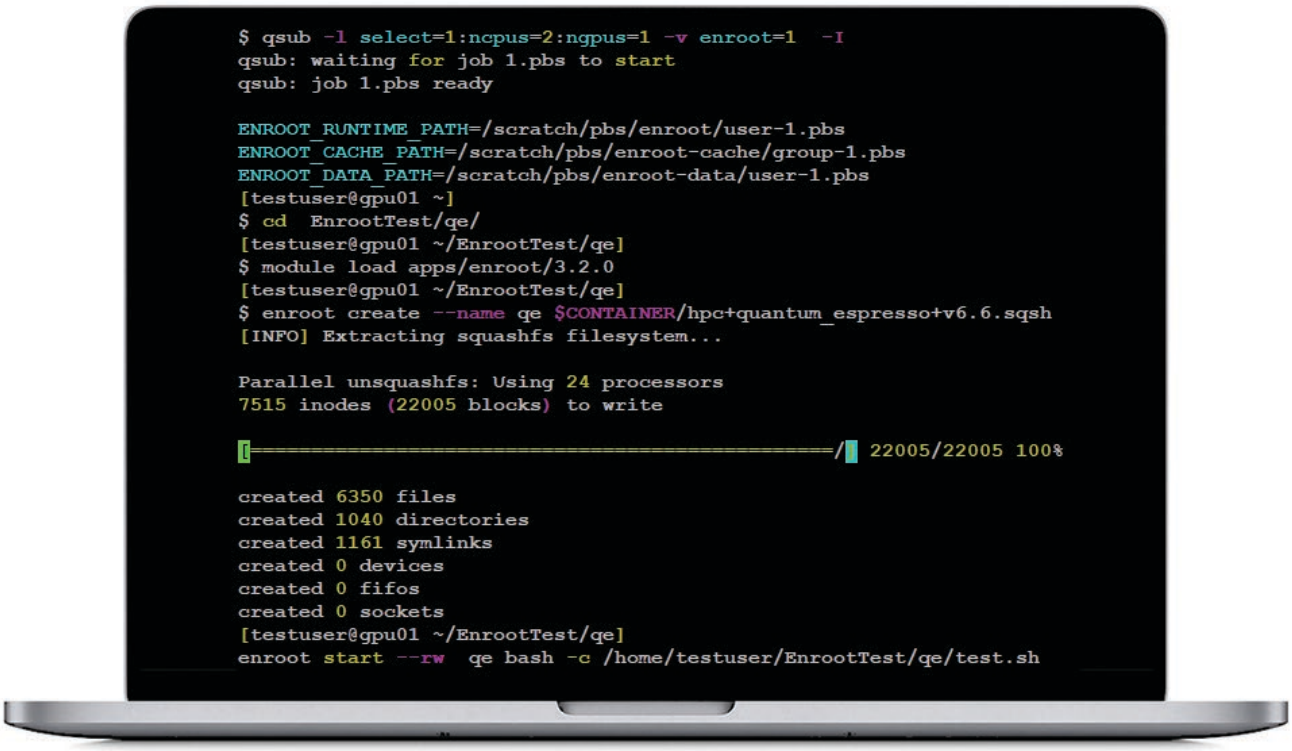
INTEGRATING ENROOT WITH ALTAIR® PBS PROFESSIONAL®

1 About PBS Professional

Altair® PBS Professional® is a fast, powerful workload manager designed to improve productivity, optimize utilization and efficiency, and simplify administration for clusters, clouds, and supercomputers — from the biggest HPC workloads to millions of small, high-throughput jobs. PBS Professional automates job scheduling, management, monitoring, and reporting, and it's the trusted solution for complex Top500 systems as well as smaller clusters. PBS Professional can be integrated with popular container technologies like Docker, Singularity, and Enroot. In this document, we will highlight PBS Professional integration with Enroot.

2 About Enroot

Enroot is a simple, yet powerful tool to turn traditional container/OS images into unprivileged sandboxes. Enroot can be thought of as an enhanced unprivileged chroot(1). It uses the same underlying technologies as containers but removes much of the isolation they inherently provide while preserving filesystem separation. This approach is generally preferred in high-performance environments and virtualized environments where portability and reproducibility are important but extra isolation is not warranted.



Enroot is similar to tools like proot(1) and fakeroot(1) but instead relies on more recent features from the Linux kernel (e.g., user and mount namespaces), and it provides facilities to import well-known container image formats (e.g., Docker).

Enroot container images are standard squashfs images (**.sqsh**), instead of **.tar**. Squashfs uses zlib, lz4, lzo, or xz compression. However, regular .tar images of Docker can also be pulled in using Enroot. It has bulit-in GPU support with the libnvidia-container. Enroot requires Linux kernel >=3.10.

