Arvados - Bug #18489

CWL: Intermittent Singularity startup failures

11/29/2021 03:41 PM - Tom Schoonjans

Status:	Resolved	**Start date:**	
Priority:	High	**Due date:**	
Assigned To:	Tom Clegg	**% Done:**	0%
Category:	Crunch	**Estimated time:**	0.00 hour
Target version:	2021-12-08 sprint		

Description

We recently started testing a simple CWL workflow in Arvados Crunch 2.3.1. The jobs are executed on a Slurm cluster consisting of 4 nodes with 4 cores each, created as OpenStack VMs.

The workflow consists of three steps:

1. Download 23 vcf.gz files from external, non-federated Arvados cluster using *arv-get*
2. Convert these files to hap/sample using *bcftools convert*
3. Upload the 23 hap/sample file pairs to the external Arvados cluster using *arv-put*

The first two steps use scattering to spread the jobs over the nodes.

The Crunch container runtime has been set to 'singularity'.

Running the Crunch workflow results in intermittent failures of jobs in stage 1 or 2. We are able to get to the desired result after restarting the workflow once or twice. The failure message is always:

```
2021-11-26T18:57:12.918958066Z ERROR [container bcftools_convert_10] (arvc1-xvhdp-ffcjthyczz8gqol) error log:
2021-11-26T18:57:12.918958066Z
2021-11-26T18:57:12.918958066Z   2021-11-26T18:56:45.272198441Z crunch-run Not starting a gateway server (GatewayAuthSecret was not provided by dispatcher)
2021-11-26T18:57:12.918958066Z   2021-11-26T18:56:45.272309584Z crunch-run crunch-run 2.3.1 (go1.17.1) started
2021-11-26T18:57:12.918958066Z   2021-11-26T18:56:45.272331939Z crunch-run Executing container 'arvc1-dz642-tlt1uw3hyppd06g' using singularity runtime
2021-11-26T18:57:12.918958066Z   2021-11-26T18:56:45.272354161Z crunch-run Executing on host 'slurm-worker-blue-4'
2021-11-26T18:57:12.918958066Z   2021-11-26T18:56:45.433328942Z crunch-run container token "token-obfuscated"
2021-11-26T18:57:12.918958066Z   2021-11-26T18:56:45.433798736Z crunch-run Running [arv-mount --foreground --read-write --storage-classes default --crunchstat-interval=10 --file-cache 268435456 --mount-by-pdh by_id --disable-event-listening --mount-by-id by_uuid /tmp/crunch-run.arvc1-dz642-tlt1uw3hyppd06g.1063908694/keep2883686336]
2021-11-26T18:57:12.918958066Z   2021-11-26T18:56:49.241177040Z crunch-run Fetching Docker image from collection '3126bcd60bc91b04916bae8dfadede7d+177'
2021-11-26T18:57:12.918958066Z   2021-11-26T18:56:49.372828520Z crunch-run Using Docker image id "sha256:c74eb4247d8550fa2df0c5275a9dd3b34cb105347cc0fcefdf5a05749faaf0a1"
2021-11-26T18:57:12.918958066Z   2021-11-26T18:56:49.372958779Z crunch-run Loading Docker image from keep
2021-11-26T18:57:12.918958066Z   2021-11-26T18:56:50.079531206Z crunch-run Starting container
2021-11-26T18:57:12.918958066Z   2021-11-26T18:56:50.080267548Z crunch-run Waiting for container to finish
2021-11-26T18:57:12.918958066Z   2021-11-26T18:56:52.939218932Z stderr FATAL:   container creation failed: mount /proc/self/fd/3->/usr/local/var/singularity/mnt/session/rootfs error: while mounting image /proc/self/fd/3: failed to find loop device: could not attach image file to loop device: failed to set loop flags on loop device: resource temporarily unavailable
2021-11-26T18:57:12.918958066Z   2021-11-26T18:56:53.365279723Z crunch-run Complete
2021-11-26T18:57:13.026434934Z ERROR [container bcftools_convert_10] unable to collect output from d41d8cd98f00b204e9800998ecf8427e+0:
```

I am attaching the CWL files (with some redacting) that were used during our test workflow.

Related issues:			
Related to Arvados - Support #18566: Document singularity loopback device con...	**Resolved**	**12/10/2021**	

History

#1 - 11/29/2021 04:23 PM - Peter Amstutz

from gitter:

"We have tried this with both Singularity 3.7.4 and 3.9.1 but the error pops up in both versions."

#2 - 11/29/2021 04:27 PM - Peter Amstutz

Just to confirm, slurm is scheduling more than one job at a time on these nodes? E.g. with 4 cores that 4 jobs scheduled per core?

