Date: 6 May, 2010

Author: Karthik Ranganathan

1. Proposed ZK based flows
2. Current operations on cluster
3. Code that needs to be revisited
4. All messages between master and region servers today

# 1. Proposed ZK based flows:

1. The basic operations are open and close. Reassignment is a close followed by an open.
2. Splits are unaided on the RS. RS splits, adds regions to META, opens daughters and then reports to the master.
3. Startup: master adds a bunch of regions to the unassigned pool in ZK. These get assigned out and marked as open once they are opened.
4. Shutdown: it follows the current scheme. No change here.

**Close region**

**RS Flow:**

M -> RS : close R

RS -> ZK : add node /UNASSIGNED/R[CLOSING @ TS1]

RS : closeRegion(R); Periodically update ZK:/UNASSIGNED/R[CLOSING @ TS]

RS -> ZK : /UNASSIGNED/R[CLOSED @ TS2]

**Master flow:**

M -> ZK : Watch /UNASSIGNED

ZK -> M : created (/UNASSIGNED/R[CLOSING @ TS1])

M -> ZK : watch /UNASSIGNED/R

ZK -> M : Updated /UNASSIGNED/R[CLOSED @ TS2]

**Open region**

**RS Flow:**

M -> RS : Open R

RS -> ZK : Update /UNASSIGNED/R[OPENING @ TS3]

RS : openRegion(R); Periodically update /UNASSIGNED/R[OPENING @ TS]

RS -> ZK : Update /UNASSIGNED/R[OPENED @ TS4]

**Master Flow:**

ZK -> M : /UNASSIGNED/R[OPENED @ TS4]

M : Insert R into meta

M -> ZK : Delete /UNASSIGNED/R

**Split region**

The RS should do the split by itself, with no intervention from the master.

Split into daughters, insert into META, bring daughters online and then inform the master.

# 2. Current operations on the cluster:

**Opening a region:**

1. M -> RS : *MSG\_REGION\_OPEN*
2. RS -> M : *MSG\_REPORT\_PROCESS\_OPEN [+]*
3. RS -> M : *MSG\_REPORT\_OPEN*
4. M : Add opened region to META

Notes:

* If master dies between (1,4), region is not opened in META.
* If message(s) in 2 are missed, the master tries to reassign region. Causes duplicate assignment. Master asks the duplicate assigning RS to close region without reporting close using MSG\_REGION\_CLOSE\_WITHOUT\_REPORT.

**Closing a region:**

1. M -> RS : *MSG\_REGION\_CLOSE*
2. RS -> M : *MSG\_REPORT\_CLOSE*
3. M : offline region in META

Notes:

* If master dies between (1,3) region is closed but not recorded in META. This is ok because this close is a permanent close.

**Reassign a region:**

1. RS1 -> M : mostLoadedRegions with regionserver report
2. M -> RS1 : *MSG\_REGION\_CLOSE*
3. RS1 -> M : *MSG\_REPORT\_CLOSE*

*<see code to revisit>*

1. M : set region unassigned in META and in Master’s memory
2. M -> RS2 : *MSG\_REGION\_OPEN*
3. RS2 -> M : *MSG\_REPORT\_PROCESS\_OPEN [+]*
4. RS2 -> M : *MSG\_REPORT\_OPEN*
5. M : Add region info to meta

Notes:

* If master dies between (2,4) region is lost. It is closed but not marked as closed.
* If master dies between (5,8) region open is not recorded.
* Duplicate assignments possible as in opening a region

**Splitting:**

RS determines when to split

After split is done, the following is done by RS:

* Offline region in RS's memory
* Add parent offline, splitA and splitB info to parent region row in META
* Add splitA row to META
* Add splitB row to META

RS -> M : *MSG\_REPORT\_SPLIT\_INCLUDES\_DAUGHTERS* (oldRegion, regionA, regionB),

M : processSplitRegion, put regionA and region in regionsInTransition

M -> RS : *MSG\_REGION\_OPEN* regionA, region

Notes:

* This is handled properly today (feeling lazy to write an explanation)

**RS dies:**

ZK -> M : ServerManager.ServerExpirer is triggered

In master:

ServerManager.ServerExpirer is triggered which splits logs

master has a list of regions per RS

master goes through all these regions and marks them as offline in meta

meta scanner takes over the assignment of these

**Shutdown cluster:**

**Start up cluster:**

**Other misc scenarios:**

# 3. Code needs to be revisited:

Calculation of most loaded regions is done in RS and sent to master, but it does not really calculate most loaded regions.

