PerformanceTuning

Improving write performance

1. Make sure your commit log and data dirs (sstables) are on different disks.
2. There isn't really a step 2. Writes go into the commitlog (fast) and straight into memory (really fast) and can't really be sped up beyond that.

If you're seeing huge latency spikes under write-heavy loads, that's due to other factors. It's a 'fix what has gone horribly wrong' problem, not a 'tune for better peak performance' problem.

Improving read performance

Be sure to review the page on Memtable Thresholds – several important configuration and system settings are described there.

Lessen overall system impact of Compactions

To lower compaction priority (thus reducing its impact on the rest of the system, and making it take longer), add these options to cassandra.in.sh in 0.6.3 or later:

```
-XX:+UseThreadPriorities \
-XX:ThreadPriorityPolicy=42 \
-Dcassandra.compaction.priority=1 \
```

The above option for setting the compaction priority (-Dcassandra.compaction.priority=1) is out of date as of 1.0. See the options for controlling compaction settings in the cassandra yaml configuration for details. The above code was originally pulled from: https://issues.apache.org/jira/browse/CASSANDRA-1181

Useful JVM options

```
-XX:+UseCompressedOops # enables compressed references, reducing memory overhead on 64bit JVMs
```

See http://wikis.sun.com/display/HotSpotInternals/CompressedOops and http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/topic/com.ibm.java.doc.diagnostics.60/diag/understanding/mm_compressed_references.html for discussion on the Sun and IBM JVMs, respectively.

Compressed references (aka OOPs = ordinary object pointers) are not stable for Sun JVM versions before 6u19.

System Settings

todo: describe how top, iostat -x, and JMX stats can help you see what is making things slow

For starters, see "Linux Performance Basics for Cassandra".

`https://c.statcounter.com/9397521/0/fe557aad/1/|stats`