Cgroups and pam_slurm_adopt

Marshall Garey
SchedMD

Slurm User Group Meeting 2019
Outline

- Cgroups overview
- Cgroups - restricting resources
- Cgroups - accounting
- Cgroups - process tracking
- PAM overview
- pam_slurm_adopt - controlling user access to nodes
What are cgroups?

- Linux “control groups”
- Associate a set of tasks with a set of parameters for one or more subsystems
- Organize processes in a hierarchy in which you can limit various types of resources
- Track processes to prevent stray programs after jobs end
- Implemented via a pseudo-filesystem called cgroupfs
  - Usually mounted at /sys/fs/cgroup
Cgroup Subsystems

- Subsystem - resource controller
  - Different subsystems restrict different resources
  - Slurm uses `cpuacct`, `cpuset`, `devices`, `freezer`, `memory`

```
marshall@voyager:/sys/fs/cgroup$ ls
blkio  cpu,cpuacct  freezer  net_cls  perf_event  systemd
cpu    cpuset       hugetlb  net_cls,net_prio  pids    unified
cpuacct devices    memory   net_prio    rdma
```
Cgroup Hierarchies

- Slurm uses cgroup hierarchies to enforce limits
- Set a limit on a directory; the children directories will inherit the limits of the parent
- Slurm’s hierarchy:
  - `slurm/uid_<uid>/job_<jobid>/step_<stepid>[/task_<taskid>]`
  - The `task_<taskid>` cgroup is used by `jobacct_gather/cgroup` in the memory and `cpuacct` subsystems
Memory Limits with Cgroups

- Memory subsystem
- slurm.conf
  - TaskPlugin=task/cgroup
- cgroup.conf
  - ConstrainRamSpace=yes
  - ConstrainSwapSpace=yes (optional)
Memory Limits with Cgroups

Other cgroup.conf parameters:

- AllowedKmemSpace
- AllowedRAMSpace
- AllowedSwapSpace
- ConstrainKmemSpace
  - Bug in older kernels (<4), do not use
- ConstrainRAMSpace
- ConstrainSwapSpace
- MaxRAMPercent
- MaxSwapPercent
- MaxKmemPercent
- MemorySwappiness
- MinKmemSpace
- MinRAMSpace
Memory Limits with Cgroups

# cgroup.conf
ConstrainRamSpace=yes

$ srun --mem=100 sleep 100&

# This is a garbage number used by Linux that means “no limit”
mmarshall@voyager:/sys/fs/cgroup/memory/slurm/uid_1017$ cat memory.limit_in_bytes
9223372036854771712

# 104857600 == 100 MB - this is our job’s limit
marshall@voyager:/sys/fs/cgroup/memory/slurm/uid_1017$ cat job_10707/memory.limit_in_bytes
104857600
marshall@voyager:/sys/fs/cgroup/memory/slurm/uid_1017$ cat job_10707/step_0/memory.limit_in_bytes
104857600
Restricting Cores with Cgroups

- slurm.conf
  - **Recommended**: TaskPlugin=task/affinity,task/cgroup

- cgroup.conf
  - ContrainCores=yes
  - TaskAffinity=no *(yes if not using task/affinity)*
  - The *task/affinity* plugin will handle affinity, the *task/cgroup* plugin will prevent jobs from using cores they aren’t assigned

- **Uses the cpuset subsystem**
Restricting Cores with Cgroups

$ cat changecpus.batch
#!/bin/bash
#SBATCH -n1 -c4
# Attempt to change my CPU affinity
(taskset -p 0xffffffff)
(taskset -p $)
sleep 600
Restricting Cores with Cgroups

# cgroup.conf
ConstrainCores=no
# Without constraining cores, a job can change its CPU affinity
# to use more CPUs than it should

$ sbatch changecpus.batch
Submitted batch job 10783
$ cat slurm-10783.out
pid 24971's current affinity mask: 303
pid 24971's new affinity mask: ffff
pid 24971's current affinity mask: ffff
Restricting Cores with Cgroups

# cgroup.conf
ConstrainCores=yes
# By constraining cores, a job cannot change its CPU affinity
# to use CPUs outside its allocation

