Arvados - Bug #10585

crunch doesn't end jobs when their arv-mount dies

11/22/2016 03:28 PM - Joshua Randall

Status:	Resolved	**Start date:**	11/22/2016
Priority:	Normal	**Due date:**	
Assigned To:	Tom Clegg	**% Done:**	67%
Category:	Crunch	**Estimated time:**	0.00 hour
Target version:	2017-03-01 sprint		

Description

We have slowly been accumulating a large number of jobs across our cluster that are still "running" although they are not doing anything. This appears to be because their corresponding arv-mount process has died for some reason (if one does a `docker exec` to get into the crunch container, `ls /keep` says "Transport endpoint not connected":

```
crunch@4e50cdfd50db:/tmp/crunch-job$ ls /keep
ls: cannot access /keep: Transport endpoint is not connected
```

My expectation would be that if arv-mount dies, the crunch container should be destroyed so that the resources can be freed up.

Subtasks:

Task # 10755: Add test	**Resolved**
Task # 10725: crunchstat option to kill child when parent dies	**Resolved**
Task # 10730: Review 10585-crunchstat-lost-parent	**Resolved**

Related issues:

Related to Arvados - Bug #10586: Python keep client (CollectionWriter) appear...	**Resolved**	**11/22/2016**
Related to Arvados - Bug #10777: [Crunch2] crunch-run: stop the container and...	**Resolved**	**02/24/2017**

Associated revisions

Revision b2cf2b9b - 12/22/2016 03:14 PM - Tom Clegg

Merge branch '10585-crunchstat-lost-parent' refs #10585

History

#1 - 11/22/2016 09:25 PM - Tom Morris

- Target version set to 2016-12-14 sprint

#2 - 11/23/2016 08:10 PM - Tom Clegg

- Assigned To set to Tom Clegg

#3 - 11/28/2016 08:15 PM - Peter Amstutz

Hi Josh, we have some questions:

- Is there an arv-mount process still running, or did it crash?
- Is there any evidence that arv-mount was out-of-memory killed?
- Are there any logs indicating what happened to arv-mount?
- By "running" do you mean crunch-job is still running and expecting the job to complete, or that crunch-job finished but the docker container did not exit?
- What processes are running inside the Docker containers that are "not doing anything"? Can you use strace to find out what they are doing / where they are stuck?
- Is crunchstat still running?

#4 - 11/28/2016 08:33 PM - Joshua Randall

This was one of two phenotypes that was causing lockups across our cluster (the other being 10586). For this one, the arv-mount process was no longer running.

I didn't see any evidence of arv-mount being OOM killed, and there is nothing at all in kern.log* from the OOM killer.

I didn't find any logs regarding what happened to arv-mount but I didn't look "everywhere" for it. Unfortunately any of the other places I would have looked (crunch-dispatch-jobs log or syslog) have been rotated out at this point.

The docker container was still running, my crunch script was still running, GATK was still running, crunchstat was still running, and crunch-job was still running and forwarding crunchstat output all the way back to the job log. It took me a while to realise that jobs were stuck because crunchstat output was still coming through in the job logs, and only on closer inspection did I find that there was no actual job output interspersed with them. From what I could tell, the only thing that was supposed to be running but wasn't was arv-mount (and as a result, the I/O on /keep from inside the container was blocking).

I killed all these containers last week, but we did a lot of strace investigating before killing them. My crunch script was waiting for output from GATK. GATK appeared to be stuck waiting on I/O of some kind (but it is hard to tell exactly what because Java). I assume it was waiting on read operations on one of its inputs (all of which were in keep).

Yes, as above, crunchstat was still running and merrily reporting the (lack of) activity.

#5 - 12/14/2016 04:27 PM - Tom Clegg

Sounds like when arv-mount dies unexpectedly we go from

```
├─dockerd───docker-container───docker-container───GATK
│
├─slurmd───slurmstepd───arv-mount───crunchstat───docker-run
```

to

```
├─dockerd───docker-container───docker-container───GATK
│
├─slurmd───slurmstepd───crunchstat───docker-run
```

...and if GATK is waiting forever for FUSE (because FUSE), then

- docker-run will wait forever for the container to exit
- the job will be "running" forever

If we assume the worst -- arv-mount gets killed so forcefully that it has no opportunity to clean up -- we could:

1. Insert *another* process before arv-mount that cleans things up after arv-mount exits.

- fusermount -u -z /keep
- kill the docker container (we already use docker run --name=$containername so we should be able to pass the same name to the watchdog so it can choose the right one)

or

2. Fork a watchdog process from arv-mount at startup (before starting any threads) which can clean up (as above) if the main process dies.

- Easy enough to communicate the mount point to the child -- but what if the mount fails and the mount point belongs to a *different* fuse process when the watchdog tries to unmount it? Should the main process sigkill the watchdog if exiting when cleanup isn't appropriate?
- How does the watchdog know which docker container to kill? crunch-job could pass an entire cleanup command to arv-mount for the watchdog to use. Seems fragile.

or

3. Have crunchstat monitor its parent PID and, if the parent dies, kill crunchstat's child with SIGTERM (which should stop the container, being the default value for docker run --stop-signal) and exit.

