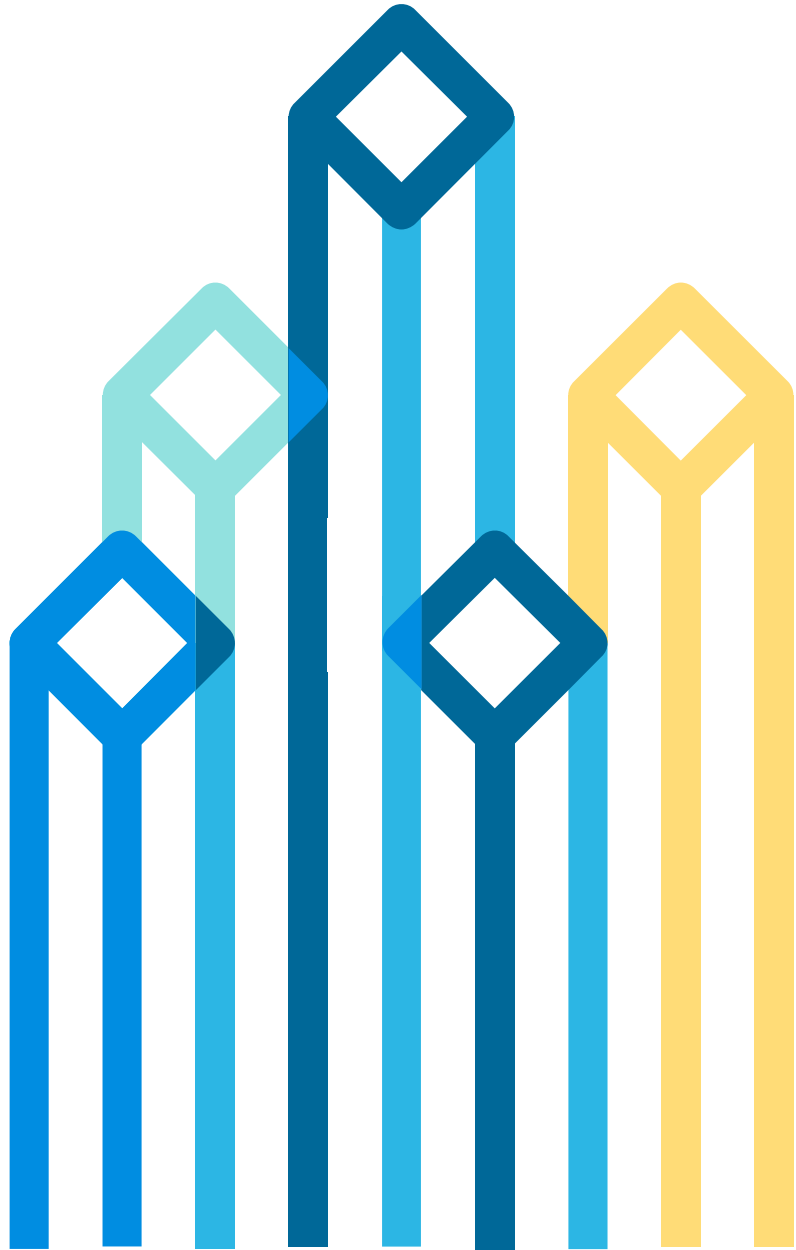
cloudera®



HBase Region Manager Tool (aka `Region Normalizer)

Rafael Arana

Introduction

- HBase automates some default management operations but this assumes that the key design follos best pracitec and data distribution is homogeneous across the set key.
- Some use cases implementation like time series with different access patterns for recent and old data are sometimes implemented in a different manner.
  - As example, for timeseries data can include the data as key (YYYYMM)
  - Tables can be created using a custom pre-split policy using three different stages (hot, warm & cold) based on the date included as first part of the HBASE key.
- This key design requires custom logic to split and merge regions.
- This document propose a custom approach to automate this activities
  - V1 was implemented as a Custom **RegionNormalizer** More info at JIRA - HBASE-13103.
  - In v2 the tool has been refactored to be used as a Command Line Tool to have from flexibility in the parametrization and scheduling.

