Enroot Implementation in IIT Delhi HPC Cluster Environment

(https://github.com/NVIDIA/enroot)

By :

Kapil Subhash Sawate

HPC Team, CSC, IIT Delhi

Contributors :

HPC Team, CSC, IIT Delhi hpchelp@iitd.ac.in

Manoj Soni, Altair manoj.soni@altair.com

Contents :

---

1. Installation from source

(**WHY ?** : As we dont want to do node specific installation of enroot on 600+ nodes, because it will be difficult to maintain versions in future as well as troubleshooting issues associated with it.)

---

a. Base Installation :

---

git clone --recurse-submodules https://github.com/NVIDIA/enroot.git

#terminate it if it get stuck at musl (unable to connect to musl repository, adding source code manually)

cd enroot

#Remove musl sections from

./.git/config:19

./.gitmodules:9

cd deps

wget https://git.musl-libc.org/cgit/musl/snapshot/musl-1.2.1.tar.gz --no-check-certificate

tar -xvf musl-1.2.1.tar.gz

mv musl-1.2.1 musl

cd ..

#Change Installation Path inside makefile

make install

---

b. Testing & dependency tracking :

#Installation configuration for enroot is set in the below mentioned module file.

module load apps/enroot/3.2.0

enroot create --name qe-v6.6a1 <path to enroot containers folder>/hpc+quantum_espresso+v6.6a1.sqsh

[ERROR] Command not found: unsquashfs

.sqsh - is a compressed file system for particular container

Squashfs is a highly compressed read-only filesystem for Linux.
(enroot create requires squashfs-tools : tools to create and extract Squashfs filesystems.)

Installation of squashfs-tools

---

squashfs-tools : https://github.com/plougher/squashfs-tools

git clone --recurse-submodules https://github.com/plougher/squashfs-tools
cd squashfs-tools/squashfs-tools
#vi Makefile
#INSTALL_DIR = <squashfs-tools installation path>
make install

#Installation configuration for squashfs-tools is set in the below mentioned module file.

module load apps/enroot/3.2.0
enroot create --name qe-v6.6a1 <path to enroot containers
folder>/hpc+quantum_espresso+v6.6a1.sqsh
[INFO] Extracting squashfs filesystem...
Filesystem uses lzo compression, this is unsupported by this version
Decompressors available:
gzip

squashfs file system can be created using different compression like : gzip, lzma2, lzo, lz4, zstd
Reinstalling squashfs-tools with support for above mentioned compressions

---

#squashfs-tools dependencies installation :

export PREFIX=<squashfs-tools dependencies common installation path>
#gzip : https://www.gnu.org/software/gzip

wget https://ftp.gnu.org/gnu/gzip/gzip-1.10.tar.gz --no-check-certificate
tar -xvf gzip-1.10.tar.gz

```
cd gzip-1.10
./configure --prefix=$PREFIX
make all -j 24
make install -j 24
cd ..
```

#lzma2 : http://tukaani.org/xz

```
wget https://excellmedia.dl.sourceforge.net/project/lzmautils/xz-5.2.5.tar.gz
--no-check-certificate
tar -xvf xz-5.2.5.tar.gz
cd xz-5.2.5
make all -j 24
make install -j 24
cd ..
```

#lzo : http://www.oberhumer.com/opensource/lzo

```
wget http://www.oberhumer.com/opensource/lzo/download/lzo-2.10.tar.gz
--no-check-certificate
tar -xvf lzo-2.10.tar.gz
cd lzo-2.10
./configure --prefix=$PREFIX --enable-shared
make all -j 24
make install -j 24
cd ..
```

#lz4 : https://github.com/lz4/lz4

```
git clone --recurse-submodules https://github.com/lz4/lz4.git
cd lz4
make
make install
cd ..
```

#zstd : https://github.com/facebook/zstd

git clone --recurse-submodules https://github.com/facebook/zstd.git
cd zstd
make all
make install
cd ..

