Arvados - Bug #4044

[Crunch] job spawns tasks but never runs them, gets stuck waiting for first task to end, even though it has already ended.

09/30/2014 09:08 PM - Ward Vandewege

Status:	Resolved	**Start date:**	10/07/2014
Priority:	Normal	**Due date:**	
Assigned To:	Tom Clegg	**% Done:**	100%
Category:	Crunch	**Estimated time:**	0.00 hour
Target version:	2014-10-29 sprint		

Description

The first task in job 9tee4-8i9sb-jykwrdj3quy00t5 create a whole lot of new tasks. Then it exits (succesfully, no docker container running, and gets marked as such in the api server) but somehow the job does not realize that, and does not get updated. The job then sits around waiting for the first task to finish.

The job record is stuck with 1 running, 0 todo.

All the job tasks are created properly though. The first one, as an example, is 9tee4-ot0gb-vmlrb41ai6iswlr.

I verified that while the job is stuck, it is not trying to read data from Keep.

Note: this particular job has a malformed input collection (the files are in a flat hierarchy, rather than split up per directory). It will never complete successfully. So the bug is that Crunch gets into this weird state with a stuck job; it should fail explicitly if it can't run the tasks.

This happened again on job 9tee4-8i9sb-50r7da1j0ks4db1

Subtasks:

Task # 4130: Review 4044-crunchstat-wait	**Resolved**

Associated revisions

Revision 4be23b41 - 10/08/2014 02:58 PM - Tom Clegg

Merge branch '4044-crunchstat-wait' refs #4044

History

#1 - 09/30/2014 09:12 PM - Ward Vandewege

- Subject changed from [Crunch] job spawns tasks but never runs them to [Crunch] job spawns tasks but never runs them, gets stuck waiting for first

#2 - 09/30/2014 09:13 PM - Ward Vandewege

- Subject changed from [Crunch] job spawns tasks but never runs them, gets stuck waiting for first to [Crunch] job spawns tasks but never runs them, gets stuck waiting for first task to end, even though it has already ended.

#3 - 09/30/2014 09:16 PM - Ward Vandewege

- Description updated

#4 - 09/30/2014 09:22 PM - Abram Connelly

Though it may be unrelated, since the files got dumped into the base directory instead of being split up into their own sub-directories, the blocks associated to individual filenames got interleaved from the collation at the end.

#5 - 10/01/2014 09:03 PM - Ward Vandewege

- Description updated

#6 - 10/01/2014 09:11 PM - Abram Connelly

Another clue: I gave 'run-command' the 'PersonToTileLibraryShim' script to run but it didn't exist. The actual script is 'PersonToTileLibraryShim.sh' (in job 9tee4-8i9sb-50r7da1j0ks4db1)

#7 - 10/01/2014 10:27 PM - Abram Connelly

On investigation, it looks like line 127 in 'run-command' is causing the issue:

```
    122 try:
    123     if "task.foreach" in jobp:
    124         if arvados.current_task()['sequence'] == 0:
    125             var = jobp["task.foreach"]
    126             items = get_items(jobp, jobp[var])
    127             logging.info("parallelizing on %s with items %s" % (var, items))
    128             if items is not None:
    129                 for i in items:
    130                     params = copy.copy(jobp)
    131                     params[var] = i
    132                     arvados.api().job_tasks().create(body={
    133                         'job_uuid': arvados.current_job()['uuid'],
```

That is, the 'logging.info("parallelizing on %s with items %s" % (var, items))' line.

Where the 'items' variable is a list of 863 length with each entry of about 60-80 characters. Taking it out causes the job to fail (as expected). Putting it back in and the job hangs. When that 'logging.info(...)' call is in place, execution in run-command does not proceed past that point.

#8 - 10/02/2014 03:22 PM - Abram Connelly

Updating to the latest 'run-command' and 'crunchutil' in the crunch scripts appears to sovle the issue. The pipeline/job is still failing but for 'valid' reasons. The job is not hanging anymore.

Using 'run-command' and 'crunchutil' from git hash 5d006b95fdcbb48afffef8272b4d2071acb5221b.

