AIPod部署的高性能作业示例

NetApp Solutions

NetApp
May 10, 2024

This PDF was generated from https://docs.netapp.com/zh-cn/netapp-solutions/ai/aicp_execute_a_single-node_ai_workload.html on May 10, 2024. Always check docs.netapp.com for the latest.

目录

AIPod部署的高性能作业示例

执行单节点 AI 工作负载

要在 Kubernetes 集群中执行单节点 AI 和 ML 作业，请从部署跳转主机执行以下任务。借助 Trident ，您可以快速轻松地使可能包含数 PB 数据的数据卷可供 Kubernetes 工作负载访问。要使此类数据卷可从 Kubernetes Pod 中访问，只需在 Pod 定义中指定 PVC 即可。

本节假定您已将尝试在 Kubernetes 集群中执行的特定 AI 和 ML 工作负载容器化（采用 Docker 容器格式）。

1. 以下示例命令显示了如何为使用 ImageNet 数据集的 TensorFlow 基准工作负载创建 Kubernetes 作业。有关 ImageNet 数据集的详细信息，请参见 "ImageNet 网站"。

   此示例作业请求八个 GPU ，因此可以在具有八个或更多 GPU 的单个 GPU 工作节点上运行。此示例作业可以在集群中提交，其中包含八个或更多 GPU 的工作节点不存在或当前占用另一个工作负载。如果是，则此作业将保持待定状态，直到此类辅助节点变为可用为止。

   此外，为了最大程度地提高存储带宽，包含所需训练数据的卷会在该作业创建的 POD 中挂载两次。此外，还会在 Pod 中挂载另一个卷。第二个卷将用于存储结果和指标。这些卷在作业定义中使用 PVC 的名称进行引用。有关 Kubernetes 作业的详细信息，请参见 "Kubernetes 官方文档"。

   在本示例作业创建的 Pod 中， `edium` 值为 `Memory` 的` emtyDir `m卷将挂载到 ` `/dev/shm` 。Docker 容器运行时自动创建的 ` /dev/shm` 虚拟卷的默认大小有时可能不足以满足 TensorFlow 的需求。按以下示例所示挂载 `emptyDir` 卷可提供足够大的 ` /dev/shm` 虚拟卷。有关 `emptyDir` 卷的详细信息，请参见 "Kubernetes 官方文档"。

   在此示例作业定义中指定的单个容器将获得 `securityContext > privileged` 值 `true` 。此值表示容器在主机上具有有效的 root 访问权限。在这种情况下使用此标注是因为要执行的特定工作负载需要 root 访问权限。具体而言，工作负载执行的清除缓存操作需要 root 访问权限。是否需要此 **特权：** `true` 标注取决于您要执行的特定工作负载的要求。

```
$ 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. 确认您在步骤 1 中创建的作业正在正确运行。以下示例命令确认已按照作业定义中的指定为此作业创建了一个 POD ，并且此 POD 当前正在其中一个 GPU 工作节点上运行。

