

KYLIN-5315 自动刷新快照 Dev Design

背景

Why

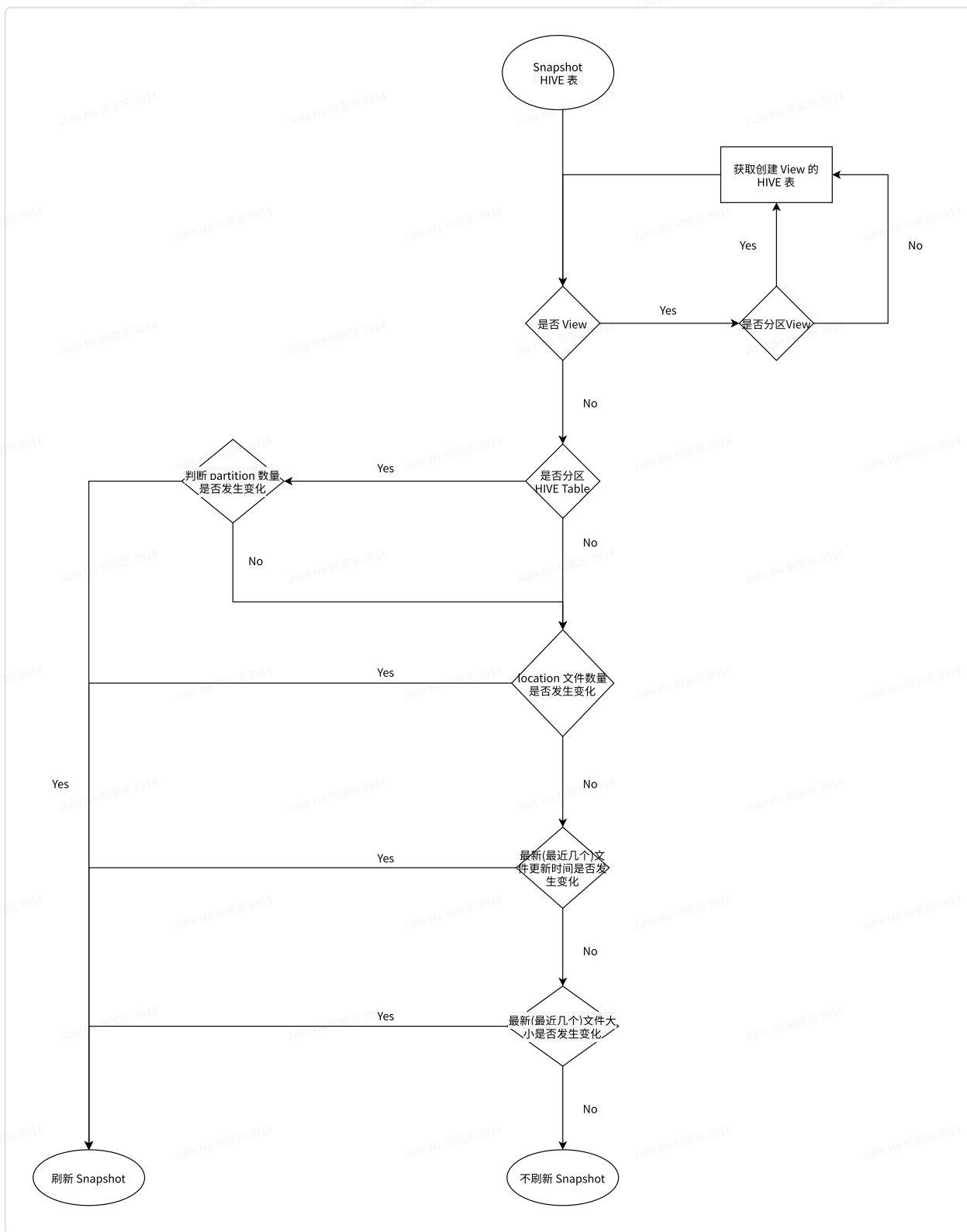
Kylin 系统中对于快照管理，有两种管理模式：系统自动管理、用户手动管理。默认是系统自动管理，可以通过在项目级开启快照管理来进入用户手动管理模式。

- 系统自动管理的优势是，用户操作简便，无需单独新建、刷新快照等。劣势是，会造成一定程度上的重复刷新，每次构建 Segment 时，都会重复刷新引用的快照，而且如果是快照是分区表，也会刷新全部分区。
- 用户手动管理的优势是，可以精准刷新快照，只需要在快照在底层数据源更新之后，主动触发刷新即可。对于分区表快照，也可以精准刷新部分分区。而缺点是管理成本高，**系统无法主动感知底层数据源刷新，需要用户主动触发**，往往需要与上游数据准备进行集成，数据源有更新或新增分区时，触发刷新快照 API。而如果不进行及时的刷新，快照数据可能滞后。

