Design of HBase Isolation and Allocation

MOST OF THE BELOW HAS BEEN DONE BY HBASE-4120.
PROJECT LOCATION ON GIT: ENTRY.

Introduction

The HBase isolation and allocation tool is designed to help users manage cluster resource among different applications and tables.
When we have a large scale of HBase cluster with many applications running on it, there will be lots of problems. In Taobao there is a cluster for many departments to test their applications' performance, while these applications are based on HBase. With one cluster which has 12 servers, there will be only one application running exclusively on this server, and many other applications must wait until the previous test finished.
After we add allocation management function to the cluster, applications can share the cluster and run concurrently. Also if the Test Engineer wants to make sure that there is no interference, he/she can move out other tables from this group.
In groups we use table priority to allocate resource, when system is busy for example: we can make sure high-priority tables are not affected by lower-priority tables
Different groups can have different region server configurations, which means that some groups optimized for reading can have large block cache size, and others optimized for writing can have large memstore size.
Tables and region servers can be moved easily between groups. After changing the configuration, a group can be restarted alone instead of restarting the whole cluster.
Though we put group and priority of tables together, users can use them separately, because there is no inseparable link between these two functions.

Structure

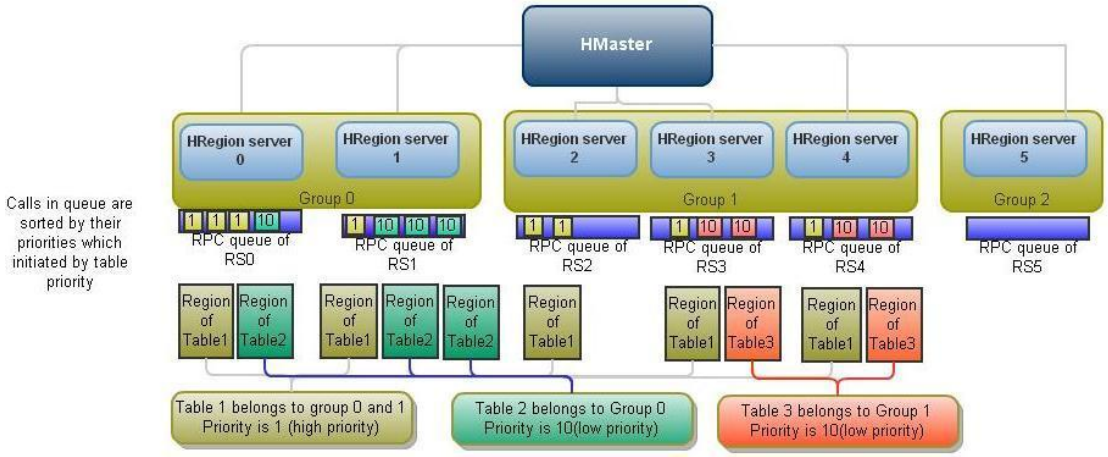

Figure 1: The System Organization

We put the server into groups, and the tables can be assigned to one or more groups. In groups (actually in region servers) requests of different tables are sorted by their priorities, which means that high priority table can take up more system resources. Tables and servers belong to different groups will not affect each other.

Rules

Default group which holds “-ROOT-“and “.META.” regions can’t be deleted or restarted, and the server holds “-ROOT-“and “.META.” regions can’t be moved into other groups.
Though when the servers in default groups are all down, system will find a random available server to hold “-ROOT-“and “.META.”, we suggest to keep some servers in default group. Because when some brutal actions like messed up the */{hbase_root}/groupinfomation.conf* file happen, if we find a table’s group has no server available, we will put this table into the default group.

If you want to delete a group, move tables and servers out of this group first, or the system will remind you “The group is not null”.

Group name must be a unique integer.

When a region server moves in or out a special configuration group, the server configuration will be updated.
If the table’s group attribute is “null”, its regions can be assigned to all groups which don’t have special configuration tag.

Table priority must be an integer and value between 1 and 10, while small number means high priority.

