Arvados - Feature #9406

[Crunch2] Crunch-run supports using existing cgroup to apply resource allocation

06/14/2016 06:39 PM - Peter Amstutz

Status:	Resolved	**Start date:**	07/28/2016
Priority:	Normal	**Due date:**	
Assigned To:	Tom Clegg	**% Done:**	100%
Category:	Crunch	**Estimated time:**	0.00 hour
Target version:	2016-08-17 sprint		

Description

Subtasks:

Task # 9675: Experiments	**Resolved**
Task # 9698: Review 9406-cgroup-parent	**Resolved**

Associated revisions

Revision 8094a4b4 - 08/04/2016 03:31 PM - Tom Clegg

Merge branch '9406-cgroup-parent'

closes #9406

History

#1 - 06/14/2016 06:39 PM - Peter Amstutz

- Assigned To set to Peter Amstutz

#2 - 07/19/2016 06:29 PM - Brett Smith

- Assigned To changed from Peter Amstutz to Brett Smith

Taking for grooming.

#3 - 07/19/2016 06:29 PM - Tom Clegg

I'm guessing TaskPlugin=task/cgroup is what we want, but I haven't looked closely.

#4 - 07/20/2016 07:31 PM - Tom Clegg

- Target version set to 2016-08-03 sprint

#5 - 07/20/2016 07:31 PM - Tom Clegg

- Category set to Crunch

- Assigned To changed from Brett Smith to Tom Clegg

#6 - 07/20/2016 07:40 PM - Tom Clegg

- Story points set to 1.0

#7 - 07/22/2016 03:41 PM - Tom Clegg

- Status changed from New to In Progress

#8 - 07/28/2016 08:30 PM - Tom Clegg

Some preliminary results:

Configure slurm to put tasks in cgroups:

```
# (in /etc/slurm-llnl/slurm.conf)
TaskPlugin=task/cgroup

# (in /etc/slurm-llnl/cgroup.conf)
```

```
CgroupMountpoint=/sys/fs/cgroup
ConstrainCores=yes
ConstrainDevices=yes
ConstrainRAMSpace=yes
ConstrainSwapSpace=yes
```

Use the `docker run --cgroup-parent` feature (HostConfig.CgroupParent in the dockerclient library) to create the container's cgroup underneath the cgroup slurm gives us.

```
# srun --pty -N 1 --tasks-per-node=1 --mem-per-cpu=1 --cpu_bind=cores --cpus-per-task=1 bash
# docker run -it --cgroup-parent=/slurm/uid_0/job_$SLURM_JOBID/step_$SLURM_STEPID debian:8 cat /proc/self/cgro
up
8:perf_event:/slurm/uid_0/job_19/step_0/2885fc65c3f3246f46d6e921f403cd32475a1850f955fb0dd62b72c92e5e904c
7:blkio:/slurm/uid_0/job_19/step_0/2885fc65c3f3246f46d6e921f403cd32475a1850f955fb0dd62b72c92e5e904c
6:net_cls,net_prio:/slurm/uid_0/job_19/step_0/2885fc65c3f3246f46d6e921f403cd32475a1850f955fb0dd62b72c92e5e904c
5:freezer:/slurm/uid_0/job_19/step_0/2885fc65c3f3246f46d6e921f403cd32475a1850f955fb0dd62b72c92e5e904c
4:devices:/slurm/uid_0/job_19/step_0/2885fc65c3f3246f46d6e921f403cd32475a1850f955fb0dd62b72c92e5e904c
3:cpu,cpuacct:/slurm/uid_0/job_19/step_0/2885fc65c3f3246f46d6e921f403cd32475a1850f955fb0dd62b72c92e5e904c
2:cpuset:/slurm/uid_0/job_19/step_0/2885fc65c3f3246f46d6e921f403cd32475a1850f955fb0dd62b72c92e5e904c
1:name=systemd:/slurm/uid_0/job_19/step_0/2885fc65c3f3246f46d6e921f403cd32475a1850f955fb0dd62b72c92e5e904c
```

slurm is enforcing the memory limit we asked for:

```
# perl -e '$x=""; for(0..128) { $x .= " " x 1000000 }'
Killed
```

