SLURM - Simple Linux Utility for Resource Management

Introduction

Slurm is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters.

It provides three key functions:

- allocating exclusive and/or non-exclusive access to resources (computer nodes) to users for some duration of time so they can perform work,
- providing a framework for starting, executing, and monitoring work (typically a parallel job such as MPI) on a set of allocated nodes, and
- arbitrating contention for resources by managing a queue of pending jobs.

Installation

Controller name: slurm-ctrl

Install slurm-wlm and tools

```
ssh slurm-ctrl
apt install slurm-wlm slurm-wlm-doc mailutils mariadb-client mariadb-server
libmariadb-dev python-dev python-mysqldb
```

Install Maria DB Server

```
apt-get install mariadb-server
systemctl start mysql
mysql -u root
create database slurm_acct_db;
create user 'slurm'@'localhost';
set password for 'slurm'@'localhost' = password('slurmdbpass');
grant usage on *.* to 'slurm'@'localhost';
grant all privileges on slurm_acct_db.* to 'slurm'@'localhost';
flush privileges;
exit
```

In the file /etc/mysql/mariadb.conf.d/50-server.cnf we should have the following setting:

```
vi /etc/mysql/mariadb.conf.d/50-server.cnf
```

```
bind-address = localhost
```

Node Authentication

First, let us configure the default options for the munge service:

```
vi /etc/default/munge
OPTIONS="--syslog --key-file /etc/munge/munge.key"
```

Central Controller

The main configuration file is /etc/slurm-llnl/slurm.conf this file has to be present in the controller and *ALL* of the compute nodes and it also has to be consistent between all of them.

```
vi /etc/slurm-llnl/slurm.conf
```

```
################################
# /etc/slurm-llnl/slurm.conf
################################
# slurm.conf file generated by configurator easy.html.
# Put this file on all nodes of your cluster.
# See the slurm.conf man page for more information.
#
ControlMachine=slurm-ctrl
#ControlAddr=10.7.20.97
#
#MailProg=/bin/mail
MpiDefault=none
#MpiParams=ports=#-#
ProctrackType=proctrack/pgid
ReturnToService=1
SlurmctldPidFile=/var/run/slurm-llnl/slurmctld.pid
##SlurmctldPidFile=/var/run/slurmctld.pid
#SlurmctldPort=6817
SlurmdPidFile=/var/run/slurm-llnl/slurmd.pid
##SlurmdPidFile=/var/run/slurmd.pid
#SlurmdPort=6818
SlurmdSpoolDir=/var/spool/slurmd
SlurmUser=slurm
#SlurmdUser=root
StateSaveLocation=/var/spool
SwitchType=switch/none
TaskPlugin=task/none
#
#
# TIMERS
#KillWait=30
#MinJobAge=300
```

```
#SlurmctldTimeout=120
#SlurmdTimeout=300
#
#
# SCHEDULING
FastSchedule=1
SchedulerType=sched/backfill
SelectType=select/linear
#SelectTypeParameters=
#
#
# LOGGING AND ACCOUNTING
AccountingStorageType=accounting_storage/none
ClusterName=cluster
#JobAcctGatherFrequency=30
JobAcctGatherType=jobacct_gather/none
#SlurmctldDebug=3
SlurmctldLogFile=/var/log/slurm-llnl/SlurmctldLogFile
#SlurmdDebug=3
SlurmdLogFile=/var/log/slurm-llnl/SlurmLogFile
#
#
# COMPUTE NODES
NodeName=linux1 NodeAddr=10.7.20.98 CPUs=1 State=UNKNOWN
```

Copy slurm.conf to compute nodes!

```
root@slurm-ctrl# scp /etc/slurm-llnl/slurm.conf csadmin@10.7.20.109:/tmp/.;
scp /etc/slurm-llnl/slurm.conf csadmin@10.7.20.110:/tmp/.
```

```
vi /lib/systemd/system/slurmctld.service
```

```
[Unit]
Description=Slurm controller daemon
After=network.target munge.service
ConditionPathExists=/etc/slurm-llnl/slurm.conf
Documentation=man:slurmctld(8)

[Service]
Type=forking
EnvironmentFile=-/etc/default/slurmctld
ExecStart=/usr/sbin/slurmctld $SLURMCTLD_OPTIONS
ExecStartPost=/bin/sleep 2
ExecReload=/bin/kill -HUP $MAINPID
PIDFile=/var/run/slurm-llnl/slurmctld.pid

[Install]
WantedBy=multi-user.target
```

```
vi /lib/systemd/system/slurmd.service
```

```
[Unit]
Description=Slurm node daemon
After=network.target munge.service
ConditionPathExists=/etc/slurm-llnl/slurm.conf
Documentation=man:slurmd(8)

[Service]
Type=forking
EnvironmentFile=-/etc/default/slurmd
ExecStart=/usr/sbin/slurmd $SLURMD_OPTIONS
ExecStartPost=/bin/sleep 2
ExecReload=/bin/kill -HUP $MAINPID
PIDFile=/var/run/slurm-llnl/slurmd.pid
KillMode=process
LimitNOFILE=51200
LimitMEMLOCK=infinity
LimitSTACK=infinity

[Install]
WantedBy=multi-user.target
```

```
root@slurm-ctrl# systemctl daemon-reload
root@slurm-ctrl# systemctl enable slurmdbd
root@slurm-ctrl# systemctl start slurmdbd
root@slurm-ctrl# systemctl enable slurmctld
root@slurm-ctrl# systemctl start slurmctld
```