#squashfs-tools

git clone --recurse-submodules https://github.com/plougher/squashfs-tools
cd squashfs-tools/squashfs-tools
#vi Makefile
#INSTALL_DIR =<squashfs-tools installation path>
make all
make install

#Testing
enroot create --name qe-v6.6a1 <path to sqsh file>/hpc+quantum_espresso+v6.6a1.sqsh
enroot list

enroot start --env PBS_NTASKS --env PBS_O_WORKDIR --rw --mount $HOME:$HOME
--mount $SCRATCH:$SCRATCH quantum_espresso:v6.6a1
== Error > enroot-nsenter: failed to create user namespace: Invalid argument

Solution :

/proc/sys/user/max_user_namespaces must be greater than 1
/proc/sys/user/max_mnt_namespaces must be greater than 1

cat /proc/sys/user/max_user_namespaces
0
cat /proc/sys/user/max_mnt_namespaces

255643

echo 100 > /proc/sys/user/max_user_namespaces

enroot import 'docker://ubuntu'

[ERROR] Command not found: jq

#jq 1.6 https://github.com/stedolan/jq/releases

export PREFIX=<jq installation path>
wget https://github.com/stedolan/jq/releases/download/jq-1.6/jq-1.6.tar.gz
--no-check-certificate
tar -xvf jq-1.6.tar.gz
cd jq-1.6
module load apps/automake/1.14/gnu
autoreconf -i
./configure --prefix=$PREFIX
make all -j 24
make install -j 24
cd ..

enroot import 'docker://ubuntu'

[ERROR] Command not found: parallel

#parallel 20201122 : https://www.gnu.org/software/parallel

export PREFIX=<parallel tool installation path>
wget https://ftp.gnu.org/gnu/parallel/parallel-20201122.tar.bz2 --no-check-certificate
tar -xvf parallel-20201122.tar.bz2
cd parallel-20201122
./configure --prefix=$PREFIX
make all -j 24
make install -j 24
cd ..

enroot start --env PBS_NTASKS --env PBS_O_WORKDIR --rw --mount $HOME:$HOME --mount $SCRATCH:$SCRATCH quantum_espresso:v6.6a1

[ERROR] Command not found: nvidia-container-cli, see https://github.com/NVIDIA/libnvidia-container
[ERROR] <path to enroot installation dir>/etc/enroot/hooks.d/98-nvidia.sh exited with return code 1

#bmake : http://www.crufty.net/help/sjg/bmake.html

export PREFIX=<bmake installation path>
wget http://www.crufty.net/ftp/pub/sjg/bmake.tar.gz
tar -xvf bmake.tar.gz
cd bmake
./configure --prefix=$PREFIX
make all
make install
cd ..

#libcap : https://mirrors.edge.kernel.org/pub/linux/libs/security/linux-privs/libcap2

export PREFIX=<libcap installation path>
wget https://mirrors.edge.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.46.tar.gz
tar -xvf libcap-2.46.tar.gz
cd libcap-2.46
#vi Make.rules, change prefix & exec_prefix to <libcap installation path>
make all
make install
cd ..

#gperf 3.1 : https://www.gnu.org/software/gperf

export GPERF=<gperf installation path>
wget http://ftp.gnu.org/pub/gnu/gperf/gperf-3.1.tar.gz

tar -xvf gperf-3.1.tar.gz
cd gperf-3.1
./configure --prefix=$GPERF
make all
make install
cd ..

#libseccomp 2.5.1 : https://github.com/seccomp/libseccomp

export PREFIX=<libseccomp installation path>
git clone --recurse-submodules https://github.com/seccomp/libseccomp.git
cd libseccomp
./autogen.sh
./configure --prefix=$PREFIX
--> Require gperf
#AFter gperf
./configure --prefix=$PREFIX
make all -j 24
make install -j 24
cd ..

#libnvidia-container : https://github.com/NVIDIA/libnvidia-container

export PREFIX=<libnvidia-container installation path>

git clone --recurse-submodules https://github.com/NVIDIA/libnvidia-container.git
cd libnvidia-container
#vi Makefile ,change prefix
make all
#/bin/sh: bmake: command not found
#After bmake installation
bmake[2]: no system rules (sys.mk).
export MAKESYSPATH=<bmake installation path>/share/mk
make all

install: cannot stat ‘libelf.so.1’: No such file or directory

cd <libnvidia-container source path >/deps/src/elftoolchain-0.7.1/libelf

mv name\ libelf.so.1 libelf.so.1

cd -

make all

#fatal error: sys/capability.h: No such file or directory

#include <sys/capability.h>

--> Require libcap

#After libcap installation

make all

21: fatal error: seccomp.h: No such file or directory

#include <seccomp.h>

--> Require libseccomp

make all

ld cannot find : -lcap

#vi Makefile : line no. 152

#Replace

#BIN_LDLIBS = -l:$(LIB_SHARED) -lcap $(LDLIBS)

#with

#BIN_LDLIBS = -l:$(LIB_SHARED) -L<libcap installation path>/lib64 -lcap $(LDLIBS)

make all

make install

cd ..