cloudera®

Region Manager Tool

Region Manager Tool scope

- The Region Manager Tool can handle:
  - a non-uniform number of regions in three different stages: hot, warm and cold, being the data density & access patterns different per stage.
    - It assumes that they key will include as first part the year and month
    - The boundaries between the stages (hot, war & cold) will be defined by the number of months per stage.
    - The region tool can be run via command line and scheduled via tools like Control-M.
  - It can also manage tables without stages.

cloudera

Functionality

- It can be run per table, adjusting the logic and parameterization for each table.
- Current implementation supports the following criteria:

Property Name	Description
Max. size per regions (DEFAULT)	*Allows to set different data density per stage being higher for less access data (old or cold data with some occasional reads and no writes) and lower for daily access data with heavy reads & writes.*
Min. number of regions	Can be used to increase parallelization in those tables (stages) with a heavy write pattern
Max. number of regions (customizable per stage)	*Use to limit the number of regions per stage as a high number of regions per regions servers will cause performance degradations*
Average regions size	Based on HFILE storage size. This allows to adjust the initial presplit policy to real key distribution based on data size per stage

cloudera

cloudera®

Region Manager Tool

Usage

Limit Region by Size (in MB)

 # Show the valid arguments
./region-tool.sh –help

 # Use default planner (StagedMaxSizeRegionPlanner) with default properties
./region-tool.sh -tablename testtable

 # Use default planner (planner.StagedMaxSizeRegionPlanner) with default properties
./region-tool.sh -tablename testtable

 # Use default planner (planner.StagedMaxSizeRegionPlanner) with custom sizes
./region-tool.sh -tablename testtable -cold_max_size 5000 -warm_max_size 10000 -hot_max_size 15000

 # Use default planner (planner.SimpleMaxSizeRegionPlanner) without stages and default size (10 GB)
./region-tool.sh -tablename testtable –no_stages

 # Use default planner (planner.SimpleMaxSizeRegionPlanner) without stages and setting the size (in MB)
./region-tool.sh -tablename testtable –no_stages –max_size 15000

cloudera

Adjust Region Size to the Average

 # Use StagedMaxSizeRegionPlanner and StagedAverageSizeRegionPlanner with default properties (split factor = 2)
./region-tool.sh -tablename testtable -use_avg_size true

 # Use StagedMaxSizeRegionPlanner and StagedAverageSizeRegionPlanner with custom split factor
./region-tool.sh -tablename testtable -use_avg_size true -split_factor 3

 # Use StagedAverageSizeRegionPlanner *with default properties and disabling regions limit by size*
./region-tool.sh -tablename testtable -use_max_size false -use_avg_size true

 # Use SimpleMaxSizeRegionPlanner and SimpleAverageSizeRegionPlanner with default properties (split factor = 2)
./region-tool.sh -tablename testtable –no_stages -use_avg_size true

 # Use SimpleMaxSizeRegionPlanner with default properties (split factor = 2)
./region-tool.sh -tablename testtable -use_max_size false -use_avg_size true -split_factor 2 –no_stages

cloudera

Limit maximum number of regions

 # Use StagedMaxSizeRegionPlanner and StagedMaxNumberRegionPlanner with default properties
./region-tool.sh -tablename testtable -use_max_num true

 # Use StagedMaxSizeRegionPlanner and StagedMaxNumberRegionPlannerwith custom split factor
./region-tool.sh -tablename testtable -use_max_num true -max_hot_num 20 -max_warm_num 25 -max_cold_num 30

 # Use StagedMaxNumberRegionPlanner with default properties and disabling regions limit by size
./region-tool.sh -tablename testtable -use_max_size false -use_max_num true