为了降低上述管理成本，希望 kylin 能自动感知快照底层 Hive 数据源的更新，并自动触发刷新快照，用户无需与上下游集成也能及时刷新快照。

详细设计(必填)

Hive 源表的变化判断依据



Snapshot 元数据

保存以下信息到 `hadoop_working_dir` 中, 文件名

- 用于保存 `location(table location / partition location)` 中:
文件数量, 最新或者最近几个文件的大小, 最新或者最近几个文件的更新时间

- HIVE Table 是 partition table 时, 保存所有的分区列信息

示例:

JSON

```
1  {
2    "location_path1":{
3      "create_time": 123,
4      "files_count": 12,
5      "files_size": [1,2,3],
6      "files_modification_time": [123,123,123]
7    },
8    "location_path2":{
9      "create_time": 1234,
10     "files_count": 1,
11     "files_size": [11,12,31],
12     "files_modification_time": [1234,1234,1234]
13   }
14 }
```

文件参数

参数	描述
create_time	Partition 的创建时间, 非分区表时为null
files_count	Location 地址中数据文件数量
files_size	最新或者最近几个文件的大小
files_modification_time	最新或者最近几个文件的更新时间

保存方式

- 定时轮询时, 判断需要刷新 snapshot 时, 更新 Snapshot 元数据 table_locations 文件
- 刷新 snapshot 任务完成时, 更新 Snapshot 元数据 table_locations 文件, 作为自动刷新的判断依据
 - 如果 Snapshot 元数据 table_locations 文件已存在, 则不进行刷新
- 地址:

Markdown

```
1 /working_dir
2 |--/${project_name}
3   |--/snapshot_auto_update
4     |--/_mark
5       |--/view_mapping
6         |--/source_table_stats
7           |--/${source_table_qualified_name}
8     |--/snapshot_job
9       |--/${snapshot_table_qualified_name}
```

HIVE Table

- 获取 Spark SessionCatalog(SparkerEnv.getSparkSession().sessionState().catalog())
- SessionCatalog 中获取 HIVE 表的元数据信息

View 表判断逻辑: **HIVE View**

非 View 表:

分区表判断逻辑: **HIVE Partitioned Table**

普通表: 获取表 Location

- 判断 HIVE 表 Location 中的文件数量和 snapshot 元数据中 file_count 是否一致
 - 文件数量不一致时, 需要刷新快照
 - 当文件数量一致时:
 - i. 获取最新的文件 (或者最近的几个) 的更新时间和 snapshot 元数据中 file_modification_time 是否一致, 不一致时, 需要刷新快照, 一致时, 比较文件大小
 - ii. 获取最新的文件 (或者最近的几个) 的大小和 snapshot 元数据中 file_size 是否一致, 不一致时, 需要刷新快照, 一致时, 无需刷新快照

HIVE Partitioned Table

- 获取 Spark SessionCatalog(SparkerEnv.getSparkSession().sessionState().catalog())
- SessionCatalog 中获取 HIVE 表 Parttition 信息(listParttitions)
- 判断 Parttition 数量和 snapshot 元数据中 table_locations 的大小是否一致
 - 不一致时, 刷新快照

- Parttition 数量和 snapshot 元数据中 table_locations 的大小一致时:
 - 判断 最新 (或者最近的几个) partttition Location path 在 snapshot 元数据 table_locations 中是否存在, 不存在时, 刷新快照
 - path 地址存在时: 判断 partition 中的文件
- 参考 **HIVE Table** 判断最新 (或者最近的几个) partttition 中的文件数量/更新时间/大小

Notice

当 partttition 数量很大, 探测的 patition 数值设置较小, 可能无法探测到 partttition 是否更新

示例:

Hive 表共有 100 个 partttition, 仅检测最新的一个 patition 是否存在更新时, 修改了更近较早的 partttition 中的文件, 此时无法探测到

HIVE View

此处默认 Kylin 的权限用户拥有权限查看/读取创建 View 的 HIVE table

现有逻辑 `SparkSqlUtil.getViewOrignalTables` 可直接使用, 获取 VIEW 的源表

获取到 HIVE Table 后, 判断是否存在更新

HIVE Partitioned View

与 HIVE View 处理方式一致

HIVE ACID Table

同 HIVE Table 一致, 获取 location 后, 进行判断

注意

当获取最新的文件(或者最近的几个)判断更新时间和文件大小时, 可能会消耗较大资源
若 snapshot 表 partition 或文件数很大, 此处可能会有性能瓶颈

