**Heap Fragmentation in HBase Region Server - 2011-01-20**

These graphs show two metrics about the old generation in an 8GB heap.

**RegionServer JVM settings:**

Sun Java6u23

-XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFraction=65 -Xms8g -Xmx8g
In this particular test, the new size is not constrained and has auto-tuned itself to 256MB.

**Metrics**

The metrics measured are:
- **free_space** - the total amount of free space (in words) in the old generation
- **max_chunk** - the size of the largest chunk (in words) in the old generation. If an object larger than this needs to be promoted from the young generation, then a promotion failure will occur and a full compacting GC (stop-the-world) will result.

These are parsed from the output of -XX:+PrintFLSStatistics=1

**Workloads**
The cluster has 5 servers running 0.90rc3. About 500 regions total, max region size 1GB, 200 RS handlers. Everything else default.

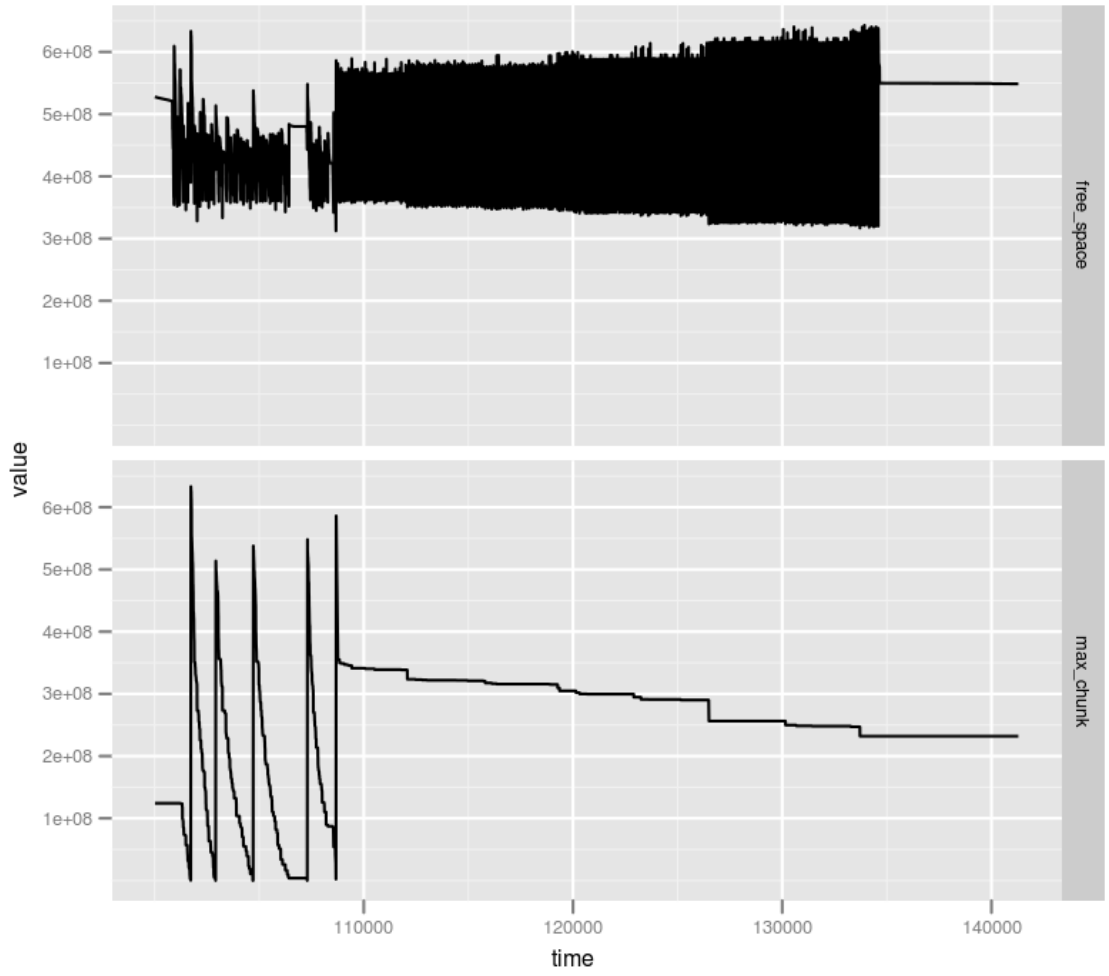
Three workloads are presented using YCSB with a zipfian distribution.

