BlockCacheReport: LruBlockCache vs CombinedBlockCache (offheap)

Report made for HBASE-11323 BucketCache all the time!

This report is like another attached to HBASE-11323 only it varies cell sizes rather than have them constant.

25 clients random reading over 10M cells of zipfian varied size from 1 to 256k bytes over 21 regions hosted by a single RegionServer.

Heaps are intentionally small to bring on any distress the sooner. The java heap was 8G and the block cache “4G”. For LruBlockCache (LruBC), it was allotted 4G. For CombinedBlockCache (CBC), we sized it at 4G offheap but CBC runs with a LruBC as L1 to host META blocks. DATA blocks are kept offheap in an L2 BucketCache. On review, only tens of blocks of some small MBs are up in L1 when testing offheap CBC.

Ran four tests: no cache misses, a few cache misses, a bunch of cache misses, then loads of cache misses. For the 2nd and 3rd cases, the filesystem cache starts to cover soon after test start so tends toward no disk seeks. For the last test, lots of disk seeking.

1 Findings

Offheap CBC GC profile is better for all but the no-cache-miss case. LruBC reads at almost twice the throughput of CBC when all out of cache with a better GC profile than CBC. Otherwise, throughput and latency are about the same for the two cache deploy types.

Use offheap CBC if your dataset does not fit in cache.

2 Notes

CBC has some not insignificant overhead. There will be slop left over because buckets will not fill to the brim especially when block sizes vary.

