Slurm Container Support
CNCF Research Users Group

Nathan Rini
10/5/22
Contents

● Slurm container support
● Slurm’s OCI runtime proxy - scrun
● Container staging with Lua plugin in scrun
● Rootless Docker support
● Podman support
● Container limitations in Slurm
● Questions
Adding support for Docker in Slurm

Steps:

1. Slurm needs to be able to:
   a. Run OCI Containers
   b. Schedule jobs with containers
   c. Track containers resources
   d. Enforce all job rules and limits upon containers

2. Docker needs way to interface with Slurm:
   a. Docker uses OCI Runtime to run containers
   b. Slurm needs an OCI Runtime interface
   c. Container images must be reliably sent to and from compute nodes
Slurm OCI Container Support

- Added `--container` (21.08) support to the following:
  - `srun`
  - `salloc`
  - `sbatch`

- Added viewing job container [bundle path] (21.08) and container-id (23.02) to the following:
  - `scontrol show jobs`
  - `scontrol show steps`
  - `sacct`
    - If passed as part of the `--format` argument using
      “Container”
  - `slurmd`, `slurmstepd`, `slurmdbd` & `slurmctld` logs (too many places to list)
OCI Container Support (21.08+)

- Slurm cgroups features apply to the OCI containers
  - All processes should be cleaned up even if the container anchor process dies or processes attempt to become daemons and detach from the session
  - Resource usage can be hard limited and monitored
- Slurm is only going to support unprivileged containers in 21.08, 22.05, 23.02
  - Use existing kernel support for containers
  - Users can already call all of these commands directly
  - Containers must be able to function in an existing host network
- Per host configuration via ‘oci.conf’ in /etc/slurm/
  - Environment variables SLURM_CONTAINER and SLURM_CONTAINER_ID (23.02) will always be set with a value (if present).
OCI Container Support (21.08+)

**srun example**

```bash
$ srun --container=/tmp/centos grep ^NAME /etc/os-release
NAME="CentOS Linux"
```

**salloc example**

```bash
$ salloc --container=/tmp/centos grep ^NAME /etc/os-release
salloc: Granted job allocation 65
NAME="CentOS Linux"
```

```bash
salloc: Relinquishing job allocation 65
```

**Note**: containers have limited permissions and can result in pseudo terminal warnings.
OCI Container Support (21.08+)

sbatch example

```bash
$ sbatch --container=/tmp/centos --wrap 'grep ^NAME /etc/os-release'
Submitted batch job 24419
$ cat slurm-24419.out
NAME="CentOS Linux"
```
Batch Job Use Case (23.02)

MUNGE Perimeter

- slurmctld

Compute Node

- slurmd
- slurmsteprd

Login Node

- sbatch/salloc
- srun
- sshd
- user

Copyright 2022 SchedMD LLC
https://schedmd.com
Container Use Case (23.02)

MUNGE Perimeter

slurmctld

Compute Node

slurmd

slurmsted

Docker / Podman / …

scrn

srun

Login Node

Login Node

ssh

user

Container/Artifact registry

Copyright 2022 SchedMD LLC
https://schedmd.com
OCI runtime proxy - scrun (23.02)

- scrun’s goal is to make containers **boring for users**
  - Users have better things to do than learn about the intricacies of containers
  - Site administrators will have to do setup and maintenance on the configuration
- Use Slurm’s existing infrastructure to run containers on compute nodes
- Automatic staging out and in of containers controlled by system administrators
  - End requirement that users manually prepare their images on compute nodes.
- Interface directly with OCI runtime clients (Docker or Podman or …)
OCI runtime proxy - scrun (23.02)

- Allow users to work with the tools they want while running work on the Slurm cluster
- scrun is a new CLI command to join srun, sbatch and salloc, but no user should ever have to call it directly or even really need to be aware of it
- scrun is still very new and we welcome tickets with requests for enhancements and especially bug reports
scrun via rootless Docker (23.02)

docker run $DOCKER_SECURITY -i ubuntu /bin/sh -c 'grep ^NAME /etc/os-release'
NAME="Ubuntu"

docker run $DOCKER_SECURITY -i centos /bin/sh -c 'grep ^NAME /etc/os-release'
NAME="CentOS Linux"
Rootless Docker Process Trees

