Co-locate Meta and Master HBASE-10569

Argument for Master always hosting the hbase:meta table

Jimmy Xiang, Matteo, and Stack

1 Preamble
2 Possible Deploys
  2.1 Master hosts hbase:meta
    2.1.1 Plusses
    2.1.2 Negatives
  2.2 Master independent of hbase:meta
    2.2.1 Plusses
    2.2.2 Negatives
  2.3 Master hosts hbase:meta, Master function is split
    2.3.1 Plusses
    2.3.2 Negatives
3 Frequently Asked Questions
  3.1 Q: So when we failover the master we'll force a move of meta. Won't that lead to bigger "blib" than before.
  3.2 Q: Load balancer will move user regions to other regionservers [...once it becomes the new Master]. Seems like this will wreak havoc to data locality?
  3.3 Q: Would this patch prevent us from ever splitting the meta in the future?

1 Preamble

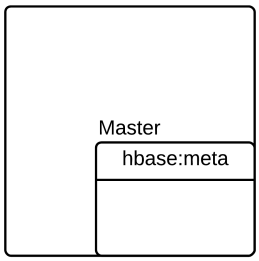
1. The master, in order to operate efficaciously, must maintain an in-memory consistent replica of the content of hbase:meta table. Ditto for other system tables: ACLs, namespaces, etc.
    a. If the Master is the only writer of said system tables, consistency is easier-to-do.
    b. State transitions across memory and hbase:meta should be atomic, prompt.
2. Updates to system tables usually also entail updating listening processes; e.g. other RegionServers want to learn near-immediately of updates on the ACL table so they can enforce the updated access rules.
    a. Currently notification is done via another system altogether, zookeeper.
    b. Enforcing quota implies being able to intercede. Interception of system table edits makes for a good ‘choke’ point.

2 Possible Deploys

HBASE-10569 implements colocated Master and hbase:meta. Lets explain why this is what we want going forward by entertaining the available Master/hbase:meta deploy options. Later in an FAQ we answer some of the objections raised in the issue.

2.1 Master hosts hbase:meta

Master hosts hbase:meta[1]. They are a unit.



1. When the master crashes, hbase:meta goes away also and vice versa. New master begins by recovering the hbase:meta table, always.
    a. Currently we designate 'backup masters'. HBASE-10569 does not change this. But lets get to a place where any regionserver can be 'backup master'. Let the notion of 'backup master' evaporate. TODO.
    b. On crash, designated backup masters (or later, any cluster member after we implement 'a.' above) rushes to become Master. Transitionally, backup masters are "lightly" loaded (if at all) enforced by the LoadBalancer (Jimmy is working on this). Also while in transition from current dedicated Master to any-node-can-be-Master, the MasterRegionServer can only host hbase:meta (and other system tables). The LoadBalancer gives the hbase:meta table a heavy weight. Other regions are offloaded for now. Later, after ensuring QoC, no need to move regions off and no need to designate lightly loaded Backup Masters.
2. The master does not need to RPC to edit the meta table; when colocated, Master can short-circuit the RPC.
3. The Master can observe edits -- A Coprocessor -- on hbase:meta (or any 'system') table and take secondary actions w/o need for the editor to do an out of band "notify of the master" on table change.

2.1.1 Plusses

1. Simpler bootstrapping, recovery
2. If we install an observer coprocessor, Master is updated when system table changes.
3. Faster updates if we can by-pass RPC

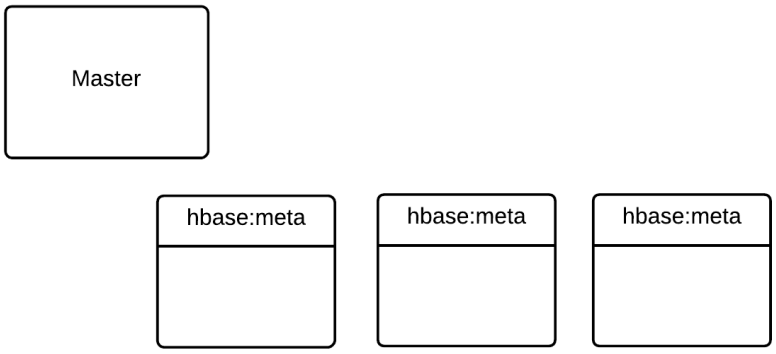
2.1.2 Negatives

1. Statistically, adding probability of Master outage to hbase:meta outage, we may be offline more (counter argument is that the simpler deploy allows for speedier recovery)

