第 5 章
Slurm 并行训练

高性能计算（HPC）通常指的是以一种方式聚合计算能力的实践。该方式提供比从典型的台式计算机或工作站获得的性能高得多的性能，以便解决科学、工程或商业中的大问题。随着数据集的增长，高性能计算对训练模型变得越来越重要。

本章首先介绍如何使用网格计算引擎 Slurm 构建 Linux 高性能计算集群，然后介绍如何实现 TensorFlow 在 Slurm 集群的运行。

5.1 网格计算引擎 Slurm 简介

Slurm（Simple Linux Utility for Resource Management）是一个开源、容错、高度可扩展的集群管理和作业调度系统，适用于大型和小型 Linux 集群。不需要为了使用 Slurm 而修改操作系统内核。作为集群工作负载管理器，Slurm 有三个关键功能：首先，它在一段时间内为用户分配对资源（计算节点）的独占和/或非独占访问，以便执行程序；其次，它提供了一个框架，用于在分配的节点集上启动、执行和监视（通常是并行作业）； 最后，它通过管理待处理工作的队列来仲裁资源争用。

Slurm 已经使用 arm64（aarch64），ppc64 和 x86_64 架构在大多数流行的 Linux 发行版上进行了全面测试。目前支持的 Linux 发行版包括 Cray Linux Environment、Debian、RedHat Enterprise Linux、CentOS、Scientific Linux、SUSE Linux Enterprise Server、Ubuntu。

5.1.1 安装 Slurm

可以通过 Docker 的 Slurm 镜像来使用 Slurm 集群。

如果操作系统还没有安装应用容器引擎 Docker，则首先安装 Docker：

```
$ sudo apt install docker.io
```

https://hub.docker.com/r/giovtorres/docker-centos7-slurm/tags/提供了多个可用的标签。要使用最新的可用镜像，请运行：

```
#docker pull giovtorres/docker-centos7-slurm:latest
```

进入容器内的 bash shell。

```
#docker run -it -h ernie giovtorres/docker-centos7-slurm:latest
```

测试 slurmd 配置：

```
#slurmd -C
NodeName=ernie CPUs=16 Boards=1 SocketsPerBoard=2 CoresPerSocket=8 ThreadsPerCore=1
RealMemory=64393
```

supervisord 是一个进程管理工具。要查看所有进程的状态，请运行：

```
[root@ernie /]#supervisorctl status
munged                           RUNNING   pid 23, uptime 0:02:35
mysqld                           RUNNING   pid 24, uptime 0:02:35
slurmctld                         RUNNING   pid 25, uptime 0:02:35
slurmd                           RUNNING   pid 22, uptime 0:02:35
slurmdbd                          RUNNING   pid 26, uptime 0:02:35
```

查看配置文件 slurm.conf：

```
#cat /etc/slurm/slurm.conf
```

显示结果如下：

```
#集群名
ClusterName=linux
#主控制器的主机名
ControlMachine=ernie
#ControlAddr=
#BackupController=
#BackupAddr=
#slurm 进程用户
SlurmUser=slurm
#SlurmdUser=root
#slurmctld 控制器端口
SlurmctldPort=6817
```

```
#slurmd 节点守护进程端口
SlurmdPort=6818
#slurm 通信认证
AuthType=auth/munge
#JobCredentialPrivateKey=
#JobCredentialPublicCertificate=
#slurm 任务状态保存目录
StateSaveLocation=/var/lib/slurmd
#slurmd 守护进程日志保存
SlurmdSpoolDir=/var/spool/slurmd
SwitchType=switch/none
MpiDefault=none
#slurmctld 的 pid 存放
SlurmctldPidFile=/var/run/slurmd/slurmctld.pid
#slurmd 守护进程的 pid 文件存放
SlurmdPidFile=/var/run/slurmd/slurmd.pid
ProctrackType=proctrack/pgid
#PluginDir=
CacheGroups=0
#FirstJobId=
ReturnToService=0
#MaxJobCount=
#PlugStackConfig=
#PropagatePrioProcess=
#PropagateResourceLimits=
#PropagateResourceLimitsExcept=
SlurmctldTimeout=300
SlurmdTimeout=300
InactiveLimit=0
MinJobAge=300
KillWait=30
Waittime=0
#
#调度
SchedulerType=sched/backfill
#SchedulerAuth=
#SchedulerPort=
#SchedulerRootFilter=
SelectType=select/cons_res
SelectTypeParameters=CR_CPU_Memory
FastSchedule=1
#PriorityType=priority/multifactor
#PriorityDecayHalfLife=14-0
#PriorityUsageResetPeriod=14-0
#PriorityWeightFairshare=100000
```

```
    #PriorityWeightAge=1000
    #PriorityWeightPartition=10000
    #PriorityWeightJobSize=1000
    #PriorityMaxAge=1-0
    #
    #日志
    SlurmctldDebug=3
    #slurmctld 控制器守护进程的日志存放
    SlurmctldLogFile=/var/log/slurm/slurmctld.log
    SlurmdDebug=3
    SlurmdLogFile=/var/log/slurm/slurmd.log
    JobCompType=jobcomp/none
    #JobCompLoc=
    #
    #审计
    #JobAcctGatherType=jobacct_gather/linux
    #JobAcctGatherFrequency=30
    #
    AccountingStorageType=accounting_storage/slurmdbd
    #AccountingStorageHost=localhost
    #AccountingStorageLoc=
    #AccountingStoragePass=
    #AccountingStorageUser=
    #
    #计算节点
    GresTypes=gpu
    NodeName=c[1-5] NodeHostName=localhost NodeAddr=127.0.0.1 RealMemory=1000
    NodeName=c[6-10] NodeHostName=localhost NodeAddr=127.0.0.1 RealMemory=1000 Gres=gpu:
titanxp:1
    #
    #分区
    PartitionName=normal Default=yes Nodes=c[1-5] Priority=50 DefMemPerCPU=500 Shared=NO
MaxNodes=5 MaxTime=5-00:00:00 DefaultTime=5-00:00:00 State=UP
    PartitionName=debug Nodes=c[6-10] Priority=50 DefMemPerCPU=500 Shared=NO MaxNodes=5
MaxTime=5-00:00:00 DefaultTime=5-00:00:00 State=UP
```

