって自動的に作成される「 /dev/shm 」仮想ボリュームのデフォルトサイズが、 TensorFlow のニーズに十分でない場合があります。次の例のように 'emptyDir' ボリュームをマウントすると '/dev/shm' 仮想ボリュームが十分に大きくなります「 emptyDir 」ボリュームの詳細については、を参照してください "Kubernetes の公式ドキュメント"。

   この例のジョブ定義で指定されている単一のコンテナには 'ecurityContext> 特権値 'true' が与えられますこの値は、コンテナにホスト上のルートアクセス権があることを意味します。このアノテーションは、実行中の特定のワークロードにルートアクセスが必要なために使用されます。具体的には、ワークロードで実行されるクリアキャッシュ処理にはルートアクセスが必要です。これが特権 : true の注釈であるかどうかは ' 実行している特定のワークロードの要件によって異なります

```
$ cat << EOF > ./netapp-tensorflow-single-imagenet.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: netapp-tensorflow-single-imagenet
spec:
  backoffLimit: 5
  template:
```

```
    spec:
      volumes:
      - name: dshm
        emptyDir:
          medium: Memory
      - name: testdata-iface1
        persistentVolumeClaim:
          claimName: pb-fg-all-iface1
      - name: testdata-iface2
        persistentVolumeClaim:
          claimName: pb-fg-all-iface2
      - name: results
        persistentVolumeClaim:
          claimName: tensorflow-results
      containers:
      - name: netapp-tensorflow-py2
        image: netapp/tensorflow-py2:19.03.0
        command: ["python", "/netapp/scripts/run.py", "--
dataset_dir=/mnt/mount_0/dataset/imagenet", "--dgx_version=dgx1", "--
num_devices=8"]
        resources:
          limits:
            nvidia.com/gpu: 8
        volumeMounts:
        - mountPath: /dev/shm
          name: dshm
        - mountPath: /mnt/mount_0
          name: testdata-iface1
        - mountPath: /mnt/mount_1
          name: testdata-iface2
        - mountPath: /tmp
          name: results
        securityContext:
          privileged: true
      restartPolicy: Never
EOF
$ kubectl create -f ./netapp-tensorflow-single-imagenet.yaml
job.batch/netapp-tensorflow-single-imagenet created
$ kubectl get jobs
NAME                                     COMPLETIONS   DURATION   AGE
netapp-tensorflow-single-imagenet        0/1           24s        24s
```

2. 手順 1 で作成したジョブが正しく実行されていることを確認します。次のコマンド例では、ジョブ定義で指定したとおりにジョブ用にポッドが 1 つ作成され、このポッドが GPU ワーカーノードの 1 つで現在実行されていることを確認します。

```
$ kubectl get pods -o wide
NAME                                                READY    STATUS
RESTARTS   AGE
IP              NODE             NOMINATED NODE
netapp-tensorflow-single-imagenet-m7x92          1/1      Running     0
3m    10.233.68.61    10.61.218.154   <none>
```

3. 手順 1 で作成したジョブが正常に完了したことを確認します。次のコマンド例は、ジョブが正常に完了したことを確認します。

```
$ kubectl get jobs
NAME                                                COMPLETIONS   DURATION
AGE
netapp-tensorflow-single-imagenet                   1/1           5m42s
10m
$ kubectl get pods
NAME                                                      READY   STATUS
RESTARTS   AGE
netapp-tensorflow-single-imagenet-m7x92                   0/1     Completed
0          11m
$ kubectl logs netapp-tensorflow-single-imagenet-m7x92
[netapp-tensorflow-single-imagenet-m7x92:00008] PMIX ERROR: NO-
PERMISSIONS in file gds_dstore.c at line 702
[netapp-tensorflow-single-imagenet-m7x92:00008] PMIX ERROR: NO-
PERMISSIONS in file gds_dstore.c at line 711
Total images/sec = 6530.59125
================ Clean Cache !!! ==================
mpirun -allow-run-as-root -np 1 -H localhost:1 bash -c 'sync; echo 1 >
/proc/sys/vm/drop_caches'
==========================================
mpirun -allow-run-as-root -np 8 -H localhost:8 -bind-to none -map-by
slot -x NCCL_DEBUG=INFO -x LD_LIBRARY_PATH -x PATH python
/netapp/tensorflow/benchmarks_190205/scripts/tf_cnn_benchmarks/tf_cnn_be
nchmarks.py --model=resnet50 --batch_size=256 --device=gpu
--force_gpu_compatible=True --num_intra_threads=1 --num_inter_threads=48
--variable_update=horovod --batch_group_size=20 --num_batches=500
--nodistortions --num_gpus=1 --data_format=NCHW --use_fp16=True
--use_tf_layers=False --data_name=imagenet --use_datasets=True
--data_dir=/mnt/mount_0/dataset/imagenet
--datasets_parallel_interleave_cycle_length=10
--datasets_sloppy_parallel_interleave=False --num_mounts=2
--mount_prefix=/mnt/mount_%d --datasets_prefetch_buffer_size=2000
--datasets_use_prefetch=True --datasets_num_private_threads=4
--horovod_device=gpu >
/tmp/20190814_105450_tensorflow_horovod_rdma_resnet50_gpu_8_256_b500_ima
genet_nodistort_fp16_r10_m2_nockpt.txt 2>&1
```