Here is a diff from the versio of 'run-command' and 'subst.py' I was using to the new version:

```
$ diff run-command $HOME/prog/arvados/crunch_scripts/run-command
4c4,9
< logging.basicConfig(level=logging.INFO, format="run-command: %(message)s")
---
>
> logger = logging.getLogger('run-command')
> log_handler = logging.StreamHandler()
> log_handler.setFormatter(logging.Formatter("run-command: %(message)s"))
> logger.addHandler(log_handler)
> logger.setLevel(logging.INFO)
127c132
<             logging.info("parallelizing on %s with items %s" % (var, items))
---
>             logger.info("parallelizing on %s with items %s" % (var, items))
168,169c173,178
<     logging.info("{}{}{}".format(' '.join(cmd), (" < " + stdinname) if stdinname is not None else "", (" > "
 + stdoutname) if stdoutname is not None else ""))
<
---
>     logger.info("{}{}{}".format(' '.join(cmd), (" < " + stdinname) if stdinname is not None else "", (" > "
+ stdoutname) if stdoutname is not None else ""))
> except subst.SubstitutionError as e:
>     logger.error(str(e))
>     logger.error("task parameters were:")
>     logger.error(pprint.pformat(taskp))
>     sys.exit(1)
171,173c180,182
<     logging.exception("caught exception")
<     logging.error("task parameters was:")
<     logging.error(pprint.pformat(taskp))
---
>     logger.exception("caught exception")
>     logger.error("task parameters were:")
>     logger.error(pprint.pformat(taskp))
189c198
<         logging.critical("terminating on signal %s" % sig.sig)
---
>         logger.critical("terminating on signal %s" % sig.sig)
192c201
<         logging.info("completed with exit code %i (%s)" % (rcode, "success" if rcode == 0 else "failed"))
---
>         logger.info("completed with exit code %i (%s)" % (rcode, "success" if rcode == 0 else "failed"))
195c204
<     logging.exception("caught exception")
---
```

```
>     logger.exception("caught exception")
205c214
< logging.info("the following output files will be saved to keep:")
---
> logger.info("the following output files will be saved to keep:")
209c218
< logging.info("start writing output to keep")
---
> logger.info("start writing output to keep")
219c228
<     outcollection = robust_put.upload(outdir)
---
>     outcollection = robust_put.upload(outdir, logger)
$
$
$ diff crunchutil/subst.py $HOME/prog/arvados/crunch_scripts/crunchutil/subst.py
3a4,6
> class SubstitutionError(Exception):
>     pass
>
31c34
<         raise Exception("Substitution error, mismatched parentheses {}".format(c))
---
>         raise SubstitutionError("Substitution error, mismatched parentheses {}".format(c))
49c52
<         raise Exception("$(glob): No match on '%s'" % v)
---
>         raise SubstitutionError("$(glob): No match on '%s'" % v)
60d62
<         #print("c is", c)
62,70c64,75
<         if m is not None:
<             v = do_substitution(p, c[m[0]+2 : m[1]])
<             var = True
<             for sub in subs:
<                 if v.startswith(sub):
<                     r = subs[sub](v[len(sub):])
<                     var = False
<                     break
<             if var:
---
>         if m is None:
>             return c
>
>         v = do_substitution(p, c[m[0]+2 : m[1]])
>         var = True
>         for sub in subs:
>             if v.startswith(sub):
>                 r = subs[sub](v[len(sub):])
>                 var = False
>                 break
>         if var:
>             if v in p:
71a77,82
>             else:
>                 raise SubstitutionError("Unknown variable or function '%s' while performing substitution on
'%s'" % (v, c))
>             if r is None:
>                 raise SubstitutionError("Substitution for '%s' is null while performing substitution on '%s'
" % (v, c))
>             if not isinstance(r, basestring):
>                 raise SubstitutionError("Substitution for '%s' must be a string while performing substitutio
n on '%s'" % (v, c))
73,75c84
<             c = c[:m[0]] + r + c[m[1]+1:]
<         else:
<             return c
---
>         c = c[:m[0]] + r + c[m[1]+1:]
```

#9 - 10/03/2014 05:38 PM - Tom Clegg

- Status changed from New to In Progress

- Story points set to 1.0

#10 - 10/03/2014 06:41 PM - Ward Vandewege

- Target version changed from Bug Triage to 2014-10-08 sprint

#11 - 10/07/2014 02:10 PM - Tom Clegg

There's still an unresolved mystery in 9tee4-8i9sb-50r7da1j0ks4db1: after doing approximately nothing for ~4 minutes, crunchstat says "user -1" and then goes silent for 10 minutes until the job is cancelled.

```
2014-10-01_19:15:46 9tee4-8i9sb-50r7da1j0ks4db1 11998 0 stderr crunchstat: cpuacct.stat user 0
2014-10-01_19:15:46 9tee4-8i9sb-50r7da1j0ks4db1 11998 0 stderr crunchstat: cpuacct.stat sys 0
2014-10-01_19:15:56 9tee4-8i9sb-50r7da1j0ks4db1 11998 0 stderr crunchstat: cpuacct.stat user -1
2014-10-01_19:15:56 9tee4-8i9sb-50r7da1j0ks4db1 11998 0 stderr crunchstat: cpuacct.stat sys 0
2014-10-01_21:25:10 9tee4-8i9sb-50r7da1j0ks4db1 11998  Job cancelled at 2014-10-01T21:25:10Z by user
9tee4-tpzed-2l9nsfm4mgltv4c
2014-10-01_21:25:10 9tee4-8i9sb-50r7da1j0ks4db1 11998  wait for last 1 children to finish
2014-10-01_21:25:10 9tee4-8i9sb-50r7da1j0ks4db1 11998 0 sending 2x signal 2 to pid 12429
```

Reading crunchstat.go it seems that if cpuacct/stat starts saying "user 0" -- or, more likely, an error occurs during os.Open, iouitl.ReadAll, or fmt.Sscanf -- crunch-stat will print a negative CPU usage and then recede into "don't output anything" mode until cpuacct/stat starts reporting non-zero numbers in the expected format.