First workload (write-only):
- 100% writes spread across 10 million records, each 10 fields at 100bytes + key

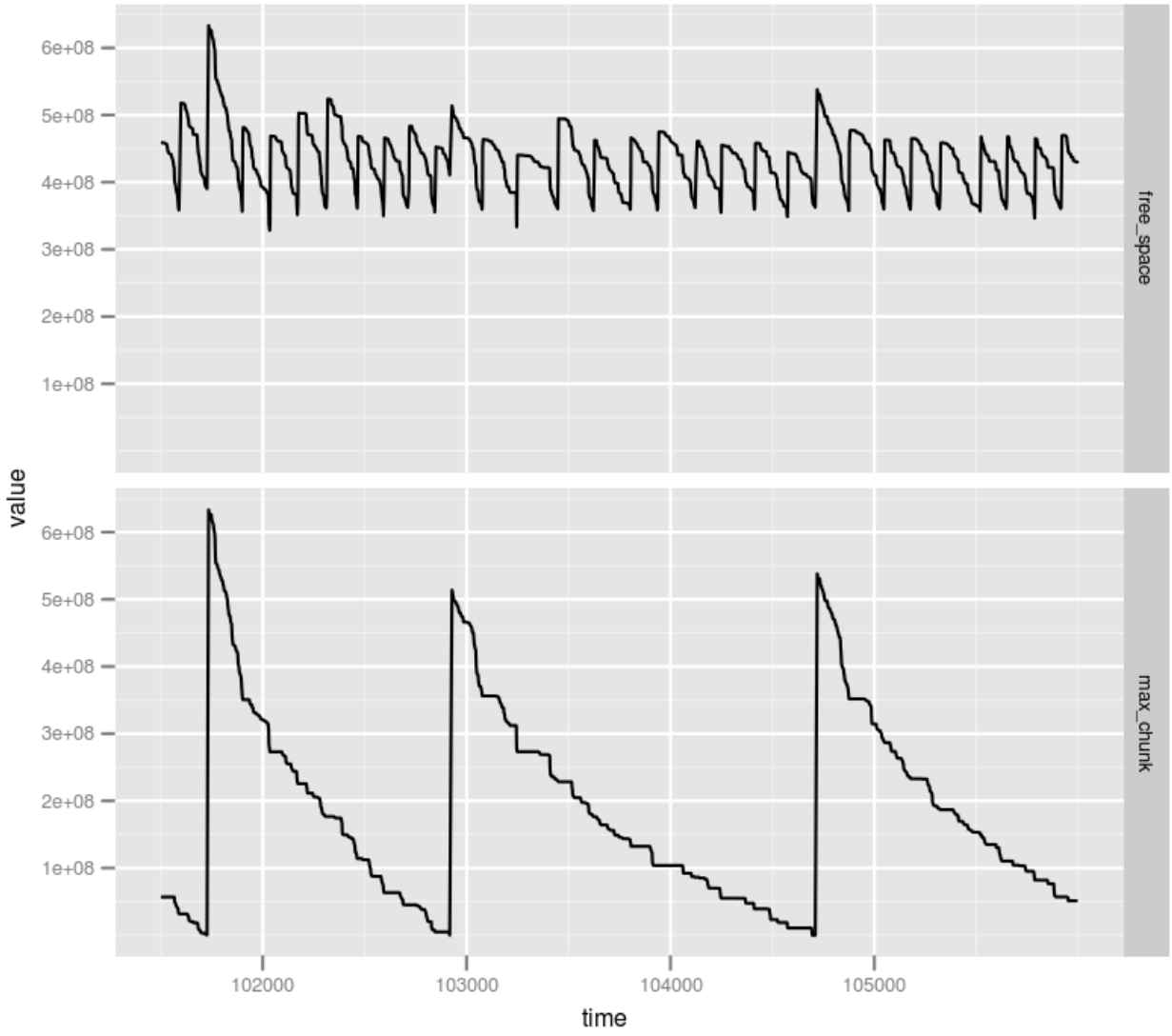
Second workload (read-only):
- 100% reads spread across same 10 million records

Third workload (read-only with working set in block cache):
- 100% reads spread across only 10 thousand records

**Overall graph**
First workload (write-only) is from the beginning of the graph up to about t=108000
Second workload (read-only with working set larger than block cache) from t=108000 to t=135000
Third workload (read-only with working set in block cache) from t=135000 to end