Accounting Storage

After we have the slurm-llnl-slurmdbd package installed we configure it, by editing the /etc/slurm-llnl/slurmdbd.conf file:

```
vi /etc/slurm-llnl/slurmdbd.conf
```

```
#
# /etc/slurm-llnl/slurmdbd.conf is an ASCII file which describes Slurm
# Database Daemon (SlurmDBD) configuration information.
# The contents of the file are case insensitive except for the names of
# nodes and files. Any text following a "#" in the configuration file is
# treated as a comment through the end of that line. The size of each
# line in the file is limited to 1024 characters. Changes to the
# configuration file take effect upon restart of SlurmDbd or daemon
# receipt of the SIGHUP signal unless otherwise noted.
#
# This file should be only on the computer where SlurmDBD executes and
# should only be readable by the user which executes SlurmDBD (e.g.
# "slurm"). This file should be protected from unauthorized access since
# it contains a database password.
```

```
###############################################################################
AuthType=auth/munge
AuthInfo=/var/run/munge/munge.socket.2
StorageHost=localhost
StoragePort=3306
StorageUser=slurm
StoragePass=slurmdbpass
StorageType=accounting_storage/mysql
StorageLoc=slurm_acct_db
LogFile=/var/log/slurm-llnl/slurmdbd.log
PidFile=/var/run/slurm-llnl/slurmdbd.pid
SlurmUser=slurm
```

```
root@slurm-ctrl# systemctl start slurmdbd
```

Authentication

Copy /etc/munge.key to all compute nodes

```
scp /etc/munge/munge.key csadmin@10.7.20.98:/tmp/.
```

Allow password-less access to slurm-ctrl

csadmin@slurm-ctrl:~$ ssh-copy-id -i .ssh/id_rsa.pub 10.7.20.102:

Run a job from slurm-ctrl

```
ssh csadmin@slurm-ctrl
srun -N 1 hostname
linux1
```

Test munge

```
munge -n | unmunge | grep STATUS
STATUS:           Success (0)
munge -n | ssh slurm-ctrl unmunge | grep STATUS
STATUS:           Success (0)
```

Test Slurm

```
sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug*       up   infinite      1   idle linux1
```

If computer node is down or drain

```
sinfo -a
```

```
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug*       up   infinite      2   down gpu[02-03]

sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
gpu*         up   infinite      1  drain gpu02
gpu*         up   infinite      1   down gpu03
```

```
scontrol update nodename=gpu02 state=idle
scontrol update nodename=gpu03 state=idle
scontrol update nodename=gpu02 state=resume
```

```
sinfo -a
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug*       up   infinite      2   idle gpu[02-03]
```

```
sinfo -o "%20N  %10c  %10m  %25f  %10G "
NODELIST              CPUS        MEMORY      AVAIL_FEATURES
GRES
gpu[02-03]            32          190000      (null)
gpu:2
gpu04                 64          1000000     (null)
gpu:4(S:0)
hpcmoi01,hpcwrk01     32+         190000+     (null)
(null)
```

Compute Nodes

A compute node is a machine which will receive jobs to execute, sent from the Controller, it runs the slurmd service.

Installation slurm and munge

```
ssh -l csadmin <compute-nodes> 10.7.20.109 10.7.20.110
sudo apt install slurm-wlm libmunge-dev libmunge2 munge
```

```
sudo vi /lib/systemd/system/slurmd.service
```

```
[Unit]
Description=Slurm node daemon
After=network.target munge.service
ConditionPathExists=/etc/slurm-llnl/slurm.conf
Documentation=man:slurmd(8)

[Service]
Type=forking
```

```
EnvironmentFile=-/etc/default/slurmd
ExecStart=/usr/sbin/slurmd $SLURMD_OPTIONS
ExecStartPost=/bin/sleep 2
ExecReload=/bin/kill -HUP $MAINPID
PIDFile=/var/run/slurm-llnl/slurmd.pid
KillMode=process
LimitNOFILE=51200
LimitMEMLOCK=infinity
LimitSTACK=infinity

[Install]
WantedBy=multi-user.target
```

```
sudo systemctl enable slurmd
sudo systemctl enable munge
sudo systemctl start slurmd
sudo systemctl start munge
```

Generate ssh keys

```
ssh-keygen
```

Copy ssh-keys to slurm-ctrl

```
ssh-copy-id -i ~/.ssh/id_rsa.pub csadmin@slurm-ctrl.inf.unibz.it:
```