```
$ 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. 确认您在步骤 1 中创建的作业已成功完成。以下示例命令确认作业已成功完成。

```
$ 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. * 可选： * 清理作业项目。以下示例命令显示了在步骤 1 中创建的作业对象的删除情况。

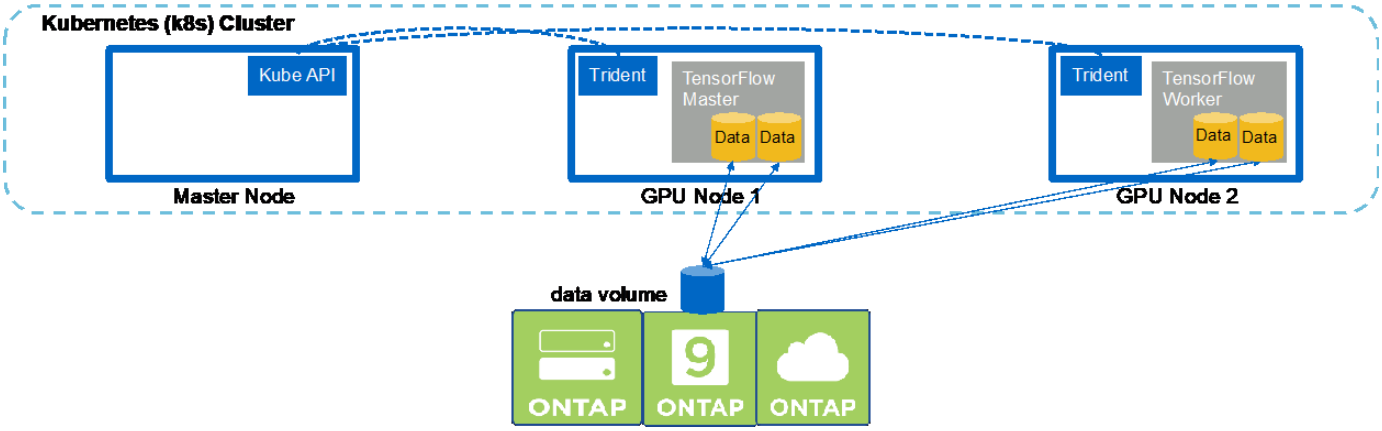
   删除作业对象时， Kubernetes 会自动删除任何关联的 Pod 。

```
$ 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.
```

执行同步分布式 AI 工作负载

要在 Kubernetes 集群中执行同步多节点 AI 和 ML 作业，请在部署跳转主机上执行以下任务。通过此过程，您可以利用存储在 NetApp 卷上的数据，并使用单个工作节点所能提供的 GPU 。有关同步分布式 AI 作业的描述，请参见下图。

与异步分布式作业相比，同步分布式作业有助于提高性能和训练准确性。本文档不会讨论同步作业与异步作业的利弊。



1. 以下示例命令显示了创建一名员工参与同步分布式执行本节中示例中在单个节点上执行的同一 TensorFlow 基准测试作业的过程 "执行单节点 AI 工作负载"。在此特定示例中，仅部署一个员工，因为此作业会在两个员工节点上执行。

   此示例员工部署请求八个 GPU ，因此可以在一个 GPU 工作节点上运行，该节点具有八个或更多 GPU 。如果您的 GPU 工作节点具有八个以上的 GPU ，则为了最大限度地提高性能，您可能需要增加此数量，使其等于您的工作节点所具有的 GPU 数量。有关 Kubernetes 部署的详细信息，请参见 "Kubernetes 官方文档

"。

在此示例中创建了 Kubernetes 部署，因为此特定容器化员工永远不会自行完成。因此，使用 Kubernetes 作业构造来部署它毫无意义。如果员工的设计或编写是为了自己完成，则可以使用此作业构建来部署员工。

在此示例部署规范中指定的 Pod 的值为 `hostNetwork` 值 `true` 。此值表示 Pod 使用主机工作节点的网络堆栈，而不是 Kubernetes 通常为每个 Pod 创建的虚拟网络堆栈。在这种情况下使用此标注是因为特定工作负载依靠 Open MPI ， NCCL 和 Horovod 以同步分布式方式执行工作负载。因此，它需要访问主机网络堆栈。有关 Open MPI ， NCCL 和 Horovod 的讨论不在本文档的讨论范围之内。是否需要此 `hostNetwork` ： `true` 标注取决于要执行的特定工作负载的要求。有关 `hostNetwork` 字段的详细信息，请参见 "Kubernetes 官方文档"。

```
$ 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. 确认您在第 1 步中创建的员工部署已成功启动。以下示例命令确认已为部署创建了一个辅助 POD ，如部署定义所示，并且此 POD 当前正在其中一个 GPU 辅助节点上运行。

```
$ 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. 为启动，参与并跟踪同步多节点作业执行的主节点创建 Kubernetes 作业。以下示例命令创建一个主节点，用于启动，参与和跟踪在一节中的示例中对单个节点执行的相同 TensorFlow 基准测试作业的同步分布式执行 "执行单节点 AI 工作负载"。

   此示例主作业请求八个 GPU ，因此可以在具有八个或更多 GPU 的单个 GPU 工作节点上运行。如果您的 GPU 工作节点具有八个以上的 GPU ，则为了最大限度地提高性能，您可能需要增加此数量，使其等于您的工作节点所具有的 GPU 数量。

   在本示例作业定义中指定的主 Pod 的值为 `hostNetwork` 值为 `true` ，就像在步骤 1 中为工作 Pod 提供了 `hostNetwork` 值 `true` 一样。有关为何需要此值的详细信息，请参见第 1 步。

