WHAT TAKES DISK SPACE

AIGARS KADIKIS
TECHNICAL SUPPORT ENGINEER

ZABBIX
WHAT TAKES DISK SPACE - AGENDA

- Tables
- Data types
- Items
- Hosts
- Tips
BIGGEST TABLES

history
history_uint

history_str
history_text
history_log

events
SELECT table_name, 
    table_rows, 
    data_length, 
    index_length, 
    round(((data_length + index_length) / 1024 / 1024 / 1024),2) "Size in GB"
FROM information_schema.tables
WHERE table_schema = "zabbix"
ORDER BY round(((data_length + index_length) / 1024 / 1024 / 1024),2) DESC
LIMIT 8;

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
<th>TABLE_ROWS</th>
<th>DATA_LENGTH</th>
<th>INDEX_LENGTH</th>
<th>Size in GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>history</td>
<td>39287666</td>
<td>2276048896</td>
<td>1389445120</td>
<td>3.41</td>
</tr>
<tr>
<td>history_uint</td>
<td>37597109</td>
<td>2179383296</td>
<td>1393065984</td>
<td>3.33</td>
</tr>
<tr>
<td>history_text</td>
<td>381535</td>
<td>871006208</td>
<td>22708224</td>
<td>0.83</td>
</tr>
<tr>
<td>trends_uint</td>
<td>2257508</td>
<td>163282944</td>
<td>0</td>
<td>0.15</td>
</tr>
<tr>
<td>trends</td>
<td>1149013</td>
<td>85590016</td>
<td>0</td>
<td>0.08</td>
</tr>
<tr>
<td>alerts</td>
<td>8196</td>
<td>4734976</td>
<td>1376256</td>
<td>0.01</td>
</tr>
<tr>
<td>items</td>
<td>7924</td>
<td>3686400</td>
<td>2424832</td>
<td>0.01</td>
</tr>
<tr>
<td>history_str</td>
<td>47559</td>
<td>5783552</td>
<td>1589248</td>
<td>0.01</td>
</tr>
</tbody>
</table>
SELECT *, pg_size_pretty(total_bytes) AS total, pg_size_pretty(index_bytes) AS index,
    pg_size_pretty(toast_bytes) AS toast, pg_size_pretty(table_bytes) AS table
FROM (SELECT *, total_bytes-index_bytes-coalesce(toast_bytes, 0) AS table_bytes
    FROM (SELECT c.oid,
                nspname AS table_schema,
                relname AS table_name,
                c.reltuples AS row_estimate,
                pg_total_relation_size(c.oid) AS total_bytes,
                pg_indexes_size(c.oid) AS index_bytes,
                pg_total_relation_size(reltoastrelid) AS toast_bytes
        FROM pg_class c
        LEFT JOIN pg_namespace n ON n.oid = c.relnamespace
        WHERE relkind = 'r' ) a) a;

In case if TimescalDB extension in place, it will tell the biggest hypertables (AKA chunks)
BUILD IN WAY TO TRACK INCOMING FLOW

<table>
<thead>
<tr>
<th>Wizard</th>
<th>Name</th>
<th>Triggers</th>
<th>Key</th>
<th>Interval</th>
<th>History</th>
<th>Trends</th>
<th>Type</th>
<th>Applications</th>
<th>Status</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>***</td>
<td>Template App Zabbix Server: Number of processed character values per second</td>
<td>zabbix[wcache.values.str]</td>
<td>6s</td>
<td>5d</td>
<td>365d</td>
<td>Zabbix internal</td>
<td>Zabbix write cache</td>
<td>Enabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>***</td>
<td>Template App Zabbix Server: Number of processed log values per second</td>
<td>zabbix[wcache.values.log]</td>
<td>6s</td>
<td>5d</td>
<td>305d</td>
<td>Zabbix internal</td>
<td>Zabbix write cache</td>
<td>Enabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>***</td>
<td>Template App Zabbix Server: Number of processed numeric (float) values per second</td>
<td>zabbix[wcache.values.float]</td>
<td>6s</td>
<td>5d</td>
<td>365d</td>
<td>Zabbix internal</td>
<td>Zabbix write cache</td>
<td>Enabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>***</td>
<td>Template App Zabbix Server: Number of processed numeric (unsigned) values per second</td>
<td>zabbix[wcache.values.uint]</td>
<td>6s</td>
<td>5d</td>
<td>365d</td>
<td>Zabbix internal</td>
<td>Zabbix write cache</td>
<td>Enabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>***</td>
<td>Template App Zabbix Server: Number of processed text values per second</td>
<td>zabbix[wcache.values.text]</td>
<td>6s</td>
<td>5d</td>
<td>365d</td>
<td>Zabbix internal</td>
<td>Zabbix write cache</td>
<td>Enabled</td>
<td></td>
</tr>
</tbody>
</table>
BIGGEST DATA COMING TO INSTANCE RIGHT NOW

SELECT hosts.host, items.itemid, items.key_,
COUNT(history_log.itemid) AS 'count', AVG(LENGTH(history_log.value)) AS 'avg size',
(COUNT(history_log.itemid) * AVG(LENGTH(history_log.value))) AS 'Count x AVG'
FROM history_log
JOIN items ON (items.itemid=history_log.itemid)
JOIN hosts ON (hosts.hostid=items.hostid)
WHERE clock > UNIX_TIMESTAMP(NOW()) - INTERVAL 30 MINUTE
GROUP BY hosts.host, history_log.itemid
ORDER BY 6 DESC
LIMIT 1