#3 - 11/29/2021 04:35 PM - Tom Schoonjans

Peter Amstutz wrote:

> Just to confirm, slurm is scheduling more than one job at a time on these nodes? E.g. with 4 cores that 4 jobs scheduled per core?

In the case of `arv-get`: yes. `bcftools convert` is using two cores per job, so two per node in the second workflow step.

#4 - 11/29/2021 04:39 PM - Joshua Randall

I can confirm that with 4 nodes (4 cores each), this minimal reproducer fails with the same error on the same system:

```
$ cat echo-many.cwl
#!/usr/bin/env cwl-runner

cwlVersion: v1.0
class: Workflow

requirements:
  ScatterFeatureRequirement: {}

inputs:
  message_array: string[]

steps:
  echo:
    run: echo-tool.cwl
    scatter: message
    in:
      message: message_array
    out: []

outputs: []
$ cat 32-inputs.cwl
message_array:
  - Hello World 1
  - Hello World 2
  - Hello World 3
  - Hello World 4
  - Hello World 5
  - Hello World 6
  - Hello World 7
  - Hello World 8
  - Hello World 9
  - Hello World 10
  - Hello World 11
  - Hello World 12
  - Hello World 13
  - Hello World 14
  - Hello World 15
  - Hello World 16
  - Hello World 17
  - Hello World 18
  - Hello World 19
  - Hello World 20
  - Hello World 21
  - Hello World 22
  - Hello World 23
```

```
  - Hello World 24
  - Hello World 25
  - Hello World 26
  - Hello World 27
  - Hello World 28
  - Hello World 29
  - Hello World 30
  - Hello World 31
  - Hello World 32
$ arvados-cwl-runner --disable-reuse echo-many.cwl 32-inputs.cwl
```

Note that it failed the first time without `--disable-reuse` but this flag is useful for making the failure reproducible. Running with these 32 inputs has failed 4/4 times I have tried it. Running with 16 inputs has failed 1/2 times, and running with 4 inputs succeeded the only time I tried.

#5 - 11/29/2021 04:55 PM - Joshua Randall

dmesg on one of the nodes during some of these runs:

```
[Mon Nov 29 16:30:53 2021] loop_set_status: loop6 () has still dirty pages (nrpages=2)
[Mon Nov 29 16:30:53 2021] loop_set_status: loop6 () has still dirty pages (nrpages=2)
[Mon Nov 29 16:30:53 2021] loop_set_status: loop6 () has still dirty pages (nrpages=2)
[Mon Nov 29 16:30:54 2021] loop_set_status: loop6 () has still dirty pages (nrpages=2)
[Mon Nov 29 16:30:54 2021] loop_set_status: loop6 () has still dirty pages (nrpages=2)
[Mon Nov 29 16:30:56 2021] blk_update_request: I/O error, dev loop6, sector 4096 op 0x0:(READ) flags 0x80700 p
hys_seg 1 prio class 0
[Mon Nov 29 16:30:56 2021] blk_update_request: I/O error, dev loop6, sector 4096 op 0x0:(READ) flags 0x0 phys_
seg 1 prio class 0
[Mon Nov 29 16:30:56 2021] Buffer I/O error on dev loop6, logical block 512, async page read
[Mon Nov 29 16:30:56 2021] blk_update_request: I/O error, dev loop6, sector 4096 op 0x0:(READ) flags 0x0 phys_
seg 1 prio class 0
[Mon Nov 29 16:30:56 2021] Buffer I/O error on dev loop6, logical block 512, async page read
[Mon Nov 29 16:34:07 2021] loop_set_status: loop6 () has still dirty pages (nrpages=2)
[Mon Nov 29 16:34:07 2021] loop_set_status: loop6 () has still dirty pages (nrpages=2)
[Mon Nov 29 16:34:07 2021] loop_set_status: loop6 () has still dirty pages (nrpages=2)
[Mon Nov 29 16:34:08 2021] loop_set_status: loop6 () has still dirty pages (nrpages=2)
[Mon Nov 29 16:34:08 2021] loop_set_status: loop6 () has still dirty pages (nrpages=128)
```

#6 - 11/29/2021 05:57 PM - Tom Clegg

The singularity code (current version at https://github.com/sylabs/singularity/blob/master/pkg/util/loop/loop.go) creates/chooses a loop device where loop_set_fd syscall succeeds, then tries the loop_set_status syscall, waiting 250s and retrying up to 5 times after EAGAIN, then gives up.