Become root to do important things:

```
sudo -i
vi /etc/hosts
```

Add those lines below to the /etc/hosts file

```
10.7.20.97      slurm-ctrl.inf.unibz.it slurm-ctrl
10.7.20.98      linux1.inf.unibz.it     linux1
```

First copy the munge keys from the slurm-ctrl to all compute nodes, now fix location, owner and permission.

```
mv /tmp/munge.key /etc/munge/.
chown munge:munge /etc/munge/munge.key
chmod 400 /etc/munge/munge.key
```

Place /etc/slurm-llnl/slurm.conf in right place,

```
mv /tmp/slurm.conf /etc/slurm-llnl/
chown root: /etc/slurm-llnl/slurm.conf
```

Directories

Be sure that the nfs mounted partitions are, all there:

```
/data
/opt/packages
/home/clusterusers
/opt/modules
/scratch
```

Modify user accounts

Display the accounts created:

```
# Show also associations in the accounts
sacctmgr show account -s
# Show all columns separated by pipe | symbol
sacctmgr show account -s -P
#
sacctmgr show user -s
```

Add user

```
sacctmgr add user <username> Account=gpu-users Partition=gpu
```

Modify user, give 12000 minutes/200 hours for usage

```
sacctmgr modify user <username> set GrpTRESMin=cpu=12000,gres/gpu=12000
```

Modify user by removing it from certain account

```
sacctmgr remove user where user=<username> and account=<account>
```

Delete user

```
sacctmgr delete user ivmilan
Deleting users...
ivmilan
Would you like to commit changes? (You have 30 seconds to decide)
(N/y): y
```

Restart the services:

```
systemctl restart slurmctld.service
systemctl restart slurmdbd.service
```

Check status:

```
systemctl status slurmctld.service
systemctl status slurmdbd.service
```

Submit a job to a specific node using Slurm's sbatch command

To run a job on a specific Node use this option in the job script

```
#SBATCH --nodelist=gpu03
```

Links

Basic Configuration and Usage

Slurm Workload Manager Overview

Steps to create a small slurm cluster with GPU enabled nodes

Slurm in Ubuntu Clusters Part1

Slurm batch queueing system

SLURM Workload Manager

Slurm Quick Start Tutorial

Linux Clusters Institute: Scheduling and Resource Management 2017

Modules

The Environment Modules package provides for the dynamic modification of a user's environment via modulefiles.

Installing Modules on Unix

Login into slurm-ctrl and become root

```
ssh slurm-ctrl
sudo -i
```

Download modules

```
curl -LJO
https://github.com/cea-hpc/modules/releases/download/v4.6.0/modules-4.6.0.ta
r.gz
tar xfz modules-4.6.0.tar.gz
```

```
cd modules-4.6.0
```

```
$ ./configure --prefix=/opt/modules
$ make
$ make install
```

https://modules.readthedocs.io/en/stable/index.html

SPACK

Add different python versions using spack!

1. First see which python versions are available:

```
root@slurm-ctrl:~# spack versions python
==> Safe versions (already checksummed):
3.8.2  3.7.7  3.7.4  3.7.1  3.6.7  3.6.4  3.6.1  3.5.2  3.4.10  3.2.6
2.7.17  2.7.14  2.7.11  2.7.8
3.8.1  3.7.6  3.7.3  3.7.0  3.6.6  3.6.3  3.6.0  3.5.1  3.4.3   3.1.5
2.7.16  2.7.13  2.7.10
3.8.0  3.7.5  3.7.2  3.6.8  3.6.5  3.6.2  3.5.7  3.5.0  3.3.6   2.7.18
2.7.15  2.7.12  2.7.9
```

==⇒ Remote versions (not yet checksummed):

```
3.10.0a6  3.8.7rc1  3.7.6rc1   3.6.8rc1   3.5.7rc1   3.4.9     3.4.0
3.1.2rc1   2.7.9rc1  2.6.6     2.4.5
3.10.0a5  3.8.7 ....
...
...
```

