



ONTAP AI

導入向けハイパフォーマンスジョブの例

NetApp Solutions

NetApp
April 10, 2024

This PDF was generated from https://docs.netapp.com/ja-jp/netapp-solutions/ai/aicp_example_high-performance_jobs_for_ontap_ai_deployments_overview.html on April 10, 2024. Always check docs.netapp.com for the latest.

目次

ONTAP AI 導入向けハイパフォーマンスジョブの例	1
ONTAP AI 導入向けハイパフォーマンスジョブの例	1
シングルノードの AI ワークロードを実行	1
分散型 AI の同期ワークロードを実行します	5

ONTAP AI 導入向けハイパフォーマンスジョブの例

このセクションでは、ONTAP AI ポッドに Kubernetes を導入する際に実行可能なさまざまなハイパフォーマンスジョブの例を示します。

ONTAP AI 導入向けハイパフォーマンスジョブの例

このセクションでは、ONTAP AI ポッドに Kubernetes を導入する際に実行可能なさまざまなハイパフォーマンスジョブの例を示します。

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。このステップは Kubernetes ネイティブの運用であり、ネットアップの専門知識は不要です。



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"](#)。

「Memory」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 'securityContext' 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
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.

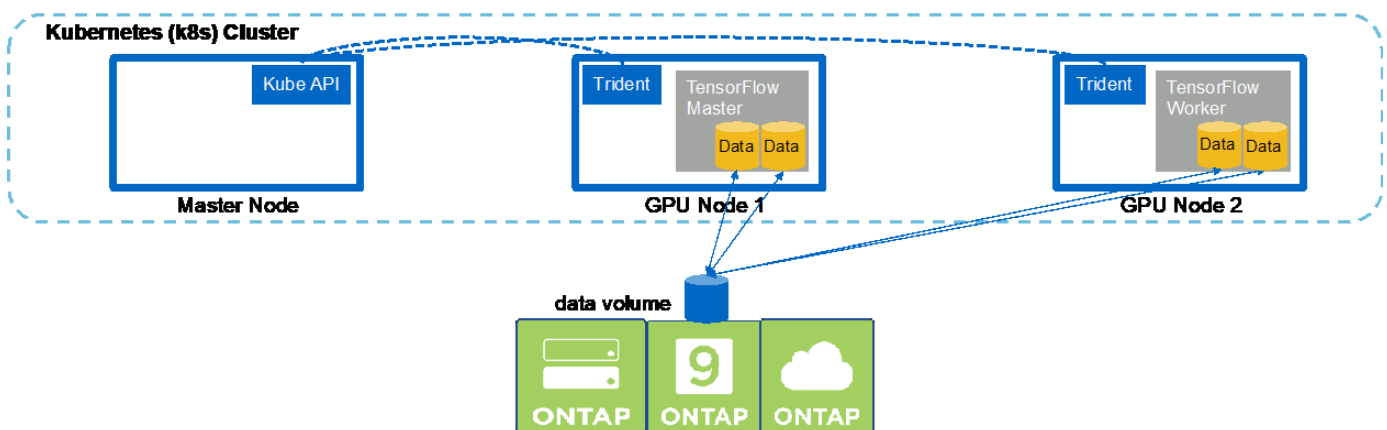
```

分散型 AI の同期ワークロードを実行します

Kubernetes クラスターでマルチノードの 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クスタックではなく、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

```

- 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

```

- 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、マスタートジョブは 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トポッドには、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 obl -mca btl ^openib -mca btl_tcp_if_include enpls0f0 -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 obl -mca btl ^openib -mca btl_tcp_if_include enpls0f0 -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

Copyright © 2024 NetApp, Inc. All Rights Reserved. Printed in the U.S. このドキュメントは著作権によって保護されています。著作権所有者の書面による事前承諾がある場合を除き、画像媒体、電子媒体、および写真複写、記録媒体、テープ媒体、電子検索システムへの組み込みを含む機械媒体など、いかなる形式および方法による複製も禁止します。

ネットアップの著作物から派生したソフトウェアは、次に示す使用許諾条項および免責条項の対象となります。

このソフトウェアは、ネットアップによって「現状のまま」提供されています。ネットアップは明示的な保証、または商品性および特定目的に対する適合性の暗示的保証を含み、かつこれに限定されないいかなる暗示的な保証も行いません。ネットアップは、代替品または代替サービスの調達、使用不能、データ損失、利益損失、業務中断を含み、かつこれに限定されない、このソフトウェアの使用により生じたすべての直接的損害、間接的損害、偶発的損害、特別損害、懲罰的損害、必然的損害の発生に対して、損失の発生の可能性が通知されていたとしても、その発生理由、根拠とする責任論、契約の有無、厳格責任、不法行為（過失またはそうでない場合を含む）にかかわらず、一切の責任を負いません。

ネットアップは、ここに記載されているすべての製品に対する変更を随時、予告なく行う権利を保有します。ネットアップによる明示的な書面による合意がある場合を除き、ここに記載されている製品の使用により生じる責任および義務に対して、ネットアップは責任を負いません。この製品の使用または購入は、ネットアップの特許権、商標権、または他の知的所有権に基づくライセンスの供与とはみなされません。

このマニュアルに記載されている製品は、1つ以上の米国特許、その他の国の特許、および出願中の特許によって保護されている場合があります。

権利の制限について：政府による使用、複製、開示は、DFARS 252.227-7013（2014年2月）およびFAR 5252.227-19（2007年12月）のRights in Technical Data -Noncommercial Items（技術データ - 非商用品目に関する諸権利）条項の(b)(3)項、に規定された制限が適用されます。

本書に含まれるデータは商用製品および / または商用サービス（FAR 2.101の定義に基づく）に関係し、データの所有権はNetApp, Inc.にあります。本契約に基づき提供されるすべてのネットアップの技術データおよびコンピュータ ソフトウェアは、商用目的であり、私費のみで開発されたものです。米国政府は本データに対し、非独占的かつ移転およびサブライセンス不可で、全世界を対象とする取り消し不能の制限付き使用权を有し、本データの提供の根拠となった米国政府契約に関連し、当該契約の裏付けとする場合にのみ本データを使用できます。前述の場合を除き、NetApp, Inc.の書面による許可を事前に得ることなく、本データを使用、開示、転載、改変するほか、上演または展示することはできません。国防総省にかかる米国政府のデータ使用权については、DFARS 252.227-7015(b)項（2014年2月）で定められた権利のみが認められます。

商標に関する情報

NetApp、NetAppのロゴ、<http://www.netapp.com/TM>に記載されているマークは、NetApp, Inc.の商標です。その他の会社名と製品名は、それを所有する各社の商標である場合があります。