#Done

#Change Installation Path from makefile : <enroot installation path>

make install

#Login Using Root (Important Step for enroot import, will set capabilities to enroot-aufs2ovlfs & enroot-mksquashovlfs)

make setcap

#Output

#setcap cap_sys_admin+pe <enroot installation path>/bin/enroot-mksquashovlfs
#setcap cap_sys_admin,cap_mknod+pe <enroot installation path>/bin/enroot-aufs2ovlfs

#Single Node Testing
enroot create --name qe-v6.6a1 <path to sqsh file>/hpc+quantum_espresso+v6.6a1.sqsh

enroot start --env PBS_NTASKS --env PBS_O_WORKDIR --rw --mount $HOME:$HOME
--mount $SCRATCH:$SCRATCH qe-v6.6a1

#Done
---------- Below are optional dependencies ----------
#libfuse 3.10.1 : https://github.com/libfuse/libfuse/releases

export PREFIX=<libfuse installation path>
wget https://github.com/libfuse/libfuse/archive/fuse-3.10.1.tar.gz
tar -xvf fuse-3.10.1.tar.gz
cd libfuse-fuse-3.10.1
mkdir build
cd build
export PATH=$PATH:<path to meson bin directory>:<path to python3 bin directory>
meson ..
meson configure -D cpp_std=c++11 -D prefix=$PREFIX -D examples=false -D useroot=false -D
utils=false
ninja
python3 -m pytest test
ninja install
cd ..

#fuse-overlayfs : https://github.com/containers/fuse-overlayfs

export PREFIX=<fuse-overlayfs installation path>
git clone --recurse-submodules https://github.com/containers/fuse-overlayfs.git
cd fuse-overlayfs
./autogen.sh
./configure --prefix=$PREFIX

configure: error: *** libfuse not found

-->Require libfuse

#After libfuse installation

./configure --prefix=$PREFIX

make all -j 24

make install -j 24

cd ..

#squashfuse : https://github.com/vasi/squashfuse

export PREFIX=<squashfuse installation path>

git clone --recurse-submodules https://github.com/vasi/squashfuse.git

cd squashfuse

./autogen.sh

./configure --prefix=$PREFIX

configure: error: in `<path to squashfuse source directory>':

configure: error: Can't find FUSE

--> Require fuse

#After fuse installation

export DEPS=<squashfs-tools dependencies installation path>

export FUSE=<libfuse installation path>

./configure --prefix=$PREFIX --with-zlib=/usr --with-xz=$DEPS --with-lzo=$DEPS

--with-lz4=$DEPS --with-zstd=$DEPS

make all

make install

cd ..

#pigz : https://zlib.net/pigz/

export PREFIX=<pigz installation path>

wget https://zlib.net/pigz/pigz-2.4.tar.gz

tar -xvf pigz-2.4.tar.gz

cd pigz-2.4

---

2. Enroot module file

#%Module 1.0

module-whatis "Enroot 3.2.0 (with squashfs-tool 4.4 + jq 1.6 + parallel 20201122 + bmake 20201212 + libcap 2.46 + gperf 3.1 + libseccomp 2.5.1 + libnvidia-container 1.3.1 + libfuse 3.10.1 + fuse-overlayfs + squashfuse + pigz 2.4 + terraform 0.14.3)"

#Enroot

set	*ENROOT*	*<enroot installation path>*
prepend-path	*PATH*	*$ENROOT/bin*

#Import Custom Environment Variable for Multi Node Job

setenv	***ENROOT_VERSION***	***3.2.0***

#squashfs-tools & its dependencies

set	*SQUASHFS*	*<squashfs-tools installation path>*
prepend-path	*PATH*	*$SQUASHFS/bin*
set	*SQUASHFSDEPS*	*<squashfs-tools deps common installation path>*
prepend-path	*PATH*	*$SQUASHFSDEPS/bin*
prepend-path	*LD_RUN_PATH*	*$SQUASHFSDEPS/lib*
prepend-path	*LD_LIBRARY_PATH*	*$SQUASHFSDEPS/lib*
prepend-path	*CPATH*	*$SQUASHFSDEPS/include*
prepend-path	*INCLUDE*	*$SQUASHFSDEPS/include*
prepend-path	*PKG_CONFIG_PATH*	*$SQUASHFSDEPS/lib/pkgconfig*
prepend-path	*MANPATH*	*$SQUASHFSDEPS/share/man*