在 slurm.conf 文件中，ControlMachine 主机名设置为 ernie。由于这是一体化安装，因此主机名必须与 ControlMachine 匹配。因此，必须在运行时将-h ernie 传递给 docker，以便主机名匹配。

如果对配置文件进行了少量更改，则可以使用 scontrol reconfig 命令让守护程序重新读取 slurm.conf。

可以运行命令 sinfo 报告由 Slurm 管理的分区和节点的状态：

```
    [root@ernie /]#sinfo
    PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
```

```
normal*      up 5-00:00:00      5   idle c[1-5]
debug        up 5-00:00:00      5   idle c[6-10]
```

scontrol 命令列出可以访问的分区：

```
[root@ernie /]#scontrol show partition
PartitionName=normal
   AllowGroups=ALL AllowAccounts=ALL AllowQos=ALL
   AllocNodes=ALL Default=YES QoS=N/A
   DefaultTime=5-00:00:00 DisableRootJobs=NO ExclusiveUser=NO GraceTime=0 Hidden=NO
  MaxNodes=1 MaxTime=5-00:00:00 MinNodes=1 LLN=NO MaxCPUsPerNode=UNLIMITED
  Nodes=c[1-5]
  PriorityJobFactor=50  PriorityTier=50  RootOnly=NO  ReqResv=NO  OverSubscribe=NO
PreemptMode=OFF
  State=UP TotalCPUs=5 TotalNodes=5 SelectTypeParameters=NONE
  DefMemPerCPU=500 MaxMemPerNode=UNLIMITED
```

如果需要从头开始安装 Slurm，则首先要确保时钟，用户和组（UID 和 GID）在群集中同步。Slurm 需要使用 MUNGE（https://dun.github.io/munge/）来认证，所以要先来安装 MUNGE。

```
#sudo apt-get install munge
```

确保群集中的所有节点都具有相同的 munge.key。确保在启动 Slurm 守护程序之前启动了 MUNGE 守护程序。

启动 MUNGE 守护进程：

```
#/etc/init.d/munge start
[ ok ] Starting munge (via systemctl): munge.service.
```

可以执行以下步骤来验证软件是否已正确安装和配置：

在 stdout 上生成凭证：

```
#munge -n
```

检查凭证是否可以在本地解码：

```
#munge -n | unmunge
```

检查凭证是否可以远程解码：

```
#munge -n | ssh <somehost> unmunge
```

如果遇到问题，请检查 munged 守护程序是否正在运行：

```
#/etc/init.d/munge status
```

安装数据库 MariaDB：

```
#sudo apt-get install mariadb-server mariadb-client
```

以 MariaDB root 用户身份登录。

```
#sudo mysql -u root -p
```

输入密码后，将进入 MariaDB shell。

如果要启动 MariaDB，可以使用以下命令。

```
#sudo systemctl start mariadb
```

可以使用以下命令停止 MariaDB：

```
#sudo systemctl stop mariadb
```

创建目录：

```
#mkdir -p  /storage
```

下载源代码：

```
#wget https://download.schedmd.com/slurm/slurm-18.08.7.tar.bz2
```

解压缩：

```
#tar xvjf slurm-18.08.7.tar.bz2
```

编译代码：

```
#cd slurm-18.08.7/
#./configure    --prefix=/tmp/slurm-build    --sysconfdir=/etc/slurm    --enable-pam
--with-pam_dir=/lib/x86_64-linux-gnu/security/ --without-shared-libslurm
```

