Arvados - Bug #6358

[SDKs] Python Keep client uses wrong probe order for put()

06/18/2015 07:10 PM - Peter Amstutz

Status:	Resolved	
Priority:	Normal	
Assigned To:		
Category:	SDKs	
Target version:		

Description

I am running tests that writes a bunch of miscellaneous blocks to Keep. I have three keep services. PUT and GET are supposed to use the same algorithm to generate the sort order used to select which keep servers to store which blocks. This means that if I PUT a set of blocks and there are no errors or changes to the keep service configuration, every GET on that same set of blocks should go to the correct server the first time; there should be no 404s.

However I am seeing 404s on one of the keepstore services in my local install using the Python SDK (it does succeed in finding the block on another keepstore), which means the it is either sometimes putting the block on the wrong keep service, or it is trying the keep services in the wrong order.

Subtasks:

Task # 7601: Review 6358-put-rendezvous	**Resolved**

Related issues:

Has duplicate Arvados - Bug #7573: Keepstore: very uneven distribution of blo...	**Resolved**	**10/15/2015**

Associated revisions

Revision 4f77c778 - 07/08/2015 09:19 PM - Brett Smith

6358: Declare FUSE driver's dependency on llfuse >= 0.40.

The workaround added in 08284382 requires this version. Below that, llfuse's Queue at least lives in a different place. We may be able to support older versions with more nuance, but for now, just codify the current reality.

Closes #6358. Refs #3198.

Revision d379c467 - 10/19/2015 07:29 PM - Tom Clegg

Merge branch '6358-put-rendezvous' closes #6358

Revision 38625a4a - 10/20/2015 09:54 PM - Tom Clegg

6358: Fix race opportunity in ThreadLimiter.

refs #6358

Revision b23dbeaa - 10/20/2015 09:54 PM - Tom Clegg

6358: Fix probe order test logic.

This request order is OK with two threads: thread "0" just took a long time to make its request.

expect 0 1 2 3 4 5 6 7
got 1 2 3 4 5 0 6 7

The inverse is not OK. This would mean 0 started before any of 1,2,3,4,5 finished.

expect 1 2 3 4 5 0 6 7
got 0 1 2 3 4 5 6 7

refs #6358

History

#1 - 06/18/2015 07:10 PM - Peter Amstutz

- Subject changed from [SDK] Not always trying the correct server first to [SDK] Not always trying the correct keep server first

#2 - 06/18/2015 07:19 PM - Peter Amstutz

- Description updated

#3 - 06/18/2015 07:37 PM - Peter Amstutz

On further investigation, I don't think weighted_service_roots is at fault, I think there might be a race condition:

One possibility is that it starts writing to server A and server B, server A is a little slow to respond but server B accepts the block, so it goes on and writes to server C while still waiting for server A. However, if that were happening, the write to server A could still go through resulting in 3 replicas, not 2.

It could also be that it is actually the thread for writing to server A that is slow to start, so the threads writing to server B and server C complete before the thread for writing to server A get going?

#4 - 06/26/2015 10:51 PM - Tom Clegg

Running arv-put and arv-get with ARVADOS_DEBUG=2 should generate enough log messages to see whether the servers are attempted in the right order, and which of arv-put, arv-get, and keepstore are doing something wrong (or slow).

When probe order is A,B,C and A is slow to respond, the correct behavior is to store on B and C as quickly as possible and move on. So it's possible this is not a client bug.

Does this happen with larger blocks, or only with small blocks? For small blocks, lookup+connection-setup time can be comparable to transfer+touch/store time, making out-of-order races more likely.

#5 - 07/08/2015 09:25 PM - Brett Smith

- Status changed from New to Resolved

- % Done changed from 0 to 100

Applied in changeset arvados|commit:4f77c7788c6fc3a6cc9cd90ff231d837fdec7cc4.

#6 - 07/22/2015 06:49 PM - Brett Smith

- Status changed from Resolved to New

#7 - 08/11/2015 02:23 PM - Brett Smith

- Project changed from 35 to Arvados

- Subject changed from [SDK] Not always trying the correct keep server first to [SDKs] Put clients giving up on Keep servers quickly enough to thwart the probe list

- Category set to SDKs

- Target version changed from Bug Triage to Arvados Future Sprints

Updating the subject to reflect what sounds like the more likely issue here. I'm doing that understanding, like Tom said, this might not be a client bug at all. The clients might be doing the right thing, and the fix to the bad block distribution might be "Write Data Manager."

#8 - 10/16/2015 09:14 PM - Tom Clegg

- Subject changed from [SDKs] Put clients giving up on Keep servers quickly enough to thwart the probe list to [SDKs] Python Keep client uses wrong probe order for put()

Reverting subject after confirming there is a real probe order bug (by writing a failing test and a fix).

(As it turns out, we were testing that the "last error from each server" list was being *reported* in the correct probe order, but we weren't testing that that we were *encountering* the errors in the correct probe order.)

#9 - 10/16/2015 11:52 PM - Tom Clegg

6358-put-rendezvous

- Fix bug, test exact ordering for single-thread case: 25af384
- Test partial ordering for multiple-thread case: f269149

#10 - 10/16/2015 11:53 PM - Tom Clegg

- Status changed from New to In Progress

#11 - 10/19/2015 07:15 PM - Peter Amstutz

6358-put-rendezvou:

As a matter of opinion, this seems like a missed opportunity to redesign multi threaded writes in keep client, e.g. a work queue design could would be simpler and avoid the need for racing writer threads to coordinate their execution order in the first place.

However, as a fix for the immediate problem at hand, it looks good to me.

#12 - 10/19/2015 07:29 PM - Tom Clegg

Peter Amstutz wrote:

> As a matter of opinion, this seems like a missed opportunity to redesign multi threaded writes in keep client, e.g. a work queue design could would be simpler and avoid the need for racing writer threads to coordinate their execution order in the first place.
>
> However, as a fix for the immediate problem at hand, it looks good to me.

I agree there are refactoring opportunities here, but yes, a bugfix seemed more important. Hopefully these new tests will make it easier/safer to refactor when we get to it.

#13 - 10/19/2015 07:35 PM - Tom Clegg

- Status changed from In Progress to Resolved

- % Done changed from 0 to 100

Applied in changeset arvados|commit:d379c467be58c66b2f1e7acafc97634b269a1542.

#14 - 11/30/2015 04:57 PM - Tom Clegg

- Target version deleted (Arvados Future Sprints)