A table can use more than one group, where group attribute like “1,2” means this table’s regions can be assigned to group 1 and group 2.

Workflow of Background

In HBase isolation and allocation, *HMaster* uses *GroupAssignmentmanager* instead of *Assignmentmanager* to manage region assignment and *HRegionServer* uses *ScheduleHBaseServer* instead of *HBaseServer* to manage the RPC request.

Workflow of Group

Initiate configuration

After "-ROOT-"and ".META." regions are assigned, system will start to initiate the table-group information.
The information was stored in "*/{hbase_root}/groupinfomation.conf"*, this file contains the region server and group information.
Each time user changes the group information like creating a new group, moving region servers between groups and so on, the web portal will invoke *GroupAssignmentManager .initValue()* to refresh the group information cached in memory.

Startup and Assignment

When start the cluster, user regions will be assigned by *GroupAssignmentManager*'s *assignAllUserRegons()* method. In this method, it will invoke *retainGroupAssignRegions()* or *groupAssignRegions()* method to assign regions to servers. *retainGroupAssignRegions()* will check the assignment in meta table first,and if the region assignment fit the table's group information, the system will reuse the assignment, or system will use *getAvailableServer()* to get a list of available servers and choose a random one in it.

Group Maintenance

There is a maintainance thread which started in static block of *GroupAssignmentManager*. This thread will do two things: one is to refresh the group information cached in memory; the second is to check region assignment, if there is some illegal assignments this thread, it will reassign them.
GroupAssignmentManager .balance(RegionPlan) will check the group information first and then decide whether to put the plan in plan map.
GroupAssignmentManager .getRegionPlan(RegionState,HServerInfo,boolean) will get a random assignment in available servers which are chosen according to group information.

Change table's group attribute

When the table's group attribute changes, the table will be disabled first and then the group information will be writen into table descriptor, after meta information of this table is changed the system will refresh the table group

information which cached in memory. Then the table will be enabled.

Workflow of Table Priority

We use a priority job queue and different priority job handlers to achieve this effect.

Initiate region server handlers

When *ScheduleHBaseServer* began to start its job handlers, we give different thread priorities to the handlers in *ScheduleHBaseServer.startThreads()*. Handler threads can only handle the jobs which have equal or higher priority.

Handle RPC calls

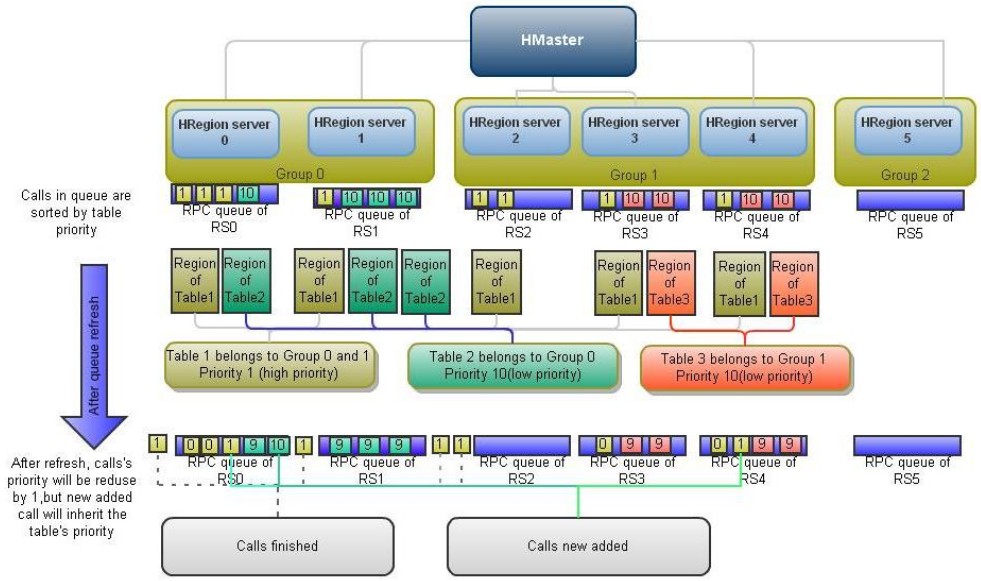

Figure 2: Refresh of RPC Queue

In *ScheduleHBaseServer.Connection.processData()*, the RPC calls will be added into a *ScheduleQueue*, the priority is decided by the region's priority cached in memory. If the cached priority is null, the system will retrieve the region's table descriptor to find the priority.