4. * オプション： * ジョブアーティファクトをクリーンアップします。次のコマンド例は、手順 1 で作成したジョブオブジェクトの削除を示しています。

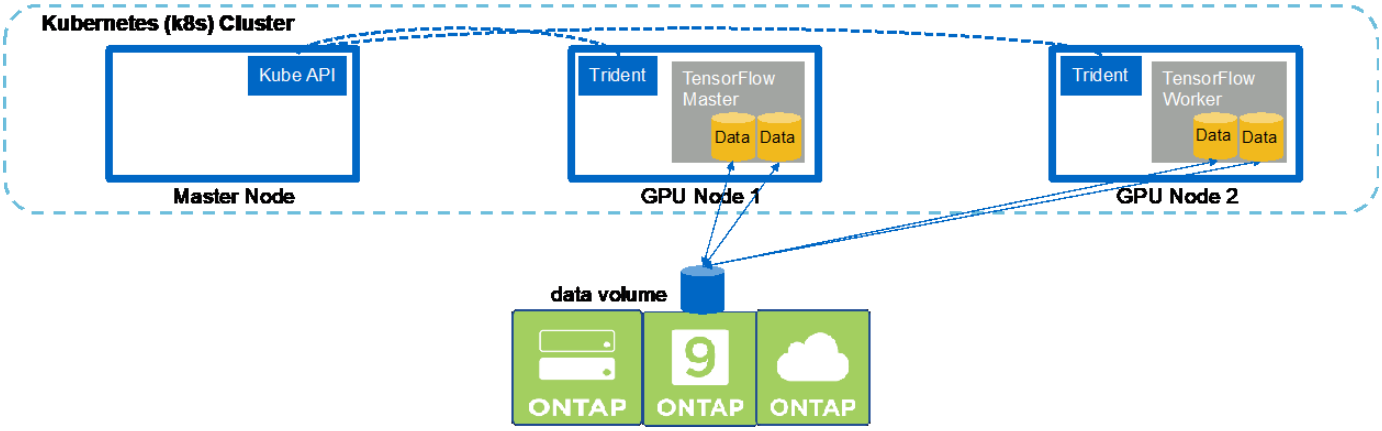
   ジョブオブジェクトを削除すると、関連付けられているポッドは Kubernetes によって自動的に削除されます。

```
$ kubectl get jobs
NAME                                                COMPLETIONS   DURATION
AGE
netapp-tensorflow-single-imagenet                   1/1           5m42s
10m
$ kubectl get pods
NAME                                                      READY   STATUS
RESTARTS   AGE
netapp-tensorflow-single-imagenet-m7x92                   0/1     Completed
0          11m
$ kubectl delete job netapp-tensorflow-single-imagenet
job.batch "netapp-tensorflow-single-imagenet" deleted
$ kubectl get jobs
No resources found.
$ kubectl get pods
No resources found.
```

分散型 AI の同期ワークロードを実行します

Kubernetes クラスタでマルチノードの AI と ML の同期ジョブを実行するには、導入ジャンプホストで次のタスクを実行します。このプロセスにより、ネットアップボリュームに格納されているデータを活用し、 1 つのワーカーノードで提供されるものよりも多くの GPU を使用することができます。同期分散 AI ジョブの説明については、次の図を参照してください。