$ sbatch changecpus.batch
Submitted batch job 10784
$ cat slurm-10784.out
pid 25238's current affinity mask: 303
pid 25238's new affinity mask: 303
pid 25238's current affinity mask: 303
Restricting Cores with Cgroups

```
# These processes only have access to CPUs 0, 1, 8, and 9
marshall@voyager:/sys/fs/cgroup/cpuset/slurm/uid_1017/job_10785/step_batch$ cat cpuset.cpus
0-1,8-9
marshall@voyager:/sys/fs/cgroup/cpuset/slurm/uid_1017/job_10785/step_batch$ cat cgroup.procs
25562
25567
25592
marshall@voyager:/sys/fs/cgroup/cpuset/slurm/uid_1017/job_10785/step_batch$ ps -elf | egrep "25562|25567|25592"
4 S root 25562 1 0 80 0 - 85162 - 15:18 ? 00:00:00 slurmstepd: [10785.batch]
4 S marshall 25567 25562 0 80 0 - 3255 wait 15:18 ? 00:00:00 /bin/bash /home/marshall/slurm/19.05/voyager/spool/slurmd-v13/job10785/slurm_script
0 S marshall 25592 25567 0 80 0 - 1868 hrtime 15:18 ? 00:00:00 sleep 600
```
Restricting Devices with Cgroups

- **slurm.conf**
  - TaskPlugin=task/cgroup
  - GresTypes=<comma-separated list of types in gres.conf>

- **cgroup.conf**
  - ConstrainDevices=yes

- **gres.conf**
  - List devices or AutoDetect=nvml
  - nvml is for newer nvidia GPUs
Restricting Devices with Cgroups

- Uses the *devices* subsystem
  - *devices.allow* and *devices.deny* control access to devices
  - All devices in gres.conf that the job does not request are added to *devices.deny* so the job can’t use them

- Must be a Unix device file. Cgroups restrict devices based on major/minor number, not file path

- GPUs are the most common use case, but any Unix device file can be restricted with cgroups
# gres.conf
# I'm calling these devices “gpu” (even though they aren’t physical GPUs)
# for testing purposes so I use the Slurm GPU plugin
NodeName=v[1-13] Name=gpu Count=1 Type=zero File=/dev/zero
NodeName=v[1-13] Name=gpu Count=1 Type=rand File=/dev/urandom

# slurm.conf
GresTypes=gpu

$ cat grestest.batch
#!/bin/bash
MY_OUTPUT_FILE="testfile" user_zero_rand.sh
Restricting Devices with Cgroups

$ cat use_zero_rand.sh
#!/bin/bash
if [ -z $MY_OUTPUT_FILE ]
then
    echo "You must specify the output file with the env var MY_OUTPUT_FILE"
    exit 1
fi
zf=${MY_OUTPUT_FILE}_zero
rf=${MY_OUTPUT_FILE}_rand
echo "Writing results to $zf and $rf"
```
    dd if=/dev/zero of=$zf count=12 bs=1024
dd if=/dev/urandom of=$rf count=12 bs=1024
```

Restricting Devices with Cgroups

# cgroup.conf
ConstrainDevices=no
# Without constraining the devices, a job can use those devices without asking for them

$ sbatch gretest.batch
Submitted batch job 10800
$ cat slurm-10800.out
Writing results to testfile_zero and testfile_rand
12+0 records in
12+0 records out
12288 bytes (12 kB, 12 KiB) copied, 0.000272917 s, 45.0 MB/s
12+0 records in
12+0 records out
12288 bytes (12 kB, 12 KiB) copied, 0.000192889 s, 63.7 MB/s
Restricting Devices with Cgroups

```bash
# cgroup.conf
ConstrainDevices=yes
# By constraining devices, the job cannot use devices outside its allocation

$ sbatch garestest.batch
Submitted batch job 10801
$ cat slurm-10801.out
Writing results to testfile_zero and testfile_rand
dd: failed to open '/dev/zero': Operation not permitted
dd: failed to open '/dev/urandom': Operation not permitted
```
Restricting Devices with Cgroups

