Tuning of MapR-DB

Tuning MapR MapR - DB

In this article, I will show you how to tune MapR - DB from the system side.
In MapR - DB, tuning is possible on both the server side and the client side, but the client side is an architecture that operates efficiently by default, such as caching by punching the Puts instruction for each tablet.
For this reason, **tuning is generally not required on the client side,** but in this article we will introduce about server side and client side respectively.

Server side tuning

Memory of MFS

MapR - DB is a function implemented in the MapR Fileserver (MFS) process, and the size of the heap used by MapR - DB is the same as the heap size set in MFS.
The MFS heap size is set `warden.conf` within `service.command.mfs.heapsize.percent` , defaulting to 35% of the memory on the node.
Increasing this value and restarting warden is the quickest tuning.

`NODE_ALARM_HIGH_MFS_MEMORY` There is an alarm as a sign to inform that MFS heap size is starting to run short .
This alarm occurs when MFS secures the heap size beyond the set heap size.
If MFS releases the heap and falls below the threshold, the alarm disappears but if it continues to occur frequently afterwards, let's restart the warden by increasing the above heap size setting.

By the way, if you do not use MapR - DB, you can reduce the heap size setting to 35% -> 20% by choosing `configure.sh` the option `-noDB` .

Region size

Changing the size of the region (called tablet in MapR - DB) may work.
If the size of tablet is small, there is a high possibility that data will be placed on more nodes, so tablets with a small size may be advantageous when handling small tables.
On the other hand, if the size of tablet is large, the number of tablets that must be cached on the client side is reduced, and the number of RPC commands required when the Put buffer flushes can be reduced.
Therefore, if the size of the table is large, you may want to increase the number of tablets.

`maprcli table edit -regionsizemb` Commands are used to change the region size .

Change size of cache area for each purpose

First of all, this tuning needs to pay extra attention, so basically it is not recommended.
I will try to write it.

`mfs.conf` For the `mfs.cache.lru.sizes` parameters in the file, set the allocation for each data in the cache area.
Specifically, it is classified as follows.

Parameter	Description
inode	Cache area of file inode
meta	Cache area of metadata block such as intermediate BTree block and allocator block
dir	Directory and kvstore area
valc	Region for Gets command to be repeated
db	For MapR-DB memindex
small	Area for file pages smaller than 64 KB
large	Area for file pages larger than 64 KB. This area is not specified, and the value obtained by subtracting the above item is assigned

Regarding the actually applied size, it is `mfs.out` displayed as follows

```
HighLatencyTraceThresh: 3000 msCidtimeout: 3600 2018-07-12 15: 36: 32, 7066 mfs using default cachePercentages: inode: 10: log: 5: dir: 10: meta:
```

Unlike the purpose of this case, the mfs.conf document contains CLDB-only nodes, recommended values for nodes of the cluster that do not have CLDB running and do not use MapR-DB If you are running such a cluster / node, let's change it.

Parameters on the server side that indirectly affect the client side

There are the following in the parameter on the server side which may affect the performance of the client.
However, as there are few references in these documents regarding these parameters, be careful when using them.
The setting destination file is `/opt/mapr/asynchbase/asynchbase-<version>/conf/asynchbase.conf` .

Parameter	Contents	Default value
[illegible]	[illegible]	[illegible]
fs.mapr.async.callback.threads	Number of threads handling callbacks	Five

Client side tuning

As for the tuning on the client side, it is written here , so I will introduce you a bit more in this article.

On the client side, set the following parameters on the node `/opt/mapr/hbase/hbase-<version>/conf/hbase-site.xml` or on the node `/opt/mapr/hadoop/hadoop-<version>/etc/hadoop/core-site.xml` .
If there is a setting with a different value for the same parameter, then `hbase-site.xml` takes precedence.

Parameter	Contents	Default value
fs.mapr.tabletlru.size.kb	Size of the metadata cache used by all tables in the client application	512 KB
fs.mapr.threads	Number of threads used when flushing put buffer	64
db.mapr.putbuffer.threshold.mb	Size of cumulative put buffer used for all tablets in the client application	32MB

Parameter	Contents	Default value
db.mapr.putbuffer.threshold.sec	The time MapR-DB waits before flushing an idle put buffer	3 sec

in conclusion

In order to derive the performance of MapR - DB, the design of key and table will be the first thing, so the chance to try the contents introduced i n this article may not be much, but as a tuning when entering the operation stage already It might be good to try.

that's all.