同期分散ジョブを使用すると、非同期分散ジョブに比べてパフォーマンスとトレーニングの精度が向上します。同期ジョブと非同期ジョブの長所と短所については、本ドキュメントでは説明していません。



1. 次のコマンド例は、 1 つのワーカーの作成を示しています これは、同じの同期分散実行に関与します 1 つのノードで実行された TensorFlow ベンチマークジョブ を参照してください "シングルノードの AI ワークロードを実行"。この例では、 2 つのワーカーノードでジョブが実行されるため、導入されるワーカーは 1 つだけです。

この例のワーカー導入では、 8 個の GPU を要求し、 8 個以上の GPU を搭載した 1 つの GPU ワーカーノードで実行できます。GPU ワーカーノードが 8 個以上の GPU を搭載している場合、パフォーマンスを最大化するには、この数をワーカーノードが機能する GPU の数と同じにすると便利です。Kubernetes の導入の詳細については、を参照してください "Kubernetes の公式ドキュメント"。

この例では、このコンテナ化された特定のワーカーが自分で完了することはないため、 Kubernetes 環境が作成されます。そのため、 Kubernetes のジョブ構造を使用して導入することは理にかなっていません。従業員が自分で設計または作成した場合、作業構成を使用して従業員を配置することが理にかなっている場合があります。

この配置例の仕様で指定されているポッドには 'hostNetwork' の値が true に設定されていますこの値は、ポッドが、 Kubernetes が各ポッドに通常作成する仮想ネットワーキングスタックではなく、ホストワーカーノードのネットワークスタックを使用することを意味します。このアノテーションは、特定のワークロードが Open MPI 、 NCCL 、 Horovod を使用して同期分散方法でワークロードを実行するために使用されます。そのため、ホストネットワークスタックにアクセスする必要があります。Open MPI 、 NCCL 、および Horovod についての説明は、本ドキュメントの範囲外です。この hostNetwork:true' 注釈が必要かどうかは ' 実行している特定のワークロードの要件によって決まります「 hostNetwork 」フィールドの詳細については、を参照してください "Kubernetes の公式ドキュメント"。

```
$ cat << EOF > ./netapp-tensorflow-multi-imagenet-worker.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: netapp-tensorflow-multi-imagenet-worker
spec:
  replicas: 1
  selector:
    matchLabels:
      app: netapp-tensorflow-multi-imagenet-worker
  template:
    metadata:
      labels:
        app: netapp-tensorflow-multi-imagenet-worker
    spec:
      hostNetwork: true
      volumes:
      - name: dshm
        emptyDir:
          medium: Memory
      - name: testdata-iface1
        persistentVolumeClaim:
          claimName: pb-fg-all-iface1
      - name: testdata-iface2
        persistentVolumeClaim:
          claimName: pb-fg-all-iface2
      - name: results
        persistentVolumeClaim:
          claimName: tensorflow-results
```

```
      containers:
      - name: netapp-tensorflow-py2
        image: netapp/tensorflow-py2:19.03.0
        command: ["bash", "/netapp/scripts/start-slave-multi.sh",
"22122"]
        resources:
          limits:
            nvidia.com/gpu: 8
        volumeMounts:
        - mountPath: /dev/shm
          name: dshm
        - mountPath: /mnt/mount_0
          name: testdata-iface1
        - mountPath: /mnt/mount_1
          name: testdata-iface2
        - mountPath: /tmp
          name: results
        securityContext:
          privileged: true
EOF
$ kubectl create -f ./netapp-tensorflow-multi-imagenet-worker.yaml
deployment.apps/netapp-tensorflow-multi-imagenet-worker created
$ kubectl get deployments
NAME                                        DESIRED   CURRENT   UP-TO-DATE
AVAILABLE   AGE
netapp-tensorflow-multi-imagenet-worker   1         1         1
1           4s
```

2. 手順 1 で作成したワーカー導入が正常に起動したことを確認します。次のコマンド例は、導入定義に示すように、単一のワーカーポッドが導入用に作成されたこと、およびこのポッドが GPU ワーカーノードの 1 つで現在実行されていることを確認します。

```
$ kubectl get pods -o wide
NAME                                                              READY
STATUS    RESTARTS   AGE
IP              NODE            NOMINATED NODE
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725   1/1
Running   0          60s   10.61.218.154   10.61.218.154   <none>
$ kubectl logs netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725
22122
```