# cgroup.conf
ConstrainDevices=yes
# By constraining devices, the job cannot use devices outside its allocation

$ sbatch --gres=gpu:zero:1,gpu:rand:1 jobscripts/grestest.batch
Submitted batch job 10802
$ cat slurm-10802.out
Writing results to testfile_zero and testfile_rand
12+0 records in
12+0 records out
12288 bytes (12 kB, 12 KiB) copied, 0.000358337 s, 34.3 MB/s
12+0 records in
12+0 records out
12288 bytes (12 kB, 12 KiB) copied, 0.0003674 s, 33.4 MB/s
Restricting Devices with Cgroups

marshall@voyager:/sys/fs/cgroup/devices/slurm/uid_1017/job_10803/step_batch$ ls -l

```
total 0
-rw-r--r-- 1 root root 0 Sep  6 08:49 cgroup.clone_children
-rw-r--r-- 1 root root 0 Sep  6 08:49 cgroup.procs
--w------- 1 root root 0 Sep  6 08:49 devices.allow
--w------- 1 root root 0 Sep  6 08:49 devices.deny
-r--r--r-- 1 root root 0 Sep  6 08:49 devices.list
-rw-r--r-- 1 root root 0 Sep  6 08:49 notify_on_release
-rw-r--r-- 1 root root 0 Sep  6 08:49 tasks
```
Accounting with Cgroups

- **slurm.conf:**
  - JobAcctGatherType=jobacct_gather/cgroup
  - JobAcctGatherFrequency=<number of seconds>
  - TaskPlugin=task/cgroup

- **jobacct_gather/cgroup** polls cpuacct.stat and memory.stat files; the remaining accounting info is the same as jobacct_gather/linux

- **Use** AcctGatherProfileType for detailed time-series profiling
Accounting with Cgroups

- Creates task cgroups as children of the step cgroups in the cpuacct and memory subsystems
- Slurm commands to view accounting information
  - `sstat` - accounting information for each step while the job is running
  - `sacct` - accounting information in the database after the job ends
- `cpuacct.stat`
  - user time, system time
- `memory.stat`
  - `total_rss`, `total_pgmajfault`
Accounting with Cgroups

● Example job:
  ○ 2 tasks
  ○ Allocate, fill, then free memory
  ○ Sleep 1 ms
  ○ Rank 0 allocates 1 MB
  ○ Rank 1 allocates 2 MB
Accounting with Cgroups

# slurm.conf
JobAcctGatherFrequency=20
JobAcctGatherType=jobacct_gather/cgroup

$ sbatch -n2 --wrap="srun eat_and_free_mem"
Submitted batch job 94
Accounting with Cgroups

marshall@voyager:/sys/fs/cgroup/cpu,cpuacct/slurm/uid_1017/job_94/step_0$ cat task_0/cgroup.procs
10276

marshall@voyager:/sys/fs/cgroup/cpu,cpuacct/slurm/uid_1017/job_94/step_0$ cat task_1/cgroup.procs
10277

marshall@voyager:/sys/fs/cgroup/cpu,cpuacct/slurm/uid_1017/job_94/step_0$ ps -elf |egrep 10276\|10277
4 R marshall 10276 10270 92 80 0 - 1356 - 18:08 ? 00:14:54
/home/marshall/tools/eat_and_free_mem
4 R marshall 10277 10270 95 80 0 - 1600 - 18:08 ? 00:15:29
/home/marshall/tools/eat_and_free_mem
Accounting with Cgroups

marshall@voyager:/sys/fs/cgroup/cpu,cpuacct/slurm/uid_1017/job_94/step_0$ cat task_0/cpuacct.stat
user 64281
system 21

marshall@voyager:/sys/fs/cgroup/cpu,cpuacct/slurm/uid_1017/job_94/step_0$ cat task_1/cpuacct.stat
user 67171
system 25
Accounting with Cgroups

marshall@voyager:/sys/fs/cgroup/memory/slurm/uid_1017/job_94/step_0$ cat task_0/cgroup.procs
10276
marshall@voyager:/sys/fs/cgroup/memory/slurm/uid_1017/job_94/step_0$ cat task_1/cgroup.procs
10277

marshall@voyager:/sys/fs/cgroup/memory/slurm/uid_1017/job_94/step_0$ ps -elf |egrep 10276|10277
4 R marshall 10276 10270 92 80 0 - 1356 - 18:08 ? 00:07:38 /home/marshall/tools/eat_and_free_mem
4 R marshall 10277 10270 95 80 0 - 1600 - 18:08 ? 00:07:56 /home/marshall/tools/eat_and_free_mem
Accounting with Cgroups

marshall@voyger:/sys/fs/cgroup/memory/slurm/uid_1017/job_94/step_0$ cat task_0/memory.stat
| egrep -i "total_rss |total_pgmajfault"
total_rss 1089536
total_pgmajfault 0

