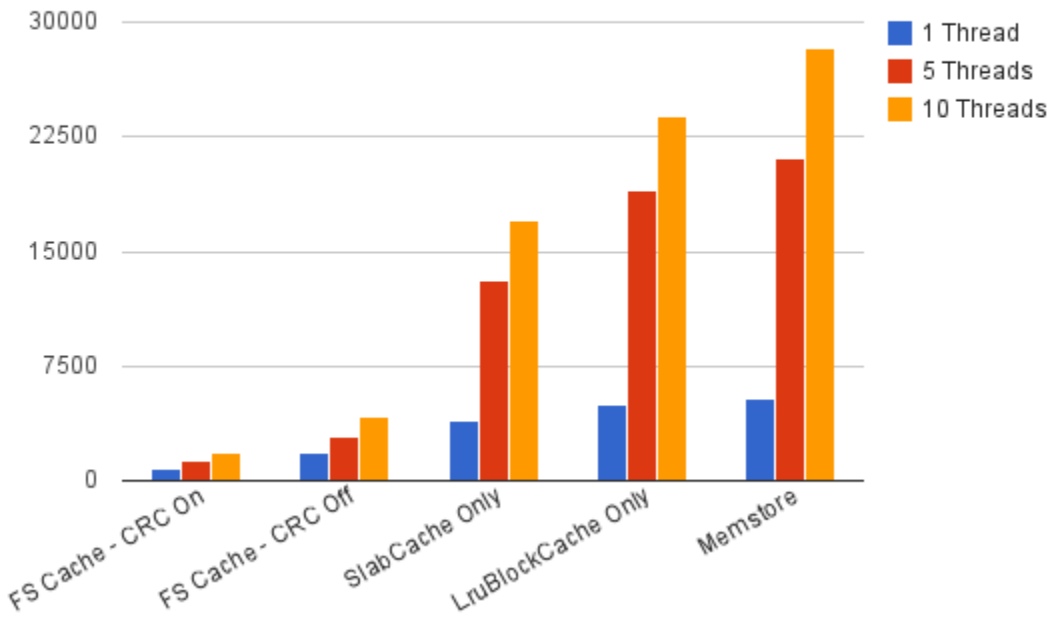
Goal: Allow for a large off heap cache that won't contribute to garbage collection pauses.

Current Situation: On a machine with substantial amounts of RAM, we limit the HeapSize, and hope the FS cache takes up the slack.

Problem: Using the fs cache is slow, even without the overhead of going over to the DataNode. Also, it only works on things which are on stored on the local disk, so pre-compaction, the FS cache won't help you very much.

Solution: Use DirectByteBuffers and Slab Allocation to create an off heap cache. We copy onto the heap if we need to retrieve a record. While there is a performance hit, as the benchmarks below show, this is far preferable to the performance hit from reading from the filesystem cache.

The tests below were conducted in standalone mode, with compression turned off and with a small enough dataset coupled with enough extraneous ram to guarantee that the files being read were in the filesystem cache. CRC was turned off to improve the speed of this. Despite all this, the filesystem cache, is at it's best, half the speed of the slab allocated cache.



If CRC is turned on, the result is far worse. If reading through a datanode, the performance should be substantially worse. The on heap cache, of course, produced the best performance, but the SlabCache keeps up very well. The memstore is included for comparison.

As the block cache's go, the LruBlockCache is close to optimal, although I have not fully measured this, I replaced the LruBlockCache with a simple ConcurrentHashMap - there was

very minimal improvement in speed.

The SlabCache avoids fragmentation via utilizing a model similar to memcached. It trades space efficiency for garbage collection overhead. We split a large directedbytebuffer or bytebuffers into smaller pieces. When we need to cache a block we choose the smallest slab that can enclose the block, and copy the block into the slab. Each slab maintains its own metrics, and is backed by a LinkedConcurrentHashMap, which is also responsible for eviction within that slab.

Currently, two slabs are allocated around the default sizes, 1.1x the default block size, and 2.1x the default block size. 80% of the off heap space is given to the former, and the remainder is to the latter. This will be made configurable, and the metrics should help you determine optimal slab sizing.