Scalable Meta

Master Topology in HBase 2.0

1 Preamble

A few of us tried proposing single meta region colocated on Master because it made our lives easier but we got pushback arguing single meta colocated (or not) just won't scale [1].

2 (Scaling)Requirement

1. 1-10M regions (50M regions?) or to put it another way, the requirement is being able to host one or more petabyte(s) on a single cluster; N hundreds or N thousands of servers x P regions of size Q gigabyte: e.g. 1k servers w/ 1k regions each of 1G in size.
2. 'Fast' startup: e.g. stop, upgrade, start cluster of 10M regions in 30minutes.

3 Related Tendency/Tenet

For a while, we have been talking of moving away from having to go in the direction of larger and larger regions (i.e. 10G, 20G++) and reverse direction and tend instead toward smaller regions -- e.g. 1G or even smaller (HDFS blocksize?) -- so we operate with tables that have region boundaries at a finer granularity. Smaller regions would mitigate write-amplification. We are more likely to compact only what has changed rather than what has changed *plus* a large chunk of the key space on either side of the changed keys just because our regions are 'big'. Smaller regions implies more regions for the master to handle.

4 A Few stats

1. From a Y! experiment with 1M row meta region fully compacted is 7G in HDFS. This is a meta with no extra 'features'/columns in meta[1]. By default meta keeps ten versions of a cell so perhaps some bloat here (To be revisited). But lets say 10G is operating size for 1M regions in a cluster or 10k per meta row.
2. From Y! experiment, a cluster of 1M regions at steady-state consumes 3G in Master keeping cluster 'state'. This does not include 'caching' cost.
3. 1M regions with say 2 columnfamilies each with say 3 HFiles in each columnfamily is about 6M files. NN does "100M files". Y! has a NN hosting 250M files in 128G.

[1] "First, we would like to keep a history of RS assignments for the region in meta (this is for debugging + new assignment design). Second, ideally we would want to keep the region files in meta as well and get away without doing reference files, etc. " [1]

5 Problems

1. If big cluster (1M regions) and/or if tendency is toward smaller regions, a single meta implies our 'hottest' region will be exceptional in size -- 7G -- with attendant write-amplification issues not to mind 'socket timeout' on startup (master handling all meta transactions, r/w), compactions during startup impinging on performance, e.g. 3minutes to scan meta when already compacted, 45 minutes when a concurrent compaction, and so on.
2. Caching a big meta region so we can respond to queries out of cache.
3. Heavy r/w i/o on single hot meta region hosted by a single server (From Y!, on startup of 1M region cluster, Master is diong 400k writes/minute assigning)
4. Lars Hofhansl notes there are also regionserver-side obstacles to scaling [2][3] that will need to be addressed though 10k regions on a single server doesn't look "*too bad*" when idle [4]

6 Solutions notes

Single meta cohosted on master won't cut it for scale case.

Split meta distributes caching across cluster, distributes iops across cluster, meta is like any other region (except prioritized -- we have this in place already) so no special-casing around sizing/compactions, and split meta means a server down means we lose a piece of meta only (more 'available'). A split meta would be easier scanning in // shortening startup time. Compaction is distributed.

Do meta with replicas (e.g.HBASE-10070) would help with read i/o, but not with write nor caching, etc.

Could work on making compactions better for the big meta region case though we tried once -- stripe compactions -- w/ inconclusive results. Improving compactions doesn't help with other dimensions: r/w i/o nor caching.

7 Notes

Who of our users has 1M and up regions? Flurry (300k, now 50k, and actively trying to keep the count down by going bigger on regions). Y! are heading here by end of the year. If POD/CELL layout (FB, SFDC), we do not run into issues with meta. If 300 regions per server and 20 nodes per POD, that makes for about 6k regions. If 200 nodes per POD, 60k. Say regions are 10G? If we want 1G or smaller, thats pushing 100k regions a small cluster and we are up at 1M for a larger POD.

HA Master and/or partitioned master seem orthogonal to the scale issues being discussed here. Let Master be such that it can be offline a while and read/writes go ahead. A non-HA Maser simplfies deploy/ops. Lets do HA Master and/or partitioned Masters later. A single Master is easier debugging and reasoning about state. It is easier than debugging state that is partitioned across N servers or state that is replicated over N servers.
Single meta happened because we thought it scale enough and were too lazy to do the work that would allow meta split.

8 FAQ

8.1 What does small-regions buy us? Anything?

Distributed i/o, cache and compaction with meta treated same as any user region rather than our having to special-case meta handling.

8.2 “We should also discuss why few, large regions are bad, and whether we can decouple the unit of distribution (a region) from whatever unit we're trying to operate on.”

Yes. Lets.

8.3 Do we need to split this conversation into what to do on master and what to do with 0.98?

No. Lets solve this for 0.98 experimental work and for master rather than have two solutions.

8.4 A directory in HDFS w/ 1M or 50M entries is untenable?

1M maybe, but more would need different layout.

8.5 Split meta makes assign and crash harder. Slows startup because more by way of prerequisite (a root and all the pieces of meta)

8.6 Do we need to bring root back? Can't we simply host all of root in zookeeper? We expect the number of meta regions in the tens / hundreds case right?