- slurmd
- slurmd
- slurmstepd
- crun
- rootlesskit
- Docker
  - containerd
    - containerd-shim-runc-v2
    - srun
    - scrun
- user
- Container/Artifact registry

Copyright 2022 SchedMD LLC
https://schedmd.com
Rootless Docker config (23.02)

~/.config/docker/daemon.json

```json
{
    "default-runtime": "slurm",
    "runtimes": {
        "slurm": {
            "path": "/usr/local/slurm/sbin/scrun"
        }
    },
    "experimental": true,
    "iptables": false,
    "bridge": "none",
    "no-new-privileges": true,
    "rootless": true,
    "selinux-enabled": false
}
```
example:

$ podman run ubuntu /bin/sh -c 'grep ^NAME /etc/os-release'
NAME="Ubuntu"

$ podman run centos /bin/sh -c 'grep ^NAME /etc/os-release'
NAME="CentOS Linux"

$ podman run centos /bin/sh -c 'printenv SLURM_JOB_ID'
77

$ podman run centos /bin/sh -c 'printenv SLURM_JOB_ID'
78
Podman Process Trees

Compute Node

- slurmd
- slurmd
- crun

Podman

- podman
- container/artifact registry
- scrun
- srun

User

Container/Artifact registry

Copyright 2022 SchedMD LLC
https://schedmd.com
Podman config for scrun (23.02)

~/.config/containers/containers.conf:

[containers]
apparmor_profile = "unconfined"
cgroupns = "host"
cgroups = "enabled"
default_sysctls = []
label = false
netns = "host"
no_hosts = true
pidns = "host"
userns = "host"

[engine]
runtime = "slurm"
runtime_supports_nocgroup = [ "slurm" ]
runtime_supports_json = [ "slurm" ]
remote = false

[engine.runtimes]
slurm = [ "/usr/local/slurm/sbin/scrun" ]
scrn - container staging

- scrn needs to stage out the image to remote host at startup
- scrn needs to stage in the image from remote host at job end
- Flexibility required as every site has a different shared file system configuration and data ingress and egress rules.
  - scrn avoids making as many assumptions about the request host vs the execution host in Slurm itself as possible.
  - Site admins must configure where and how images are staged.
scrunch - container staging via Lua

- scrunch’s Lua staging plugin allows site to write custom and simple scripts to move the image to and back from the remote storage.
- scrunch’s staging lua script is located at:
  - /etc/slurm/staging.lua
- Lua script runs as user avoiding any additional privilege escalation risk
- Lua already has JSON support via libraries
- Sites can write a native Slurm plugin if desired instead of using the Lua plugin.
Simplified stage in (to compute node) hook:

```lua
function slurm_stage_in_allocator(id, bundle, spool_path, config_path)
    os.execute(string.format("/usr/bin/env rsync --numeric-ids --delete-after --ignore-errors -a -- %s/ %s/", rootfs, dstfs))
    slurm.set_bundle_path(p)
    slurm.set_root_path(p.."rootfs")
    write_file(jc, json.encode(c))
    return slurm.SUCCESS
end
```
Simplified stage out (from compute node) hook: (this example only deletes the container)

```lua
function slurm_stage_out_allocator(id, bundle, spool_path, config_path)
    os.execute("rm --one-file-system --preserve-root=all -rf ".bundle)
    return slurm.SUCCESS
end
```

See Slurm’s documentation for a full and functional example of the Lua script when slurm-23.02 is officially released.
scrun - limitations (23.02)

- No network namespaces support
  - All containers must run under host network

- No cgroup/apparmor/selinux support via Docker/Podman
  - All the containers are executed remotely making the local system’s security systems irrelevant to the container. Podman allows easy configuration disablement while Docker requires command line arguments

- No container annotation support implemented yet

- No automatic resource selections implemented yet
  - Use of Slurm environment variables allow job property control
  - scrun will currently run the default job with default resources requested

- Container failures may require examining slurmd logs and/or syslogs to determine root cause
Lua must either be compiled with JSON support or the library must be installed.