marshall@voyager:/sys/fs/cgroup/memory/slurm/uid_1017/job_94/step_0$ cat task_1/memory.stat
| egrep -i "total_rss |total_pgmajfault"
total_rss 2088960
total_pgmajfault 0
### Accounting with Cgroups

```bash
$ sstat --format jobid,maxrss,MaxRSSTask,MinCPU,mincputask,pids 94.0
    JobID  MaxRSS  MaxRSSTask  MinCPU  MinCPUTask                  Pids
         ------------ ---------- ---------- ---------- ---------- ----------------------
         94.0        2040K          1  17:10.000          0          10276,10277

$ sacct -j94.0 --format=jobid,maxrss,maxrsstask,mincpu,mincputask
    JobID  MaxRSS  MaxRSSTask  MinCPU  MinCPUTask
         ------------   ----------   ----------   ----------
         94.0        2040K          1   00:18:45          0
```
Slurm Process Tracking with Cgroups

- Slurm uses the *freezer* subsystem to do process tracking
- `slurm.conf`
  - `ProctrackType=proctrack/cgroup`
- Subprocesses are also added to the cgroup
  - Process tracking cannot be escaped by users.
  - When the job ends, all processes created by the job are killed.
  - With proctrack/linuxproc or proctrack/pgid, processes can escape process tracking and therefore won’t be killed when the job ends.
Slurm Process Tracking with Cgroups

# slurm.conf
ProctrackType=proctrack/cgroup

marshall@voyager:~/slurm-local/19.05/voyager$ srun my_fork
Parent: pid: 14169
Parent: child pid = 14187
Parent: pid: 14169
Child: my pid: 14187; parent pid: 14169
Child: grandchild pid: 14188; now exit
Parent: pid: 14169
Child: my pid: 14187; parent pid: 14169
Grandchild: my pid: 14188; parent pid: 14187; wait for my parent to exit
Grandchild: my pid: 14188; parent pid: 1
Slurm Process Tracking with Cgroups

marshall@voyager:/sys/fs/cgroup/freezer/slurm/uid_1017/job_12777/step_0$ cat cgroup.procs
14169
14188
What is PAM?

- Linux Pluggable Authentication Modules (PAM) are libraries that authenticate applications or services
- Four management groups
  - auth, account, session, password
  - Allows modules to do different things depending on context
- Stack structure
  - Modules are processed from top to bottom
PAM Control Flags

- **Requisite**
  - Upon failure, stop loading other modules and return a failure

- **Required**
  - Upon failure, load other modules but return failure

- **Sufficient**
  - Upon success, don’t process the rest of the modules and return success
  - Upon failure, continue processing other modules

- **Optional**
  - Failure is ignored
PAM Example

# /etc/pam.d/sshd
# PAM configuration for the Secure Shell service
@include common-auth
account      required     pam_nologin.so
@include common-account
session     [success=ok ignore=ignore module_unknown=ignore default=bad]  pam_selinux.so
close
session     required     pam_loginuid.so
session     optional     pam_keyinit.so force revoke
@include common-session
# /etc/pam.d/sshd continued from previous slide

# /etc/pam.d/sshd continued from previous slide

**session**    optional    pam_motd.so  motd=/run/motd.dynamic

**session**    optional    pam_motd.so noupdate

**session**    optional    pam_mail.so standard noenv # [1]

**session**    required    pam_limits.so

**session**    required    pam_env.so  # [1]

**session**    required    pam_env.so user_readenv=1 envfile=/etc/default/locale

**session**    [success=ok ignore=ignore module_unknown=ignore default=bad]    pam_selinux.so

open

@include common-password
pam_slurm_adopt

- pam_slurm_adopt is a PAM plugin that prevents users from sshing into nodes on which they don’t have a running job.
- The user’s connection is “adopted” into the extern step cgroup of the job so that they cannot exceed cgroup limits.
- All processes created by the user and the user’s connection are killed when the job ends.
pam_slurm_adopt

- Build from source:
  - cd /path/to/slurm/build/directory/contribs/pam_slurm_adopt
  - make && make install

- Build from RPM:
  - slurm.spec will build a slurm-pam_slurm RPM

- Default installation location:
  - /lib/security on Debian systems
  - /lib64/security on RHEL/CentOS or SUSE

- Configure option `--with-pam_dir` changes installation directory
pam_slurm_adopt Configuration