#jq 1.6

set	*JQ*	*<jq installation path>*
prepend-path	*PATH*	*$JQ/bin*

prepend-path	*LD_RUN_PATH*	*$JQ/lib*
prepend-path	*LD_LIBRARY_PATH*	*$JQ/lib*
prepend-path	*CPATH*	*$JQ/include*
prepend-path	*INCLUDE*	*$JQ/include*
prepend-path	*PKG_CONFIG_PATH*	*$JQ/lib/pkgconfig*
prepend-path	*MANPATH*	*$JQ/share/man*

#parallel 20201122

set	*PARALLEL*	*<parallel tool installation path>*
prepend-path	*PATH*	*$PARALLEL/bin*
prepend-path	*LD_RUN_PATH*	*$PARALLEL/lib*
prepend-path	*LD_LIBRARY_PATH*	*$PARALLEL/lib*
prepend-path	*CPATH*	*$PARALLEL/include*
prepend-path	*INCLUDE*	*$PARALLEL/include*
prepend-path	*PKG_CONFIG_PATH*	*$PARALLEL/lib/pkgconfig*
prepend-path	*MANPATH*	*$PARALLEL/share/man*

#bmake 20201212

set	*BMAKE*	*<bmake installation path>*
prepend-path	*PATH*	*$BMAKE/bin*
prepend-path	*MANPATH*	*$BMAKE/share/man*
setenv	*MAKESYSPATH*	*$BMAKE/share/mk*

#libcap 2.46

set	*LIBCAP*	*<libcap installation path>*
prepend-path	*PATH*	*$LIBCAP/sbin*
prepend-path	*LD_RUN_PATH*	*$LIBCAP/lib64*
prepend-path	*LD_LIBRARY_PATH*	*$LIBCAP/lib64*
prepend-path	*CPATH*	*$LIBCAP/include*
prepend-path	*INCLUDE*	*$LIBCAP/include*
prepend-path	*PKG_CONFIG_PATH*	*$LIBCAP/lib/pkgconfig*

prepend-path *MANPATH* *$LIBCAP/share/man*

#gperf 3.1

set *GPERF* *<gperf installation path>*
prepend-path *PATH* *$GPERF/bin*
prepend-path *MANPATH* *$GPERF/share/man*

#libseccomp 2.5.1

set *LIBSECCOMP* *<libseccomp installation path>*
prepend-path *PATH* *$LIBSECCOMP/bin*
prepend-path *LD_RUN_PATH* *$LIBSECCOMP/lib*
prepend-path *LD_LIBRARY_PATH* *$LIBSECCOMP/lib*
prepend-path *CPATH* *$LIBSECCOMP/include*
prepend-path *INCLUDE* *$LIBSECCOMP/include*
prepend-path *PKG_CONFIG_PATH* *$LIBSECCOMP/lib/pkgconfig*
prepend-path *MANPATH* *$LIBSECCOMP/share/man*

#libnvidia-container 1.3.1

set *LIBNVIDIA* *<libnvidia-container installation path>*
prepend-path *PATH* *$LIBNVIDIA/bin*
prepend-path *LD_RUN_PATH* *$LIBNVIDIA/lib*
prepend-path *LD_LIBRARY_PATH* *$LIBNVIDIA/lib*
prepend-path *CPATH* *$LIBNVIDIA/include*
prepend-path *INCLUDE* *$LIBNVIDIA/include*
prepend-path *PKG_CONFIG_PATH* *$LIBNVIDIA/lib/pkgconfig*
prepend-path *MANPATH* *$LIBNVIDIA/share/man*

