## Zabbix 数据库分区优化

目前支持mysql分区的zabbix版本有2.0 ,2.2, 2.4,3.0以及3.2，zabbix 2.0+版本的数据库表trends和history没有用外键，因此这些表支持分区。

1. 连接到数据库

mysql -h <database\_ip/fqdn> -u<user> -p

mysql> use zabbix;

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

1. 更改索引（zabbix 3.2版本跳过该步骤）

mysql> Alter table history\_text drop primary key, add index (id), drop index history\_text\_2, add index history\_text\_2 (itemid, id);

Query OK, 0 rows affected (0.49 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> Alter table history\_log drop primary key, add index (id), drop index history\_log\_2, add index history\_log\_2 (itemid, id);

Query OK, 0 rows affected (2.71 sec)

Records: 0 Duplicates: 0 Warnings: 0

1. 添加存储过程

DELIMITER $$

CREATE PROCEDURE `partition\_create`(SCHEMANAME varchar(64), TABLENAME varchar(64), PARTITIONNAME varchar(64), CLOCK int)

BEGIN

/\*

SCHEMANAME = The DB schema in which to make changes

TABLENAME = The table with partitions to potentially delete

PARTITIONNAME = The name of the partition to create

\*/

/\*

Verify that the partition does not already exist

\*/

DECLARE RETROWS INT;

SELECT COUNT(1) INTO RETROWS

FROM information\_schema.partitions

WHERE table\_schema = SCHEMANAME AND table\_name = TABLENAME AND partition\_description >= CLOCK;

IF RETROWS = 0 THEN

/\*

1. Print a message indicating that a partition was created.

2. Create the SQL to create the partition.

3. Execute the SQL from #2.

\*/

SELECT CONCAT( "partition\_create(", SCHEMANAME, ",", TABLENAME, ",", PARTITIONNAME, ",", CLOCK, ")" ) AS msg;

SET @sql = CONCAT( 'ALTER TABLE ', SCHEMANAME, '.', TABLENAME, ' ADD PARTITION (PARTITION ', PARTITIONNAME, ' VALUES LESS THAN (', CLOCK, '));' );

PREPARE STMT FROM @sql;

EXECUTE STMT;

DEALLOCATE PREPARE STMT;

END IF;

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE `partition\_drop`(SCHEMANAME VARCHAR(64), TABLENAME VARCHAR(64), DELETE\_BELOW\_PARTITION\_DATE BIGINT)

BEGIN

/\*

SCHEMANAME = The DB schema in which to make changes

TABLENAME = The table with partitions to potentially delete

DELETE\_BELOW\_PARTITION\_DATE = Delete any partitions with names that are dates older than this one (yyyy-mm-dd)

\*/

DECLARE done INT DEFAULT FALSE;

DECLARE drop\_part\_name VARCHAR(16);

/\*

Get a list of all the partitions that are older than the date

in DELETE\_BELOW\_PARTITION\_DATE. All partitions are prefixed with

a "p", so use SUBSTRING TO get rid of that character.

\*/

DECLARE myCursor CURSOR FOR

SELECT partition\_name

FROM information\_schema.partitions

WHERE table\_schema = SCHEMANAME AND table\_name = TABLENAME AND CAST(SUBSTRING(partition\_name FROM 2) AS UNSIGNED) < DELETE\_BELOW\_PARTITION\_DATE;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

/\*

Create the basics for when we need to drop the partition. Also, create

@drop\_partitions to hold a comma-delimited list of all partitions that

should be deleted.

\*/

SET @alter\_header = CONCAT("ALTER TABLE ", SCHEMANAME, ".", TABLENAME, " DROP PARTITION ");

SET @drop\_partitions = "";

/\*

Start looping through all the partitions that are too old.