The "has still dirty pages" kernel message comes with the EAGAIN response.
https://github.com/torvalds/linux/blob/master/drivers/block/loop.c#L1268-L1271

I wonder whether singularity would do better to handle EAGAIN by trying to use/create the next loop device, only resorting to a delay-and-retry loop when reaching MaxLoopDevices. It seems unnecessary to wait around for the kernel to clean up loop6 if we could start using loop7 right now.

As an aside, the EAGAIN kernel behavior itself seems to be in flux.

https://lkml.org/lkml/2020/2/27/2267

I haven't thought of a reasonable way to work around this from the Arvados end.

#7 - 11/29/2021 06:49 PM - Peter Amstutz

You meant 250ms not 250s.

So retrying 5 times is only a total of 1.25s of waiting.

Perhaps there is a race condition aspect, if a multiple singularity processes are trying to start at once, could they all decide they want to take the same loop device?

#8 - 11/29/2021 10:23 PM - Peter Amstutz

- Target version set to 2021-12-08 sprint

#9 - 11/29/2021 10:27 PM - Peter Amstutz

This is pure speculation but if the problem is singularity racing with other singularity processes to acquire a loopback device, maybe we could use some kind of locking strategy to prevent multiple singularity processes starting within some time window (10s?).

#10 - 11/30/2021 03:46 PM - Tom Clegg

- Status changed from New to In Progress

Rephrasing note-6 above, I think the problem is that singularity chooses the first /dev/loopN device (sorted by N in the filename) that meets nearly all (but not all) of the criteria it needs:

- exists or can be created
- not already in use
- ~~can be configured with the desired parameters (offset, size limit, flags, etc)~~

Once it has chosen a device that meets the first two criteria, it waits 1.5s for it to meet the third criterion, then gives up.

It could be fixed by choosing the first device that meets all three criteria. With that change, the 1.5s waiting period is probably not even needed.

So far my only idea for working around this from the outside (without modifying singularity itself) is to detect the nearly-but-not-quite-usable /dev/loopN device files and replace them with device files that point to different loop devices -- e.g., if /dev/loop1 points to loop device number 1 which is not usable, delete /dev/loop1 and create a new /dev/loop1 that actually maps to loop device number 8 which is usable. This seems considerably more painful than patching singularity.

#11 - 11/30/2021 04:31 PM - Tom Clegg

Another possibility is for crunch-run to test for the situation where singularity will fail, and wait (longer than 1.5s) for it to clear up. Maybe 60 seconds? But this would achieve exactly the same thing as increasing "maxRetries" in singularity, so patching singularity seems more reasonable.

#12 - 11/30/2021 05:18 PM - Tom Clegg

Summary so far:

Easiest thing that could get your workflows running reliably is to patch singularity to wait up to 150 seconds before giving up instead of 1.5 seconds. Just change maxRetries from 5 to 500 here: https://github.com/sylabs/singularity/blob/master/pkg/util/loop/loop.go#L177

Regardless of whether that gets things running, I think it is worth refactoring the retry logic in that function, and upstreaming it:

- Move the F_SETFD and CmdSetStatus64 syscalls into the "try each /dev/loopN" loop
- After exhausting all loop devices, return EAGAIN instead of "no loop devices available" if any of them ended in EAGAIN
- Rename AttachFromFile to attachFromFile and add an AttachFromFile wrapper that retries attachFromFile up to 5x after EAGAIN

The benefits would be

- if /dev/loop6 says EAGAIN but /dev/loop7 works, use /dev/loop7 right away instead of waiting for /dev/loop6
- /dev isn't flock()ed while waiting, as it is in the current implementation

I expect that with this change, delay-and-retry will be extremely rare, but removing it could conceivably break a case that could have worked with the current code, so seems best to leave it in.

#13 - 11/30/2021 09:57 PM - Peter Amstutz

- Assigned To set to Tom Clegg

#14 - 12/07/2021 09:35 AM - Tom Schoonjans

A PR has been opened at https://github.com/sylabs/singularity/pull/458

Hope it gets merged in soon.

Many thanks Tom for all your help investigating this issue!

#15 - 12/08/2021 04:15 PM - Peter Amstutz

- Status changed from In Progress to Resolved

#16 - 12/08/2021 04:15 PM - Peter Amstutz

- Related to Support #18566: Document singularity loopback device conflict bug added

#17 - 12/08/2021 04:15 PM - Peter Amstutz

Will document "if you get this error, check for a newer version of singularity, see this PR" in #18566

Files

arv-put.cwl	1.14 KB	11/29/2021	Tom Schoonjans
arv-get.cwl	718 Bytes	11/29/2021	Tom Schoonjans
bcftools-convert-vcf2hapsample.cwl	747 Bytes	11/29/2021	Tom Schoonjans