However, the memory limit doesn't cover our docker container, even though the container's cgroup is underneath the limited cgroup:

```
# docker run -it --cgroup-parent=/slurm/uid_0/job_$SLURM_JOBID/step_$SLURM_STEPID debian:8 perl -e '$x=""; for
(0..128) { $x .= " " x 1000000 }'
(not killed)
# grep memory /proc/self/cgroup
4:memory:/slurm/uid_0/job_27/step_0
# docker run -it --cgroup-parent=/slurm/uid_0/job_$SLURM_JOBID/step_$SLURM_STEPID debian:8 grep memory /proc/s
elf/cgroup
4:memory:/slurm/uid_0/job_27/step_0/441e8ec95191a639500c1a091395b17c76e1985b1c1fbc33aa7f71440616e717
```

#9 - 07/28/2016 08:34 PM - Tom Clegg

Some versions of docker don't work properly with --cgroup-parent and systemd.

- 1.6 -- no
- 1.9.1 -- ~~tbd~~ yes, but only if docker daemon started with --exec-opt=native.cgroupdriver=cgroupfs
- 1.11.2 -- yes (above tests used 1.11.2)

#10 - 07/28/2016 09:00 PM - Tom Clegg

I thought maybe memory.use_hierarchy was at fault, but /sys/fs/cgroup/memory/slurm/uid_0/job_27/step_0/memory.use_hierarchy is 1 (ditto the child cgroup created by docker).

#11 - 07/28/2016 09:33 PM - Tom Clegg

In the following experiments, the memory limits seem to apply to child procs in the container, but not the *first* pid in the container (the one that looks like pid=1 from inside the container).

docker run ...	RAM effectively limited?
perlscript	no
bash -c perlscript	yes
bash -c 'exec perlscript'	no
xargs -E {} perlscript	yes

Examples

```
# echo foo | docker run -i --cgroup-parent=/slurm/uid_0/job_$SLURM_JOBID/step_$SLURM_STEPID debian:8 xargs -E
{} perl -e '$x=""; for(0..512) { $x .= " " x 1000000 }'
xargs: perl: terminated by signal 9
```

```
# docker run -it --cgroup-parent=/slurm/uid_0/job_$SLURM_JOBID/step_$SLURM_STEPID debian:8 bash -c 'perl -e '\
''$x=""; for(0..512) { $x .= " " x 1000000 }'\'
bash: line 1:     6 Killed                  perl -e '$x=""; for(0..512) { $x .= " " x 1000000 }'
```

```
# docker run -it --cgroup-parent=/slurm/uid_0/job_$SLURM_JOBID/step_$SLURM_STEPID debian:8 bash -c 'exec perl -e '\''$x=""; for(0..512) { $x .= " " x 1000000 }'\''
(ok)
```

#12 - 07/29/2016 03:39 PM - Tom Clegg

Putting aside for a moment the question of whether it's 100% effective, here are some ways to implement this feature:

1. New option to crunch-run, "-slurm-cgroup". If given, set docker container CgroupParent to "/slurm/uid_0/job_$SLURM_JOBID/step_$SLURM_STEPID" (where $SLURM_* come from crunch-run's environment).
2. New options to crunch-run, "-cgroup-parent-subsystem=memory". If given, look up crunch-run's own cgroup for the given subsystem, and use that as CgroupParent for the docker container.

In either case, add a corresponding config option for crunch-dispatch-slurm that causes it to pass the option to crunch-run in its sbatch script.

Alternatively, the behavior could be triggered automatically when the string ":/slurm/" appears in /proc/self/cgroup. In debian8, for example, /proc/self/cgroup in a slurm job looks like this:

- [redacted]

```
8:blkio:/
7:net_cls,net_prio:/
6:freezer:/
5:devices:/slurm/uid_1000/job_28/step_0
4:memory:/slurm/uid_1000/job_28/step_0
3:cpu,cpuacct:/
2:cpuset:/slurm/uid_1000/job_28/step_0
1:name=systemd:/system.slice/slurmd.service
```

#13 - 07/29/2016 03:53 PM - Peter Amstutz

#11 builds the case that we probably want to be providing an "init" process in our containers (which would make some other features easier, such as stdin/stdout redirection).

Would it make sense foe crunch-run to always try to assign the container to crunch-run's cgroup parents? This wouldn't be slurm specific, and would preserve the intended behavior of hierarchical cgroups.