3 Enroot Command Line Overview

Make sure you are able to create and start the Enroot container using the commands below before proceeding with PBS Professional integration.

To import/pull the container image:


```
enroot import docker://ubuntu
```

To create the container:

```
enroot create --name
<USER_SPECIFIED_CONTAINER_NAME>
<PATH_TO_SQSH_TF1_FILE>
```

To start the container:

```
enroot start
<USER_SPECIFIED_CONTAINER_NAME>
```

Enroot configuration file:

```
/etc/enroot/enroot.conf
```

4 Enroot PBS Professional Integration

Enroot requires the creation of three directories:

1. Enroot Runtime
2. Enroot Cache
3. Enroot Data

Through PBS Professional, we automate the creation of these directories for each new job and ensure the deletion of these directories on job completion. This requires a PBS MoM-level hook that creates Enroot-specific directories for each Enroot job submitted through PBS Professional on the compute nodes before the job starts. The hook also ensures the deletion of these directories upon job completion.

4.1.1 Enable the PBS Professional Enroot Hook

After installation of Enroot, it creates the **/etc/enroot/enroot.conf** configuration file. We need to update this file on each compute node to control the root directory under which these Enroot-specific subdirectories are to be created. Update the /etc/enroot/enroot.conf file on each compute node with these lines:

```
ENROOT_RUNTIME_PATH
/scratch/pbs/enroot/user-$PBS_JOBID

ENROOT_CACHE_PATH
/scratch/pbs/enroot-cache/group-$PBS_JOBID

ENROOT_DATA_PATH
/scratch/pbs/enroot-data/user-$PBS_JOBID
```

Import pbs_enroot hook syntax:

```
..........
qmgr -c "create hook enroot"
qmgr -c "set hook enroot enabled = true"
qmgr -c "set hook enroot event = execjob_prologue"
qmgr -c "set hook enroot event += execjob_end"
qmgr -c "set hook enroot alarm = 30"
qmgr -c "set hook enroot order = 1"
qmgr -c "import   hook    enroot    application/x-python    default    pbs_enroot.py"
..........
```

Visit the Altair Community for pbs_enroot.py code.

4.1.2 Enable the PBS Professional Cgroup Hook

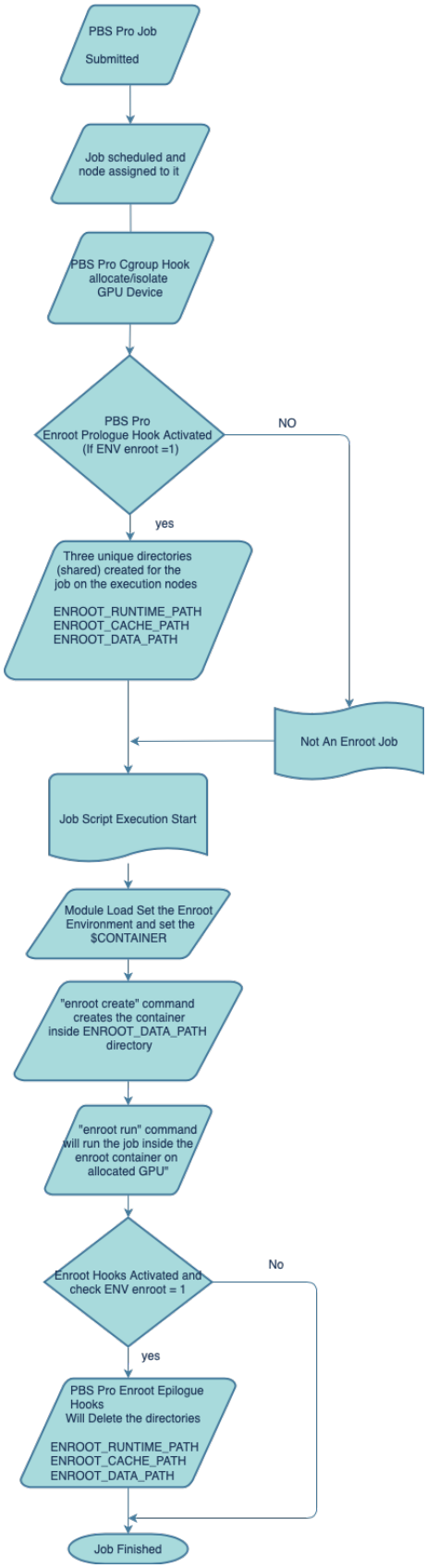
The PBS Professional cgroup hook is used to allocate/isolate GPU cards/devices which are used by Enroot. Enroot inherits the allocation provided by PBS Professional's cgroup hook. Enable the cgroup hook to achieve GPU device isolation. Syntax:

```
..........
create hook pbs_cgroups
set hook pbs_cgroups enabled = true
set hook pbs_cgroups event = execjob_begin
set hook pbs_cgroups event += execjob_epilogue
set hook pbs_cgroups event += execjob_end
set hook pbs_cgroups event += execjob_launch
set hook pbs_cgroups event += execjob_attach
set hook pbs_cgroups event += execjob_resize
set hook pbs_cgroups event += execjob_abort
set hook pbs_cgroups event += execjob_postsuspend
set hook pbs_cgroups event += execjob_preresume
set hook pbs_cgroups event += exechost_periodic
set hook pbs_cgroups event += exechost_startup
set hook pbs_cgroups alarm = 90
set hook pbs_cgroups freq = 120
set hook pbs_cgroups order = 100
set hook pbs_cgroups fail_action = offline_vnodes
import hook pbs_cgroups application/x-python  default
pbs_cgroups.py
import hook pbs_cgroups application/x-config default
pbs_cgroup.json
..........
```