Slurm may need to be compiled after the JSON library is installed in Lua in order to be able to use it.

scrun will not currently kill or stop the lua script while it is running.

If the Lua staging scripts hang, then the job time limit may be triggered and kill the job.

scrun has the relevant SPANK and clifilter support.

These hooks are not a security device and any user may override them same as srun/sbatch/salloc.

scrun uses standard Slurm RPCs and user permissions. Any user may modify or ptrace their own processes. Any security must be applied at the controller.
scrun - limitations (23.02)

- One podman/docker instance per user per host
  - scrun does not provide information for jobs other than its own
    - Jobs will be visible via squeue/sacct/slurmrestd
  - docker / podman will be blind to any externally started containers

- MUNGE Authentication
  - scrun currently only works via MUNGE
  - Job submission host must have Slurm installed and be in MUNGE perimeter

- JWT Authentication
  - Not currently implemented

- Container IDs must be unique per user
  - Docker or Podman will hand the container ID to scrun verbatim.
  - scrun will try to search for the container by ID
    If the local anchor process is dead.
scrun - limitations (23.02)

- All existing limitations for running containers in Slurm still apply:
  - Containers must have a compatible version of Slurm installed to call Slurm commands
  - MUNGE's socket must be mounted in container to use MUNGE based authentication
  - JWT authentication is possible from container but there are no secrets functionality currently available.
    - Slurm does not support step controls/commands via JWT currently.

- User environment must be explicitly set
  - The environment at time of calling docker/podman will not be inherited by the container unless environment variables are supported by Docker/Podman.
scrun - limitations (23.02)

- scrun will create a local process that must remain alive for the duration of the Job
  - If the local process is killed, then the job will be killed by Slurm. This is the same requirement as any job run via srun
  - scrun can be started from a batch job to avoid submission host uptime requirements
- scrun supports output of Docker JSON formatted log files
  - All output to set to STDOUT instead of being split between STDOUT and STDERR
- Docker current uses an event and poll based system for determining if a container is alive
  - This may result in higher CPU usage on the submission host than only running a container directly via srun
scrn - limitations (23.02)

- scrn requires oci.conf to be fully configured
- I/O restrictions and other limitations from the submission host will affect staging containers in and out
- Slurm (scrn) is run as one of the last steps of starting the container in Docker/Podman
- Slurm has no control over Docker/Podman
  - Docker and podman will need to configured independently of Slurm
  - Only rootless Docker/Podman is supported
    - rootless docker has varying levels of support with older kernels
    - Sites are recommended to run the latest version of their distro and docker to avoid issues
- Not all functionality of Docker/Podman is implemented
scrun - limitations (23.02)

- Online image repositories exist independently of Slurm and may apply bandwidth or usage restrictions
  - These limitations can falsely imply scrun (and Slurm) being slow
  - Sites are suggested to set up local caching proxies if possible
  - scrun does not cache images

- scrun is not a security solution or antivirus or a new security layer
  - It does not scan or reason about the contents of the container images beyond enforcing basic OCI image formatting
  - It will push the images to the execution hosts where the configured and the OCI runtime in oci.conf will be executed to start the containers
  - Users are responsible to ensure the container images are following site policies and procedures while being free of malicious code
scrun - limitations (23.02)

- scrun will only run under the POSIX user/group neither adding or removing abilities/capabilities/permissions from the user and therefore the container processes
- Sites must configure the stage in and stage out Lua scripts to clean up cached images
  - Failure to cleanup the images may result in massive wasteful usage of the filesystems.
- Sites must configure docker/podman to cleanup cached images independently of Slurm
  - Dockers build cache can get very large!
Questions?