IIRC it was reported that the docker container was not running during this 10-minute period. This would be consistent with the "cpuacct/stat cannot be opened+read+parsed" behavior. This suggests crunchstat failed to notice that its child had exited.

Should we put the cmd.Wait() block before the 2× <-finish_chan?

#12 - 10/07/2014 08:47 PM - Tom Clegg

4044-crunchstat-wait @ 292856a

This branch fixes/simplifies a bunch of pipe/channel stuff in crunchstat. I haven't found a way to reproduce the hang, but I did fix a few bugs that will *at least* prevent crunchstat from going silent (and reporting negative CPU usage) due to unreadable cgroup files. I also removed some unnecessary attempts to synchronize stdout/stderr channels that could conceivably contribute to a deadlock condition.

Here's a quick example of testing crunchstat using the "ubuntu" docker image, in the absence of a proper test suite:

```
GOPATH="/tmp/tmp.O0vXh97Iqx" go get git.curoverse.com/arvados.git/services/crunchstat
rm /tmp/cid
echo "I am stdin" | \
  /tmp/tmp.O0vXh97Iqx/bin/crunchstat -cgroup-root=/sys/fs/cgroup -cgroup-parent=docker -cgroup-cid=/tmp/cid -p
oll=1000 \
  docker run --rm=true --cidfile=/tmp/cid --attach=stdin --attach=stdout --attach=stderr -i ubuntu \
  stdbuf --output=0 --error=0 --input=0 \
  sh -c 'wc >&2; sleep 3; echo ok'; echo DONE
```

#13 - 10/07/2014 08:47 PM - Tom Clegg

- Category set to Crunch

- Assigned To set to Tom Clegg

#14 - 10/08/2014 01:19 PM - Tom Clegg

Still looking for consistent results sending data into docker run on stdin:

```
tom.shell.9tee4:~# docker -v
Docker version 1.1.2, build d84a070

tom.shell.9tee4:~# echo foo | docker run -a stdin -a stdout ubuntu wc
      0       0       0

tom.shell.9tee4:~# echo foo | docker run -i -a stdout ubuntu wc
[hang, ^C ignored, ^P ignored -> docker kill]

tom.shell.9tee4:~# echo foo | docker run -i -a stdout ubuntu wc
[hang, ^C ignored, ^P ignored -> docker kill]

tom.shell.9tee4:~# echo foo | docker run -i -a stdout ubuntu wc
[hang, ^C ignored, ^P ignored -> docker kill]

tom.shell.9tee4:~# echo foo | docker run -a stdin -a stdout ubuntu wc
      0       0       0
```

Aha(?):

```
tom.shell.9tee4:~# echo foo | docker run -i -a stdin -a stdout ubuntu wc
      1       1       4
```

It seems that:

docker run flags	result
-i	hang
-a stdin	stdin is empty in container
-i -a stdin	stdin works

#15 - 10/08/2014 02:39 PM - Peter Amstutz

Looking at 292856a

- I suspect that we're going to get stderr and stdout stepping on each other in the logs, at least until we do #4028 so that the "synchronize lines" logic can happen downstream. I guess we will find out if this is actually a problem or not.
- If there's an Open() error reading any of the cgroup stat files, it uses "continue" to go back to the top of the loop and won't do any of the subsequent stats.

#16 - 10/08/2014 02:55 PM - Tom Clegg

Peter Amstutz wrote:

> Looking at 292856a
>
> - I suspect that we're going to get stderr and stdout stepping on each other in the logs, at least until we do #4028 so that the "synchronize lines" logic can happen downstream. I guess we will find out if this is actually a problem or not.

Indeed, there's a risk that log lines will get mangled by crunch-job when tasks write to stdout. But crunchstat is the wrong place to tackle that issue. (For one thing, even if crunchstat is careful to write whole lines to stderr and stdout, there's still no guarantee that crunch-job won't mangle them.)

> - If there's an Open() error reading any of the cgroup stat files, it uses "continue" to go back to the top of the loop and won't do any of the subsequent stats.

Yes. This probably means the cgroup no longer exists (or something else is terribly wrong) and there's no point writing N error messages per time interval. Added comments @ 5ed4ae4

#17 - 10/08/2014 07:05 PM - Tom Clegg

- Target version changed from 2014-10-08 sprint to 2014-10-29 sprint

#18 - 10/08/2014 07:10 PM - Tom Clegg

- Status changed from In Progress to Feedback

#19 - 10/30/2014 05:11 PM - Tom Clegg

- Status changed from Feedback to Resolved