hosted by Alibaba Group 阿里巴巴集团 APACHE HBASE

Use CCSMap to Improve HBase YGC Time & Efforts on SLA improvements from Alibaba search in 2018

Chance Li, Xiang Wang
Lijin Bin

Agenda

01 Why we need CCSMap

02 CCSMap structure

03 How to use CCSMap

04 Future work

Why we need CCSMap

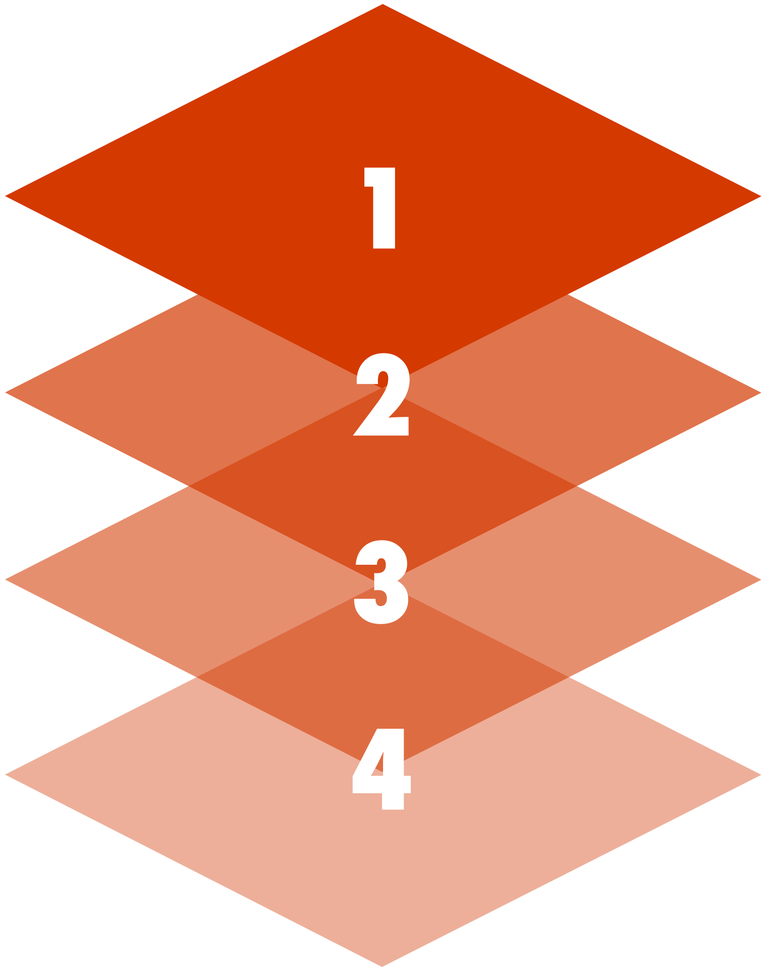
Overview

Heap is huge

Big memory preserved for HBase, BucketCache for read path but what about write path

CSLM not GC-friendly

Thousands of millions objects, long time to scan card table/old space leads to slow YGC