2. now select the python version you would like to install:

```
root@slurm-ctrl:~# spack install python@3.8.2
==> 23834: Installing libiconv
==> Using cached archive: /opt/packages/spack/var/spack/cache/_source-
cache/archive/e6/e6a1b1b589654277ee790cce3734f07876ac4ccfaecbee8afa0b649cf52
9cc04.tar.gz
==> Staging archive: /tmp/root/spack-stage/spack-stage-libiconv-1.16-
b2wenwxf2widzewcvnhsxtjyisz3bcmc/libiconv-1.16.tar.gz
==> Created stage in /tmp/root/spack-stage/spack-stage-libiconv-1.16-
b2wenwxf2widzewcvnhsxtjyisz3bcmc
==> No patches needed for libiconv
==> 23834: libiconv: Building libiconv [AutotoolsPackage]
==> 23834: libiconv: Executing phase: 'autoreconf'
==> 23834: libiconv: Executing phase: 'configure'
==> 23834: libiconv: Executing phase: 'build'
```

```
==> 23834: libiconv: Executing phase: 'install'
==> 23834: libiconv: Successfully installed libiconv
Fetch: 0.04s.  Build: 24.36s.  Total: 24.40s.
[+] /opt/packages/spack/opt/spack/linux-ubuntu18.04-
skylake_avx512/gcc-9.3.0/libiconv-1.16-b2wenwxf2widzewcvnhsxtjyisz3bcmc
==> 23834: Installing libbsd
...
...
...
==> 23834: Installing python
==> Fetching https://www.python.org/ftp/python/3.8.2/Python-3.8.2.tgz
######################################################################
################################## 100.0%
==> Staging archive: /tmp/root/spack-stage/spack-stage-python-3.8.2-
vmyztzplzddt2arrsx7d7koebyuzvk6s/Python-3.8.2.tgz
==> Created stage in /tmp/root/spack-stage/spack-stage-python-3.8.2-
vmyztzplzddt2arrsx7d7koebyuzvk6s
==> Ran patch() for python
==> 23834: python: Building python [AutotoolsPackage]
==> 23834: python: Executing phase: 'autoreconf'
==> 23834: python: Executing phase: 'configure'
==> 23834: python: Executing phase: 'build'
==> 23834: python: Executing phase: 'install'
==> 23834: python: Successfully installed python
Fetch: 1.81s.  Build: 1m 42.11s.  Total: 1m 43.91s.
[+] /opt/packages/spack/opt/spack/linux-ubuntu18.04-
skylake_avx512/gcc-9.3.0/python-3.8.2-vmyztzplzddt2arrsx7d7koebyuzvk6s
```

This will take some minutes time, depending on the type of version

3. Now you need to add a modules file

```
root@slurm-ctrl:~# vi /opt/modules/modulefiles/python-3.8.2
```

```
#%Module1.0
proc ModulesHelp { } {
global dotversion
puts stderr "\tPython 3.8.2"
}

module-whatis "Python 3.8.2"

set     main_root       /opt/packages/spack/opt/spack/linux-ubuntu18.04-
skylake_avx512/gcc-9.3.0/python-3.8.2-vmyztzplzddt2arrsx7d7koebyuzvk6s
set-alias       python3.8       /opt/packages/spack/opt/spack/linux-
ubuntu18.04-skylake_avx512/gcc-9.3.0/python-3.8.2-
vmyztzplzddt2arrsx7d7koebyuzvk6s/bin/python3.8

prepend-path    PATH    $main_root/bin
prepend-path    LIBRARY_PATH    $main_root/lib
```

4. New module should now be available:

```
root@slurm-ctrl:~# module avail
------------------------------------------- /opt/modules/modulefiles ------
-----------------------------------
anaconda3  cuda-11.2.1  intel-mpi             module-info  py-mpi4py
python-3.7.7       use.own
bzip       dot          intel-mpi-benchmarks  modules      python-2.7.18
python-3.8.2
cuda-10.2  gcc-6.5.0    miniconda3            null         python-3.5.7
python-3.9.2
cuda-11.0  go-1.15.3    module-git            openmpi      python-3.6.10
singularity-3.6.4
```

5. Load the new module

```
root@slurm-ctrl:~# module load python-3.8.2
```

6. Verify it works

```
root@slurm-ctrl:~# python3.8
Python 3.8.2 (default, Mar 19 2021, 11:05:37)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
```

7. Unload the new module

```
module unload python-3.8.2
```

Python

Python 3.7.7

```
cd /opt/packages
mkdir /opt/packages/python/3.7.7
wget https://www.python.org/ftp/python/3.7.7/Python-3.7.7.tar.xz
tar xfJ Python-3.7.7.tar.xz
cd Python-3.7.7/
./configure --prefix=/opt/packages/python/3.7.7/ --enable-optimizations
make
make install
```

Python 2.7.18

```
cd /opt/packages
mkdir /opt/packages/python/2.7.18
```

```
wget https://www.python.org/ftp/python/2.7.18/Python-2.7.18.tar.xz
cd Python-2.7.18
./configure --prefix=/opt/packages/python/2.7.18/ --enable-optimizations
make
make install
```

Create modules file

PYTHON

```
cd /opt/modules/modulefiles/
vi python-2.7.18
```

```
#%Module1.0
proc ModulesHelp { } {
global dotversion

puts stderr "\tPython 2.7.18"
}

module-whatis "Python 2.7.18"
prepend-path PATH /opt/packages/python/2.7.18/bin
```

CUDA

```
vi /opt/modules/modulefiles/cuda-10.2
```

```
#%Module1.0
proc ModulesHelp { } {
global dotversion

puts stderr "\tcuda-10.2"
}

module-whatis "cuda-10.2"

set     prefix  /usr/local/cuda-10.2

setenv          CUDA_HOME       $prefix
prepend-path    PATH            $prefix/bin
prepend-path    LD_LIBRARY_PATH $prefix/lib64
```

GCC

This takes a long time!