In *ScheduleQueue*, the calls are sorted by their priorities. So when the handler thread invokes the queue's *get()*, it will get RPC call with the highest priority.

When the number of times which one handler invoke the *ScheduleQueue .get()* reached a threshold value, the handler will interrupt the queue's refresh thread and the refresh thread will reduce the priorities of the calls which are in this queue by 1(small number means high priority), which will ensure the low priority calls to execute after a while of waiting.

When *ScheduleQueue*'s size exceeds its capacity, the thread which adds calls to the queue will wait for a while and check the size again. After the wait times exceeded the threshold, the call will be added regardless of the capacity.

Workflow of Web Portal

There are many JSP pages which are used to provide a GUI to manage the group division and table priorities. The detailed use cases of the GUI portal referred to the user guide which you can download from https://github.com/ICT-Ope/HBase_allocation/raw/master/doc/HBase%20allocation.pdf .

Move server between groups

The *showgroup.jsp* page will read group configuration from HDFS first. In this step, dead servers will be removed from the group, and new servers will be added to default group. This is achieved by functions *ServerWithGroup .readGroupInfo (HMaster master, final String confpath), ServerWithGroup.initGroupMap (HMaster master, String confline) and ServerWithGroup.initGroupPropertyMap (HMaster master, String confline).*

Before moving servers between groups, the system will check if this server contains "-ROOT-"and ".META." region, then it will construct a *MoveGroupPlan* instance, which contains region server name, original group name and target group name. The action of moving server is processed by *processgroup.jsp.*

If there is a *MoveGroupPlan*, this page will set system at the state of "busy" and start a thread to move region servers to other group. This is achieved by Class *ProcessMove.* It will check available servers in this group by *GroupAssignmentManager.getAvailableServer (tablename)* and move regions which belong to the *MoveGroupPlan's* region server to the available servers. Then it will check if the source group or the target group is special configured. If so we must update the server's configuration to the new one which includes hbase.jar, lib fold and conf fold. This is achieved by Class *MoveConfImpl* which contains 3 static methods.

public boolean ScpConf(String server, String command);
public boolean ImplRegionServer(String server, String command);
public boolean DistributeConf(String server, String command);

These methods will run local bash scripts to finish these operations. There are two bash scripts *moveremoteconf.sh a*nd *restartserver.sh.*

Load Balance

In this *groupinfo.jsp* page we provide an reference linked to *dogroupbalacne.jsp;* it will start a thread to balance all regions in this group. This function is implemented by *GroupAssignmentManager.balanceGroup (group).*
Then this page will list detail information of all tables in this group. It contains region numbers of the table, table priority, table group and an reference to *dotablebalacne.jsp.* When the balance reference is clicked, a thread used to balance all regions of this table will be started. The balance table function is implemented by *GroupAssignmentManager.balanceTable (table);*
You can change table priority and table group by click the button at the end of each row. It will start a thread to

invoke the methods *GroupAssignmentManager.setGroup (groups, tablename) or GroupAssignmentManager.setPriority(priporty, tablename);*