Overhead of JDK CSLM

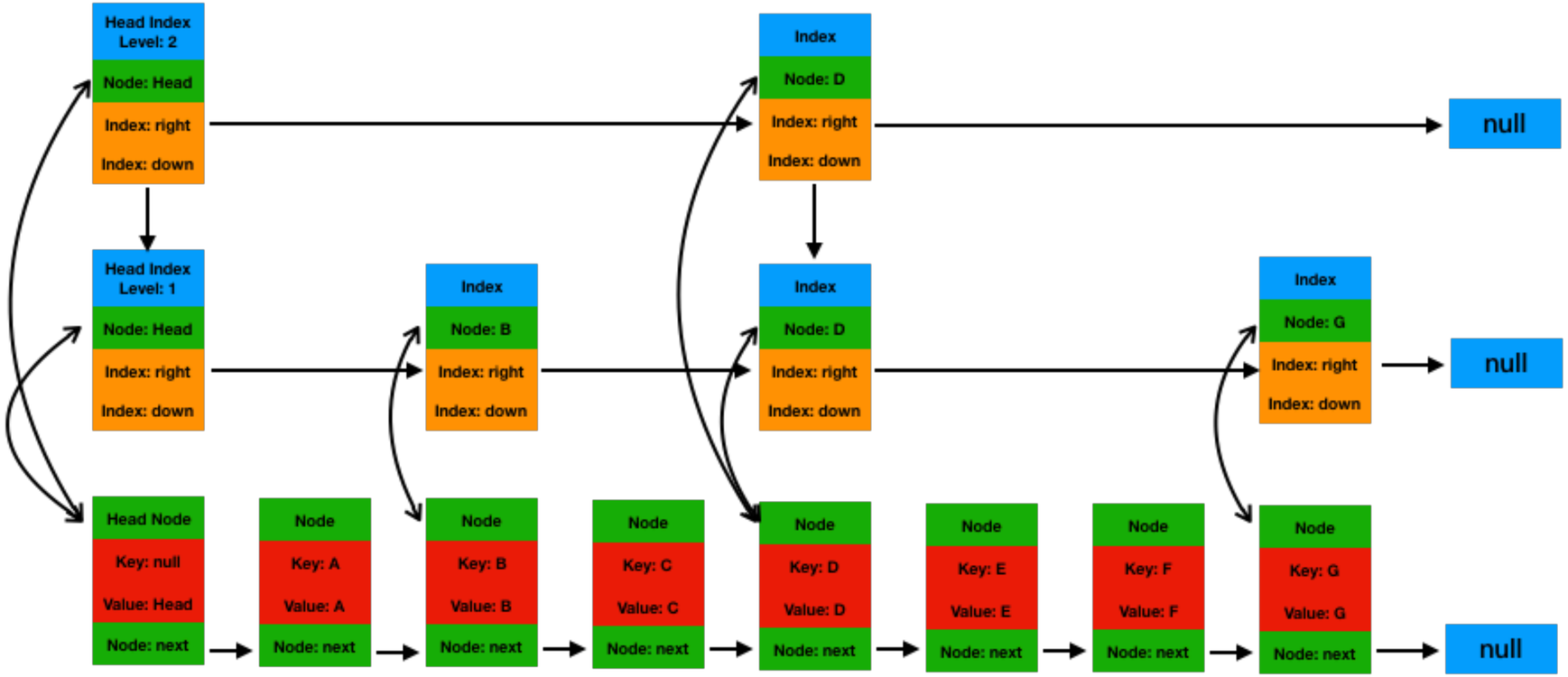
Needs 3 objects to store one key-value: index, Node and Cell, plenty of objects required than those for real data.

Memory fragment

Need SLAB to prevent CSLM from fragment.