Commands to run to compile gcc-6.1.0

```
wget https://ftp.gnu.org/gnu/gcc/gcc-6.1.0/gcc-6.1.0.tar.bz2
tar xfj gcc-6.1.0.tar.bz2
cd gcc-6.1.0
./contrib/download_prerequisites
./configure --prefix=/opt/package/gcc/6.1.0 --disable-multilib
make
```

After some time an error occurs, and the make process stops!

```
...
In file included from ../.././libgcc/unwind-dw2.c:401:0:
./md-unwind-support.h: In function ‘x86_64_fallback_frame_state’:
./md-unwind-support.h:65:47: error: dereferencing pointer to incomplete type
‘struct ucontext’
       sc = (struct sigcontext *) (void *) &uc_->uc_mcontext;
                                                ^~
../.././libgcc/shared-object.mk:14: recipe for target 'unwind-dw2.o' failed
```

To fix do: solution

```
vi /opt/packages/gcc-6.1.0/x86_64-pc-linux-gnu/libgcc/md-unwind-support.h
```

and replace/comment out line 61 with this:

```
struct ucontext_t *uc_ = context->cfa;
```

old line: /* struct ucontext *uc_ = context→cfa; */

```
make
```

Next error:

```
../../../../libsanitizer/sanitizer_common/sanitizer_stoptheworld_linux_libcde
p.cc:270:22: error: aggregate ‘sigaltstack handler_stack’ has incomplete
type and cannot be defined
   struct sigaltstack handler_stack;
```

To fix see: solution or https://gcc.gnu.org/bugzilla/show_bug.cgi?id=81066

Amend the files according to solution above!

Next error:

```
...
checking for unzip... unzip
configure: error: cannot find neither zip nor jar, cannot continue
Makefile:23048: recipe for target 'configure-target-libjava' failed
...
...
```

```
apt install unzip zip
```

and run make again!

```
make
```

Next error:

```
...
In file included from ../.././libjava/prims.cc:26:0:
../.././libjava/prims.cc: In function ‘void _Jv_catch_fpe(int, siginfo_t*,
void*)’:
./include/java-signal.h:32:26: error: invalid use of incomplete type ‘struct
_Jv_catch_fpe(int, siginfo_t*, void*)::ucontext’
   gregset_t &_gregs = _uc->uc_mcontext.gregs;    \
...
```

Edit the file: /opt/packages/gcc-6.1.0/x86_64-pc-linux-gnu/libjava/include/java-signal.h

```
vi /opt/packages/gcc-6.1.0/x86_64-pc-linux-gnu/libjava/include/java-signal.h
```

Not enough more errors!

```
// kh
  ucontext_t *_uc = (ucontext_t *);                                  \
  //struct ucontext *_uc = (struct ucontext *)_p;
\
  // kh
```

Next error:

```
...
In file included from ../.././libjava/prims.cc:26:0:
./include/java-signal.h:32:3: warning: multi-line comment [-Wcomment]
   //struct ucontext *_uc = (struct ucontext *)_p;   \
   ^
../.././libjava/prims.cc: In function ‘void _Jv_catch_fpe(int, siginfo_t*,
void*)’:
./include/java-signal.h:31:15: warning: unused variable ‘_uc’ [-Wunused-
variable]
   ucontext_t *_uc = (ucontext_t *)_p;     \
               ^
../.././libjava/prims.cc:192:3: note: in expansion of macro
‘HANDLE_DIVIDE_OVERFLOW’
   HANDLE_DIVIDE_OVERFLOW;
   ^~~~~~~~~~~~~~~~~~~~~~
../.././libjava/prims.cc:203:1: error: expected ‘while’ before ‘jboolean’
 jboolean
 ^~~~~~~
../.././libjava/prims.cc:203:1: error: expected ‘(’ before ‘jboolean’
```

```
../../../libjava/prims.cc:204:1: error: expected primary-expression before
‘_Jv_equalUtf8Consts’
 _Jv_equalUtf8Consts (const Utf8Const* a, const Utf8Const *b)
 ^~~~~~~~~~~~~~~~~~
../../../libjava/prims.cc:204:1: error: expected ‘)’ before
‘_Jv_equalUtf8Consts’
../../../libjava/prims.cc:204:1: error: expected ‘;’ before
‘_Jv_equalUtf8Consts’
../../../libjava/prims.cc:204:22: error: expected primary-expression before
‘const’
 _Jv_equalUtf8Consts (const Utf8Const* a, const Utf8Const *b)
...
```

Examples

Example mnist

An simple example to use nvidia GPU!