  /\*\*

  \* Get the top N most loaded regions this server is serving so we can

  \* tell the master which regions it can reallocate if we're overloaded.

  \* TODO: actually calculate which regions are most loaded. (Right now, we're

  \* just grabbing the first N regions being served regardless of load.)

  \*/

RegionManager.java:1420

  int numRegionsToClose = balanceFromOverloaded(servLoad, avg);

  // check if we can unload server by low loaded servers

  if(numRegionsToClose <= 0) {

  numRegionsToClose = balanceToLowloaded(info.getServerName(), servLoad,

  avg);

  }

# 4. All the messages between master and the region server today:

// Message types sent from master to region server

/\*\* Start serving the specified region \*/

*MSG\_REGION\_OPEN*,

/\*\* Stop serving the specified region \*/

*MSG\_REGION\_CLOSE*,

/\*\* Split the specified region \*/

*MSG\_REGION\_SPLIT*, // This is user triggered

/\*\* Compact the specified region \*/

*MSG\_REGION\_COMPACT*, // This is user triggered

/\*\* Region server is unknown to master. Restart \*/

*MSG\_CALL\_SERVER\_STARTUP*, // Message comes in from an RS but we don’t know of it. Happens when DNS resolution changes, or long GC pause.

/\*\* Master tells region server to stop \*/

*MSG\_REGIONSERVER\_STOP*, // This shuts down the RS, sent after the quiesce

/\*\* Stop serving the specified region and don't report back that it's

\* closed

\*/

*MSG\_REGION\_CLOSE\_WITHOUT\_REPORT*, // This closes the region because of duplicate assignment

/\*\* Stop serving user regions \*/

*MSG\_REGIONSERVER\_QUIESCE*, // This asks RS to give up all regions EXCEPT meta. Need this because closing updates meta

// Message types sent from the region server to the master

/\*\* region server is now serving the specified region \*/

*MSG\_REPORT\_OPEN*,

/\*\* region server is no longer serving the specified region \*/

*MSG\_REPORT\_CLOSE*,

/\*\* region server is processing open request \*/

*MSG\_REPORT\_PROCESS\_OPEN*,

/\*\*

\* Region server split the region associated with this message.

\*

\* Note that this message is immediately followed by two MSG\_REPORT\_OPEN

\* messages, one for each of the new regions resulting from the split

\* **@deprecated** See MSG\_REPORT\_SPLIT\_INCLUDES\_DAUGHTERS

\*/

~~MSG\_REPORT\_SPLIT~~,

/\*\*

\* Region server is shutting down

\*

\* Note that this message is followed by MSG\_REPORT\_CLOSE messages for each

\* region the region server was serving, unless it was told to quiesce.

\*/

*MSG\_REPORT\_EXITING*,

/\*\* Region server has closed all user regions but is still serving meta

\* regions

\*/

*MSG\_REPORT\_QUIESCED*,

/\*\*

\* Flush

\*/

*MSG\_REGION\_FLUSH*,

/\*\*

\* Run Major Compaction

\*/

*MSG\_REGION\_MAJOR\_COMPACT*,

/\*\*

\* Region server split the region associated with this message.

\*

\* Its like MSG\_REPORT\_SPLIT only it carries the daughters in the message

\* rather than send them individually in MSG\_REPORT\_OPEN messages.

\*/

*MSG\_REPORT\_SPLIT\_INCLUDES\_DAUGHTERS*,