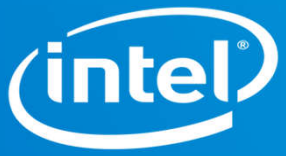
HBase Off heaping

Anoop Sam John (Intel)
Ramkrishna S Vasudevan (Intel)

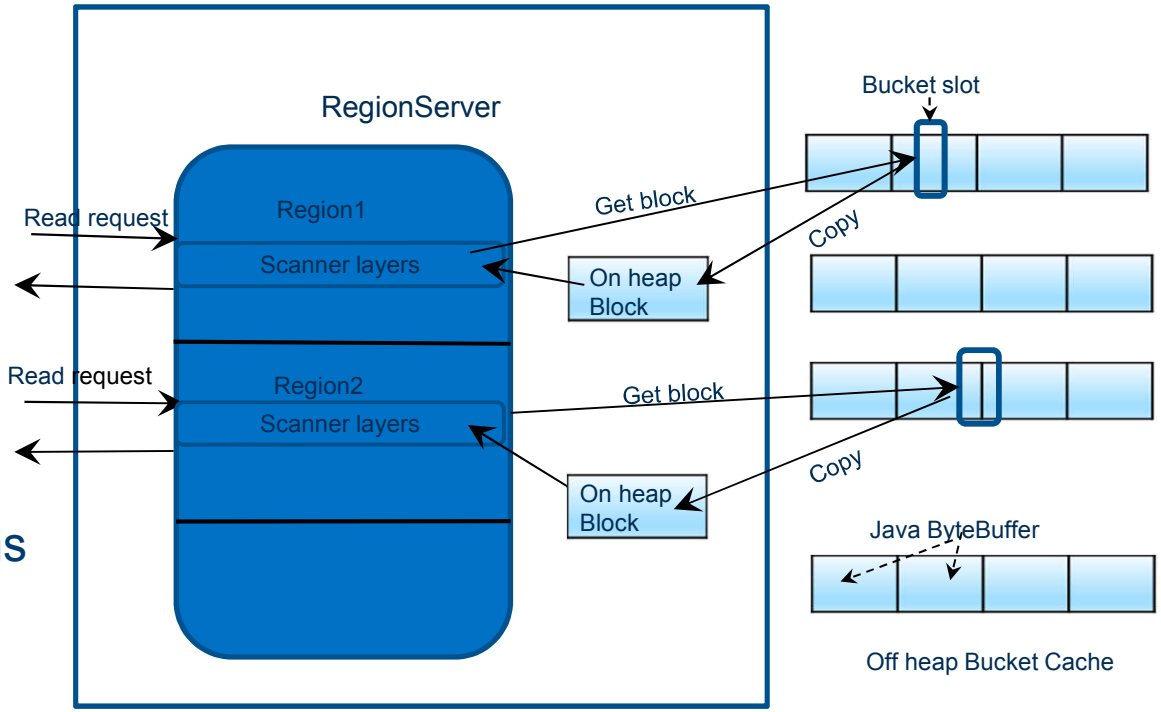


HBase Cache

- HBase Cache for better read performance
  - Keep data local to the process
- *L1* on heap LRU Cache
  - Larger Cache Sizes => Larger Java Heap => GC issues
- *L2* Bucket Cache
  - LRU
  - Backed by off heap memory / File
  - Can be larger than L1 allocation
  - Not constrained by Java Heap Size