Why we need CCSMap

Anatomy of JDK CSLM



Extra Objects

- Index: Almost ¼ of Nodes
- Nodes

Memory overhead

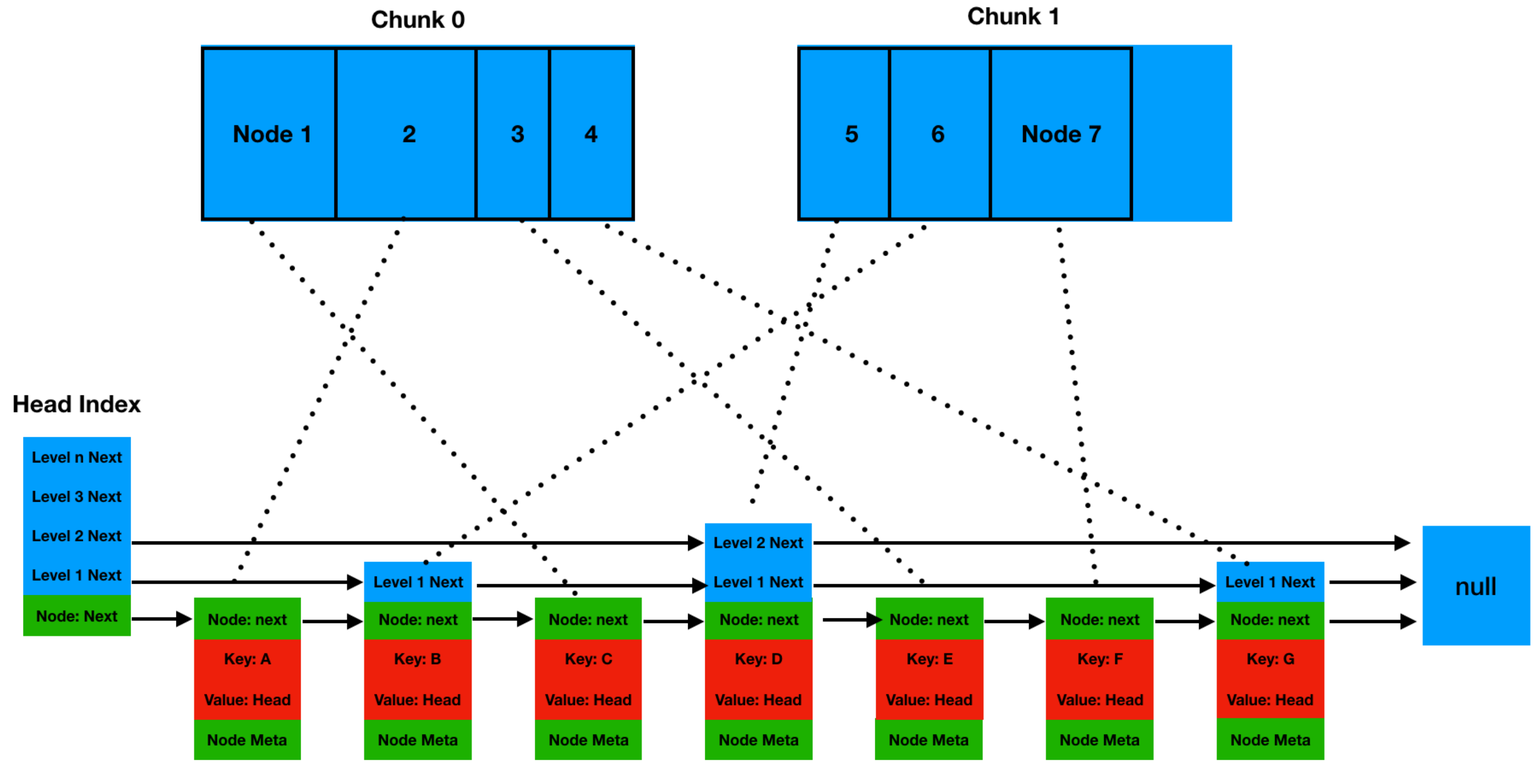
- Overhead 40B per Index
- Overhead 40B per Node