Add the following code snippet to PBS Professional's cgroup JSON configuration file:

```
..........
"devices" : {
     "enabled"      : true,
     "exclude_hosts" : [],
     "exclude_vntypes": [],
     "allow"
     "b *:* rwm",
     "c 136:* rwm",
     "c 231:* rwm",

["infiniband/rdma_cm","rwm"],
["cpu/0/cpuid","rwm","*"],
["full","rwm"],
["fuse","rwm"],
["zero","rwm"],
["null","rwm"],
["ptmx","rwm"],
["tty","rwm"],
["random","rwm"],
["urandom","rwm"],
["obd","rwm"],
["net/tun","rwm"],
["mic/scif","rwm"],
["nvidiactl","rwm"],
["nvidia-uvm","rwm"]
          ],
```



PBS Professional Enroot Workflow


```
"cuda_visible_from_zero" : true
}
..........
```

For more details, refer to section 16.5.5.1.ii, Isolating NVIDIA GPUs, in the PBS Professional 2021.1 Administrator's Guide.

Enroot also has the option to enable cgroups but it doesn't work as it is a non-privileged environment. We used device isolation using cgroups at the PBS Professional level. PBS Professional provides the device isolation and Enroot inherits it. Use cgroups at the PBS level and use --no-cgroups and --no-devbind in the file enroot/hooks.d/98-nvidia.sh:

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

4.1.3 Enable MPI Integration for Enroot on PBS Professional

For multi-node jobs, it is essential to pass `$PBS_JOBID` to all the participating compute nodes, as we are using the `$PBS_JOBID` variable in the enroot.conf file to create a unique directory for each Enroot job. MPI integration with PBS Professional is the key.

Ssh-based MPI integration can be enabled at the site just by enabling the mounting of **/etc/ssh** in the Enroot environment.

Steps:

1. Update the **/etc/ssh/sshd_config** and **/etc/ssh/ssh_config** file to pass the `PBS_JOBID` along with the each ssh session done through within the PBS job.

```
..........
grep PBS_JOBID  /etc/ssh/sshd_config
AcceptEnv PBS_JOBID
grep PBS_JOBID  /etc/ssh/ssh_config
SendEnv PBS_JOBID
..........
```

2. Create a file /etc/ssh/sshrc as below in each compute node:

```
...........
#!/bin/sh

if read proto cookie && [ -n "$DISPLAY" ]; then
       if [ 'echo $DISPLAY | cut -c1-10' = 'localhost:' ]; then
             # X11UseLocalhost=yes
             echo add unix:'echo $DISPLAY |
                  cut -c11-' $proto $cookie
       else
             # X11UseLocalhost=no
             echo add $DISPLAY $proto $cookie
       fi | xauth -q -
fi

conf=${PBS_CONF_FILE:-/etc/pbs.conf}
if [ -f "$conf" ] ; then
       source $conf
else
       exit 1
fi

string=$*
attach_cmd="pbs_attach"

if [ -n "$PBS_JOBID" ]; then
```

```
# Check to see whether the command is already calling pbs_attach
       if [ "${string/$attach_cmd}" = "$string" ] ; then
              echo "Attaching $PPID to $PBS_JOBID"
              $PBS_EXEC/bin/$attach_cmd -j $PBS_JOBID -p $$ 2> /dev/null
              exit 0
       fi
fi
...........
```

4.2 Configuration at the Enroot Level

1. To launch the Open MPI orted agent on multiple nodes:

File path:

```
...........
/soft/enroot/3.2.0/etc/enroot/hooks.d/ompi.sh
#! /bin/bash
echo "OMPI_MCA_orte_launch_agent=enroot start ${ENROOT_ROOTFS##*/} orted" >>
"${ENROOT_ENVIRON}"
...........
```

`ompi.sh` will execute **`enroot start ${ENROOT_ROOTFS##*/} orted`** on multiple nodes in order to start the orted agent for a multi-node job.