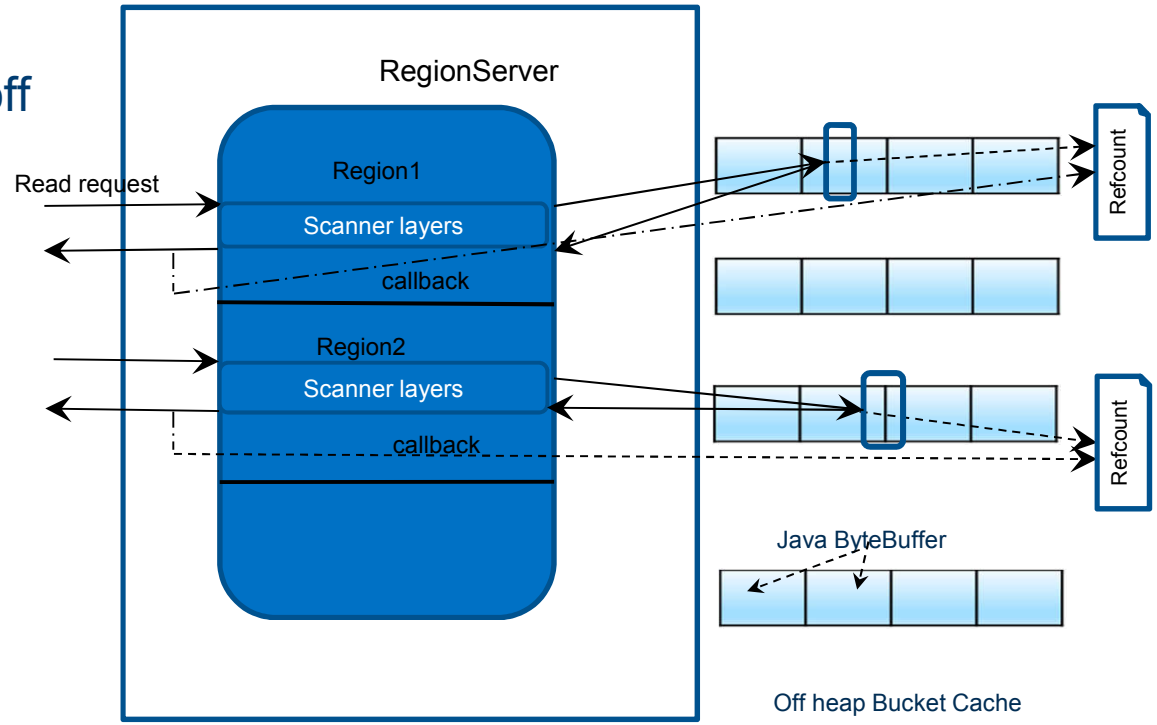
Bucket Cache - Reads

- Read block by block
- On heap block copy
- Less throughput *vs* L1
- GC Impacts
- Predictable latency problems

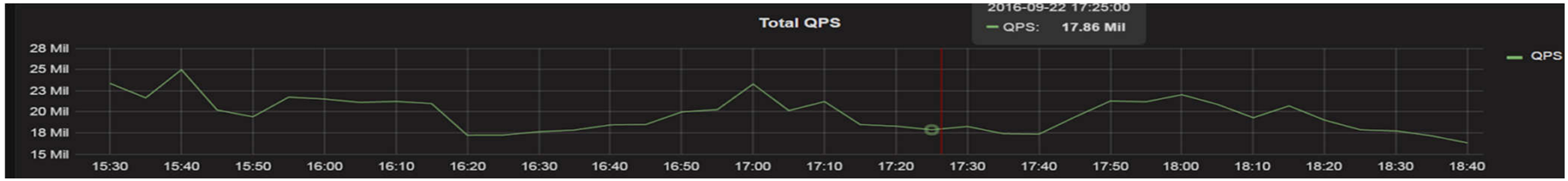


Bucket Cache – Optimized Off heap Reads

- Delivers data directly from off heap cache
- Less Heap needed
- Better throughput
- Predictable latency
- Jira # HBASE-11425
- Available in HBase 2.0