#14 - 08/01/2016 07:35 PM - Tom Clegg

Problem in note-11 solved: I wasn't really measuring whether the OOM-killer killed the program. I was only measuring whether anything printed a "Killed" message. xargs and bash print a message when a program is killed by a signal, but docker run doesn't. Re-testing with this in mind, all cases work.

#15 - 08/01/2016 07:44 PM - Tom Clegg

Peter Amstutz wrote:

> Would it make sense foe crunch-run to always try to assign the container to crunch-run's cgroup parents? This wouldn't be slurm specific, and would preserve the intended behavior of hierarchical cgroups.

I didn't list this option for two reasons:

- It departs from normal docker behavior. Normally, docker uses the "docker" cgroup as the parent for all containers. I expect various things (like container monitoring tools and resource-limiting configurations) assume nobody messes with that. I don't think we should mess with it unless asked.
- It doesn't necessarily answer the question: we'd still have to decide which subsystem has a cgroup worth using as a parent. In the example in note-12, there are three choices: "/", "systemd:/system.slice/slurmd.service", and "/slurm/uid_1000/job_28/step_0". (Docker only lets us specify one parent -- not one parent per subsystem.)

#16 - 08/01/2016 09:06 PM - Tom Clegg

Going with the "-cgroup-parent-subsystem=memory" option.

It's less sensitive to slurm's choice of cgroup name; could potentially be useful for other custom-cgroup setups (e.g., you want a cgroup under "/" instead of "docker"); and leaves default/normal docker behavior alone unless you ask for something different.

#17 - 08/02/2016 02:09 PM - Tom Clegg

9406-cgroup-parent @ 5df1299

Added section to Crunch2 installation

#18 - 08/03/2016 07:01 PM - Radhika Chippada

- Target version changed from 2016-08-03 sprint to Arvados Future Sprints

#19 - 08/03/2016 07:02 PM - Radhika Chippada

- Target version changed from Arvados Future Sprints to 2016-08-17 sprint

#20 - 08/04/2016 02:07 PM - Lucas Di Pentima

The are my comments/questions as far as my limited Go knowledge permits:

services/crunch-dispatch-slurm/crunch-dispach-slurm.go:

- It appears that the crunch-run command cannot be passed as an argument now, why is that? In that case, wouldn't the conditional on line 70 be superfluous?
- Is it convenient to remove crunch-run full path when assigning a default value?

services/crunch-run/cgroup.go:

- Can be the final 'return ""' be eliminated? the Go docs say that log.Fatalf() already do an os.Exit(1), or maybe that return is to avoid compilation errors?

services/crunch-run/crunchrun.go:

- What's the difference between setCgroupParent & expectCgroupParent? I think it would be nice to have a comment on the ContainerRunner struct declaration

#21 - 08/04/2016 02:56 PM - Tom Clegg

Lucas Di Pentima wrote:

> services/crunch-dispatch-slurm/crunch-dispach-slurm.go:
>
> - It appears that the crunch-run command cannot be passed as an argument now, why is that?

Passing an array in CLI args is awkward, and we want to replace command line args with config files anyway, so I figured we might as well drop support now.

> In that case, wouldn't the conditional on line 70 be superfluous?

That covers the case where the config file doesn't mention CrunchRunCommand, or there's no config file at all.

> - Is it convenient to remove crunch-run full path when assigning a default value?

When I've asked ops they've said they would rather let $PATH do its thing, so I avoid specifying /usr/bin/.

> services/crunch-run/cgroup.go:
>
> - Can be the final 'return ""' be eliminated? the Go docs say that log.Fatalf() already do an os.Exit(1), or maybe that return is to avoid compilation errors?

Yes, this is only here because the compiler isn't aware that log.Fatalf exits.

> services/crunch-run/crunchrun.go:
>
> - What's the difference between setCgroupParent & expectCgroupParent? I think it would be nice to have a comment on the ContainerRunner struct declaration

Good call, added comments in 8183ec3... make sense?

#22 - 08/04/2016 03:24 PM - Lucas Di Pentima

Thanks for the explanations!

LGTM.

#23 - 08/04/2016 03:45 PM - Tom Clegg

- Status changed from In Progress to Resolved

- % Done changed from 50 to 100

Applied in changeset arvados|commit:8094a4b4914d892461b2a6fbcbeb10b938a6733b.