The example consists of the following files:

- README.md
- requirements.txt
- main.job
- main.py

Create a folder mnist and place the 4 files in there.

```
mkdir mnist
```

cat README.md

````
# Basic MNIST Example

```bash
pip install -r requirements.txt
python main.py
# CUDA_VISIBLE_DEVICES=2 python main.py  # to specify GPU id to ex. 2
```
````

```
cat requirements.txt
```

```
torch
torchvision
```

```
cat main.job
```

```
#!/bin/bash
```

```
#SBATCH --job-name=mnist
#SBATCH --output=mnist.out
#SBATCH --error=mnist.err

#SBATCH --partition gpu
#SBATCH --gres=gpu
#SBATCH --mem-per-cpu=4gb
#SBATCH --nodes 2
#SBATCH --time=00:08:00

#SBATCH --ntasks=10

#SBATCH --mail-type=ALL
#SBATCH --mail-user=<your-email@address.com>

ml load miniconda3
python3 main.py
```

Remove your-email@address.com and add your e-mail address.

main.py

from __future__ import print_function import argparse import torch import torch.nn as nn import torch.nn.functional as F import torch.optim as optim from torchvision import datasets, transforms from torch.optim.lr_scheduler import StepLR class Net(nn.Module): def __init__(self): super(Net, self).__init__() self.conv1 = nn.Conv2d(1, 32, 3, 1) self.conv2 = nn.Conv2d(32, 64, 3, 1) self.dropout1 = nn.Dropout2d(0.25) self.dropout2 = nn.Dropout2d(0.5) self.fc1 = nn.Linear(9216, 128) self.fc2 = nn.Linear(128, 10) def forward(self, x): x = self.conv1(x) x = F.relu(x) x = self.conv2(x) x = F.max_pool2d(x, 2) x = self.dropout1(x) x = torch.flatten(x, 1) x = self.fc1(x) x = F.relu(x) x = self.dropout2(x) x = self.fc2(x) output = F.log_softmax(x, dim=1) return output def train(args, model, device, train_loader, optimizer, epoch): model.train() for batch_idx, (data, target) in enumerate(train_loader): data, target = data.to(device), target.to(device) optimizer.zero_grad() output = model(data) loss = F.nll_loss(output, target) loss.backward() optimizer.step() if batch_idx % args.log_interval == 0: print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(epoch, batch_idx * len(data), len(train_loader.dataset), 100. * batch_idx / len(train_loader), loss.item())) def test(args, model, device, test_loader): model.eval() test_loss = 0 correct = 0 with torch.no_grad(): for data, target in test_loader: data, target = data.to(device), target.to(device) output = model(data) test_loss += F.nll_loss(output, target, reduction='sum').item() # sum up batch loss pred = output.argmax(dim=1, keepdim=True) # get the index of the max log-probability correct += pred.eq(target.view_as(pred)).sum().item() test_loss /= len(test_loader.dataset) print('\nTest set: Average loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)\n'.format(test_loss, correct, len(test_loader.dataset), 100. * correct / len(test_loader.dataset))) def main(): # Training settings parser = argparse.ArgumentParser(description='PyTorch MNIST Example') parser.add_argument('--batch-size', type=int, default=64, metavar='N', help='input batch size for training (default: 64)') parser.add_argument('--test-batch-size', type=int, default=1000, metavar='N', help='input batch size for testing (default: 1000)') parser.add_argument('--epochs', type=int, default=14, metavar='N', help='number of epochs to train (default: 14)') parser.add_argument('--lr', type=float, default=1.0, metavar='LR', help='learning rate (default: 1.0)') parser.add_argument('--gamma', type=float, default=0.7, metavar='M', help='Learning rate step gamma (default: 0.7)') parser.add_argument('--no-cuda', action='store_true',