Alibaba : Old Performance Data With L2 Cache



From A/B test cluster with 400+ nodes and on simulated data

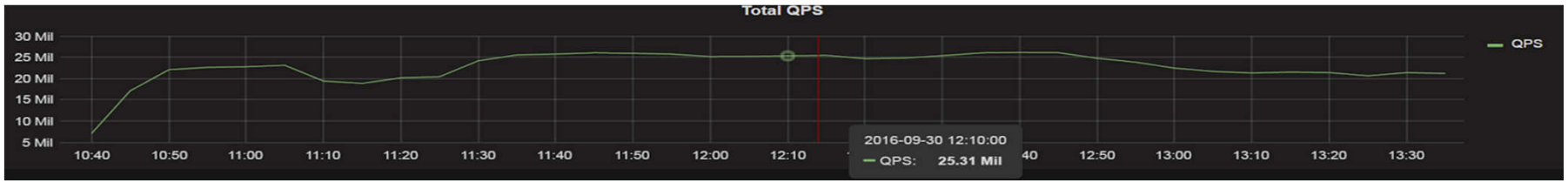
12 GB off heap L2 cache

Inconsistent throughput. Dips in latency

HBase read from L2 does data copy to on heap

- More garbage and GC workload

Performance Data After Off heaping read path



Backported patches to hbase-1.2 based codebase

Average throughput improved from 18 to 25 million QPS – 30% improvement

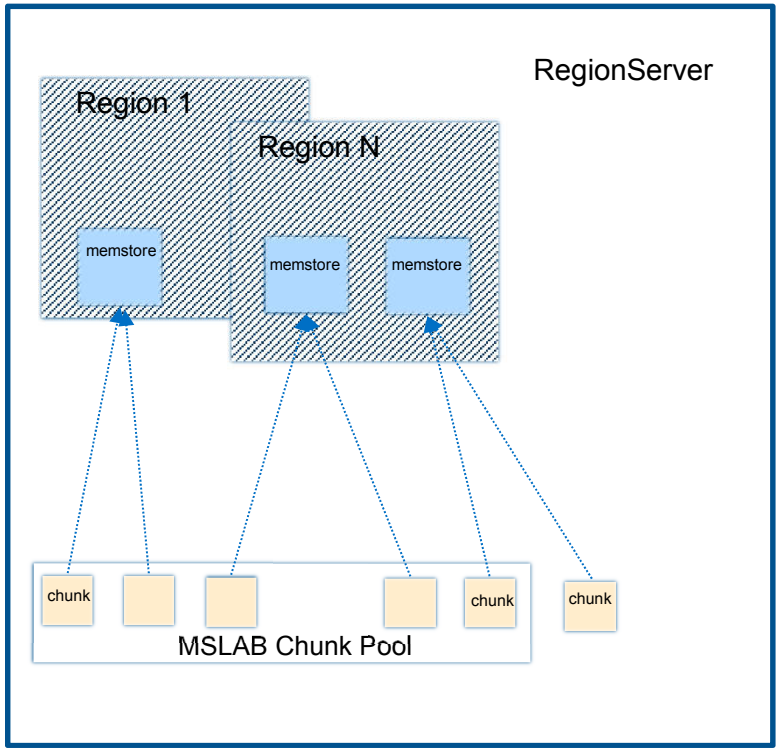
Linear throughput. Predictable latency.

In production since Oct 2016

Used in 2016 Singles' Day, with peaks at 100K QPS per single RegionServer

HBase Writes

- Accumulate cells in memstore
  - Default flushes at size >= 128 MB
- Global memstore heap size
  - Defaults to 40% of Xmx
- MSLAB Chunks & Pool
  - On heap Chunks of 2 MB
  - Cells in memstore have data copied over to chunks
  - Reduce fragmentation
  - Pool to Reuse chunks



Write path Offheaping

HBASE-15179 In HBase 2.0

Reading request data into reusable off heap ByteBuffers

- ByteBuffer pool. Response Cell blocks already use this
- Using the same off heap ByteBuffers – Each of 64 KB size
- Reduce garbage and GC
- Usage of off heap avoid temp off heap copy in NIO.

Protobuf upgrade

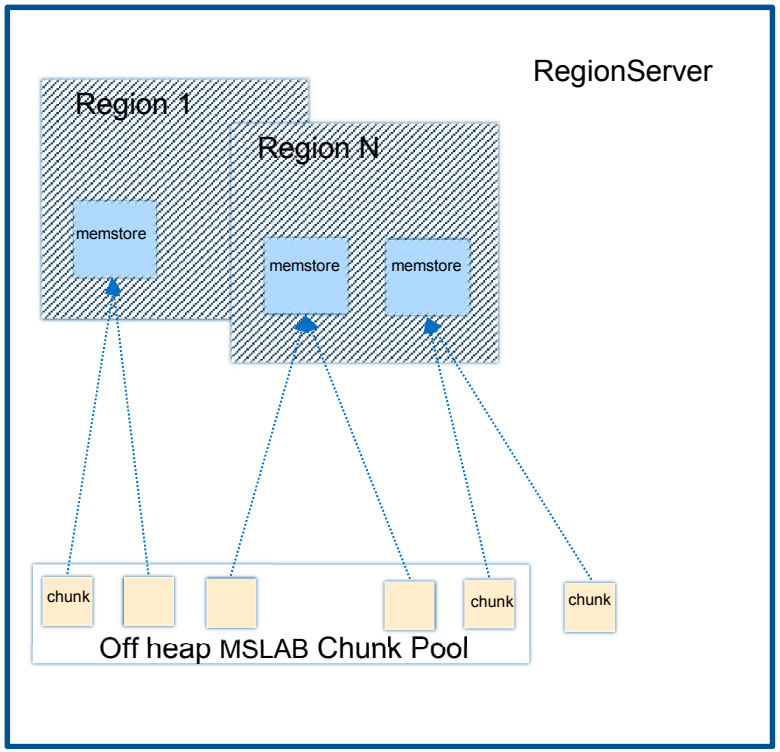
- PB 3.x latest version support off heap ByteBuffers throughout
- Shaded PB jars to avoid possible version conflict
- New PB ByteStream to work over 1 or more Off heap ByteBuffers