- slurm.conf
  - PrologFlags=contain
    - Enables the creation of the extern step
  - ProctrackType=proctrack/cgroup or proctrack/cray_aries
  - TaskPlugin=task/cgroup
pam_slurm_adopt Configuration

- In /etc/pam.d/, add pam_slurm_adopt.so to sshd or system-auth (depending on the OS)

  ```
  account    required    pam_slurm_adopt.so
  ```

  - Prepend a ‘-’ sign if pam_slurm_adopt is on a shared filesystem. This allows PAM to fail gracefully if pam_slurm_adopt isn’t found so you aren’t locked out of the node while the shared filesystem is mounting or down.

- pam_slurm_adopt is typically the last plugin in the account stack
pam_slurm_adopt Configuration

- Comment out pam_systemd in all files included in the pam stack - it will steal cgroups from Slurm
  - Bug 5920 is an enhancement to work around this issue
- You may need to disable SELinux and comment out pam_selinux
- You may need to stop and mask systemd-logind
  - systemctl stop systemd-logind
  - systemctl mask systemd-logind
pam_slurm_adopt Configuration

- Make sure a different PAM module isn't unintentionally short-circuiting the account stack before pam_slurm_adopt
  - pam_localuser.so
- Intentionally skipping pam_slurm_adopt can be useful to allow privileged users access to the node without a job on the node
- Be careful to not accidentally lock yourself out of a node while configuring pam_slurm_adopt
pam_slurm_adopt Configuration Options

- action_no_jobs
- action_unknown
- action_adopt_failure
- action_generic_failure
- disable_x11
- log_level
- nodename
- service
# /etc/pam.d/sshd
@include common-auth
account    required     pam_nologin.so
@include common-account

# nodename is required if the nodename in slurm.conf is not the same as the hostname
# action_adopt_failure=deny - reject the connection if it can’t be adopted in cgroups
# action_generic_failure=deny - reject the connection if something else goes wrong

account required pam_slurm_adopt.so log_level=debug5 nodename=voyager2 \
  action_generic_failure=deny action_adopt_failure=deny

...
pam_slurm_adopt Configuration

# /etc/pam.d/sshd
@include common-auth
account    required     pam_nologin.so
@include common-account

account sufficient pam_slurm_adopt.so log_level=debug5 nodename=voyager2 \
        action_generic_failure=deny action_adopt_failure=deny

# List users/groups in /etc/security/access.conf that you want to allow or deny.
# Example /etc/security/access.conf that allows group “marshall” and denies everybody else
# +:marshall:ALL
# -:ALL:ALL
account required pam_access.so
# slurm.conf
NodeName=voyager2 Port=33100 CoresPerSocket=1

marshall@voyager:~$ ssh voyager2
Access denied by pam_slurm_adopt: you have no active jobs on this node
Connection closed by 192.168.1.237 port 22
marshall@voyager:~$ srun --nodelist=voyager2 sleep 7890&
[1] 3299
marshall@voyager:~$ squeue

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODES</th>
<th>NODELIST(REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12788</td>
<td>debug</td>
<td>sleep</td>
<td>marshall</td>
<td>R</td>
<td>6:42</td>
<td>1</td>
<td>voyager2</td>
</tr>
</tbody>
</table>

marshall@voyager:~$ ssh voyager2
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-58-generic x86_64)
...
```
marshall@voyager2:~$ cat /proc/self/cgroup
12:hugetlb:/
11:cpu,cpuacct:/slurm/uid_1017/job_12788/stepExtern/task_0
10:memory:/slurm/uid_1017/job_12788/stepExtern/task_0
  9:net_cls,net_prio:/
  8:pids:/system.slice/ssh.service
  7:devices:/slurm/uid_1017/job_12788/stepExtern
  6:cpuset:/slurm/uid_1017/job_12788/stepExtern
  5:perf_event:/
  4:freezer:/slurm/uid_1017/job_12788/stepExtern
  3:rdma:/
  2:blkio:/
  1:name=systemd:/system.slice/ssh.service
  0::/system.slice/ssh.service
```
Slurm documentation

- https://slurm.schedmd.com/slurm.conf.html
- https://slurm.schedmd.com/cgroup.conf.html
- https://slurm.schedmd.com/cgroups.html
- https://slurm.schedmd.com/pam_slurm_adopt.html
Questions?