3. マルチノード同期ジョブの実行を開始して参加させ、追跡するマスター用の Kubernetes ジョブを作成します。次のコマンド例では、 1 つのマスターを作成します。このマスターは、セクションの例で 1 つのノード上で実行された、同じ TensorFlow ベンチマークジョブの同期分散実行を追跡し、開始します "シングルノードの AI ワークロードを実行"。

この例では、マスタージョブは 8 個の GPU を要求するため、 8 個以上の GPU を搭載した 1 つの GPU ワーカーノードで実行できます。GPU ワーカーノードが 8 個以上の GPU を搭載している場合、パフォーマンスを最大化するには、この数をワーカーノードが機能する GPU の数と同じにすると便利です。

この例のジョブ定義で指定されているマスターポッドには、手順 1 でワーカーポッドに「 hostNetwork 」の値「 true 」が与えられたのと同様に、「 hostNetwork 」の値が「 true 」に設定されます。この値が必要な理由については、手順 1 を参照してください。

```
$ cat << EOF > ./netapp-tensorflow-multi-imagenet-master.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: netapp-tensorflow-multi-imagenet-master
spec:
  backoffLimit: 5
  template:
    spec:
      hostNetwork: true
      volumes:
      - name: dshm
        emptyDir:
          medium: Memory
      - name: testdata-iface1
        persistentVolumeClaim:
          claimName: pb-fg-all-iface1
      - name: testdata-iface2
        persistentVolumeClaim:
          claimName: pb-fg-all-iface2
      - name: results
        persistentVolumeClaim:
          claimName: tensorflow-results
      containers:
      - name: netapp-tensorflow-py2
        image: netapp/tensorflow-py2:19.03.0
        command: ["python", "/netapp/scripts/run.py", "--
dataset_dir=/mnt/mount_0/dataset/imagenet", "--port=22122", "--
num_devices=16", "--dgx_version=dgx1", "--
nodes=10.61.218.152,10.61.218.154"]
        resources:
          limits:
            nvidia.com/gpu: 8
        volumeMounts:
        - mountPath: /dev/shm
          name: dshm
        - mountPath: /mnt/mount_0
          name: testdata-iface1
        - mountPath: /mnt/mount_1
```

```
          name: testdata-iface2
        - mountPath: /tmp
          name: results
        securityContext:
          privileged: true
      restartPolicy: Never
EOF
$ kubectl create -f ./netapp-tensorflow-multi-imagenet-master.yaml
job.batch/netapp-tensorflow-multi-imagenet-master created
$ kubectl get jobs
NAME                                        COMPLETIONS   DURATION   AGE
netapp-tensorflow-multi-imagenet-master   0/1           25s        25s
```

4. 手順 3 で作成したマスタージョブが正しく実行されていることを確認します。次のコマンド例では、ジョブ定義に示されているように、ジョブに対して単一のマスターポッドが作成され、このポッドが GPU ワーカーノードの 1 つで現在実行されていることを確認します。また、手順 1 で最初に確認したワーカーポッドがまだ実行中で、マスターポッドとワーカーポッドが別々のノードで実行されていることも確認する必要があります。

```
$ kubectl get pods -o wide
NAME                                                              READY
STATUS    RESTARTS   AGE
IP              NODE            NOMINATED NODE
netapp-tensorflow-multi-imagenet-master-ppwwj                     1/1
Running   0          45s   10.61.218.152   10.61.218.152   <none>
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725   1/1
Running   0          26m   10.61.218.154   10.61.218.154   <none>
```

