How to use System Cube in Kylin 4

1. Background

System cube is a set of cubes created by Kylin for better self-monitoring, which is supported from Kylin-2.3.0.

In Kylin 3.x and Kylin 2.x, the built segment data is stored in HBase, so the query metrics collected by the system cube are basically related to HBase RPC; while Kylin 4 implements a new build and query engine, HBase storage has been replaced by the new parquet storage, and the original metrics no longer exist in Kylin 4.

In order to make the system cube work normally in kylin4 and help users monitor the build and query, we need to refine the new system cube query metrics, and the structure of the three query-related hive tables that the corresponding system cube depends on will also change.

After the system cube is enabled, every query or build operation in Kylin will be recorded in the hive table. There are five hive tables, which correspond to the fact tables of the five system cubes:

Hive Table Name	Description	System Cube Name
hive_metrics_query_execution_qa	Collect query level and spark execution level related metrics	KYLIN_HIVE_METRICS_QUERY_EXECUTION_QA
hive_metrics_query_spark_job_qa	Collect query spark job level related information	KYLIN_HIVE_METRICS_QUERY_SPARK_JOB_QA
hive_metrics_query_spark_stage_qa	Collect query spark stage level information	KYLIN_HIVE_METRICS_QUERY_SPARK_STAGE_QA
hive_metrics_job_qa	Collect job-related metrics	KYLIN_HIVE_METRICS_JOB_QA
hive_metrics_job_exception_qa	Collect job-related metrics	KYLIN_HIVE_METRICS_JOB_EXCEPTION_QA

2. Configuration

By default, the system cube is disabled. To enable system cube, you need to make the following configuration:

```
kylin.metrics.monitor-enabled=true
kylin.metrics.reporter-query-enabled=true
kylin.metrics.reporter-job-enabled=true
```

Generally, the system cube is used together with the Dashboard. You can do the following configuration to open the Dashboard:

```
kylin.web.dashboard-enabled=true
```

When Kylin 4 collects query-related metrics, it will temporarily save each query-related metrics record as a piece of data in the memory cache. When the storage time of the records in the cache exceeds the expiration time or the total number of records exceeds the maximum capacity, the records that need to be removed from the cache will be packaged in a certain format and passed to the metrics system. The expiration time and maximum capacity are determined by the following configurations. Their default values are 300 (seconds) and 10000 (pieces). You can find them in the "conf/kylin.properties" file. Modify their values:

```
kylin.metrics.query-cache.expire-seconds=300
kylin.metrics.query-cache.max-entries=10000
```

The records in the metrics system will be saved to HDFS after a certain period of time or a certain amount of time. Here, the default time is 10 (minutes), and the default number is 10. You can modify these two values by modifying the configuration item in the configuration file $KYLIN_HOME/tomcat/webapps/kylin/WEB-INF/classes/kylinMetrics.xml, the configuration item of "index = 1" indicates how many pieces of data will be inserted into "hive", and the configuration item of "index = 2" indicates how long it will be inserted into "hive", in minutes:

```
<!-- A Reservoir which staged metrics message in memory, and emit them in fixed rate. -->
<bean id="blockingReservoir" class="org.apache.kylin.metrics.lib.impl.BlockingReservoir">
    <!-- minReportSize, only if currently count of staged message exceed minReportSize, will Reservoir try to write message-->
    <constructor-arg index="0">
        <value>10</value>
    </constructor-arg>

    <!-- maxReportSize, max size of report in one time -->
    <constructor-arg index="1">
        <value>500</value>
    </constructor-arg>
```

3. Create the system cube

Before using the system cube, you need to prepare the hive table and cube mentioned in the above table. You can choose to **create the system cube manually** or **create automatically using system-cube.sh.**

3.2 Create system cube manually

Step1 Prepare the configuration file

Create a configuration file SCSinkTools.json in the $KYLIN_HOME directory. For example:

SCSinkTools.json

```
[
    {
       "sink": "hive",
       "storage_type": 4,
       "cube_desc_override_properties": {
         "kylin.cube.max-building-segments": "1"
       }
    }
]
```

Step2 Generate metadata

Run the following command in $KYLIN_HOME directory to generate related metadata:

Generate Metadata

```
./bin/kylin.sh org.apache.kylin.tool.metrics.systemcube.SCCreator \
-inputConfig SCSinkTools.json \
-output <output_forder>
```