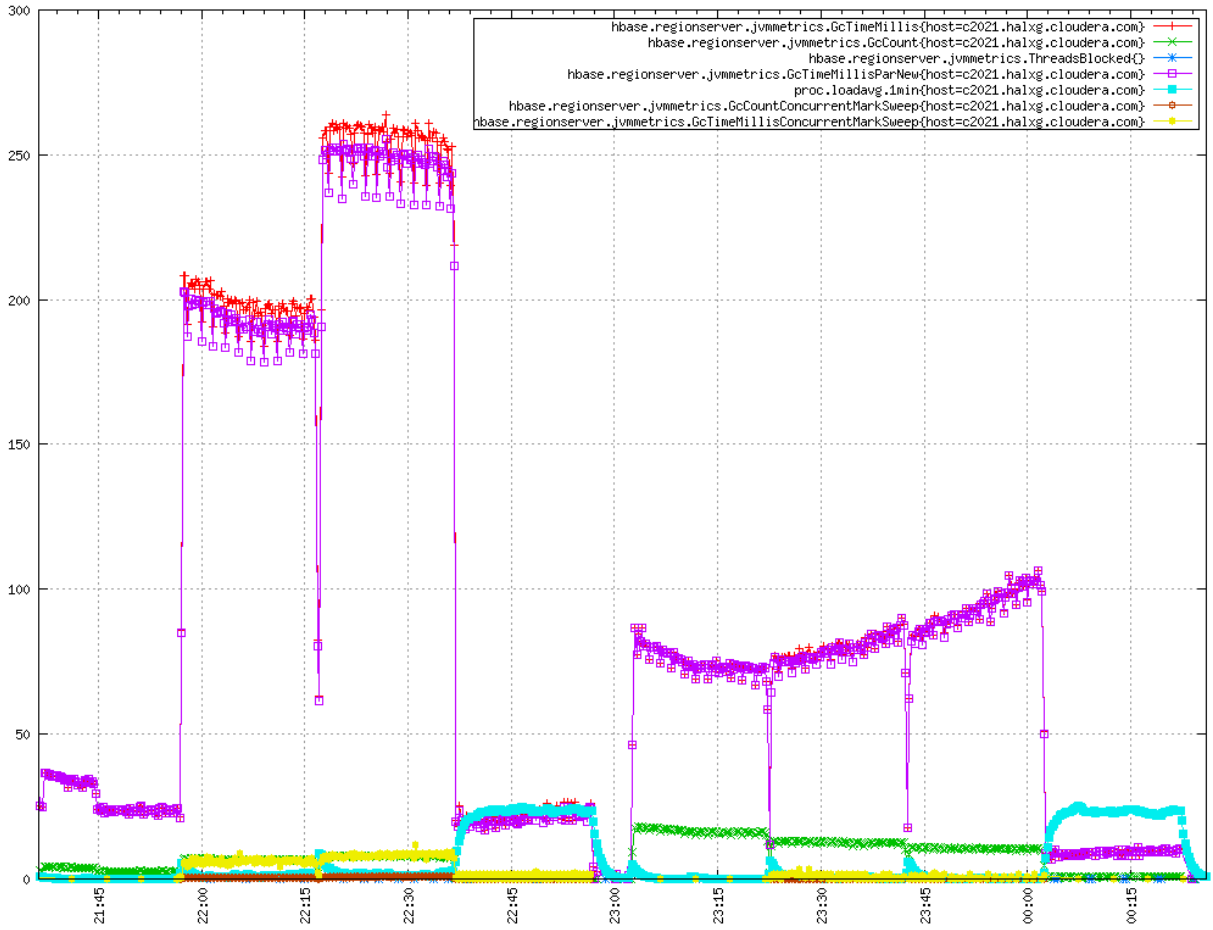
3 Graphs

All graphs correlate on their x-axis. See below start times for each test (size=32 is all-in-cache up to size=1000 for lots of cache misses):

```
Tue Jul 22 21:36:39 PDT 2014 run size=32, clients=25 ; lrubc time=1200 size=32
Tue Jul 22 21:56:39 PDT 2014 run size=72, clients=25 ; lrubc time=1200 size=72
Tue Jul 22 22:16:40 PDT 2014 run size=144, clients=25 ; lrubc time=1200 size=144
Tue Jul 22 22:36:40 PDT 2014 run size=1000, clients=25 ; lrubc time=1200 size=1000
Tue Jul 22 23:02:16 PDT 2014 run size=32, clients=25 ; bucket time=1200 size=32
Tue Jul 22 23:22:16 PDT 2014 run size=72, clients=25 ; bucket time=1200 size=72
Tue Jul 22 23:42:17 PDT 2014 run size=144, clients=25 ; bucket time=1200 size=144
Wed Jul 23 00:02:17 PDT 2014 run size=1000, clients=25 ; bucket time=1200 size=1000
```