5. 手順 3 で作成したマスタージョブが正常に完了したことを確認します。次のコマンド例は、ジョブが正常に完了したことを確認します。

```
$ kubectl get jobs
NAME                                        COMPLETIONS   DURATION   AGE
netapp-tensorflow-multi-imagenet-master   1/1           5m50s      9m18s
$ kubectl get pods
NAME                                                              READY
STATUS      RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj                     0/1
Completed   0          9m38s
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725   1/1
Running     0          35m
$ kubectl logs netapp-tensorflow-multi-imagenet-master-ppwwj
[10.61.218.152:00008] WARNING: local probe returned unhandled
shell:unknown assuming bash
rm: cannot remove '/lib': Is a directory
```

```
[10.61.218.154:00033] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 702
[10.61.218.154:00033] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 711
[10.61.218.152:00008] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 702
[10.61.218.152:00008] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 711
Total images/sec = 12881.33875
================ Clean Cache !!! ==================
mpirun -allow-run-as-root -np 2 -H 10.61.218.152:1,10.61.218.154:1 -mca
pml ob1 -mca btl ^openib -mca btl_tcp_if_include enp1s0f0 -mca
plm_rsh_agent ssh -mca plm_rsh_args "-p 22122" bash -c 'sync; echo 1 >
/proc/sys/vm/drop_caches'
==========================================
mpirun -allow-run-as-root -np 16 -H 10.61.218.152:8,10.61.218.154:8
-bind-to none -map-by slot -x NCCL_DEBUG=INFO -x LD_LIBRARY_PATH -x PATH
-mca pml ob1 -mca btl ^openib -mca btl_tcp_if_include enp1s0f0 -x
NCCL_IB_HCA=mlx5 -x NCCL_NET_GDR_READ=1 -x NCCL_IB_SL=3 -x
NCCL_IB_GID_INDEX=3 -x
NCCL_SOCKET_IFNAME=enp5s0.3091,enp12s0.3092,enp132s0.3093,enp139s0.3094
-x NCCL_IB_CUDA_SUPPORT=1 -mca orte_base_help_aggregate 0 -mca
plm_rsh_agent ssh -mca plm_rsh_args "-p 22122" python
/netapp/tensorflow/benchmarks_190205/scripts/tf_cnn_benchmarks/tf_cnn_be
nchmarks.py --model=resnet50 --batch_size=256 --device=gpu
--force_gpu_compatible=True --num_intra_threads=1 --num_inter_threads=48
--variable_update=horovod --batch_group_size=20 --num_batches=500
--nodistortions --num_gpus=1 --data_format=NCHW --use_fp16=True
--use_tf_layers=False --data_name=imagenet --use_datasets=True
--data_dir=/mnt/mount_0/dataset/imagenet
--datasets_parallel_interleave_cycle_length=10
--datasets_sloppy_parallel_interleave=False --num_mounts=2
--mount_prefix=/mnt/mount_%d --datasets_prefetch_buffer_size=2000 --
datasets_use_prefetch=True --datasets_num_private_threads=4
--horovod_device=gpu >
/tmp/20190814_161609_tensorflow_horovod_rdma_resnet50_gpu_16_256_b500_im
agenet_nodistort_fp16_r10_m2_nockpt.txt 2>&1
```

6. 不要になったワーカー配置を削除します。次のコマンド例は、手順 1 で作成したワーカー配置オブジェクトの削除を示しています。

   ワーカー導入オブジェクトを削除すると、関連付けられているワーカーポッドは Kubernetes によって自動的に削除されます。

```
$ kubectl get deployments
NAME                                          DESIRED   CURRENT   UP-TO-DATE
AVAILABLE   AGE
netapp-tensorflow-multi-imagenet-worker   1         1         1
1           43m
$ kubectl get pods
NAME                                                              READY
STATUS      RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj                     0/1
Completed   0          17m
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725   1/1
Running     0          43m
$ kubectl delete deployment netapp-tensorflow-multi-imagenet-worker
deployment.extensions "netapp-tensorflow-multi-imagenet-worker" deleted
$ kubectl get deployments
No resources found.
$ kubectl get pods
NAME                                               READY   STATUS
RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj   0/1     Completed   0
18m
```

7. * オプション： * マスタージョブアーティファクトをクリーンアップします。次のコマンド例は、手順 3 で作成したマスタージョブオブジェクトの削除を示しています。

   マスタージョブオブジェクトを削除すると、関連付けられているマスターポッドは Kubernetes によって自動的に削除されます。

```
$ kubectl get jobs
NAME                                       COMPLETIONS   DURATION   AGE
netapp-tensorflow-multi-imagenet-master   1/1           5m50s      19m
$ kubectl get pods
NAME                                             READY   STATUS
RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj   0/1     Completed   0
19m
$ kubectl delete job netapp-tensorflow-multi-imagenet-master
job.batch "netapp-tensorflow-multi-imagenet-master" deleted
$ kubectl get jobs
No resources found.
$ kubectl get pods
No resources found.
```

著作権に関する情報

商標に関する情報