Arvados - Bug #10586

Python keep client (CollectionWriter) appears to deadlock

11/22/2016 03:33 PM - Joshua Randall

Status:	Resolved	**Start date:**	11/22/2016
Priority:	Normal	**Due date:**	
Assigned To:	Tom Clegg	**% Done:**	100%
Category:	SDKs	**Estimated time:**	0.00 hour
Target version:	2016-12-14 sprint		

Description

Some of the jobs across our cluster (those that are neither stuck due to #10585 nor one of the handful that are still running) appear to be stuck in our python crunch script in one of the calls to arvados.CollectionWriter()

Our crunch script is stuck after printing "...writing output to keep" but before "...validating it", which means it is in one of these three calls: https://github.com/wtsi-hgi/arvados-pipelines/blob/master/crunch_scripts/gatk-haplotypecaller-cram.py#L73-L80

It seems likely that issue 10585 and this one could be due to the same underlying issue, which would be some sort of deadlock in the Python keep client, assuming that arv-mount has some supervisor process that eventually notices things are hung and kills them off (whereas our crunch script doesn't have that).

Subtasks:

Task # 10617: Review 10586-writer-pool-deadlock **Resolved**

Related issues:

Related to Arvados - Bug #10585: crunch doesn't end jobs when their arv-mount... **Resolved** **11/22/2016**

Associated revisions

Revision b738c7e7 - 11/28/2016 07:46 PM - Tom Clegg

Merge branch '10586-writer-pool-deadlock' refs #10586

History

#1 - 11/22/2016 09:27 PM - Tom Morris

- Subject changed from python keep client appears to deadlock to Python keep client (CollectionWriter) appears to deadlock

- Target version set to 2016-12-14 sprint

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

- Assigned To set to Tom Clegg

#3 - 11/25/2016 11:16 PM - Tom Clegg

10586-writer-pool-deadlock

test 86f04235021d84afa0d28d105111422e0dd15738

#4 - 11/28/2016 05:18 PM - Peter Amstutz

I would feel better if this exit path also called self.pending_tries_notification.notify_all() (or maybe wrap the whole method in a try/finally that calls notify_all()) since spurious wakeups are preferrable to spurious deadlocks.

keep.py:529

```
                    if self.pending_copies() < 1:
                        # Drain the queue and then raise Queue.Empty
                        while True:
                            self.get_nowait()
                            self.task_done()
```

This should be if e is not self.TaskFailed ("is" is the object identity operator in Python).

keep.py:598

```
                except Exception as e:
                    if e != self.TaskFailed:
                        _logger.exception("Exception in KeepWriterThread")
```

This method is supposed to return a 2-tuple, but if service.finished() it will return None, which will raise an iteration error when it tries to unpack into locator, copies (maybe you want this to raise TaskFailed?)

keep.py:606

```
        def do_task(self, service, service_root):
            if service.finished():
                return
```

#5 - 11/28/2016 06:46 PM - Tom Clegg

Peter Amstutz wrote:

> I would feel better if this exit path also called self.pending_tries_notification.notify_all() (or maybe wrap the whole method in a try/finally that calls notify_all()) since spurious wakeups are preferrable to spurious deadlocks.
>
> keep.py:529
> [...]

It's impossible for any other thread to be stuck in wait() at this point, because write_success() has already called notify_all() after pending_copies() dropped to zero (or it was always zero, in which case no thread could have even called wait()). But I agree that an extra notify_all() is cheap insurance so I've added that.

> This should be if e is not self.TaskFailed ("is" is the object identity operator in Python).
>
> keep.py:598
> [...]

Fixed.

> This method is supposed to return a 2-tuple, but if service.finished() it will return None, which will raise an iteration error when it tries to unpack into locator, copies (maybe you want this to raise TaskFailed?)
>
> keep.py:606
> [...]

Indeed. Fixed this by addressing it back in get_next_task(), so we don't pile on more "last result was an error" outcomes after a service starts reporting finished() (which includes "success"!).

e4664336c420836bf26f423faf2af9316302da93

#6 - 11/28/2016 07:19 PM - Peter Amstutz

10586-writer-pool-deadlock LGTM @ e4664336c420836bf26f423faf2af9316302da93

#7 - 12/01/2016 04:25 PM - Tom Clegg

- Status changed from New to Feedback

#8 - 12/14/2016 08:11 PM - Tom Clegg

- Status changed from Feedback to Resolved

Can't be certain that the reported deadlock is the one that got fixed, but it seems likely enough, so closing.