---------- Basic things done ----------

#libfuse 3.10.1

set *LIBFUSE* *<libfuse installation path>*
prepend-path *PATH* *$LIBFUSE/bin*
prepend-path *PATH* *$LIBFUSE/sbin*

prepend-path *LD_RUN_PATH* *$LIBFUSE/lib*
prepend-path *LD_LIBRARY_PATH* *$LIBFUSE/lib*
prepend-path *CPATH* *$LIBFUSE/include*
prepend-path *INCLUDE* *$LIBFUSE/include*
prepend-path *PKG_CONFIG_PATH* *$LIBFUSE/lib/pkgconfig*
prepend-path *MANPATH* *$LIBFUSE/share/man*

#fuse-overlayfs

set *FUSEOVERLAYFS* *<fuse-overlayfs installation path>*
prepend-path *PATH* *$FUSEOVERLAYFS/bin*
prepend-path *MANPATH* *$FUSEOVERLAYFS/share/man*

#squashfuse

set *SQUASHFUSE* *<squashfuse installation path>*
prepend-path *PATH* *$SQUASHFUSE/bin*
prepend-path *PATH* *$SQUASHFUSE/sbin*
prepend-path *LD_RUN_PATH* *$SQUASHFUSE/lib*
prepend-path *LD_LIBRARY_PATH* *$SQUASHFUSE/lib*
prepend-path *CPATH* *$SQUASHFUSE/include*
prepend-path *INCLUDE* *$SQUASHFUSE/include*
prepend-path *PKG_CONFIG_PATH* *$SQUASHFUSE/lib/pkgconfig*
prepend-path *MANPATH* *$SQUASHFUSE/share/man*

#pigz : 2.4

set *PIGZ* *<pigz installation path>*
prepend-path *PATH* *$PIGZ/bin*

#central container sqsh files path
setenv ***CONTAINER*** ***<path to enroot containers directory>***
---

3. Enroot configuration file

File Path : < enroot installation path>/etc/enroot/enroot.conf

File : enroot.conf

ENROOT_LIBRARY_PATH ***<enroot installation path>/lib/enroot***
ENROOT_SYSCONF_PATH ***<enroot installation path>/etc/enroot***
ENROOT_RUNTIME_PATH ***<shared path for enroot specific dirs>/enroot/user-${PBS_JOBID}***
ENROOT_CONFIG_PATH ***<enroot installation path>/etc/enroot***
ENROOT_CACHE_PATH ***<shared path for enroot specific dirs>/enroot-cache/group-${PBS_JOBID}***
ENROOT_DATA_PATH ***<shared path for enroot specific dirs>/enroot-data/user-${PBS_JOBID}***
ENROOT_TEMP_PATH ***<shared path for enroot specific dirs>/tmp/pbs.${PBS_JOBID}***

 # Gzip program used to uncompress digest layers.
#ENROOT_GZIP_PROGRAM gzip

 # Options passed to zstd to compress digest layers.
#ENROOT_ZSTD_OPTIONS -1

 # Options passed to mksquashfs to produce container images.
ENROOT_SQUASH_OPTIONS -comp lzo -noD

 # Make the container root filesystem writable by default.
ENROOT_ROOTFS_WRITABLE yes

 # Remap the current user to root inside containers by default.
#ENROOT_REMAP_ROOT no

 # Maximum number of processors to use for parallel tasks (0 means unlimited).
#ENROOT_MAX_PROCESSORS $(nproc)

 # Maximum number of concurrent connections (0 means unlimited).

#ENROOT_MAX_CONNECTIONS 10

 # Maximum time in seconds to wait for connections establishment (0 means unlimited).
#ENROOT_CONNECT_TIMEOUT 30

 # Maximum time in seconds to wait for network operations to complete (0 means unlimited).
#ENROOT_TRANSFER_TIMEOUT 0

 # Number of times network operations should be retried.
#ENROOT_TRANSFER_RETRIES 0

 # Use a login shell to run the container initialization.
#ENROOT_LOGIN_SHELL yes

 # Allow root to retain his superuser privileges inside containers.
#ENROOT_ALLOW_SUPERUSER no

 # Use HTTP for outgoing requests instead of HTTPS (UNSECURE!).
#ENROOT_ALLOW_HTTP no

 # Include user-specific configuration inside bundles by default.
#ENROOT_BUNDLE_ALL no

 # Generate an embedded checksum inside bundles by default.
#ENROOT_BUNDLE_CHECKSUM no

 # Mount the current user's home directory by default.
ENROOT_MOUNT_HOME yes

 # Mount the current user's scratch directory by default.
ENROOT_MOUNT_SCRATCH yes

 # Restrict /dev inside the container to a minimal set of devices.
#ENROOT_RESTRICT_DEV no

 # Always use --force on command invocations.

#ENROOT_FORCE_OVERRIDE no

 # SSL certificates settings:

#SSL_CERT_DIR

#SSL_CERT_FILE

 # Proxy settings:

#all_proxy

#no_proxy

#http_proxy

#https_proxy

---

4. Customization as per our requirements

---

a) Requirements

---

i) Setting some important environment variables by default inside the container

---

Solution :