With this command, the related metadata will be generated and its location is <output_forder>. The details are as follows, here's system_cube is our <output_forder>:

```
[root@cdh-worker-2 apache-kylin-4.0.0-SNAPSHOT-bin]# ll system_cube/
总用量 28
-rw-rw-r--. 1 root root 3954 1月  15 18:48 create_hive_tables_for_system_cubes.sql
drwxrwxr-x. 2 root root 4096 1月  15 18:48 cube
drwxrwxr-x. 2 root root 4096 1月  15 18:48 cube_desc
drwxrwxr-x. 2 root root 4096 1月  15 18:48 model_desc
drwxrwxr-x. 2 root root 4096 1月  15 18:48 project
drwxrwxr-x. 2 root root 4096 1月  15 18:48 table
-rw-rw-r--. 1 root root   38 1月   1 1970 UUID
```

Step3 Create hive tables

Run the following command to generate five hive tables in the above table:

create hive tables

```
hive -f <output_forder>/create_hive_tables_for_system_cubes.sql
```

By default, these tables will be created in the database named "kylin" in hive, and the default value can be modified through the configuration item "kylin.metrics.prefix".

```
hive> use kylin;
OK
Time taken: 0.009 seconds
hive> show tables;
OK
hive_metrics_job_exception_qa
hive_metrics_job_qa
hive_metrics_query_execution_qa
hive_metrics_query_spark_job_qa
hive_metrics_query_spark_stage_qa
Time taken: 0.014 seconds, Fetched: 5 row(s)
```

Step4 Restore metadata

Then we need to restore system cube metadata to Kylin metastore through the following command:

create system cube

```
bin/metastore.sh restore <output_forder>
```

Step5 Reload metadata

Finally, Reload metadata in Kylin Web UI and you can see a group of system cubes appear in the project named KYLIN_SYSTEM.

Step6 Build system cube regularly

After creating system cubes, as the metrics information is written into hive, these cubes need to be built regularly so that the metrics information written into hive can be quickly queried in Kylin.

You can use the following methods to build the system cube on a regular basis:

1 Create a shell script by calling org.apache.kylin.tool.job.CubeBuildingCLI to build the system cube. For example:

shell

```
#!/bin/bash

dir=$(dirname ${0})
export KYLIN_HOME=${dir}/../

CUBE=$1
INTERVAL=$2
DELAY=$3
CURRENT_TIME_IN_SECOND=`date +%s`
CURRENT_TIME=$((CURRENT_TIME_IN_SECOND * 1000))
END_TIME=$((CURRENT_TIME-DELAY))
END=$((END_TIME - END_TIME%INTERVAL))

ID="$END"
echo "building for ${CUBE}_${ID}" >> ${KYLIN_HOME}/logs/build_trace.log
sh ${KYLIN_HOME}/bin/kylin.sh org.apache.kylin.tool.job.CubeBuildingCLI --cube ${CUBE} --endTime ${END} >
${KYLIN_HOME}/logs/system_cube_${CUBE}_${END}.log 2>&1 &
```

2 Run the shell script on a regular basis. This can be achieved by adding a cron job as follows:

cron job

```
0 */2 * * * sh ${KYLIN_HOME}/bin/system_cube_build.sh KYLIN_HIVE_METRICS_QUERY_QA 3600000 1200000

20 */2 * * * sh ${KYLIN_HOME}/bin/system_cube_build.sh KYLIN_HIVE_METRICS_QUERY_CUBE_QA 3600000 1200000

40 */4 * * * sh ${KYLIN_HOME}/bin/system_cube_build.sh KYLIN_HIVE_METRICS_QUERY_RPC_QA 3600000 1200000

30 */4 * * * sh ${KYLIN_HOME}/bin/system_cube_build.sh KYLIN_HIVE_METRICS_JOB_QA 3600000 1200000

50 */12 * * * sh ${KYLIN_HOME}/bin/system_cube_build.sh KYLIN_HIVE_METRICS_JOB_EXCEPTION_QA 3600000 12000
```

3.2 Automatically create system cube

You can use ${KYLIN_HOME}/bin/system-cube.sh to help you automatically complete the above operations:

- Create System Cube`sh system-cube.sh setup`
- Build System Cube`sh bin/system-cube.sh build`
- Add scheduled build jobs to the system cube`bin/system.sh cron`