default=False, help='disables CUDA training') parser.add_argument('--seed', type=int, default=1, metavar='S', help='random seed (default: 1)') parser.add_argument('--log-interval', type=int, default=10, metavar='N', help='how many batches to wait before logging training status') parser.add_argument('--save-model', action='store_true', default=False, help='For Saving the current Model') args = parser.parse_args() use_cuda = not args.no_cuda and torch.cuda.is_available() torch.manual_seed(args.seed) device = torch.device("cuda" if use_cuda else "cpu") kwargs = {'num_workers': 1, 'pin_memory': True} if use_cuda else {} train_loader = torch.utils.data.DataLoader(datasets.MNIST('../data', train=True, download=True, transform=transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.1307,), (0.3081,))])), batch_size=args.batch_size, shuffle=True, **kwargs) test_loader = torch.utils.data.DataLoader(datasets.MNIST('../data', train=False, transform=transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.1307,), (0.3081,))])), batch_size=args.test_batch_size, shuffle=True, **kwargs) model = Net().to(device) optimizer = optim.Adadelta(model.parameters(), lr=args.lr) scheduler = StepLR(optimizer, step_size=1, gamma=args.gamma) for epoch in range(1, args.epochs + 1): train(args, model, device, train_loader, optimizer, epoch) test(args, model, device, test_loader) scheduler.step() if args.save_model: torch.save(model.state_dict(), "mnist_cnn.pt") if __name__ == '__main__': main()

```
1.
2.
3. from __future__ import print_function
4. import argparse
5. import torch
6. import torch.nn as nn
7. import torch.nn.functional as F
8. import torch.optim as optim
9. from torchvision import datasets, transforms
10. from torch.optim.lr_scheduler import StepLR
11.
12.
13. class Net(nn.Module):
14.     def __init__(self):
15.         super(Net, self).__init__()
16.         self.conv1 = nn.Conv2d(1, 32, 3, 1)
17.         self.conv2 = nn.Conv2d(32, 64, 3, 1)
18.         self.dropout1 = nn.Dropout2d(0.25)
19.         self.dropout2 = nn.Dropout2d(0.5)
20.         self.fc1 = nn.Linear(9216, 128)
21.         self.fc2 = nn.Linear(128, 10)
22.
23.     def forward(self, x):
24.         x = self.conv1(x)
25.         x = F.relu(x)
26.         x = self.conv2(x)
27.         x = F.max_pool2d(x, 2)
28.         x = self.dropout1(x)
29.         x = torch.flatten(x, 1)
30.         x = self.fc1(x)
31.         x = F.relu(x)
32.         x = self.dropout2(x)
33.         x = self.fc2(x)
34.         output = F.log_softmax(x, dim=1)
35.         return output
36.
```

```
37.
38. def train(args, model, device, train_loader, optimizer, epoch):
39.     model.train()
40.     for batch_idx, (data, target) in enumerate(train_loader):
41.         data, target = data.to(device), target.to(device)
42.         optimizer.zero_grad()
43.         output = model(data)
44.         loss = F.nll_loss(output, target)
45.         loss.backward()
46.         optimizer.step()
47.         if batch_idx % args.log_interval == 0:
48.             print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss:
    {:.6f}'.format(
49.                 epoch, batch_idx * len(data), len(train_loader.dataset),
50.                 100. * batch_idx / len(train_loader), loss.item()))
51.
52.
53. def test(args, model, device, test_loader):
54.     model.eval()
55.     test_loss = 0
56.     correct = 0
57.     with torch.no_grad():
58.         for data, target in test_loader:
59.             data, target = data.to(device), target.to(device)
60.             output = model(data)
61.             test_loss += F.nll_loss(output, target,
    reduction='sum').item()  # sum up batch loss
62.             pred = output.argmax(dim=1, keepdim=True)  # get the index
    of the max log-probability
63.             correct += pred.eq(target.view_as(pred)).sum().item()
64.
65.     test_loss /= len(test_loader.dataset)
66.
67.     print('\nTest set: Average loss: {:.4f}, Accuracy: {}/{}
    ({:.0f}%)\n'.format(
68.         test_loss, correct, len(test_loader.dataset),
69.         100. * correct / len(test_loader.dataset)))
70.
71.
72. def main():
73.     # Training settings
74.     parser = argparse.ArgumentParser(description='PyTorch MNIST
    Example')
75.     parser.add_argument('--batch-size', type=int, default=64,
    metavar='N',
76.                         help='input batch size for training (default:
    64)')
77.     parser.add_argument('--test-batch-size', type=int, default=1000,
    metavar='N',
78.                         help='input batch size for testing (default:
    1000)')
79.     parser.add_argument('--epochs', type=int, default=14, metavar='N',
80.                         help='number of epochs to train (default: 14)')
81.     parser.add_argument('--lr', type=float, default=1.0, metavar='LR',
82.                         help='learning rate (default: 1.0)')
83.     parser.add_argument('--gamma', type=float, default=0.7, metavar='M',
```

```
                        help='Learning rate step gamma (default: 0.7)')
    parser.add_argument('--no-cuda', action='store_true', default=False,
                        help='disables CUDA training')
    parser.add_argument('--seed', type=int, default=1, metavar='S',
                        help='random seed (default: 1)')
    parser.add_argument('--log-interval', type=int, default=10, metavar='N',
                        help='how many batches to wait before logging training status')

    parser.add_argument('--save-model', action='store_true', default=False,
                        help='For Saving the current Model')
    args = parser.parse_args()
    use_cuda = not args.no_cuda and torch.cuda.is_available()

    torch.manual_seed(args.seed)

    device = torch.device("cuda" if use_cuda else "cpu")

    kwargs = {'num_workers': 1, 'pin_memory': True} if use_cuda else {}
    train_loader = torch.utils.data.DataLoader(
        datasets.MNIST('../data', train=True, download=True,
                       transform=transforms.Compose([
                           transforms.ToTensor(),
                           transforms.Normalize((0.1307,), (0.3081,))
                       ])),
        batch_size=args.batch_size, shuffle=True, **kwargs)
    test_loader = torch.utils.data.DataLoader(
        datasets.MNIST('../data', train=False, transform=transforms.Compose([
                           transforms.ToTensor(),
                           transforms.Normalize((0.1307,), (0.3081,))
                       ])),
        batch_size=args.test_batch_size, shuffle=True, **kwargs)

    model = Net().to(device)
    optimizer = optim.Adadelta(model.parameters(), lr=args.lr)

    scheduler = StepLR(optimizer, step_size=1, gamma=args.gamma)
    for epoch in range(1, args.epochs + 1):
        train(args, model, device, train_loader, optimizer, epoch)
        test(args, model, device, test_loader)
        scheduler.step()

    if args.save_model:
        torch.save(model.state_dict(), "mnist_cnn.pt")


if __name__ == '__main__':
    main()