 # Use SimpleMaxSizeRegionPlanner and SimpleMaxNumberRegionPlannerwith default properties (split factor = 2)
./region-tool.sh -tablename testtable –no_stages -use_max_num true

 # Use SimpleMaxNumberRegionPlanner with default properties (split factor = 2)
./region-tool.sh -tablename testtable -use_max_size false -use_max_num true–no_stages

cloudera

Setting a minimum number of regions

 # Use StagedMaxSizeRegionPlanner and StagedMinNumberRegionPlanner with default properties (split factor = 2)
./region-tool.sh -tablename testtable –use_min_num true

 # Use StagedMaxSizeRegionPlanner and StagedMinNumberRegionPlanner with custom properties
./region-tool.sh -tablename testtable –use_min_num true -min_cold_num 10 -min_warm_num 15 -min_hot_num 30

 # Use StagedAverageSizeRegionPlanner with default properties and disabling regions limit by size
./region-tool.sh -tablename testtable –use_max_size false -use_min_num true

 # Use SimpleMaxSizeRegionPlanner and SimpleMinNumberRegionPlanner with default properties
./region-tool.sh -tablename testtable –no_stages –use_min_num true

 # Use SimpleMinNumberRegionPlanner with custom properties
./region-tool.sh -tablename testtable -use_max_size false –use_min_num true -min_hot_num 30 -no_stages

cloudera

Bonus - Iterations

 # Enabling all
./region-tool.sh -tablename testtable –use_min_num true -use_max_num true -use_avg_size true

 # Run 3 iterations sleeping 30 secs between each iteration
./region-tool.sh -tablename testtable -iterations 2 -sleep 30

 # Debugging - Calculate & print the Normalization Plans but DON'T EXECUTE
./region-tool.sh -tablename testtable –report

 # Setting custom stage split boundaries
./region-tool.sh -tablename testtable -num_hot_months 8 -num_warm_months 12 –num_cold_months 12

cloudera

Which arguments can I use?

- Get the full list of argument with–h or –help to list all:

```
[rarana@rarana-hbase-1 ~]$ java -Dlog4j.configuration=tool-log4j.properties admin.RegionsTool -h
usage: java admin.RegionsTool <options>
Options:
 -cold_max_size <arg>   Max size per region at cold stage, in MB (defaults to 20 GB).
 -h,--help              Show usage
 -hot_max_size <arg>    Max.size per region at hot stage, in MB (defaults to 5 GB).
 -iterations <arg>      Number of iterations it will run the normalization process (defaults to 1).
 -max_cold_num <arg>    Number used to set a max number of regions in the cold stage. Default=10
 -max_hot_num <arg>     Number used to set a max number of regions in the hot stage. Default=10
 -max_warm_num <arg>    Number used to set a max number of regions in the warm stage. Default=10
 -min_cold_num <arg>    Number used to set a min number of regions in the cold stage. Default=3
 -min_hot_num <arg>     Number used to set a min number of regions in the hot stage. Default=3
 -min_warm_num <arg>    Number used to set a min number of regions in the warm stage. Default=3
 -plan_only             Disables plan execution. Only compute the normalization plans.
 -sleep <arg>           Number of seconds to sleep between iterations  (defaults to 300 secs)
 -split_factor <arg>    Factor used to split regions with size N times over the average (default=2).
 -tablename <arg>       Name of the table to normalize
 -use_avg_size <arg>    Computes the average size per stage. Use split_factor to customize.
 -use_max_num <arg>     Applies a minimum number of regions per stage. Set (max_cold_num,max_warm_num,max_hot_num)
                        customize.
 -use_max_size <arg>    [DEFAULT] Split the regions based on the max size per regions. Use the following args to set the
                        limits in MB: + cold_max_size:warm_max_size:hot_max_size
 -use_min_num <arg>     Applies a minimum number of regions per stage. Set (min_cold_num,min_warm_num,min_hot_num)
                        customize.
 -warm_max_size <arg>   Max size per region at warm stage, in MB (defaults to 10 GB).
```