Write path Offheaping

HBASE-15179

Off heap MSLAB Chunk Pool

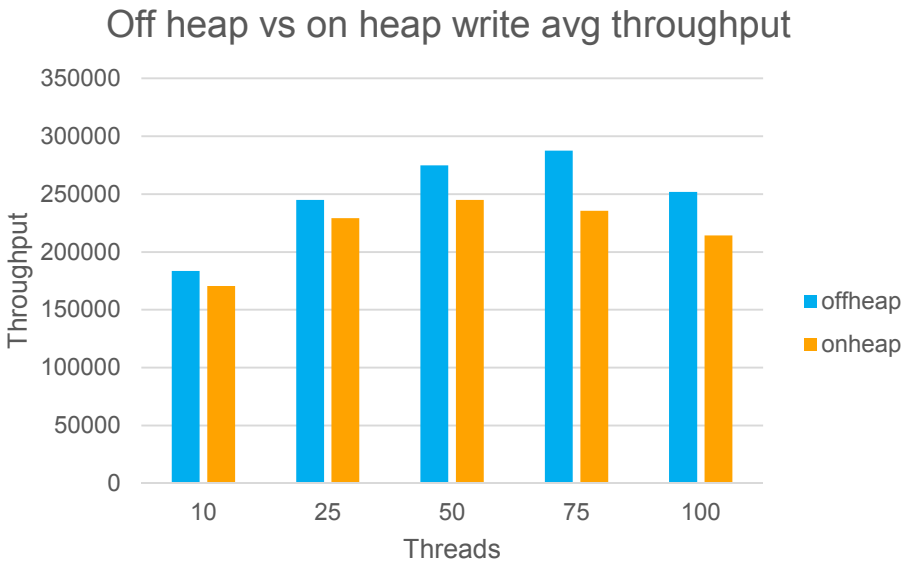
- 2 MB off heap chunks
- Global off heap memstore size – New config *hbase.regionserver.offheap.global.memstore .size*
- Separation of Cell data size, heap overhead size tracking
- Region flush decision on cell size alone (128 MB default flush decision)
- Global level checking with data size and heap overhead



