****

**Professional Services**

**HBase Tuning Basics**

University of Maryland | June 2018

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision History** | | | |
| **Version** | **Author** | **Description** | **Date** |
| V1 | Michael Arnold | Initial Version | 06/03/2018 |

The information transmitted in this document is intended only for the person or entity to which it is addressed, and contains material confidential to Clairvoyant, LLC. Any review, retransmission, dissemination or other use of, or taking of any action in reliance upon, this information by persons or entities other than the intended recipient(s) is prohibited. If you received this document in error, please contact the sender and delete the material from your files.**T A B L E O F C O N T E N T S**

[**SECTION 1.0 RegionServer Tuning**](#_b4z4vkewwz24) **3**

[**SECTION 2.0 Other Tuning**](#_do9kpr3m5m4) **5**

[**SECTION 3.0 Data Tuning**](#_4mhf11agj9qo) **5**

# 

# SECTION 1.0 RegionServer Tuning

These are basic HBase RegionServer tuning parameters that switch from using the CMS to the G1 garbage collector and which moves data out of the JVM heap and into off-heap storage.

1. Use Oracle JDK 8.
2. Enable GC logging.  
   **Java Configuration Options for HBase RegionServer: -XX:+PrintGC -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -XX:+PrintAdaptiveSizePolicy -XX:+PrintReferenceGC -XX:+PrintFlagsFinal -Xloggc:/var/log/hbase/regionserver-gc.log**
3. Enable G1GC for RegionServer.  
   The Garbage-First (G1) collector is a server-style garbage collector, targeted for multi-processor machines with large memories. It meets garbage collection (GC) pause time goals with a high probability, while achieving high throughput.   
   **Java Configuration Options for HBase RegionServer: -XX:+UseG1GC -XX:MaxGCPauseMillis=100**
4. Tune G1GC.  
   <https://www.cloudera.com/documentation/enterprise/latest/topics/admin_hbase_garbage_collection.html>  
   Garbage collection (memory cleanup) by the JVM can cause HBase clients to experience excessive latency. To tune the garbage collection settings, you pass the relevant parameters to the JVM.  
   **Java Configuration Options for HBase RegionServer: -XX:+ParallelRefProcEnabled -XX:-ResizePLAB -XX:ParallelGCThreads=18**

**-XX:+UnlockExperimentalVMOptions**

**-XX:G1NewSizePercent=2**

1. Increase Java Heap of HBase RegionServer.  
   <https://blog.codecentric.de/en/2014/02/35gb-heap-less-32gb-java-jvm-memory-oddities/>  
   **Java Heap Size of HBase RegionServer in Bytes: 48 GiB**
2. Enable HBase BucketCache.  
   <https://www.cloudera.com/documentation/enterprise/latest/topics/admin_hbase_blockcache_configure.html>  
   In the default configuration, HBase uses a single on-heap cache. If you configure the off-heap BucketCache, the on-heap cache is used for Bloom filters and indexes, and the off-heap BucketCache is used to cache data blocks. This is called the Combined Blockcache configuration. The Combined BlockCache allows you to use a larger in-memory cache while reducing the negative impact of garbage collection in the heap, because HBase manages the BucketCache instead of relying on the garbage collector.  
   **hbase.bucketcache.combinedcache.enabled: true  
   hbase.bucketcache.ioengine: offheap  
   hbase.bucketcache.size: 48 GiB  
   hfile.block.cache.size: 0.2  
   HBase Client Environment Advanced Configuration Snippet for hbase-env.sh:  
   HBASE\_OFFHEAPSIZE=64G  
   HBASE\_OPTS=-XX:MaxDirectMemorySize=64G**
3. Enable HBase MultiWAL Support.  
   <https://www.cloudera.com/documentation/enterprise/latest/topics/admin_configure_multiwal.html>  
   MultiWAL allows a RegionServer to write multiple WAL streams in parallel, by using multiple pipelines in the underlying HDFS instance, which increases total throughput during writes. This parallelization is done by partitioning incoming edits by their Region.  
   **hbase.wal.provider: Multiple HDFS WAL**  
   **hbase.wal.regiongrouping.numgroups: 8**
4. Enable HDFS Hedged Reads  
   <https://www.cloudera.com/documentation/enterprise/latest/topics/admin_hedged_reads.html>  
   If a read from a block is slow, the HDFS client starts up another parallel, 'hedged' read against a different block replica. The result of whichever read returns first is used, and the outstanding read is cancelled. This feature helps in situations where a read occasionally takes a long time rather than when there is a systemic problem.   
   **dfs.client.hedged.read.threadpool.size: 10  
   dfs.client.hedged.read.threshold.millis: 500 milliseconds**

# SECTION 2.0 Other Tuning

1. Recommended # of regions per RegionServer is 100-200. Cleanup and Reduce regions to a recommended range.
2. Size of the data.
3. Tune DataNode Transceivers
4. Increase HFiles (for major compactions).
5. Disable automatic compactions and switch to manual compactions.
6. HBase Memstore Flush Size: 128 to 512 MB - best value?
7. HBase Region Split Limit?
8. Increase hbase.regionserver.global.memstore.upperLimit to 0.5
9. Increase hbase.hregion.memstore.block.multiplier to 4
10. Increase hbase.hstore.compactionThreshold to 10

# SECTION 3.0 Data Tuning

1. Pre-split tables
2. Regions per RS are balanced but requests per region is not consistent - hotspotting on several RS
3. Review Key Salting Algorithm / Check if rowkeys are properly hashed and salted
4. Enable compression on all tables.