定时轮询 HIVE 源表

新增配置/属性

配置	默认值	描述
kylin.snapshot.auto-refresh-enabled	false	快照是否自动刷新. 默认值为 false表示不自动刷新, 项目级配置
kylin.snapshot.auto-refresh-fetch-files-count	1	快照检测是否需要刷新时, fetch files 的数量. 默认值为 1, 表示只检测最新的一个文件
kylin.snapshot.auto-refresh-fetch-partitions-count	1	快照检测是否需要刷新时, fetch partitions 的数量. 默认值为 1, 表示只检测最新的 partition
kylin.snapshot.auto-refresh-max-concurrent-jobs	20	并发检测快照是否需要刷新的并发度,项目级配置
kylin.snapshot.first-auto-refresh-enabled	false	首次检测表信息后, 是否需要刷新快照 默认值为false, 不刷新快照
kylin.snapshot.null-location-auto-refresh-enabled	false	Table location 为空时, 是否刷新快照 默认值为false, 不刷新快照
kylin.snapshot.auto-refresh-task-timeout	30min	判断Hive表是否需要更新超时时间 默认值为 30min, 请求超过 30min 后中断本次请求
kylin.env.write-hdfs-working-dir	""	读写分离环境中, 写集群的 hdfs working dir, 用于获取写集群的 file system, 进行校验 tbale 中的数据文件
项目属性(proposal 1 不需要)	默认值	描述
snapshot_automatic_refresh_time_mode	DAY	快照刷新时间单位: 默认 DAY, 可选项: DAY, HOURS, MINUTE

snapshot_automatic_refresh_time_interval	1	快照刷新时间间隔: 默认值为1 单位配置为分钟时, 可配置值为 1~59; 单位为小时时, 可配置值为 1~23; 单位为天时, 可配置值为 >1。 只可配置整数, 不可配置小数。
snapshot_automatic_refresh_trigger_hours	0	单位为天时, 触发的时间点, 可配置值为 0~23 默认值为 0 表示 0 点触发
snapshot_automatic_refresh_trigger_minute	0	可配置值为 0~59 默认值为 0 表示 0 分触发
snapshot_automatic_refresh_trigger_second	0	可配置值为 0~59 默认值为 0 表示 0 秒触发

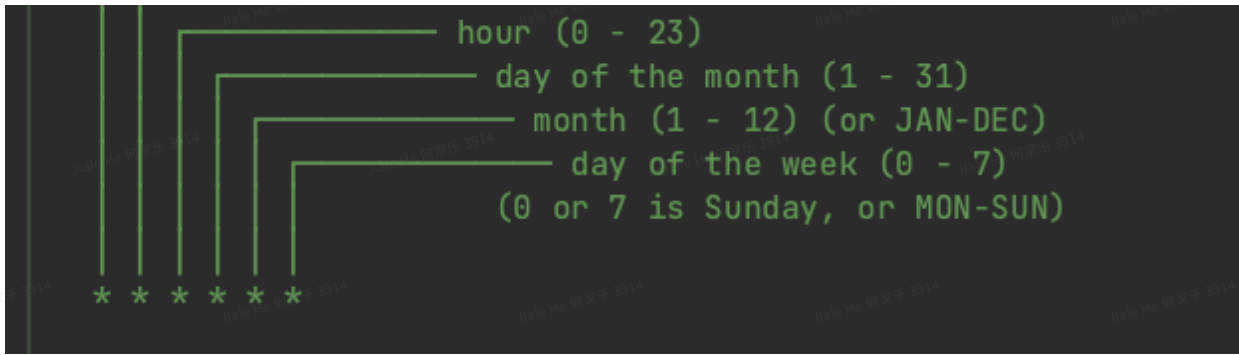
Proposal 1 使用 Cron 表达式

项目配置	默认值	描述
kylin.snapshot.auto-refresh-cron	"0 0 0 */1 * ?"	快照刷新频率 Cron , 通过页面选择参数生成

Cron 生成逻辑:

触发时间单位	选择范围	生成 Cron	描述
MINUTE	1 - 59	0 */x * * * ?	每 x 分钟执行一次
HOURS	1 - 23	0 0 */x * * ?	每 x 小时执行一次
DAY	> 1	u z y */x * ?	每月 1 号开始每 x 天 Y 时 z 分 u 秒执行一次

```
Parse the given crontab expression string into a CronExpression. The string has six single space-separated time and date fields:  
    _____ second (0-59)  
    |_____|  
    |_____| minute (0 - 59)
```



优点:

Spring 原生支持 Cron, 直接使用 spring scheduler 轮询

Cron 灵活, 扩展方便

缺点:

Spring Cron 对于 Day 级别支持为: day of the month , day of the week ; 不能支持传统意义上每 x 天运行一次, 此处使用的是 day of the month: 表示每月一号开始隔x天运行一次

Proposal 2 使用 Quartz 执行

项目配置	默认值	描述
kylin.snapshot.auto-refresh-time-mode	DAY	快照刷新时间单位: 默认 DAY, 可选项: DAY, HOURS, MINUTE
kylin.snapshot.auto-refresh-time-interval	1	快照刷新时间间隔: 默认值为1 单位配置为分钟时, 可配置值为 1~59; 单位为小时时, 可配置值为 1~23; 单位为天时, 可配置值为 >1。 只可配置整数, 不可配置小数。

优点:

可支持全部时间单位,可使用 quartz-scheduler 轮询

缺点:

针对需求开发

与现有定时逻辑保持一致, 使用 **Proposal 1 使用 Cron 表达式**

定时触发时间结果

Proposal 1 使用 Cron 表达式

单位为分钟

每 x 分钟执行一次

示例:

单位为分钟, 设置为 10

当前时间	第一次执行时间	下下一次执行时间	...
2202-08-08 18:18:18	2202-08-08 18:30:00	2202-08-08 18:40:00	...
2202-08-08 18:18:00	2202-08-08 18:20:00	2202-08-08 18:30:00	...
2202-08-31 23:58:50	2202-09-01 00:00:00	2202-09-01 00:10:00	...

单位为小时

每 x 小时执行一次

示例:

单位为小时, 设置为 2

当前时间	第一次执行时间	下下一次执行时间	...
2202-08-08 18:18:18	2202-08-08 20:00:00	2202-08-08 22:00:00	...

2202-08-08 19:18:00	2202-08-08 20:00:00	2202-08-08 22:00:00	...
2202-08-31 23:58:50	2202-09-01 00:00:00	2202-09-01 02:00:00	...

单位为天

频率为每月 1 号开始每 x 天 Y 时执行一次

示例:

单位为天, 设置为 2 , 触发时间设置为 15:15:15

当前时间	第一次执行时间	下下一次执行时间	...
2202-08-08 18:18:18	2202-08-09 15:15:15	2202-08-11 15:15:15	...
2202-08-07 18:18:00	2202-08-09 15:15:15	2202-08-11 15:15:15	...
2202-08-31 23:58:50	2202-09-01 15:15:15	2202-09-03 15:15:15	...
2202-08-08 11:18:18	2202-08-09 15:15:15	2202-08-11 15:15:15	...

Proposal 2 使用 Quartz 执行

单位为分钟

以当前时间为基准, 每 x 分钟执行一次

示例:

单位为分钟, 设置为 10

当前时间	第一次执行时间	下下一次执行时间	...
2202-08-08 18:18:18	2202-08-08 18:28:00	2202-08-08 18:38:00	...
2202-08-08 18:18:00	2202-08-08 18:28:00	2202-08-08 18:38:00	...
2202-08-31 23:58:50	2202-09-01 00:08:00	2202-09-01 00:18:00	...

单位为小时

以当前时间为基准, 每 x 小时执行一次

示例:

单位为小时, 设置为 2

当前时间	第一次执行时间	下下一次执行时间	...
2202-08-08 18:18:18	2202-08-08 20:00:00	2202-08-08 22:00:00	...
2202-08-08 19:18:00	2202-08-08 21:00:00	2202-08-08 23:00:00	...
2202-08-31 23:58:50	2202-09-01 01:00:00	2202-09-01 03:00:00	...

单位为天

以当前时间为基准, 从明天开始执行, 频率为每 x 天 Y 时执行一次

示例:

单位为天, 设置为 2, 触发时间设置为 15:15:15

当前时间	第一次执行时间	下下一次执行时间	...
2202-08-08 18:18:18	2202-08-09 15:15:15	2202-08-11 15:15:15	...
2202-08-07 18:18:00	2202-08-08 15:15:15	2202-08-10 15:15:15	...
2202-08-31 23:58:50	2202-09-01 15:15:15	2202-09-03 15:15:15	...
2202-08-08 11:18:18	2202-08-09 15:15:15	2202-08-11 15:15:15	...

