Arvados - Feature #10541

[Keep] Share buffers between overlapping/consecutive GET requests for the same block

11/15/2016 09:59 PM - Tom Clegg

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	Arvados Future Sprints		

Description

Background

Certain common scenarios -- for example, starting up a job using cause many clients to retrieve the same data block at around the same time.

Proposed improvement

Extend the memory buffer allocation system to act as a cache of recently fetched data.

- When performing a GET operation, if another goroutine is already processing a GET request for the same block, wait for it to finish finding/retrieving the block data, and then respond with that data instead of re-fetching.
- When returning a buffer to the pool, attach the relevant block hash so the data can be reused by a future GET request.
- When getting a buffer from the pool to service a GET request, first try to get a buffer that already has data for the requested block. If that is available, use the cached data instead of reading it from a volume.
- When getting a buffer for a PUT request, or a GET request for un-cached data, allocate a new one or (if max buffers are already allocated) use the least recently used buffer.
- As before, if all buffers are already in use, wait until one is available or (after reaching max clients) return 503.

Implementation ideas

Instead of a sync.Pool of byte slices, add a buffer type (containing a []byte, a reference counter, and whatever else is needed for synchronization) and use a []*buffer as an LRU list. Maintain a map[string]*buffer mapping hashes to buffers (some buffers won't have a corresponding hash, e.g., after errors).

Add a bufferPool type to manage buffer sharing and repurposing.

- All overlapping GET requests for a given hash should share the same buffer.
- If requests are sharing a buffer and the data isn't already present, one handler attempts to fetch the data while the others wait. If an error occurs, all of the waiting requests return an error.

Related issues:

Related to Arvados - Feature #2960: Keep can stream GET and PUT requests	**New**

History

#1 - 11/16/2016 02:28 AM - Ward Vandewege

- Description updated

#2 - 11/16/2016 04:40 PM - Peter Amstutz

Recommend refactoring keepstore to use arvados/sdk/go/streamer to disconnect data fetch from providing data to clients.

This implements a buffered Reader. It accepts a source Reader which pulls data into the buffer via a goroutine. It also allows for the creation of any number of async readers that read from the buffer. Async readers start from the beginning of the buffer and read up to the "frontier", then block on further incoming data from the source. On EOF from source, the buffer is terminated, and readers get EOF. However, new readers can still be created (they will read entirely from the buffer.)

This has the added advantage that multiple requests for the same block can be served out of the same buffer even if the block hasn't finished fetching.

The async streaming reader was written for KeepProxy and is used by the Go SDK for block upload, for the explicit use case of efficiently copying a data stream to multiple destinations.

#3 - 11/16/2016 04:45 PM - Tom Clegg

- Description updated

#4 - 11/16/2016 05:29 PM - Tom Clegg

Peter Amstutz wrote:

> Recommend refactoring keepstore to use arvados/sdk/go/streamer to disconnect data fetch from providing data to clients.

Refactoring the volume interface to use readers/writers instead of buffers would give us the option of streaming data to clients without store-and-forward, which would surely be better in some setups. But that refactoring is a significant task in its own right and it doesn't seem to make the caching implementation noticeably easier/different (most of the code here will be about tracking hashes and timestamps and knowing when to reuse an existing buffer/stream and when to create a new one), so I think we should consider the streaming feature separately (#2960?).

> This has the added advantage that multiple requests for the same block can be served out of the same buffer even if the block hasn't finished fetching.

Sharing buffers between overlapping requests should be easy enough with the current store-and-forward code: one goroutine reads the data, then they all wake up and copy data from the shared buffer to their network clients.

#5 - 08/29/2017 02:18 PM - Tom Morris

- Target version set to Arvados Future Sprints