编译 Slurm。

```
#make -j 4
#make contrib
#make install
```

安装 fpm：

```
#apt-get install ruby ruby-dev rubygems build-essential
#gem install --no-ri --no-rdoc fpm
```

用 fpm 制作安装包：

```
#fpm -s dir -t deb -v 1.0 -n slurm-18.08.7 --prefix=/usr -C /tmp/slurm-build .
```

用 dpkg 命令安装 deb 包：

```
#dpkg -i slurm-18.08.7_1.0_amd64.deb
```

测试 slurmd 配置：

```
#slurmd -C
```

使用 https://slurm.schedmd.com/configurator.easy.html 上的在线工具，您可以生成 slurm.conf 配置文件。

在主节点启动控制守护进程：

```
#slurmctld -c
```

在计算节点上启动 slurm 守护进程：

```
#slurmd -c
```

5.1.2　Slurm 脚本编程

使用 srun 命令直接提交适用于快速简单的单个任务。例如，在 3 个节点上执行/bin/hostname，并输出任务编号。

```
#srun -N3 -l /bin/hostname
```

输出结果如下：

```
0: ernie
2: ernie
1: ernie
```

这里使用默认分区。默认情况下，每个节点一个任务。srun 命令有许多选项可用于控制分配的资源以及如何在这些资源之间分配任务。

使用 sbatch 命令将批处理脚本提交给 Slurm。显示 sbatch 帮助信息：

```
#sbatch --help
```

例如，批处理脚本 my.script 内容如下：

```
adev0: cat my.script
#!/bin/sh
#SBATCH --time=1
/bin/hostname
srun -l /bin/hostname
srun -l /bin/pwd
```

此脚本包含嵌入其自身作业的时间限制。可以通过使用前缀“#SBATCH”后跟脚本开头的选项（在脚本中执行任何命令之前）来根据需要提供其他选项。例如，如果需要重新启动失败的工作作业，可以使用--requeue 选项。

```
#SBATCH --requeue                ###失败时，重新排队进行另一次尝试
```

在节点 adev0 上提交作业：

```
adev0: sbatch -n4 -w "adev[9-10]" -o my.stdout my.script
```

```
sbatch: Submitted batch job 469
```

在此示例中，脚本名称为 my.script，声明使用节点 adev9 和 adev10（-w "adev[9-10]"，请注意使用节点范围表达式）。还明确声明后续作业步骤将分别产生 4 个任务，这将确保分配包含至少 4 个处理器（每个任务要启动一个处理器）。输出将出现在文件 my.stdout（-o my.stdout）中。

my.script 包含在分配中的第一个节点上（脚本运行的位置）执行的命令/bin/hostname，加上使用 srun 命令发起的两个作业步骤并按顺序执行。

使用 squeue 命令查看作业队列中作业的信息。例如：

```
#squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
```

使用 scancel 命令取消已经提交的作业。例如：

```
#scancel 123456
```

Slurm 不会自动将可执行文件或数据文件迁移到分配给作业的节点。文件必须存在于本地磁盘或某些全局文件系统（例如 NFS 或 Lustre）中。 Slurm 提供工具 sbcast，使用 Slurm 的分层通信将文件传输到分配节点上的本地存储。

Toil 是一个完全用 Python 编写的可扩展、高效、动态的跨平台管道管理系统。Toil 曾用于在 4 天内使用 32000 个可抢占计算核心的商业云集群处理 20000 多个 RNA-seq 样本。Toil 还可以运行 Common Workflow Language 中描述的工作流，并支持各种调度程序，例如 Mesos、GridEngine、LSF、Parasol 和 Slurm。

5.2 TensorFlow 集群

为了让 TensorFlow 运行在 Slurm 集群，可以使用 sbatch 命令将批处理脚本简单地传递给 Slurm：

```
#sbatch --partition=part start.sh
```

可以使用 sinfo 列出可用分区。

start.sh 中可能的配置如下：

```
#!/bin/sh
#SBATCH -N 1                #请求的节点
#SBATCH -n 1                #请求的任务
#SBATCH -c 10               #请求的 CPU 核
#SBATCH --mem=32000         #以 MB 为单位的内存
#SBATCH -o outfile          #将 stdout 发送到 outfile 文件
```

```
#SBATCH -e errfile  #将 stderr 发送到 errfile 文件
python run.py
```

这里的 run.py 包含想用 slurm 执行的脚本，即 TensorFlow 代码。

5.3　本章小结

本章介绍了如何使用网格计算引擎 Slurm 构建 Linux 高性能计算集群和如何实现 TensorFlow 在 Slurm 集群的运行。