定时轮询

项目设置 `kylin.snapshot.auto-refresh-enabled` 设置为 `true` 且开启 快照管理 时, 开启项目自动刷新快照

针对每个 project 创建一个 scheduler 进行轮询 snapshot 是否需要更新

在轮询任务中使用线程池(线程数由 `kylin.snapshot.auto-refresh-max-concurrent-jobs` 控制), 并行轮询, 提高并发度

使用 `spring @Scheduled` 轮询所有项目, 判断是否需要自动刷新

当自动刷新 snapshot 触发配置发生改变时, 修改 scheduler 的 trigger, 改变自动刷新快照的触发时间

刷新 snapshot 任务

刷新 snapshot

- 刷新前判断是否存在正在运行中的 snapshot 任务, 如果存在, 则取消此次自动刷新, 保留现有的 snapshot 任务
- 通过调用现有的刷新 sanpshot 的API

刷新逻辑

HIVE Table / HIVE View / HIVE Partitioned View

表发生变化时, 直接刷新 snapshot

HIVE Partitioned Table

表发生变化时, 获取对应的 partition, 刷新 partition

- 多级分区发生变化时, 获取对应的一级分区, 刷新一级分区 snapshot

API

新增内部使用 API

检测 Snapshot 源表数据

用于校验表的数据是否存在更新

仅非 Job 节点处理对应逻辑

URL: /api/snapshots/source_table_stats

参数: SnapshotTableLocationRequest

参数	描述
project	项目名
table	Table 名
database	Database 名
snapshot_partition_col	Snapshot 构建的分区字段名

返回信息: SnapshotTableLocationResponse

参数	描述
need_refresh	是否需要刷新
need_refresh_partitions_value	分区 Snapshot HIVE Table更新数据的分区

保存 view_mapping 文件

用于保存 snapshot 表为 view 时, view 表和源表的对应关系

仅非 Job 节点处理对应逻辑

URL: /api/snapshots/view_mapping

参数: SnapshotTableLocationRequest

参数	描述
project	项目名

返回信息: boolean

成功: true, 失败: false

刷新 Snapshot

用于cron 任务: 刷新快照

URL: /api/snapshots/automic

内部使用, 与 [快照管理 API · 刷新快照](#) 参数一致, 不需要认证, 可直接调用

页面交互 API

修改 snapshot 配置

URL: /api/projects/{project:.+}/snapshot_config

新增请求参数:

用于保存自动更新快照的配置信息

参数	默认值	描述
snapshot_automatic_refresh_enabled	false	快照是否自动刷新. 默认值为 false表示不自动刷新
snapshot_automatic_refresh_time_mod e	DAY	快照刷新时间单位: 默认 DAY, 可选项: DAY, HOURS, MINUTE
snapshot_automatic_refresh_time_inter val	1	快照刷新时间间隔: 默认值为1 单位配置为分钟时, 可配置值为 1~59; 单位为小时时, 可配置值为 1~23; 单位为天时, 可配置值为 1~31。 只可配置整数, 不可配置小数。
snapshot_automatic_refresh_trigger_ho urs	0	单位为天时, 触发的时间点, 可配置值为 0~23 默认值为 0 表示 0 点触发

snapshot_automatic_refresh_trigger_minute	0	可配置值为 0~59 默认值为 0 表示 0 分触发
snapshot_automatic_refresh_trigger_second	0	可配置值为 0~59 默认值为 0 表示 0 秒触发

获取 project config 信息

URL: /api/projects/{project:.+}/project_config

新增返回信息:

用于显示自动更新快照的配置信息

参数	默认值	描述
snapshot_automatic_refresh_enabled	false	快照是否自动刷新. 默认值为 false表示不自动刷新
snapshot_automatic_refresh_time_mode	DAY	快照刷新时间单位: 默认 DAY, 可选项: DAY, HOURS, MINUTE
snapshot_automatic_refresh_time_interval	1	快照刷新时间间隔: 默认值为1 单位配置为分钟时, 可配置值为 1~59; 单位为小时时, 可配置值为 1~23; 单位为天时, 可配置值为 >1。 只可配置整数, 不可配置小数。
snapshot_automatic_refresh_trigger_hour	0	单位为天时, 触发的时间点, 可配置值为 0~23 默认值为 0 表示 0 点触发
snapshot_automatic_refresh_trigger_minute	0	可配置值为 0~59 默认值为 0 表示 0 分触发
snapshot_automatic_refresh_trigger_second	0	可配置值为 0~59 默认值为 0 表示 0 秒触发

任务诊断包

当下载任务诊断包时:

- snapshot 任务

开启 snapshot 自动刷新时, 下载对应的 table_location 文件

- segment / index 构建任务

开启 snapshot 自动刷新时, 下载模型使用的相关 snapshot 的 table_location 文件

垃圾清理

在删除 project , 关闭 project 时, 由后台轮询线程控制删除垃圾文件