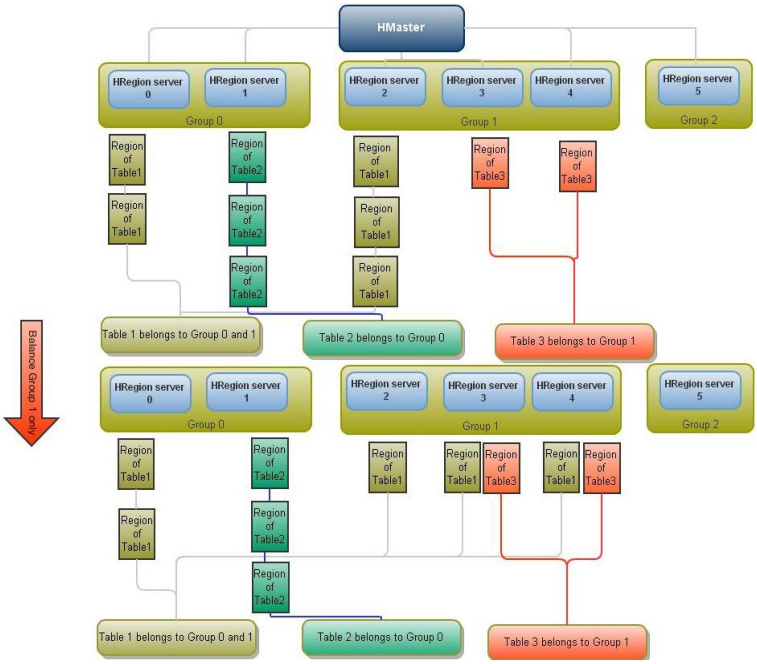

Figure 3: Balance One Group

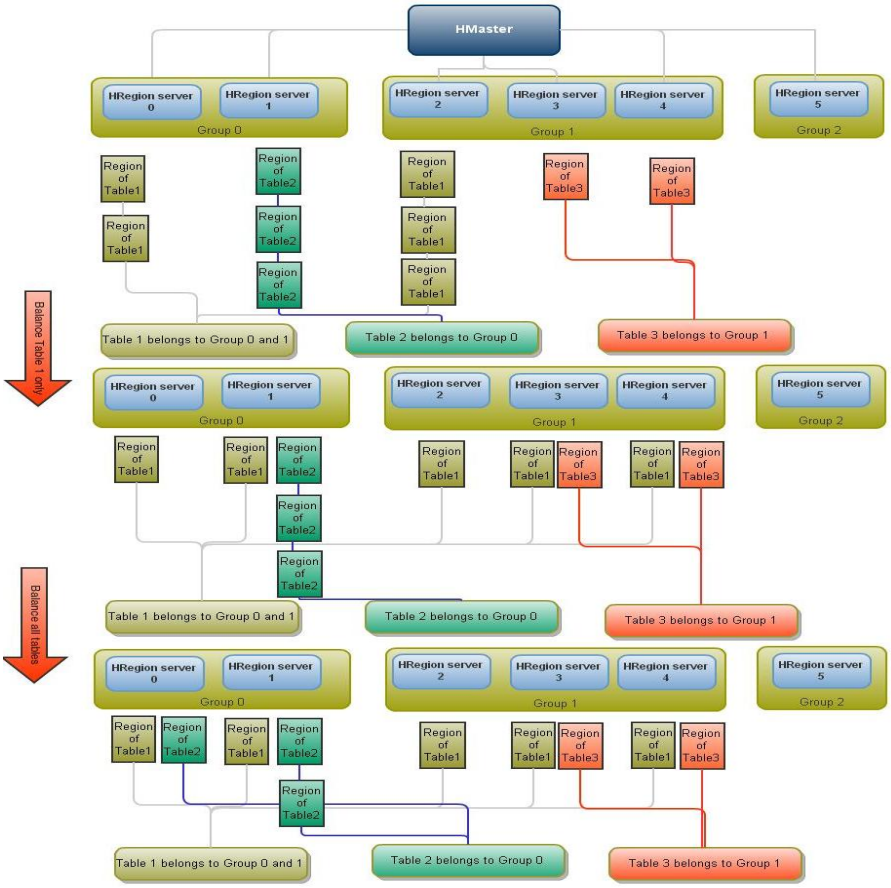

Figure 4: Balance Tables