\*/

OPEN myCursor;

read\_loop: LOOP

FETCH myCursor INTO drop\_part\_name;

IF done THEN

LEAVE read\_loop;

END IF;

SET @drop\_partitions = IF(@drop\_partitions = "", drop\_part\_name, CONCAT(@drop\_partitions, ",", drop\_part\_name));

END LOOP;

IF @drop\_partitions != "" THEN

/\*

1. Build the SQL to drop all the necessary partitions.

2. Run the SQL to drop the partitions.

3. Print out the table partitions that were deleted.

\*/

SET @full\_sql = CONCAT(@alter\_header, @drop\_partitions, ";");

PREPARE STMT FROM @full\_sql;

EXECUTE STMT;

DEALLOCATE PREPARE STMT;

SELECT CONCAT(SCHEMANAME, ".", TABLENAME) AS `table`, @drop\_partitions AS `partitions\_deleted`;

ELSE

/\*

No partitions are being deleted, so print out "N/A" (Not applicable) to indicate

that no changes were made.

\*/

SELECT CONCAT(SCHEMANAME, ".", TABLENAME) AS `table`, "N/A" AS `partitions\_deleted`;

END IF;

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE `partition\_maintenance`(SCHEMA\_NAME VARCHAR(32), TABLE\_NAME VARCHAR(32), KEEP\_DATA\_DAYS INT, HOURLY\_INTERVAL INT, CREATE\_NEXT\_INTERVALS INT)

BEGIN

DECLARE OLDER\_THAN\_PARTITION\_DATE VARCHAR(16);

DECLARE PARTITION\_NAME VARCHAR(16);

DECLARE OLD\_PARTITION\_NAME VARCHAR(16);

DECLARE LESS\_THAN\_TIMESTAMP INT;

DECLARE CUR\_TIME INT;

CALL partition\_verify(SCHEMA\_NAME, TABLE\_NAME, HOURLY\_INTERVAL);

SET CUR\_TIME = UNIX\_TIMESTAMP(DATE\_FORMAT(NOW(), '%Y-%m-%d 00:00:00'));

SET @\_\_interval = 1;

create\_loop: LOOP

IF @\_\_interval > CREATE\_NEXT\_INTERVALS THEN

LEAVE create\_loop;

END IF;

SET LESS\_THAN\_TIMESTAMP = CUR\_TIME + (HOURLY\_INTERVAL \* @\_\_interval \* 3600);

SET PARTITION\_NAME = FROM\_UNIXTIME(CUR\_TIME + HOURLY\_INTERVAL \* (@\_\_interval - 1) \* 3600, 'p%Y%m%d%H00');

IF(PARTITION\_NAME != OLD\_PARTITION\_NAME) THEN

CALL partition\_create(SCHEMA\_NAME, TABLE\_NAME, PARTITION\_NAME, LESS\_THAN\_TIMESTAMP);

END IF;

SET @\_\_interval=@\_\_interval+1;

SET OLD\_PARTITION\_NAME = PARTITION\_NAME;

END LOOP;

SET OLDER\_THAN\_PARTITION\_DATE=DATE\_FORMAT(DATE\_SUB(NOW(), INTERVAL KEEP\_DATA\_DAYS DAY), '%Y%m%d0000');

CALL partition\_drop(SCHEMA\_NAME, TABLE\_NAME, OLDER\_THAN\_PARTITION\_DATE);

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE `partition\_verify`(SCHEMANAME VARCHAR(64), TABLENAME VARCHAR(64), HOURLYINTERVAL INT(11))

BEGIN

DECLARE PARTITION\_NAME VARCHAR(16);

DECLARE RETROWS INT(11);

DECLARE FUTURE\_TIMESTAMP TIMESTAMP;

/\*

\* Check if any partitions exist for the given SCHEMANAME.TABLENAME.

\*/

SELECT COUNT(1) INTO RETROWS

FROM information\_schema.partitions

WHERE table\_schema = SCHEMANAME AND table\_name = TABLENAME AND partition\_name IS NULL;

/\*

\* If partitions do not exist, go ahead and partition the table

\*/

IF RETROWS = 1 THEN

/\*

\* Take the current date at 00:00:00 and add HOURLYINTERVAL to it. This is the timestamp below which we will store values.

