Comparison of Bulk assignment using ZK and ZK-less assignment (HBASE-11059, HBASE-11546) on Hbase 0.98

Author: Virag Kothari, Francis Liu

Cluster size: 160 Region servers with 1M regions

Zk-assignment:

a) forceSync=yes - Untested on this cluster (Previous setup had 1.5 hrs for forcesync=no and 7 hrs for forcesync=yes. The previous setup with forceSync=no was for a smaller cluster (Look at prevous benchmark) and the only purpose of mentioning this here is for relative performance comparison between forceSync=yes and forceSync=no. It should not be used to compare with the current setup.)

b) forceSync=no - 10 mins 46 secs

Zk-less assignment:
a) Write to remote META with single HTable (original patch) - 1 hr 16 mins
b) Write to remote META using HBASE-11610 - 12 mins
c) Not write to META at all - 10 mins 34 secs (This was just an experiment and was done by commenting the synchronized block doing htable.put() in RegionStateStore.)

Resource utilization:

a) CPU utilization on master - ~50%

Master Configuration:
Intel® Xeon® Processor E5620
Memory 23Gb

Size of META:
7GB (META has 10 versions)

Settings:

On master
hbase.regionserver.handler.count=100
hbase.bulk.assignment.threadpool.size=30

On regionserver:
hbase.regionserver.executor.openregion.threads=50

Observations:
No performance gain when doing zk-less assignment when compared to using zk-assignment with force-sync=no
Expirement(c) shows that overhead of writing to META is 1.5 mins
From the profiler, cpu spends around 50% of time in locking region states (HBASE-11290).
After disabling and enabling table multiple times, performance degraded by around 20% likely due to compaction

Thoughts:
Advantage of zk less assignment
1) Better API's (ls on znode vs scan table)
2) Scalability (If META is split)
  a) Read/Write throughput
  b) Storage (1M child znodes vs 1M rows in META)