2. Module load apps/enroot/3.2.0 will set the system environment variables (e.g., `PATH`, `LD_LIBRARY_PATH`, `INCLUDE`, etc.) on the first allocated node only, so to launch the orted agent on multiple nodes, Enroot version-specific entries need to be available in these environment variables on the other allocated nodes. Enroot version-specific information can be maintained inside the file with the name version at the parent folder of `ENROOT_ROOTFS` (e.g., `ENROOT_DATA_PATH`) which is specific to a particular job.

File path:

```
...........
/soft/enroot/3.2.0/etc/enroot/hooks.d/version.sh
echo "${ENROOT_VERSION}" > ${ENROOT_ROOTFS}/../version
...........
```

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

3. To set Enroot version-specific entries in the system environment variable (e.g., `PATH & LD_LIBRARY_PATH`, `/etc/bashrc`) can be used (source **`/soft/enroot/enrootVersionSetupForMultiNode`**).

Script: `enrootVersionSetupForMultiNode`

`#/scratch/pbs/enroot-data/user-$PBS_JOBID` is a value of `ENROOT_DATA_PATH`

```
...........
if [ -n "$PBS_JOBID" ] && [ -f /scratch/pbs/enroot-data/user-$PBS_JOBID/version ]; then
     version='cat /scratch/pbs/enroot-data/user-$PBS_JOBID/version'
     if [ $version = "3.2.0" ] ; then
          export PATH=/soft/enroot-deps/libnvidia-container/1.3.1/bin:/soft/enroot/3.2.0/bin:$PATH
          export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/soft/enroot-deps/libnvidia-container/1.3.1/lib
     fi
fi
...........
```

In order to run all the above-mentioned customization/hooks/scripts successfully, in addition to the default entries the following entries need to be added in the `fstab` file under the `mounts.d` directory.

Path:/soft/enroot/3.2.0/etc/enroot/mounts.d
File name: 10-system.fstab

```
...........
/soft/enroot/3.2.0 /soft/enroot/3.2.0 none   x-create=dir,rbind,rw,nosuid,nodev,noexec,rslave 0 -1
/scratch/pbs    /scratch/pbs none x-create=dir,rbind,rw,nosuid,nodev,noexec,rslave 0 -1
/var/spool/PBS/aux    /var/spool/PBS/aux none 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 Running an Enroot Job through PBS Professional

Users can use the qsub command to submit a job to PBS Professional. To create the Enroot-specific directory, use "-v enroot=1" as an argument to qsub. You can specify commands within the jobs to create and start the Enroot container on the allocated node by PBS Professional scheduler. You can submit interactive as well as batch jobs, and single-node as well as multi-node jobs. Multi-node jobs require Open MPI.

5.1 Batch Enroot Job

User can use qsub.

```
...........
#!/usr/bin/env bash
#PBS -N Enroot_Batch
#PBS -l select=1:ncpus=12:mpiprocs=12:ngpus=1
#PBS -l walltime=10:00:00
export OMP_NUM_THREADS=1
module load apps/enroot/3.2.0
enroot create --name qe6.6a1 $CONTAINER/hpc+quantum_espresso+v6.6a1.sqsh
enroot start --rw qe6.6a1 bash -c /home/testuser/EnrootTesting/qe/test.sh

or

enroot start qe6.6a1 'cd $PBS_O_WORKDIR && mpirun -np 12 pw.x -inp test.in'
```

5.2 Interactive Enroot Job

Quantum ESPRESSO is a suite for first-principles electronic-structure calculations and materials modeling. To submit a single-node Quantum ESPRESSO application job using Enroot through PBS Professional:

```
qsub   -l select=1:ncpus=20:ngpus=1 -N EnrootTest -v enroot=1 -I

export OMP_NUM_THREADS=1

module load apps/enroot/3.2.0

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

enroot start --rw qe6.6a1

cd  $PBS_O_WORKDIR

mpirun  -np 20 pw.x -inp test.in
```

Note: $CONTAINER refers to the directory path having enroot images (.sqsh).

6 Troubleshooting

For Enroot prolog, epilog, and hook-related errors, refer to $PBS_HOME/ mom_logs on execution nodes.

Use the nvidia-smi command to ensure your job is running on the allocated GPU device and to track GPU utilization.

Use the qstat -f command to see if the Enroot environment variable is set for the job.

For a job's Enroot runtime, data, and cache directories, refer to the job's stderr file.

7 Reference

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