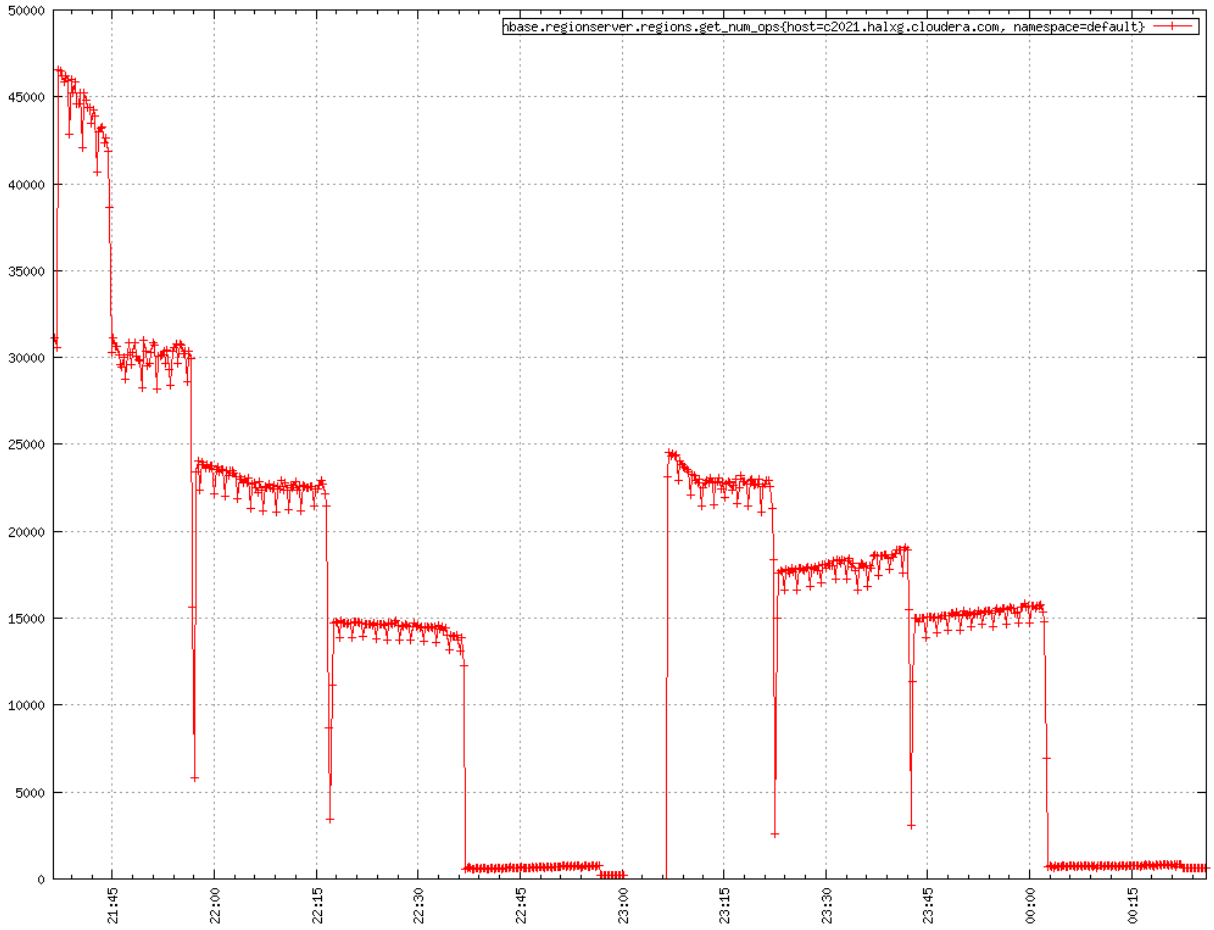
3.1 GC

Size up the document to get a better view on the graph. The CBC tests started at 23:02. Comparing the no-cache-misses run (size=32), we see ParNew GC times of ~25ms for LruBC vs about 75ms for CBC. For all other cases, CBC does better taking 1/3rd of the GC time.