Example of 5M cell

- Extra 6.25M objects
- 250M memory overhead

CCSMap Structure

Anatomy of CCSMap



Self-managed memory

- Thousands of Chunks(4M)
- Multi nodes in a chunk

Memory overhead

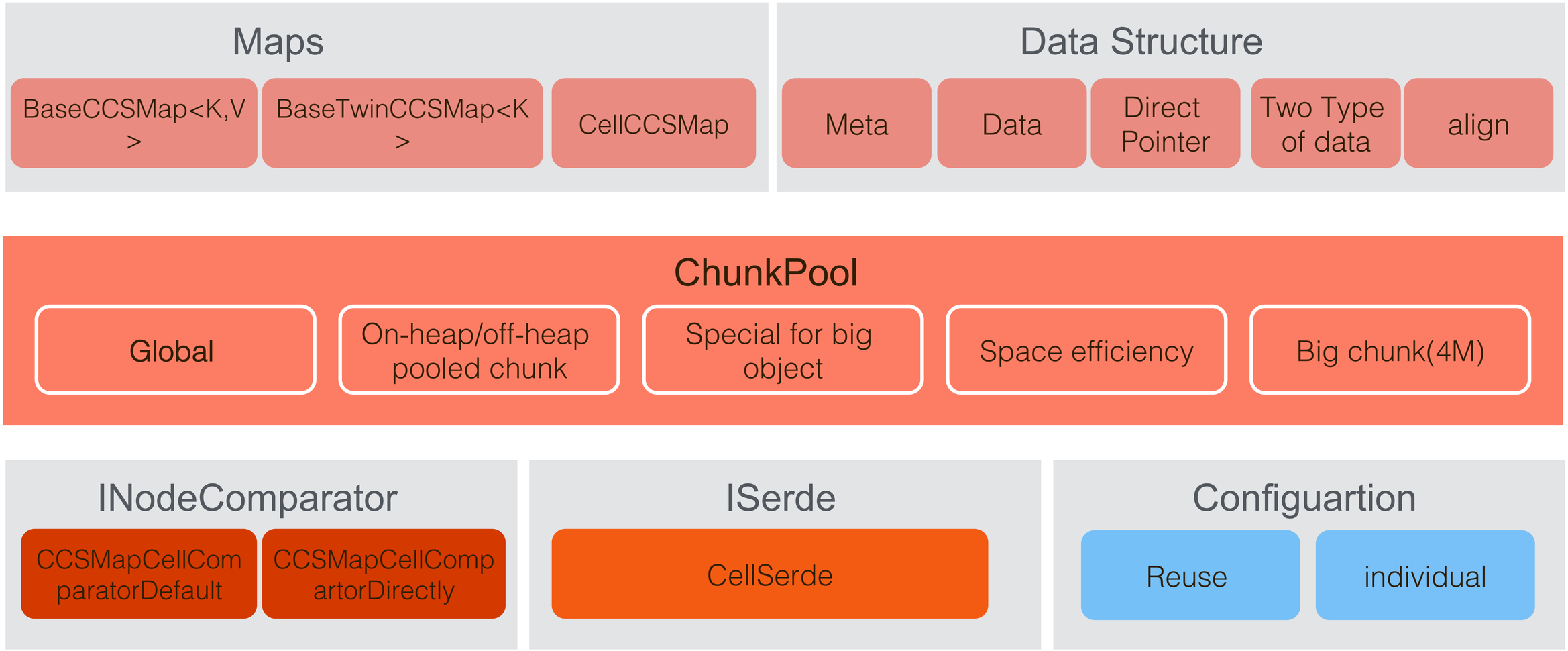
- Overhead 8B each Index
- Overhead 16B per Node Meta

Example of 5M cell

- No Extra objects
- 90M memory overhead

CCSMap Structure

Overall look



CCSMap Structure

Data structure of Node

```
* meta:
        * int level && lock:
          -- byte1 level  (maximum is 255)
          -- byte2 NodeStat
          -- byte3 NodeLock for preserve
          -- byte4 preserve
        * int dataLen
          -- for hbase Tags feature,  adding this dataLen to support fastly generating Cell
        * long getNextNode;    CAS update
        * long[] levelIndex;    CAS update
* data:    (if key and value is same object, only byte[])
        * int keyLen
        * int valueLen
        * byte[] key
        * byte[] value
```

For Hbase, key and value are all Cell, so data is byte[] of Cell, and we can deserialize the data to Cell(ByteBufferKeyValue) directly