Write path Offheaping – Performance results

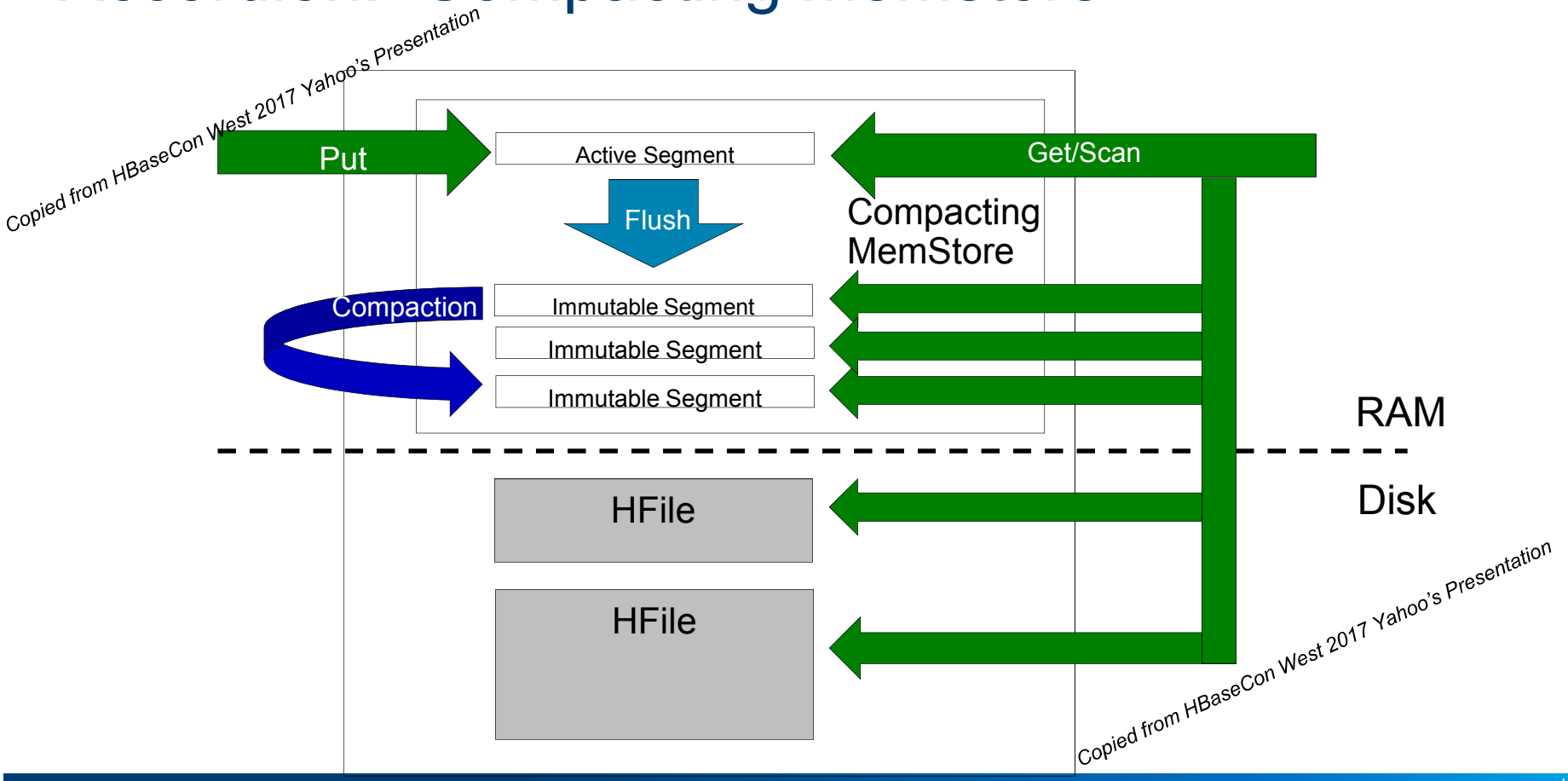
Performance Evaluation tool

- 200 regions – Single node
- 3 cells per row
- 300 bytes value size / cell
- 100 GB data write
- G1GC
  - 65% IHOP
  - On heap tests
    - 40 GB max heap
    - 60% Global memstore size
    - MSLAB pool 100%
  - Off heap tests
    - 24 GB global off heap memstores (60% of 40 GB)
    - 16 GB max heap



6% - 22% avg throughput gain

Accordion: Compacting memstore

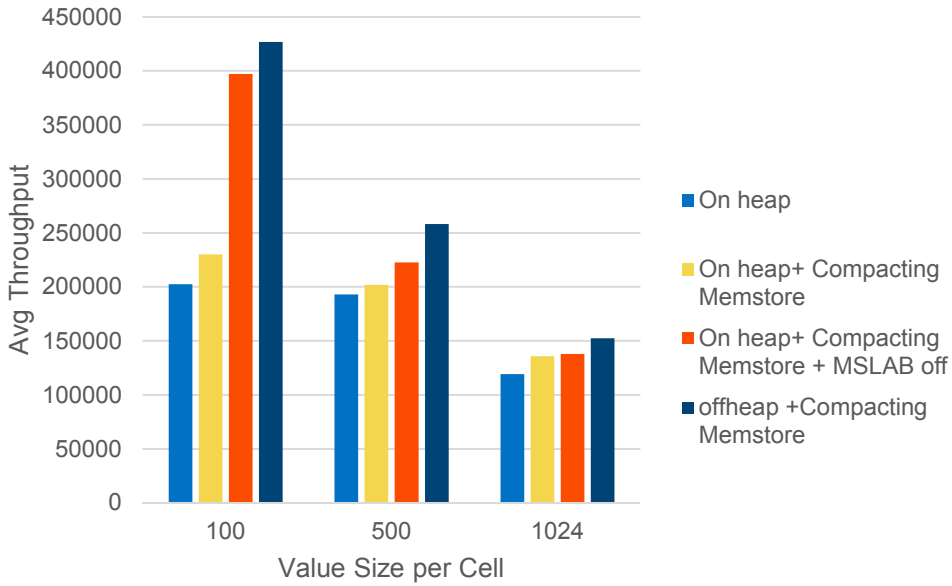


Write path Offheaping – Performance results

Off heap with Compacting memstore

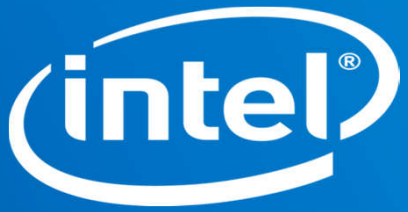
- In memory flushes/flattening reduces heap overhead.
- Allow memstore to grow bigger
- Off heap MSLAB pool with compacting memstores perform the best.

PE Tool – 50 threads and writing 25GB



- References
  - https://blogs.apache.org/hbase/
  - https://blogs.apache.org/hbase/entry/offheaping_the_read_path_in

Questions?


intel