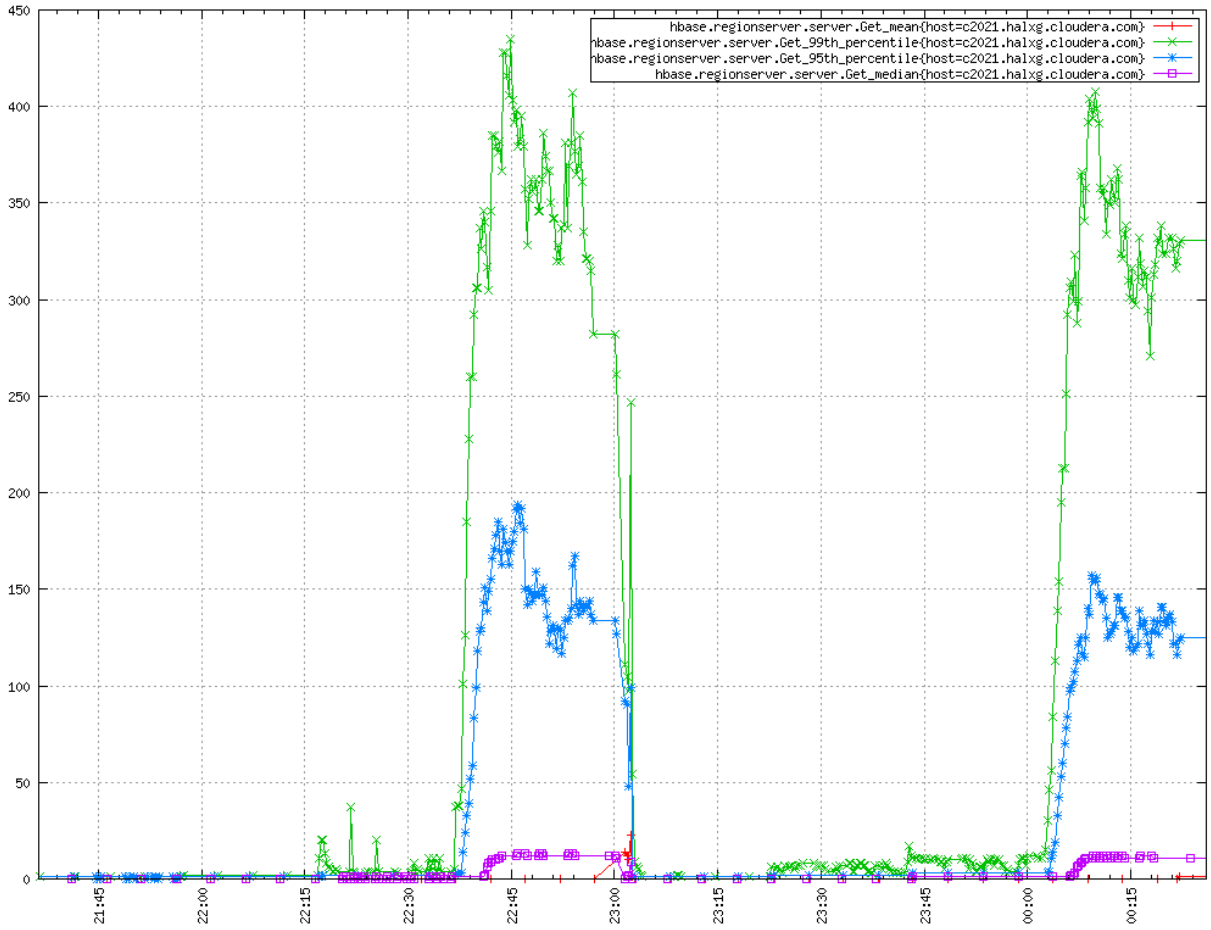
3.2 Throughput

We do more random reads per second out of LruBC than we do out of the offheap CBC, almost twice the reads (I missed why the step in the LruBC all-in-cache test). Thereafter, read rates are about the same for the two cache deploy types.



3.3 Latency

About the same for both deploy types. CBC is a bit worse the less cache misses involved.



3.4 CPU

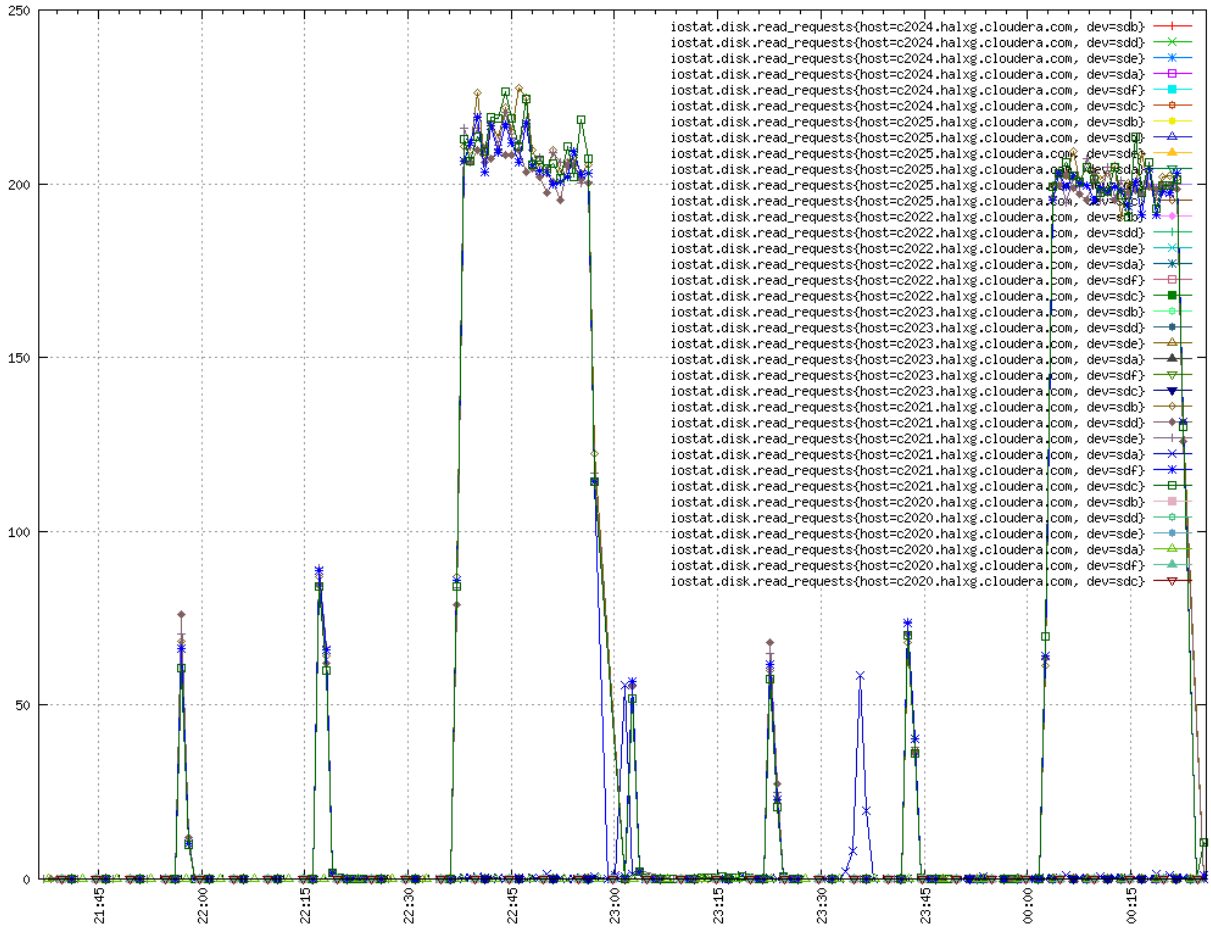
CBC does more work when we are mostly in-cache.