Creating own environment file at : <enroot installation path>/etc/enroot/environ.d

File name : 10-terminal.env

(Format : <environment variable inside the container>=<value you want to assign>)

Here we are using whatever the value environment variable outside the container reflects the same inside the container with the same environment variable name.)

Reference : https://github.com/NVIDIA/enroot/blob/master/doc/image-format.md

File : 10-terminal.env

TERM=${TERM}

HOME=${HOME}

DISPLAY=${DISPLAY}

OMP_NUM_THREADS=${OMP_NUM_THREADS}

SCRATCH=${HOME/home/scratch}

PBS_ENVIRONMENT=${PBS_ENVIRONMENT}

PBS_JOBCOOKIE=${PBS_JOBCOOKIE}

PBS_JOBID=${PBS_JOBID}

PBS_JOBDIR=${PBS_JOBDIR}

PBS_JOBNAME=${PBS_JOBNAME}

PBS_MOMPORT=${PBS_MOMPORT}

PBS_NCHUNKS=${PBS_NCHUNKS}

PBS_NCPUS=${PBS_NCPUS}

PBS_NODEFILE=${PBS_NODEFILE}

PBS_NODENUM=${PBS_NODENUM}

PBS_NTASKS=${PBS_NTASKS}

PBS_O_HOME=${PBS_O_HOME}

PBS_O_HOST=${PBS_O_HOST}

PBS_O_LANG=${PBS_O_LANG}

PBS_O_LOGNAME=${PBS_O_LOGNAME}

PBS_O_MAIL=${PBS_O_MAIL}

PBS_O_PATH=${PBS_O_PATH}

PBS_O_QUEUE=${PBS_O_QUEUE}

PBS_O_SHELL=${PBS_O_SHELL}

PBS_O_SYSTEM=${PBS_O_SYSTEM}

PBS_O_WORKDIR=${PBS_O_WORKDIR}

PBS_QUEUE=${PBS_QUEUE}

PBS_TASKNUM=${PBS_TASKNUM}

http_proxy=${http_proxy}

https_proxy=${https_proxy}

SSL_CERT_FILE=${SSL_CERT_FILE}

---

ii) Mount HOME & SCRATCH locations by default inside the container & make the container

---

Solution :

Reference : https://github.com/NVIDIA/enroot/blob/master/doc/standard-hooks.md

For HOME location :

Setting ENROOT_MOUNT_HOME to yes in enroot.conf file which will use default hook available : <enroot installation path>/etc/enroot/hooks.d/**10-home.sh**

For SCRATCH Location :

Creating own env variable ENROOT_MOUNT_SCRATCH in enroot.conf file & defining our own hook for scratch : <enroot installation path>/etc/enroot/hooks.d/**10-scratch.sh**

FIle : 10-scratch.sh

```
#! /usr/bin/env bash
set -eu
if [ -z "${ENROOT_MOUNT_SCRATCH-}" ]; then
        exit o
fi
source "${ENROOT_LIBRARY_PATH}/common.sh"
common::checkcmd getent

if [ -z "${SCRATCH-}" ]; then
        export "SCRATCH=${HOME/home/scratch}"
fi
if [ -n "${ENROOT_REMAP_ROOT-}" ]; then
        printf "%s /root none x-create=dir,bind,rw,nosuid\n" "${SCRATCH}" >>
"${ENROOT_MOUNTS}"
        printf "SCRATCH=/root\n" >> "${ENROOT_ENVIRON}"
else
        printf "%s %s none x-create=dir,bind,rw,nosuid\n" "${SCRATCH}" "${SCRATCH}" >>
"${ENROOT_MOUNTS}"
        printf "SCRATCH=%s\n" "${SCRATCH}" >> "${ENROOT_ENVIRON}"
fi
```

---

b) Challenges

---

i) Problem in enroot working with linux cgroups

Solution :

Complete description available at : https://github.com/NVIDIA/enroot/issues/54

In Brief :

We were getting the following error with enroot while using linux cgroups.

enroot start lammps

nvidia-container-cli: mount error: open failed: ***<shared path for enroot specific dirs>***/enroot-data/user-${PBS_JOBID}/lammps/sys/fs/cgroup/devices/user.slice/devices.allow: permission denied

<shared path for enroot specific dirs>/enroot-data/user-$PBS_JOBID/lammps was our enroot data path where the user was having correct permissions.