```
$ 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. 确认您在步骤 3 中创建的主作业正在正确运行。以下示例命令确认已为作业创建了一个主 Pod ，如作业定义所示，并且此 Pod 当前正在其中一个 GPU 工作节点上运行。您还应看到，您最初在步骤 1 中看到的辅助 POD 仍在运行，并且主节点和辅助节点正在不同的节点上运行。

```
$ 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. 确认您在步骤 3 中创建的主作业已成功完成。以下示例命令确认作业已成功完成。

```
$ 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. 如果您不再需要此员工部署，请将其删除。以下示例命令显示了删除在步骤 1 中创建的工作部署对象的过程。

   删除 worker 部署对象时， Kubernetes 会自动删除任何关联的 worker Pod 。

```
$ 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. * 可选： * 清理主作业项目。以下示例命令显示了删除在步骤 3 中创建的主作业对象的过程。

   删除主作业对象时， Kubernetes 会自动删除任何关联的主 Pod 。

```
$ 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.
```

版权信息

版权所有 © 2024 NetApp, Inc.。保留所有权利。中国印刷。未经版权所有者事先书面许可，本文档中受版权保护的任何部分不得以任何形式或通过任何手段（图片、电子或机械方式，包括影印、录音、录像或存储在电子检索系统中）进行复制。

从受版权保护的 NetApp 资料派生的软件受以下许可和免责声明的约束：

本软件由 NetApp 按“原样”提供，不含任何明示或暗示担保，包括但不限于适销性以及针对特定用途的适用性的隐含担保，特此声明不承担任何责任。在任何情况下，对于因使用本软件而以任何方式造成的任何直接性、间接性、偶然性、特殊性、惩罚性或后果性损失（包括但不限于购买替代商品或服务；使用、数据或利润方面的损失；或者业务中断），无论原因如何以及基于何种责任理论，无论出于合同、严格责任或侵权行为（包括疏忽或其他行为），NetApp 均不承担责任，即使已被告知存在上述损失的可能性。

NetApp 保留在不另行通知的情况下随时对本文档所述的任何产品进行更改的权利。除非 NetApp 以书面形式明确同意，否则 NetApp 不承担因使用本文档所述产品而产生的任何责任或义务。使用或购买本产品不表示获得 NetApp 的任何专利权、商标权或任何其他知识产权许可。

本手册中描述的产品可能受一项或多项美国专利、外国专利或正在申请的专利的保护。

有限权利说明：政府使用、复制或公开本文档受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中“技术数据权利 — 非商用”条款第 (b)(3) 条规定的限制条件的约束。

本文档中所含数据与商业产品和/或商业服务（定义见 FAR 2.101）相关，属于 NetApp, Inc. 的专有信息。根据本协议提供的所有 NetApp 技术数据和计算机软件具有商业性质，并完全由私人出资开发。 美国政府对这些数据的使用权具有非排他性、全球性、受限且不可撤销的许可，该许可既不可转让，也不可再许可，但仅限在与交付数据所依据的美国政府合同有关且受合同支持的情况下使用。除本文档规定的情形外，未经 NetApp, Inc. 事先书面批准，不得使用、披露、复制、修改、操作或显示这些数据。美国政府对国防部的授权仅限于 DFARS 的第 252.227-7015(b)（2014 年 2 月）条款中明确的权利。

商标信息

NetApp、NetApp 标识和 http://www.netapp.com/TM 上所列的商标是 NetApp, Inc. 的商标。其他公司和产品名称可能是其各自所有者的商标。