CCSMap Structure

Classes for HBase internal

1. CCSMapMemstore
2. ImmutableSegmentOnCCSMap
3. MutableSegmentOnCCSMap
4. MemstoreLABProxyForCCSMap

How to use CCSMap

To enable CCSMap

CCSMap will be enable by default
after turning off CompactingMemstore

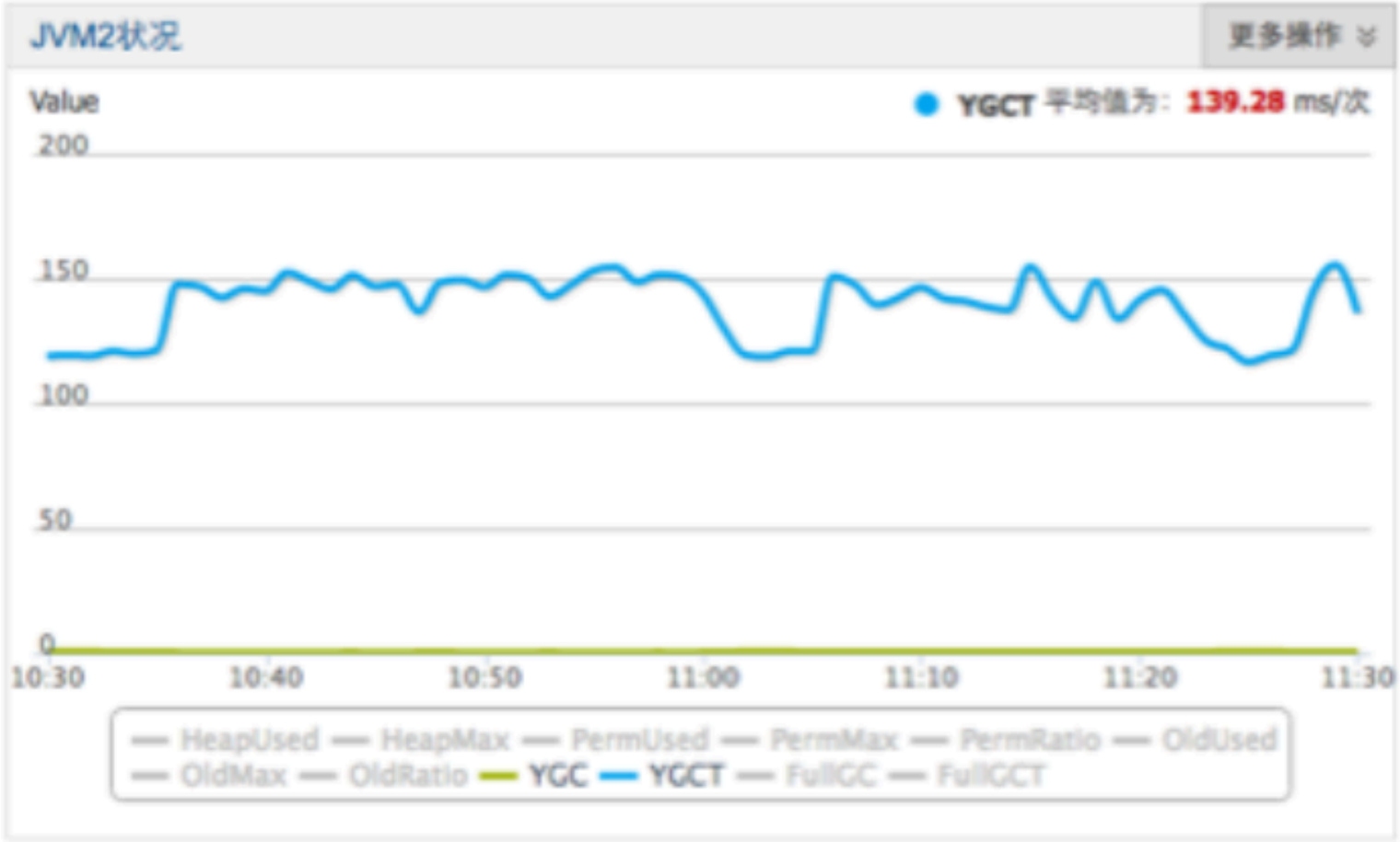
on-heap or off-heap

CCSMap individual configuration, or
original configuration used by
DefaultMemstore