- -signal-on-dead-ppid=15 ?

#6 - 12/14/2016 06:07 PM - Peter Amstutz

If GATK is in an uninterruptible I/O wait, killing the container might not be enough, keep still needs to be unmounted (possibly with umount -f)

#7 - 12/14/2016 08:03 PM - Tom Clegg

- Status changed from New to In Progress

- Target version changed from 2016-12-14 sprint to 2017-01-04 sprint

- Story points set to 0.5

#8 - 12/16/2016 09:19 PM - Tom Clegg

10585-crunchstat-lost-parent @ fc390927833d14b6c439db8ea72d3d52b60a5e6d

Worst test case ever. :(

> If GATK is in an uninterruptible I/O wait, killing the container might not be enough, keep still needs to be unmounted (possibly with umount -f)

It does seem plausible that docker won't do a good job in some situations, but even so I think this feature makes sense for the situations where docker run --stop-signal does what it claims to do.

Even if the container is still wedged in some fuse state, just ending the docker run process itself should be a big improvement over the reported behavior:

- The affected job will end (so it can be retried) instead of hanging forever.
- Next time we try to run a job on this node, crunch-job's "fusermount -u -z" stuff will potentially help clean things up.

#9 - 12/19/2016 06:48 PM - Lucas Di Pentima

- Is there a reason for the default values of signalOnDeadPPID & ppidCheckInterval vars to be defined in different places?
- If the default behaviour is to signal the child process with 15, maybe adding a note on the argument documentation string that say something like “Use zero to disable” would be helpful.
- On func sendSignalOnDeadPPID(), the monitored PID is passed by argument but the check interval is read from the global variable, wouldn’t be more consistent to give the same treatment to both values?
- What happens if the user passes a negative signal number? I’ve searched golang’s docs but couldn’t find a clear answer to this case.
- Is it OK to break from the loop leaving the Ticker running? (Line 126)

#10 - 12/22/2016 06:34 AM - Tom Clegg

Lucas Di Pentima wrote:

> - Is there a reason for the default values of signalOnDeadPPID & ppidCheckInterval vars to be defined in different places?
> - On func sendSignalOnDeadPPID(), the monitored PID is passed by argument but the check interval is read from the global variable, wouldn’t be more consistent to give the same treatment to both values?

Good points, thanks. Fixed both of these to be consistent.

> - If the default behaviour is to signal the child process with 15, maybe adding a note on the argument documentation string that say something like “Use zero to disable” would be helpful.

Good call, added.

> - What happens if the user passes a negative signal number? I’ve searched golang’s docs but couldn’t find a clear answer to this case.

I think we'd get "error: sending signal -4: invalid argument" or something. I've added a check for negative numbers in case someone tries to use -1 to disable. I considered checking for too-large numbers too, but that starts to feel a bit too much like second-guessing the OS's list of signals.

> - Is it OK to break from the loop leaving the Ticker running? (Line 126)

I don't think that would break anything (tickers don't accumulate unbounded backlogs of ticks etc) but it does seem like good form to clean up anyway so I've updated that.

cc94954f69ed2d26451bae6610b38de260d2252f

#11 - 12/22/2016 03:11 PM - Lucas Di Pentima

LGTM. Ran services/crunchstat tests locally without issues.

#12 - 12/22/2016 03:23 PM - Tom Clegg

Now that -signal-on-dead-ppid=15 is the default, we should see an improvement here:

- If arv-mount dies, crunchstat will signal docker to stop the container.
- *Assuming things aren't too wedged for even SIGTERM to make `docker run` stop,* this will cause slurmstepd to finish the task, and return arv-mount's non-zero exit status to crunch-job
- crunch-job will either reattempt the task or give up, as with other failures.

TODO:

- In crunch2, crunch-run could do the analogous thing, "notice if arv-mount dies, and kill the container." (In crunch2, crunch-run starts arv-mount, instead of the other way around.) (this is now #10777)

#13 - 12/28/2016 03:17 PM - Tom Clegg

- Status changed from In Progress to Feedback

#14 - 01/04/2017 08:06 PM - Tom Clegg

- Target version changed from 2017-01-04 sprint to 2017-01-18 sprint

- Story points changed from 0.5 to 0.0

#15 - 01/18/2017 06:53 PM - Tom Clegg

- Target version changed from 2017-01-18 sprint to 2017-02-01 sprint

#16 - 02/01/2017 08:08 PM - Tom Clegg

- Target version changed from 2017-02-01 sprint to 2017-02-15 sprint

#17 - 02/15/2017 08:01 PM - Tom Clegg

- Target version changed from 2017-02-15 sprint to 2017-03-01 sprint

#18 - 03/01/2017 08:22 PM - Tom Clegg

- Status changed from Feedback to Resolved