Arvados - Story #9446

[SDK] Refactor keep parallel write strategy

06/20/2016 07:40 PM - Peter Amstutz

Status:	Resolved	**Start date:**	06/27/2016
Priority:	Normal	**Due date:**	
Assigned To:	Lucas Di Pentima	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	2016-07-06 sprint		

Description

Currently arvados.keep.KeepClient.put creates one KeepWriterThread per keep server and then uses a complex (and error prone) locking strategy implemented in ThreadLimiter to ensure that only certain threads perform uploads in a certain order.

Refactor this code to use the following alternate strategy:

- For N wanted copies create N upload threads
- Start a new upload to the next server in sorted_roots when an upload fails
- The server response may include a header x-keep-replicas-stored, if so the client's replica count should be incremented by the number in the header
- If the service lists consists only of "disk" type services, start up N parallel write threads for each copy; if the service list has non-"disk" services (such as "proxy") perform one upload at a time and look for the x-keep-replicas-stored header.
- Don't write to servers that are read only (this should already be handled by map_new_services(need_writable=True))

You may want to use a Queue (instead of explicit locks) to communicate between the main thread and the upload threads.

Consider setting up a thread pool attached to the keep client object and dispatching work instead of spawning new threads.

Two things to keep in mind:

- The order that it tries to upload to each server matters (because the earlier servers are preferred over the later ones)
- If it goes through the entire list without uploading sufficient replicas, it should try again. However, when it does so, it should:
    1. remember how many replicas have already been uploaded (so if it wanted 3 and got 2 on the first pass, it only needs 1 more)
    2. not try to upload to services to which it did a successful upload or did not get a retryable failure code in the previous round. Retryable failure codes (from the Go SDK) are 408 (Request Timeout), 429 (Too Many Requests), and any 5xx error except 503 (which means "Server Full").

Subtasks:

Task # 9469: Review 9446-refactor-keep-parallel-write-strategy	**Resolved**

Related issues:

Related to Arvados - Story #9180: [PySDK] Avoid overreplication in KeepClient	**Resolved**	**06/08/2016**

Associated revisions

Revision 0ea1f67f - 06/27/2016 06:36 PM - Lucas Di Pentima

9446: Applying Peter's review suggestions. refs #9446

Revision 1b44d67f - 06/27/2016 08:03 PM - Lucas Di Pentima

Merge branch '9446-refactor-keep-parallel-write-strategy'

9446: PySDK Keep concurrent write refactoring. Simplified thread management using a thread pool and a queue.
ThreadLimiter not needed anymore, so related tests were rewritten for the new code.
Closes #9446

History

#1 - 06/20/2016 07:52 PM - Peter Amstutz

- Description updated

#2 - 06/20/2016 07:59 PM - Peter Amstutz

- Assigned To set to Lucas Di Pentima

#3 - 06/20/2016 08:07 PM - Peter Amstutz

- Description updated

#4 - 06/20/2016 08:22 PM - Peter Amstutz

- Description updated

#5 - 06/20/2016 08:28 PM - Lucas Di Pentima

- Status changed from New to In Progress

#6 - 06/21/2016 01:13 PM - Peter Amstutz

- Description updated

- Status changed from In Progress to New

#7 - 06/21/2016 01:25 PM - Peter Amstutz

- Description updated

#8 - 06/21/2016 01:30 PM - Peter Amstutz

- Description updated

#9 - 06/22/2016 07:30 PM - Radhika Chippada

- Target version set to 2016-07-06 sprint

- Story points set to 1.0

#10 - 06/24/2016 02:43 PM - Lucas Di Pentima

- Status changed from New to In Progress

#11 - 06/27/2016 02:40 PM - Peter Amstutz

KeepWriterQueue.pending_copies() should take successful_copies_lock.

```
        def pending_copies(self):
            with self.successful_copies_lock:
                return self.wanted_copies - self.successful_copies
```

This is confusing:

```
if not self.queue.retries > 0:
```

It would be clearer as:

```
if self.queue.retries <= 0:
```

Don't call self.queue.task_done() unless a task was actually removed from the queue. If there is a race and Queue.Empty exception is thrown, the task could get double-counted.

Add task_done to the bottom of the if self.queue.pending_copies() > 0: block and inside the exception handler of the else branch:

```
                if self.queue.pending_copies() > 0:
                    ....
                    # Mark as done so the queue can be join()ed
                    self.queue.task_done()
                else:
                    # Remove the task from the queue anyways
                    try:
                        self.queue.get_nowait()
                        # Mark as done so the queue can be join()ed
                        self.queue.task_done()
                    except Queue.Empty:
                        continue
```

Using "retry" for "try the next service in the queue" is a little confusing, because the term "retry" is used a lot in the Arvados SDK to mean "try the same service again". Considering using a different term, maybe "pending_tries" or "pending_attempts".

The rest of it looks good! Thanks!

#12 - 06/27/2016 06:38 PM - Lucas Di Pentima

0ea1f67f70f942c4732f2269e31c3ddb7d63fc9e

Applied Peter's suggestions, thanks!
All local tests run OK.

#13 - 06/27/2016 07:45 PM - Peter Amstutz

LGTM, please merge

#14 - 06/27/2016 08:06 PM - Lucas Di Pentima

- Status changed from In Progress to Resolved