Important File : <enroot installation path>/etc/enroot/hooks.d/98-nvidia.sh

Line No 30 : *cli_args=("--no-cgroups" "--ldconfig=@$(command -v ldconfig.real || command -v ldconfig)")*

Scenarios :

1. Without linux cgroups & with --no-cgroups in nvidia hook, everything was working fine, but the problem was, enroot was interfering with the other resources which are not assigned to it that's why we want to use cgroups.

2. With linux cgroups & removing --no-cgroups from nvidia hook, enroot start will give the following error,

enroot start lammps

nvidia-container-cli: mount error: open failed: ***<shared path for enroot specific dirs>***/enroot-data/user-613.chas052/lammps/sys/fs/cgroup/devices/pbspro.slice/pbspro-613.chas052.slice/devices.allow: permission denied

[ERROR] <enroot installation path>/etc/enroot/hooks.d/98-nvidia.sh exited with return code 1

Important Note : Enroot is unprivileged so you won't be able to set up cgroups from it. Enroot will inherit the parent cgroups.

Solution :

Use linux cgroups & use **--no-cgroups --no-devbind** line no. 30 of file <enroot installation path>/etc/enroot/hooks.d/98-nvidia.sh

*cli_args=("--**no-cgroups**" "--**no-devbind**" "--ldconfig=@$(command -v ldconfig.real || command -v ldconfig)")*

Conclusion :

- **--no-devbind** is required because of bug in libnvidia-container (this was the actual cause of the above mentioned error, no issue with cgroups)
  see : https://github.com/NVIDIA/libnvidia-container/issues/125

Note : In case of IIT Delhi Cluster Environment PBS Pro hook is instructing linux cgroups which is parent cgroups for enroot.

---

ii) How to run enroot instances on multiple nodes, in order to run multi-node jobs using containers

---

Complete Description is available at : https://github.com/NVIDIA/enroot/issues/49
(Last two comments) https://github.com/NVIDIA/enroot/issues/54
Scenario :
(As most of the containers available at NGC site use openmpi, this solution is for applications which are using openmpi to run applications parallelly inside the container on multiple nodes.)

We have two nodes, say node1 & node2 (40 processors each), where password less ssh is enabled which is required for mpirun to run jobs on multiple nodes.

Let's say, I am on node1 from where I am starting the job which is expected to run on node1, node2.

machines is a file containing name node1 & node2 (host file for mpirun)

enroot create --name qe6.6a1 $CONTAINER/hpc+quantum_espresso+v6.6a1.sqsh

enroot start --rw qe6.6a1

mpirun -np 80 -hostfile machines pw.x -inp test1.in

Will end up with "**bash: orted: command not found**" error, because the container environment is not visible on other node i.e node2,

We are having working multinode non container applications (as all nodes are sharing the common storage path).

That's why we want to know how the file system inside the container will be visible to other nodes.

Single node job using container is working absolutely fine, i.e mpirun -np 40 pw.x -inp test1.in

Solution :

Customized hook to launch orted agent of openmpi on multiple nodes.

File Path : <enroot installation path>/etc/enroot/hooks.d/ompi.sh

(make sure its an executable)

File : ompi.sh

#! /bin/bash

echo "OMPI_MCA_orte_launch_agent=enroot start ${ENROOT_ROOTFS##*/} orted" >> "${ENROOT_ENVIRON}"

---

iv) Setting required system environment variables i.e PATH, LD_LIBRARY_PATH specific to enroot for the ompi hook & setting version specific enroot environment for multi node job

---

As ompi hook is executing ***enroot start ${ENROOT_ROOTFS##*/} orted*** on multiple nodes in order to start **orted** agent for a multi node job. In case of IIT Delhi cluster environment **module load apps/enroot/3.2.0** will set the system environment variables i.e PATH, LD_LIBRARY_PATH, INCLUDE etc. on the first allocated node only so to launch orted agent on multiple nodes, enroot version specific entries need to be available in these environment variables on the other allocated nodes otherwise it will give an error like ***enroot command not found.***

Enroot version specific information will be maintained inside the file with the name **version** at the parent folder of ENROOT_ROOTFS i.e ENROOT_DATA_PATH which is specific to a particular job.