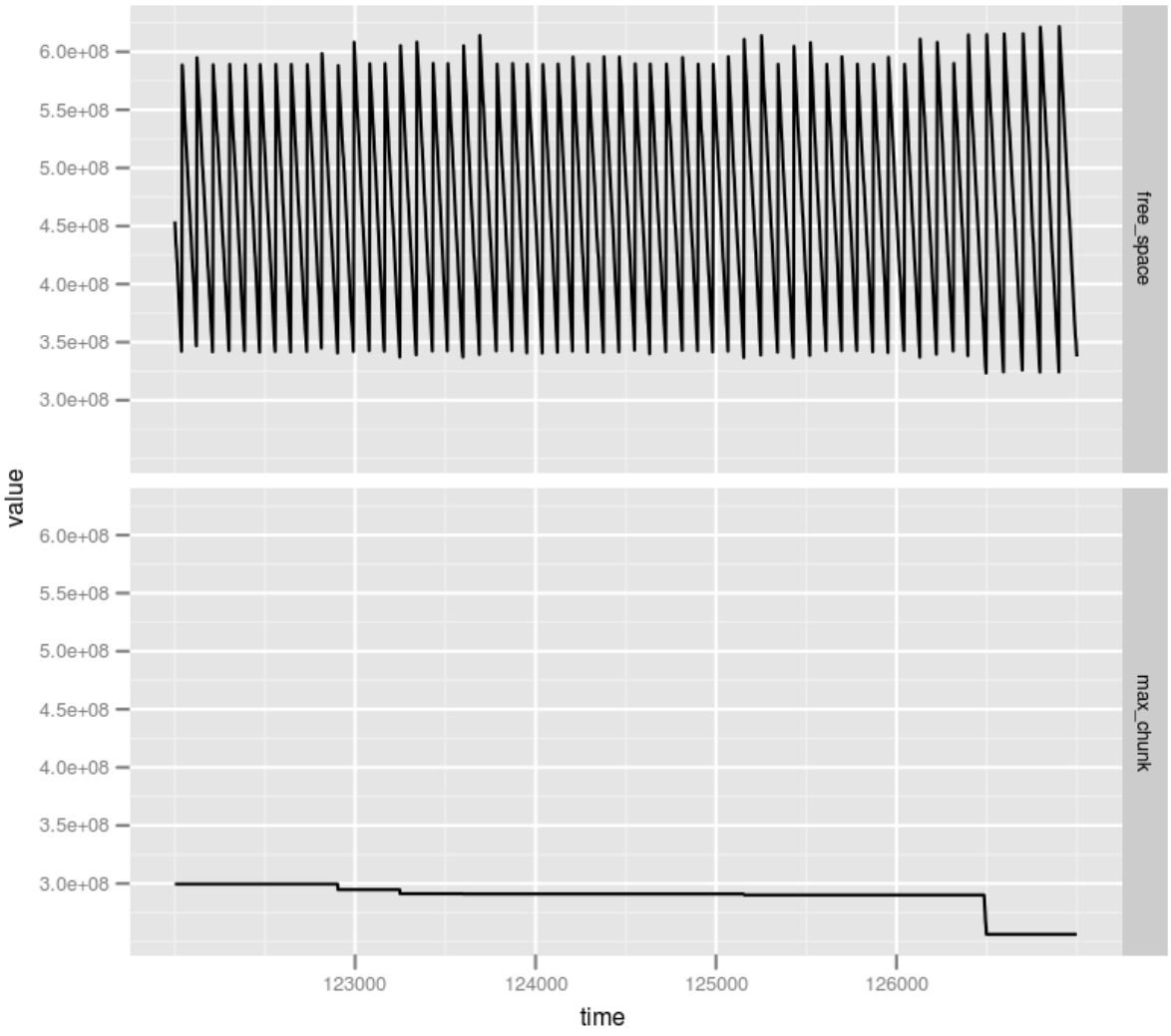



**Zoom in on first workload (write-only):**




The free_space line is hovering nicely between 350-500megawords (2.8GB-4GB). Each bump in free_space is a CMS concurrent collection.
max_chunk however slowly drops until a minimum of 30kilowords (240KB). At this point there is a promotion failure causing a full compacting GC, and max_chunk jumps back up to 4GB.

**Zoom in on second workload (read-only with active set larger than block cache)**




In this workload the LRU is constantly evicting blocks because the working set doesn't fit. This results in CMS collections happening much more frequently (somewhere around twice the frequency of collections compared to the write-only workload)
But, max_chunk doesn't decrease nearly as fast, indicating that the LRU cache doesn't cause a lot of fragmentation even when being evicted quickly.

**Third workload (read-only with active set in LRU cache)**

No need to zoom in here - there are zero old gen collections and hence no fragmentation caused by this workload. This makes sense - only short-lived data structures are allocated to copy data out of block cache to IPC, and they all fit in the young generation.