[1] A side-effect of HBASE-10569 is that the Master is now also a RegionServer, it can host regions.

2.2 Master independent of hbase:meta

As it was before HBASE-10569, the Master and the hbase:meta roam independent of each other. Also allow that hbase:meta can split.



2.2.1 Plusses

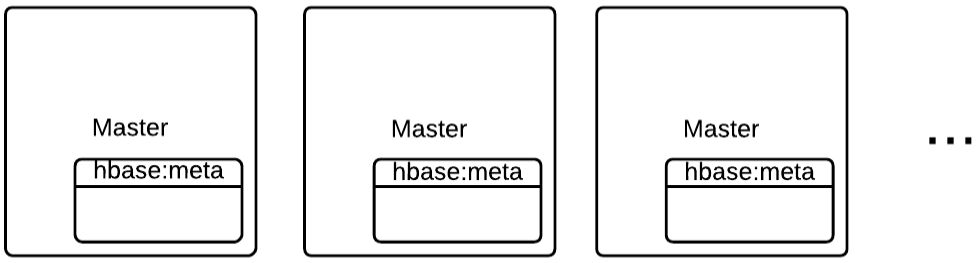
1. If the server hosting hbase:meta goes down, we can still talk to Master
2. If Master is down, edits to hbase:meta can proceed; hbase:meta is available for reads
3. This is the devil we know.

2.2.2 Negatives

1. If the server hosting hbase:meta goes down, we can still talk to Master only Master can't do anything w/o an available hbase:meta!.
2. If Master is down, edits to hbase:meta can proceed though cannot complete if no Master; e.g. split. If Master is enforcing quota, quota can be violated if no Master to enforce.
3. Complex recovery and deploy of hbase:meta hosted 'elsewhere'
4. RPC to update hbase:meta makes consistency harder, makes hbase:meta update 'slower' having to cross RPC
5. Master (or zk) needs its chain pulled explicitly -- a client must update the table AND signal the Master/zk -- if a system table is changed, one that it is 'watching'.

2.3 Master hosts hbase:meta, Master function is split

I'm not sure how this work (where would the heartbeats go, how do the masters not clash running cluster-wide operations unless a global cluster lock, etc…Perhaps we partition by 'table')



2.3.1 Plusses

1. Distributed master function, distributed hbase:meta

2.3.2 Negatives

1. Too much to do for now

3 Frequently Asked Questions

A few questions that came up in the issue.

3.1 Q: So when we failover the master we'll force a move of meta. Won't that lead to bigger "blib" than before.

Yes. If Master goes away, so does the hbase:meta. A new Master will assume the Master role and recover hbase:meta and redeploy it.

Failing over the Master in the old setup went unnoticed because the Master and the new Master weren't doing anything. Two machines were standing by effectively idle. Putting these previously idle resources to work will cost.

3.2 Q: Load balancer will move user regions to other regionservers [...once it becomes the new Master]. Seems like this will wreak havoc to data locality?

Yes. These regions will lose their locality.

Making it so the Master only carries hbase:meta and other system tables exclusively is a transitional effect while we get comfortable with the new order and are sure that an hbase:meta region hosted on a busy regionserver has its requests serviced ahead all user regions.

3.3 Q: Would this patch prevent us from ever splitting the meta in the future?

Maybe, or, it might be hard if we go the route of distributing the Master function w/ each Master hosting some subset of the table space. Other outs would be hosting multiple (consistent) replicas of the hbase:meta table distributing load out this way.