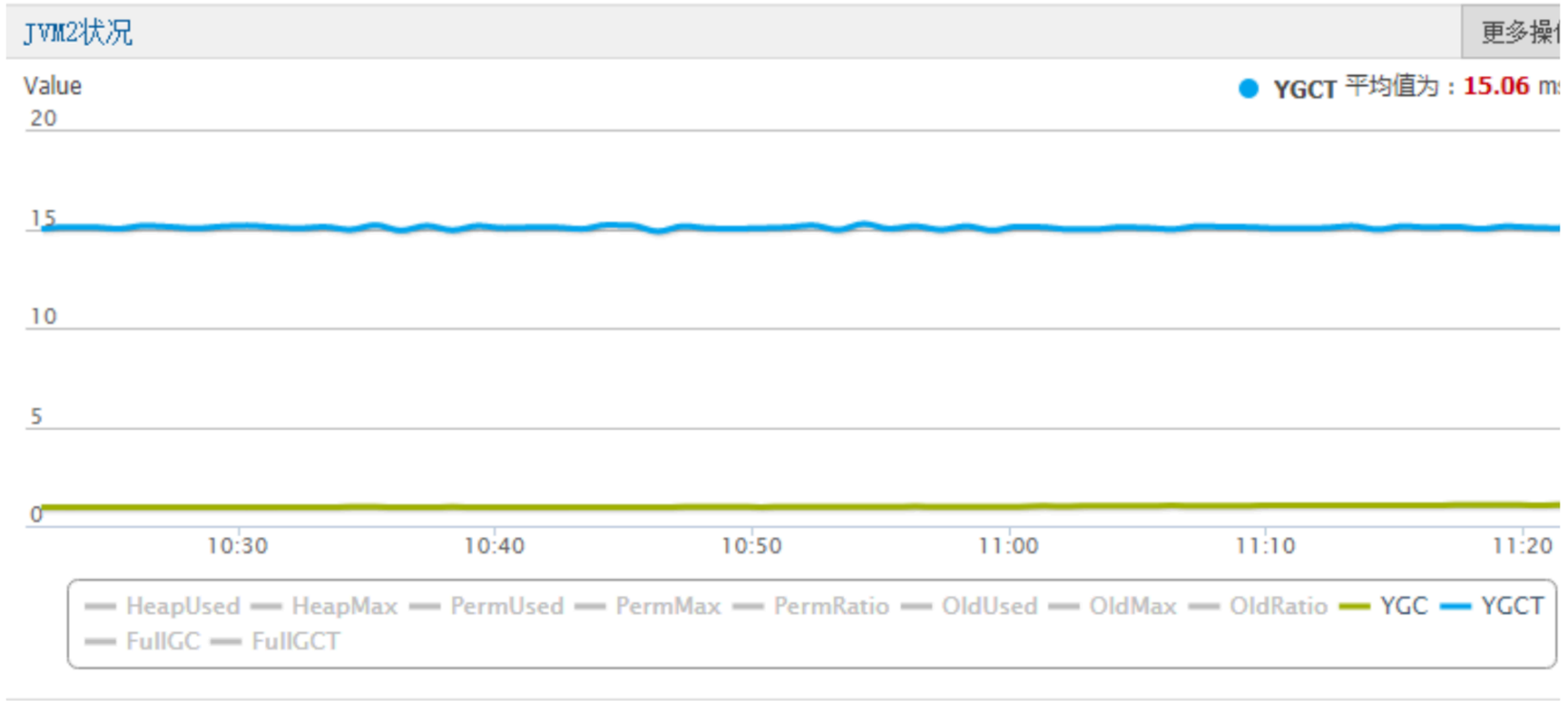
Other configurations

Chunk pool capacity, Chunk size,
count of pooled chunk to be initialized

Before using CCSMap



After using CCSMap



Future work

01 Combine CompactingMemstore

To support in-memory flush/compaction

02 To further improve ability of Memstore

To improve performance of put with dense columns

To improve performance of get with dense columns,
especially hot key

To support on Persistent Memory

Agenda

01 Client Connection Flood

02 HDFS Failure Affection

03 Disaster Recovery

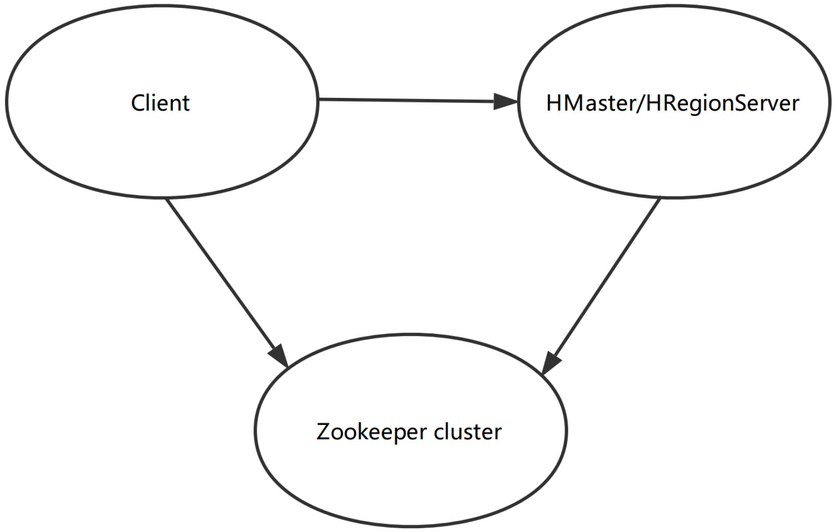
04 Stall Server

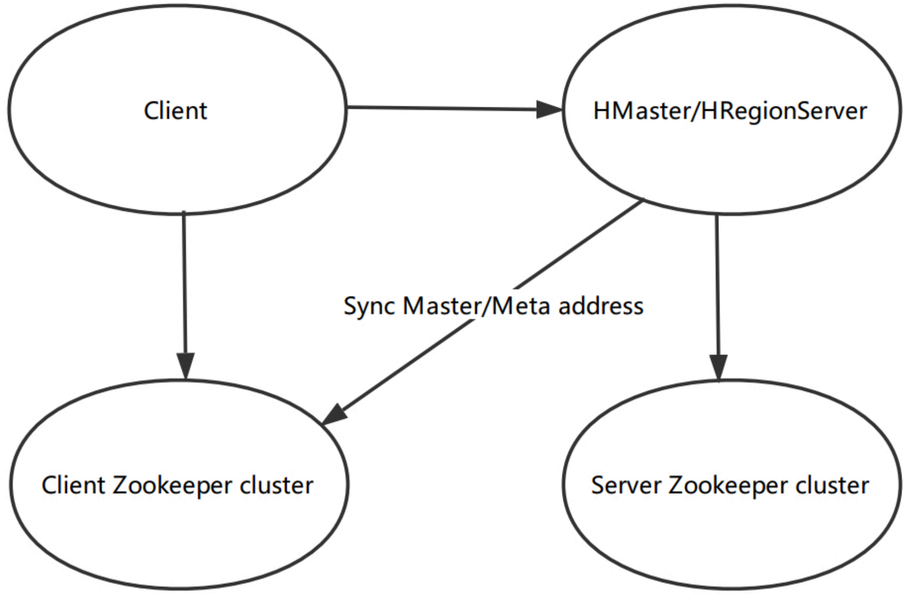
05 Dense Columns

06 Client Meta Acess Flood

Client Connection Flood

- Separate client and server ZK quorum to avoid client connection boost exhausted zookeeper and cause RegionServer ZK session loss(HBASE-20159)





- Configurations to enable this feature
  - hbase.client.zookeeper.quorum
  - hbase.client.zookeeper.property.clientPort
  - hbase.client.zookeeper.observer.mode

HDFS Failure Affection

- Allow RegionServer to live during HDFS failure(HBASE-20156)
  - Not abort when HDFS failure
    - Check FS and retry if flush fail
    - Postpone WAL rolling when FileSystem not available
  - RegionServer service will recover after HDFS comes back
  - Upstreaming in progress