cloudera

cloudera®

Region Manager Tool

Implementation Details

Implementation – The NormalizationPlanner

- Check the code at https://github.com/rafaelarana/HBase-RegionTool
- The tool is implemented in Java using the HBase Java API
  - Tested with
    - Java 1.7
    - HBase 1.2 CDH 5.x
  - Main class: admin.R*egionsTool*
  - The tool use Apache Commons CLI to parse input arguments

cloudera

Implementation – The NormalizationPlanner

- The different logic is implemented via *RegionPlanner,* can be enabled/disabled via command line per execution/table.
- Current implementation includes 4 types of planners (8 implementations)
  - (Simple|Staged)MaxSizeRegionPlanner (DEFAULT)
  - (Simple|Staged)MaxNumberRegionPlanner
  - (Simple|Staged)AverageSizeRegionPlanner
  - (Simple|Staged)MinNumberRegionPlanner
- All planners can be run in one iterations in the following order:
  - Max Size > Min Num > Max Num > Average Size

cloudera

Stage Split Points

- Stage split points are calculated based on the length of the different stages:
  - Hot Split Point(YYYYMM) = Current Month - hbase.normalizer.nonuniform.months.hot
  - Warm Split Point(YYYYMM) = Hot Split Point - hbase.normalizer.nonuniform.months.warm
  - Cold Split Point(YYYYMM) = Warm Split Point - hbase.normalizer.nonuniform.months.cold
- Properties set in RegionsTool via command line arguments or

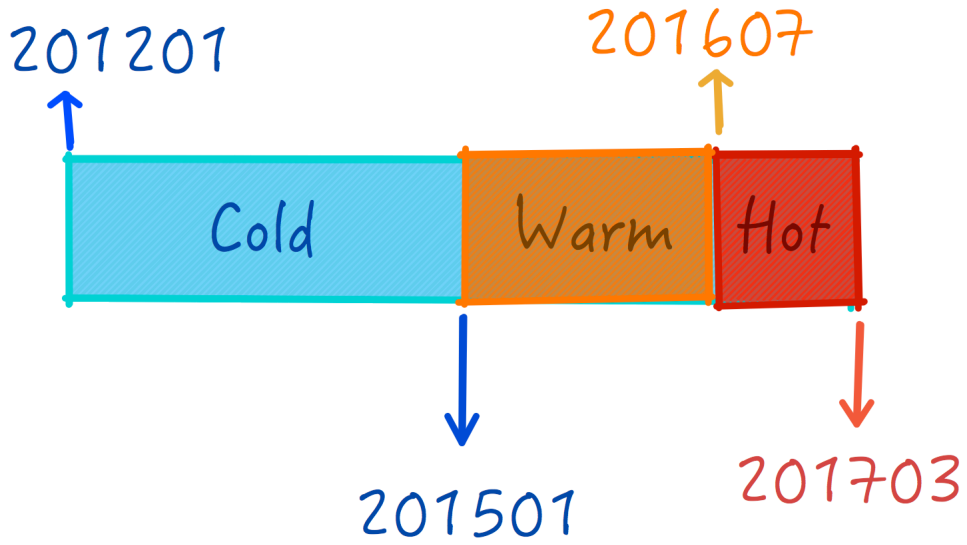
HBase Service Advanced Configuration Snippet (Safety Valve) for *hbase-site.xml*