\* We begin partitioning based on the beginning of a day. This is because we don't want to generate a random partition

\* that won't necessarily fall in line with the desired partition naming (ie: if the hour interval is 24 hours, we could

\* end up creating a partition now named "p201403270600" when all other partitions will be like "p201403280000").

\*/

SET FUTURE\_TIMESTAMP = TIMESTAMPADD(HOUR, HOURLYINTERVAL, CONCAT(CURDATE(), " ", '00:00:00'));

SET PARTITION\_NAME = DATE\_FORMAT(CURDATE(), 'p%Y%m%d%H00');

-- Create the partitioning query

SET @\_\_PARTITION\_SQL = CONCAT("ALTER TABLE ", SCHEMANAME, ".", TABLENAME, " PARTITION BY RANGE(`clock`)");

SET @\_\_PARTITION\_SQL = CONCAT(@\_\_PARTITION\_SQL, "(PARTITION ", PARTITION\_NAME, " VALUES LESS THAN (", UNIX\_TIMESTAMP(FUTURE\_TIMESTAMP), "));");

-- Run the partitioning query

PREPARE STMT FROM @\_\_PARTITION\_SQL;

EXECUTE STMT;

DEALLOCATE PREPARE STMT;

END IF;

END$$

DELIMITER ;

备注：以上代码可以保存成sql文件直接导入数据库：

mysql  -uroot -p'123456' zabbix < mysql\_partition.sql

1. 自定义过程

存储过程解释：

mysql>  CALL partition\_maintenance('<zabbix\_db\_name>', '<table\_name>', <days\_to\_keep\_data>, <hourly\_interval>, <num\_future\_intervals\_to\_create>)

zabbix\_db\_name：库名

table\_name：表名

days\_to\_keep\_data：保存多少天的数据

hourly\_interval：每隔多久生成一个分区

num\_future\_intervals\_to\_create：本次一共生成多少个分

自定义分区周期：

DELIMITER $$

CREATE PROCEDURE `partition\_maintenance\_all`(SCHEMA\_NAME VARCHAR(32))

BEGIN

CALL partition\_maintenance(SCHEMA\_NAME, 'history', 28, 24, 14);

CALL partition\_maintenance(SCHEMA\_NAME, 'history\_log', 28, 24, 14);

CALL partition\_maintenance(SCHEMA\_NAME, 'history\_str', 28, 24, 14);

CALL partition\_maintenance(SCHEMA\_NAME, 'history\_text', 28, 24, 14);

CALL partition\_maintenance(SCHEMA\_NAME, 'history\_uint', 28, 24, 14);

CALL partition\_maintenance(SCHEMA\_NAME, 'trends', 730, 24, 14);

CALL partition\_maintenance(SCHEMA\_NAME, 'trends\_uint', 730, 24, 14);

END$$

DELIMITER ;

备注：以上sql也可以保存成sql文件导入数据库：

mysql  -uroot -p'123456' zabbix < partition\_call.sql

五、设置定时任务：

每天执行调用一次存储过程，crontab设置

mysql -uzabbix -pzabbix zabbix -e "CALL partition\_maintenance\_all('zabbix');"

六、关闭housekeeper自动清理

Zabbix控制台 "Administration" -> "General"，选择housekeeper关闭

**注意事项：例子中分区创建了14天，当过了14天没有创建新的分区，数据将会丢失，因此，要定时去调用存储过程创建新的分区。**

**操作步骤：**

1. **修改自定义分区周期，可以用默认**
2. **执行脚本添加存储过程：bash +x mysql\_partitions.sh**
3. **Zabbix控制台暂停housekeeper**
4. **添加定时任务每日执行一次**mysql -uzabbix -pzabbix zabbix -e "CALL partition\_maintenance\_all('zabbix');"