Disaster Recovery

- Reduce ZK request to prevent ZK suspend during disaster recovery (HBASE-19290)
  - Before: request flood to ZK during disaster recovery
    - Get all SplitTasks from splitLogZNode
    - For each SplitTask
      - getChildren of rsZNode's to get number of available RSs
      - try grab SplitTask's lock break if no splitter available
    - Try to grab task if no splitter available
  - After
    - getChildren of rsZNode's to get number of available RSs
    - For each SplitTask: try grab SplitTask's lock break if no splitter available
    - Won't try grab task if no splitter available

Stall Server

- Enhance RegionServer self health check to avoid stall server (HBASE-20158)
  - Before
    - Heartbeat still works when resource exhausted, RS regarded as alive but inaccessible
    - Request hang on Disk failure cannot be captured by existing metrics
  - After (w/ DirectHealthChecker)
    - Pre-launched chore thread, not affected if physical resource exhausted
    - Periodically pick some regions on the RS and send request to itself
      - Shortcut requests, don’t need to access zookeeper/master/meta
  - Upstreaming in progress

Dense Columns

- Limit concurrency of put with dense columns to prevent write handler exhausted(HBASE-19389)
  - MemStore insertion is at KV granularity, dense columns will cause CPU bottleneck

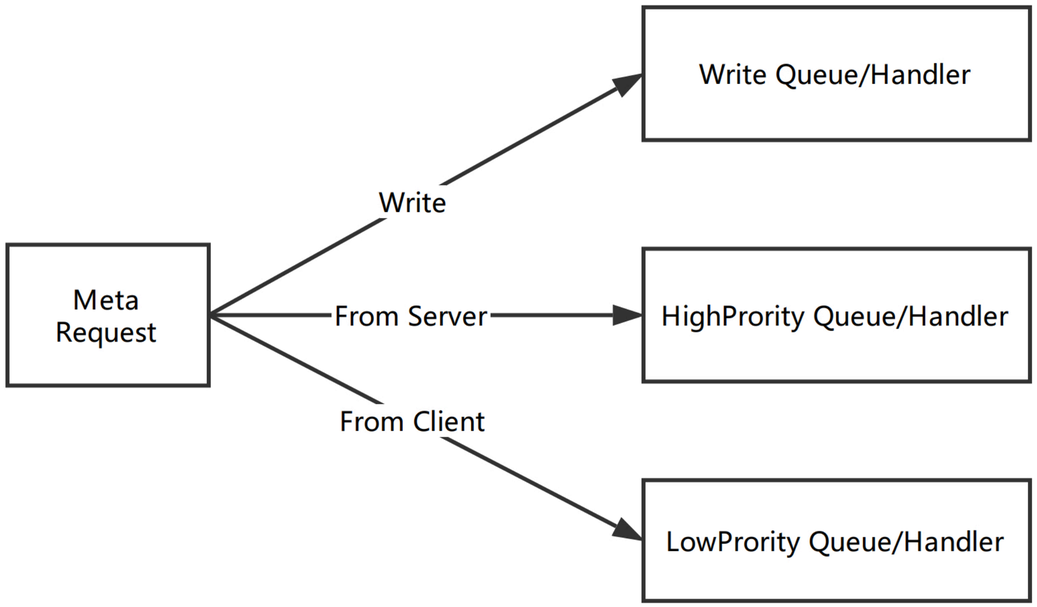
Average RT of 100K operations

	mslab=true			
	Only 1 thread put	**With 10 threads put**	**With 50 threads put**	**With 150 thread put**
1 column	2us	7us	21 ~ 48 us	87us
1000 column	2467 us	12409 us	70169 us	175942 us

  - We must limit the concurrency of such puts
  - Configurations
    - hbase.region.store.parallel.put.limit.min.column.count
    - hbase.region.store.parallel.put.limit

Client Meta Acess Flood

- Separate client and server request of meta (to be upstreamed)



hosted by Alibaba Group 阿里巴巴集团 APACHE HBASE

Thanks

HBase Dingding Group

Personal Wechat