This achieved using customized hook :<enroot installation path>/etc/enroot/hooks.d/**version.sh** (make sure its an executable)

File version.sh : echo "${ENROOT_VERSION}" > ${ENROOT_ROOTFS}/../version

(Note : ENROOT_VERSION environment variable is set by module apps/enroot/3.2.0)

To set enroot version specific entries in the system environment variable i.e PATH & LD_LIBRARY_PATH we are using /etc/bashrc i.e global bashrc which will execute a script available at shared path. (source <path to the script>/enrootVersionSetupForMultiNode)

Script : enrootVersionSetupForMultiNode

#<shared path for enroot specific dirs>/enroot-data/user-$PBS_JOBID is a value of ENROOT_DATA_PATH

if [-n "$PBS_JOBID"] && [-f <shared path for enroot specific dirs>/enroot-data/user-$PBS_JOBID/version]; then

version=`cat <shared path for enroot specific dirs>/enroot-data/user-$PBS_JOBID/version`

if [$version = "3.2.0"] ; then

export PATH=<libnvidia-container installation path>/bin:<enroot installation path>/bin:$PATH

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<libnvidia-container installation path>/lib

fi

fi

(Note : In case of IIT Delhi cluster environment, global bashrc i.e /etc/bashrc is sourced in users local .bashrc i.e $HOME/.bashrc file**)**

Important :

Reference : https://github.com/NVIDIA/enroot/blob/master/doc/image-format.md

In order to run all the above mentioned customization/hooks/scripts successfully in addition to the default entries following entries need to be added in enroot's mounts.d fstab files.

Path : <enroot installation path>/etc/enroot/mounts.d

File Name : 10-system.fstab

<enroot installation path> <enroot installation path> none x-create=dir,rbind,rw,nosuid,nodev,noexec,rslave 0 -1

<shared path for enroot specific dirs> <shared path for enroot specific dirs> none x-create=dir,rbind,rw,nosuid,nodev,noexec,rslave 0 -1

<PBSHOME path> <PBSHOME path> x-create=dir,rbind,rw,nosuid,nodev,noexec,rslave 0 -1

/etc/ssh /etc/ssh none x-create=dir,rbind,ro,nosuid,nodev,noexec,rslave 0 -1

File Name : 20-config.fstab

/etc/pbs.conf /etc/pbs.conf none x-create=file,bind,ro,nosuid,nodev,noexec,private 0 -1

---

5. Job Submission with Enroot in IIT Delhi Cluster Environment

(Example : Quantum Espresso)

---

1. Interactive Job Submission Example :

---

qsub -I -P cc -lselect=2:ncpus=10:ngpus=1 **-v enroot=1** -lplace=scatter -lwalltime=01:00:00

#On the allocated node

export OMP_NUM_THREADS=1

module load apps/enroot/3.2.0

enroot create --name qe $CONTAINER/hpc+quantum_espresso+v6.6a1.sqsh

enroot start qe

cd $PBS_O_WORKDIR

mpirun -np $PBS_NTASKS pw.x -inp test.in

---

2. Batch Job Submission Script Example :

#!/usr/bin/env bash

#PBS -N EnrootQE

#PBS -P cc

#PBS -m bea

#PBS -M $USER@iitd.ac.in

#PBS -l select=2:ncpus=10:mpiprocs=10:ngpus=2

#PBS -v enroot=1

#PBS -l place=scatter

#PBS -l walltime=01:00:00

export OMP_NUM_THREADS=1

module load apps/enroot/3.2.0

enroot create --name qe $CONTAINER/hpc+quantum_espresso+v6.6a1.sqsh

enroot start qe 'cd $PBS_O_WORKDIR && mpirun -np $PBS_NTASKS pw.x -inp test.in'

To know about Enroot Integration with PBS Pro Contact: ***Manoj Soni, Altair*** *manoj.soni@altair.com*
