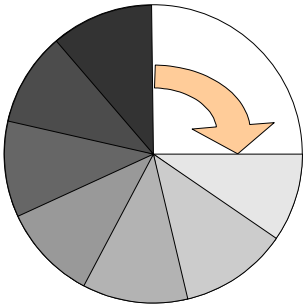
Round-Robin Database Storage Engine (RRD)

Oli Sennhauser (© GNU FDL)

Rebenweg 6
CH – 8610 Uster
Switzerland
oli.sennhauser@bluewin.ch

Introduction

In a round-robin database (RRD) usually time-series data like network bandwidth, temperatures, CPU load etc. is stored. The data is stored in the way that system storage footprint remains constant over time. This avoids resource expensive purge jobs and reduces complexity:



MySQL does NOT yet provide this kind of storage engine. Although some people were thinking about and some prototypes exists.
Nevertheless in this white paper it is shown how you can build your own RRD tables.

Implementation

Let's assume this is your table which you want to convert into a RRD table:

```
CREATE TABLE statistic (
    attribute_key     INT UNSIGNED    NOT NULL DEFAULT '0'
  , start_utime       INT UNSIGNED    NOT NULL DEFAULT '0'
  , end_utime         INT UNSIGNED             DEFAULT NULL
  , logging_interval  INT UNSIGNED    NOT NULL DEFAULT '0'
  , value             BIGINT UNSIGNED NOT NULL DEFAULT '0'
  , PRIMARY KEY (attribute_key, start_utime)
  , KEY start_time (start_utime)
)
;
```

What you have to do now is adding a rrd_key which is used to simulate the RRD behavior. You should also consider to chose the FIXED MySQL row format to avoid holes, increase speed and enable concurrent insert (in MyISAM). Let us assume we want to store 25 mio rows in this table:

```
CREATE TABLE statistic_rrd (
    rrd_key           INT UNSIGNED    NOT NULL AUTO_INCREMENT PRIMARY KEY
  , attribute_key     INT UNSIGNED    NOT NULL DEFAULT '0'
  , start_utime       INT UNSIGNED    NOT NULL DEFAULT '0'
  , end_utime         INT UNSIGNED             DEFAULT NULL
  , logging_interval  INT UNSIGNED    NOT NULL DEFAULT '0'
  , value             BIGINT UNSIGNED NOT NULL DEFAULT '0'
  , UNIQUE KEY (attribute_key, start_utime)
  , KEY start_time (start_utime)
) ROW_FORMAT = FIXED
, MAX_ROWS = 25000000
;
```

The you have to add a table where your rrd_key is stored and initialize the key:

```
CREATE TABLE statistic_rrd_key (
    rrd_key BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY
)
;

INSERT INTO statistic_rrd_key VALUES (0);
```

RRD logic

To simulate the RRD behavior you need a trigger on every INSERT. Do not forget to also set the number of rows here:

```
DROP TRIGGER IF EXISTS statistic_rrd_ins;

DELIMITER $$

CREATE TRIGGER  statistic_rrd_ins
BEFORE INSERT ON statistic_rrd
FOR EACH ROW
BEGIN

  SET @rrd_key = 0;
  SET @rows = 25000000;

  -- PK is NULL
  IF NEW.rrd_key = 0 THEN

    SELECT rrd_key + 1
      FROM statistic_rrd_key
      INTO @rrd_key;

    SET NEW.rrd_key = @rrd_key;
  END IF;

  IF (NEW.rrd_key % @rows) THEN
    SET NEW.rrd_key = NEW.rrd_key % @rows;
```

```
  ELSE
    SET NEW.rrd_key = @rows;
  END IF;

  UPDATE statistic_rrd_key SET rrd_key = NEW.rrd_key;
END;
$$

DELIMITER ;
```

Testing

Now we have to change all your INSERT statements into REPLACE and adapt all your UPDATE and DELETE statements to the new table structure and it should work as usual:

```
REPLACE INTO statistic_rrd
  (attribute_key, start_utime, end_utime, logging_interval, value)
 VALUES
  (ROUND(RAND()*100), UNIX_TIMESTAMP(NOW()), NULL, 100, 123456789)
;

SELECT * FROM statistic_rrd;

SELECT * FROM statistic_rrd_key;
```

Some performance metrics

With the following statement the table was filled:

```
REPLACE INTO statistic_rrd
    (attribute_key, start_utime, end_utime, logging_interval, value)
 VALUES
    (ROUND(RAND()*100000), UNIX_TIMESTAMP(NOW()), NULL, 100, 123456789)
;
```

The a file was created which contains 53248 rows. This file was running against the database. It results in around 50 to 53 thousand rows inserted.

A average insertion rate of around 600 INSERT/s was achieved (on a 1350 Mhz AMD Athlon with 1 CPU and 1 GB RAM, IDE disk 7200 rpm).