3.5 BlockCache Stats and I/O

These two graphs just show that the loading was about the same for both deploy types: the block eviction rates are about the same for both (looks worse for CBC because it aggregates numbers over L1 and L2) and the seeks are about same for both tests; you can see the fs cache move under the seeks for the second and third runs for each deploy type.

3.5.1 BlockCache Stats

3.5.2 Seeks



4 Setup

Master branch. 2.0.0-SNAPSHOT, r69039f8620f51444d9c62cfca9922baffe093610. Hadoop 2.4.1-SNAPSHOT.

5 nodes all the same w/ 48G and six disks. One master and one regionserver, each on distinct nodes, with 21 regions of 10M rows of zipf varied size -- 0 to 256k -- created as follows:

```
$ export HADOOP_CLASSPATH=/home/stack/conf_hbase:`./hbase-2.0.0-SNAPSHOT/bin/hbase
classpath`
$ nohup  ./hadoop-2.4.1-SNAPSHOT/bin/hadoop --config /home/stack/conf_hadoop
org.apache.hadoop.hbase.PerformanceEvaluation --valueSize=262144 --valueZipf
--rows=100000 sequentialWrite 100 &
```

Here is my loading script. Does 4 loadings. All in cache, just out of cache, a good bit out of cache and then mostly out of cache:

```
[stack@c2020 ~]$ more bin/bc_in_mem.sh
```

```
#!/bin/sh
HOME=/home/stack
testtype=$1
date=`date -u +"%Y-%m-%dT%H:%M:%SZ"
echo testtype=$testtype $date` >> nohup.out
HBASE_HOME=$HOME/hbase-2.0.0-SNAPSHOT
runtime=1200
clients=25
cycles=1000000
#for i in 38 76 304 1000; do
for i in 32 72 144 1000; do
  echo "`date` run size=${i}, clients=$clients ; $testtype time=$runtime size=$i"
>> nohup.out
  timeout $runtime nohup ${HBASE_HOME}/bin/hbase --config /home/stack/conf_hbase
org.apache.hadoop.hbase.PerformanceEvaluation --nomapred --valueSize=110000
--size=$i --cycles=$cycles randomRead $clients
done
#mv $HOME/nohup.out $HOME/nohup.$testtype.$date
```