```

Once you have all files launch this command on slurm-ctrl:

```
sbatch main.job
```

Check your job with

```
squeue
```

CUDA NVIDIA TESLA Infos

nvidia-smi

```
root@gpu02:~# watch nvidia-smi
```

```
Every 2.0s: nvidia-smi                                          gpu02: Mon
Jun 22 17:49:14 2020

Mon Jun 22 17:49:14 2020
+-----------------------------------------------------------------------------
--+
| NVIDIA-SMI 440.64.00    Driver Version: 440.64.00    CUDA Version: 10.2
|
|-------------------------------+----------------------+----------------------
--+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr.
ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute
M. |
|===============================+======================+======================
==|
|   0  Tesla V100-PCIE...  On   | 00000000:3B:00.0 Off |
0 |
| N/A   53C    P0   139W / 250W |  31385MiB / 32510MiB |     69%
Default |
+-------------------------------+----------------------+----------------------
--+
|   1  Tesla V100-PCIE...  On   | 00000000:AF:00.0 Off |
0 |
| N/A   35C    P0    26W / 250W |      0MiB / 32510MiB |      0%
Default |
+-------------------------------+----------------------+----------------------
--+

+-----------------------------------------------------------------------------
--+
| Processes:                                                       GPU
Memory |
|  GPU       PID   Type   Process name                             Usage
|
```

```
|=============================================================================
==|
|    0      8627      C   /opt/anaconda3/bin/python3
31373MiB |
+-----------------------------------------------------------------------------
--+
```

deviceQuery

To run the deviceQuery it is necessary to make it first!

```
root@gpu03:~# cd /usr/local/cuda/samples/1_Utilities/deviceQuery
make
```

Add PATH to the system wide environment

```
vi /etc/environment
```

Add this to the end

```
/usr/local/cuda/samples/bin/x86_64/linux/release
```

Next enable/source it:

```
source /etc/environment
```

```
root@gpu03:~# deviceQuery
deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 2 CUDA Capable device(s)

Device 0: "Tesla V100-PCIE-32GB"
  CUDA Driver Version / Runtime Version          10.2 / 10.2
  CUDA Capability Major/Minor version number:    7.0
  Total amount of global memory:                 32510 MBytes (34089730048
bytes)
  (80) Multiprocessors, ( 64) CUDA Cores/MP:     5120 CUDA Cores
  GPU Max Clock rate:                            1380 MHz (1.38 GHz)
  Memory Clock rate:                             877 Mhz
  Memory Bus Width:                              4096-bit
  L2 Cache Size:                                 6291456 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072,
65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048
layers
  Total amount of constant memory:               65536 bytes
```

```
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 7 copy engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Enabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Supports MultiDevice Co-op Kernel Launch:      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 59 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device
simultaneously) >

Device 1: "Tesla V100-PCIE-32GB"
  CUDA Driver Version / Runtime Version          10.2 / 10.2
  CUDA Capability Major/Minor version number:    7.0
  Total amount of global memory:                 32510 MBytes (34089730048
bytes)
  (80) Multiprocessors, ( 64) CUDA Cores/MP:     5120 CUDA Cores
  GPU Max Clock rate:                            1380 MHz (1.38 GHz)
  Memory Clock rate:                             877 Mhz
  Memory Bus Width:                              4096-bit
  L2 Cache Size:                                 6291456 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072,
65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048
layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 7 copy engine(s)
  Run time limit on kernels:                     No
```

```
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Enabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Supports MultiDevice Co-op Kernel Launch:      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 175 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device
simultaneously) >
> Peer access from Tesla V100-PCIE-32GB (GPU0) -> Tesla V100-PCIE-32GB
(GPU1) : Yes
> Peer access from Tesla V100-PCIE-32GB (GPU1) -> Tesla V100-PCIE-32GB
(GPU0) : Yes

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 10.2, CUDA Runtime
Version = 10.2, NumDevs = 2
Result = PASS
```

Links

CUDA Toolkit

NVIDIA CUDA Installation Guide for Linux

https://www.admin-magazine.com/HPC/Articles/Warewulf-Cluster-Manager-Development-and-Run-Time/Warewulf-3-Code/MPICH2

https://proteusmaster.urcf.drexel.edu/urcfwiki/index.php/Environment_Modules_Quick_Start_Guide

https://en.wikipedia.org/wiki/Environment_Modules_(software)

http://www.walkingrandomly.com/?p=5680

https://modules.readthedocs.io/en/latest/index.html

From:
https://wiki.inf.unibz.it/ - **Engineering-Tech Wiki**

Permanent link:
https://wiki.inf.unibz.it/doku.php?id=tech:slurm

Last update: **2022/11/24 16:17**