Property Name	Description	Default Value
num_hot_months	*Length of the hot data interval in months*	8
num_warm_months	*Length of the warm data interval in months*	18
num_cold_months	*Length of the cold data interval in months*	36

cloudera

Stage Split Points Example

Property Name	Value
hbase.normalizer.nonuniform.months.hot	8
hbase.normalizer.nonuniform.months.warm	18
hbase.normalizer.nonuniform.months.cold	36



cloudera

Min Number of Regions Planner

- Implementations:
  - SimpleMinNumberRegionPlanner
  - StagedMinNumberRegionPlanner:
    - It can handle a non-uniform number of regions per stage.
    - It will adjust the # of regions to an different interval per stage.
- The number of regions needs to be decided based on the data density & access patterns different per stage.
- For each stage (hot, warm, cold) it will adjust dynamically the number of regions per table based on:.

cloudera

Stage min & max factor properties

Property Name	Description	Default Value
min_num	Number used to set a min number of regions	3
min_cold_num	Number used to set a min number of regions in the cold stage.	3
min_warm_num	Number used to set a min number of regions in the warm stage	3
min_hot_num	Number used to set a min number of regions in the hot stage	3
max_num	Number used to set a max number of regions	100
max_cold_num	Number used to set a max number of regions in the cold stage	100
max_warm_num	Number used to set a max number of regions in the warm stage	100
max_hot_num	Number used to set a max number of regions in the hotstage	100

cloudera

Average Planner

- Implementations: SimpleMinNumberRegionPlanner & StagedMinNumberRegionPlanner:
- Criteria to adjust the number of regions to the intervals per stage:
  - If the number of regions with a stage < min #Regions it will split the regions with SIZE (MB) > Average Size.
  - If the number of regions with a stage (Hot) > max #Regions it will merge contiguous regions (Ri, Rj) when the (Size of Region i +Size of Region j) < Average Size of all regions in that area.
  - If the number of regions is within the interval of that stage **it will normalize regions by size**:
    - Split those regions bigger than N times than the average.
    - Merge contiguous regions (Ri, Rj) when the (Size of Region i +Size of Region j) < Average Size of all regions in that stage

Property Name	Description	Default Value
split_factor	Factor used to split those regions with size N times over the average size for the stage interval.	2

cloudera

cloudera®

Region Manager Tool

Debug and Trouble Shooting

How to Debug ?

- Use the argument –report

./region-tool.sh -tablename testtable –report

- Review the logs
  - Implementation based in Apache Commons Logging / Log4j
  - Default log configuration at resources/ tool-log4j.properties
  - Default path at TOOL_HOME_DIR/log/
- Increase trace level

```
log4j.rootLogger=INFO, stdout, file

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
# Pattern to output the caller's file name and line number.
log4j.appender.stdout.layout.ConversionPattern=%d{yy/MM/dd
HH:mm:ss} %5p [%t] (%F:%L) %c{2}: %m%n

log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=../log/region-tool.log
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yy/MM/dd
HH:mm:ss} %5p [%t] (%F:%L) %c{2}: %m%n
log4j.appender.file.MaxFileSize=100MB
log4j.appender.file.MaxBackupIndex=5
log4j.appender.file.append=true

log4j.logger.admin=DEBUG
log4j.logger.org.apache.hadoop=WARN
log4j.logger.org.apache.zookeeper=ERROR
log4j.logger.org.apache.hadoop.hbase=WARN
```

cloudera

cloudera®

Packaging & Installation

Compilation & Packaging

- Maven file included with the project.

```
$ cd [HBase-Extensions-project]
$ mvn package
```

- 2 JAR file at [HBase-Extensions-project]/target
  - hbase-extensions-0.1-hbase1.2-cdh5.8.4.jar
  - hbase-extensions-0.1-hbase1.2-cdh5.8.4-jar-with-dependencies.jar

cloudera

Installation

- Copy the jar file:
  - Edge Nodes: *hbase-extensions-0.1-hbase1.2-cdh5.8.4.jar*
  - *External Nodes without Gateway Role:use the jar including the dependencies*
- *Update the environment dependencies in the shell scripts:*
  - export JAVA_HOME = /usr/java/jdk1.7.0_67-cloudera
  - export CDH_VERSION = cdh5.8.4