Arvados - Feature #8228

[SDKs] [FUSE] Python SDK and arv-mount use Range requests when a caller requests part of a block that has been ejected from the cache

01/19/2016 06:23 PM - Tom Clegg

Status:	New	**Start date:**	01/19/2016
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	0%
Category:	SDKs	**Estimated time:**	0.00 hour
Target version:	Arvados Future Sprints		

Description

Background

In crunch jobs (and elsewhere) it's easy to demand more random access than arv-mount's cache can handle (e.g., by reading from 100 files in order to do a merge sort). Currently, the only way to deal with this is to increase arv-mount's block cache: even without readahead/prefetch, this means 6.4GiB of RAM is needed to support a 100-way merge. If your cache is smaller, arv-mount can easily end up reading each 64 MiB block hundreds of times as it enters and leaves the cache. To make matters worse, in order to detect that arv-mount's cache is thrashing, you need to pay attention to crunchstat logs.

This story aims to reduce cache requirements such that a reasonable data cache size (like the default 256 MiB) is enough to support a reasonable number of concurrent readers (like 512).

There are two basic changes:

1. Make it possible to eject *most* of a 64 MiB block from the cache without ejecting *all* of it. If each of 10 processes is reading 64 KiB at a time, a 256 MiB cache is more effective if it holds 500 x 512 Kib chunks rather than 4 x 64 MiB chunks.
2. Make it possible to retrieve *part* of a 64 MiB block from Keepstore without retrieving *all* of it. In a situation where we've already ejected a 64 MiB block from the cache to reclaim space, if we need part of that block again, ejecting another 64 MiB block from the cache to make room for this one again is just making the problem worse. And if we're not going to cache it, we shouldn't bother transferring it over the network or asking Keepstore to read it from disk. Instead, we should request a portion small enough to put in our cache.

We want to get these advantages without sacrificing our use of the MD5 hash to check data integrity. (In fact, it is *more* important to check data integrity when doing partial reads: Keepstore can't do it for us if it's not reading the whole block from disk.)

Implementation

In the Python SDK, KeepClient.get() should

- have a CACHE_CHUNK_SIZE const = 524288
- accept a range argument (e.g., range=[1234,11234]), and return only the specified byte range.
- instead of inserting an entire block in the cache, chunk it into CACHE_CHUNK_SIZE-byte pieces and insert each chunk into the cache (the cache key will now be a tuple (md5, byte_offset))
- maintain a separate cache of chunk hashes: (md5, byte_offset) → chunk_md5 (capped at data_cache_cap_bytes÷32768 entries, which is approximately 1/1024 of the size of the data cache and will accept 16x)

If asked for a range that can't be satisfied by getting all of the necessary chunks from the data cache, but the MD5 of each necessary chunk *is* cached:

- do a range request that returns all of the required pieces (i.e., the Range header boundaries sent to Keepstore will be a multiple of CACHE_CHUNK_SIZE, even if the boundaries in the range argument to get() aren't)
- if the Keepstore service ignores the Range header and returns the entire block, discard the excess parts. (Sure, it *might* be better to cache them just in case they're needed, but that might also cause more cache churn and wreck performance, and it's a temporary measure to accommodate older services anyway.)

If asked for a range that isn't fully cached, and the MD5 of one or more relevant chunks is not known,

- fetch the whole block

In both cases:

- Insert/freshen all of the returned chunks in the data cache
- Insert/freshen all of the chunk MD5s in the chunk-hash cache

KeepClient.get_from_cache() should also accept a range argument, and look up the appropriate entries in the data cache to put together a response.

ArvadosFile.readfrom should pass a range argument to get_block_contents, and get_block_contents should pass that range argument along to self._keep.get() and self._keep.get_from_cache().

ArvadosFile._BlockManager._block_prefetch_worker should call self._keep.get(b, range=(0,1)) instead of self._keep.get(b) to avoid evicting a lot of useful data from the cache unless that's necessary to fill the chunk-MD5 cache.

Related issues:

Related to Arvados - Feature #3734: [Keep] Keepstore and keepproxy support HT...	**New**	**08/27/2014**
Related to Arvados - Feature #6310: [FUSE] Support scaling the internal block...	**New**	
Related to Arvados - Story #3640: [SDKs] Add runtime option to SDKs (esp Pyth...	**New**	

History

#1 - 01/19/2016 08:27 PM - Tom Clegg

- Description updated

- Category set to SDKs

#2 - 01/20/2016 06:14 PM - Brett Smith

- Target version set to Arvados Future Sprints