Possible tables to analyze history_text, history_log, history_str
```sql
SELECT value FROM history_log WHERE itemid=123780 LIMIT 1;
```

```sql
SET SESSION SQL_LOG_BIN=0; DELETE FROM history_log WHERE itemid=123780;
```
BIGGEST DATA PER HOST PER ONE DATA TYPE

```
SELECT ho.hostid, ho.name, count(*) AS records, 
(count(*)) * (SELECT AVG_ROW_LENGTH FROM information_schema.tables 
WHERE TABLE_NAME = 'history_text' and TABLE_SCHEMA = 'zabbix')/1024/1024) 
AS 'Total size average (Mb)', sum(length(history_text.value))/1024/1024 + 
sum(length(history_text.clock))/1024/1024 + sum(length(history_text.ns))/1024/1024 + 
sum(length(history_text.itemid))/1024/1024 AS 'history_text Column Size (Mb)'
FROM history_text 
LEFT OUTER JOIN items i on history_text.itemid = i.itemid 
LEFT OUTER JOIN hosts ho on i.hostid = ho.hostid 
WHERE ho.status IN (0,1) 
AND clock > UNIX_TIMESTAMP(now() - INTERVAL 1 DAY - INTERVAL 6 MINUTE) 
AND clock < UNIX_TIMESTAMP(now() - INTERVAL 1 DAY) 
GROUP BY ho.hostid ORDER BY 4 DESC LIMIT 5;
```

Repeat process with tables «history_log» and «history_str»
TRACK SIZE ON PARTITIONS (MYSQL)

Locate what is the data directory with command:

```
select @@datadir;
```

Most of times it prints ‘/var/lib/mysql’. In case the database name is ‘zabbix’, then we need to navigate:

```
cd /var/lib/mysql/zabbix
ls -lh history#*
ls -lh history_uint#*
ls -lh history_str#*
ls -lh history_text#*
ls -lh history_log#*
ls -lh trends#*
ls -lh trends_uint#*
```

```
[root@demo zabbix]# ls -lh history_log#*
-rw-r----- 1 mysql mysql  44M Jan 22 14:16 history_log#p#p2021_02w.ibd
-rw-r----- 1 mysql mysql  13M Jan 22 14:56 history_log#p#p2021_03w.ibd
```
ANALYZE A PARTITION (MYSQL)

```sql
SELECT ho.hostid, ho.name, count(*) AS records,
       (count(*)* (SELECT AVG_ROW_LENGTH FROM information_schema.tables
                   WHERE TABLE_NAME = 'history_log' and TABLE_SCHEMA = 'zabbix')/1024/1024)
       AS 'Total size average (Mb)', sum(length(history_log.value))/1024/1024 +
       sum(length(history_log.clock))/1024/1024 + sum(length(history_log.ns))/1024/1024 +
       sum(length(history_log.itemid))/1024/1024 AS 'history_log Column Size (Mb)'
FROM history_log PARTITION (p2021_02w)
LEFT OUTER JOIN items i on history_log.itemid = i.itemid
LEFT OUTER JOIN hosts ho on i.hostid = ho.hostid
WHERE ho.status IN (0,1)
GROUP BY ho.hostid ORDER BY 4 DESC LIMIT 10;
```

<table>
<thead>
<tr>
<th>hostid</th>
<th>name</th>
<th>records</th>
<th>Total size average (Mb)</th>
<th>history_log Column Size (Mb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10461</td>
<td>VMware host</td>
<td>117938</td>
<td>20.92024612</td>
<td>13.35941887</td>
</tr>
<tr>
<td>10084</td>
<td>Zabbix server</td>
<td>41535</td>
<td>7.36762047</td>
<td>7.34180927</td>
</tr>
</tbody>
</table>

Repeat process with tables «history_text» and «history_str»
It’s required to rebuild partitions. This can be a timeconsuming process:

```sql
SHOW CREATE TABLE history;
ALTER TABLE history REBUILD PARTITION p202101160000;
```

If there is not enough free disk space, we can crash the DB engine.
In the background it will copy the data from one partition to new partition.
It’s a lot of I/O operations.
Execute command through the «screen» utility.
TO FREE UP SPACE ON POSTGRES

Postgres is using autovacuum functionality. It is a separate process which cleans up dead tuples:

```sql
SELECT schemaname, relname, n_live_tup, n_dead_tup, last_autovacuum
FROM pg_stat_all_tables
WHERE n_dead_tup > 0
ORDER BY n_dead_tup DESC;
```

Query will tell how many dead tuples are in each table and when the last autovacuum has occurred.
TO FREE UP SPACE ON POSTGRES

If vacuum has not occurred in last 10 days, it's bad:

<table>
<thead>
<tr>
<th>schemaname</th>
<th>relname</th>
<th>n_live_tup</th>
<th>n_dead_tup</th>
<th>last_autovacuum</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>history_uint</td>
<td>1228819782</td>
<td>577423316</td>
<td>2020-08-13 04:55:11.54239-07</td>
</tr>
</tbody>
</table>

Must change settings and increase the priority for vacuum process:

- vacuum_cost_page_miss = 10
- vacuum_cost_page_dirty = 20
- autovacuum_vacuum_threshold = 50
- autovacuum_vacuum_scale_factor = 0.01
- autovacuum_vacuum_cost_delay = 20ms
- autovacuum_vacuum_cost_limit = 3000
- autovacuum_max_workers = 6
LOG FILE MONITORING

Log file monitoring has been confirmed per application.

However, when an application (JMX) encounters an error, it generates a lot of long lines per second.

Solution: use 'log.count' instead of log item and seek the occurrence of patterns.
‘Zabbix raw items’ works in tandem with dependable items.

A good template solution will use «a master item» and multiple «dependable items».

By default, a master item must have a historical period 0.

Sometimes for troubleshooting purpose we enable «History» for «Zabbix raw items».

It will put an unnecessary content in database.
QUESTIONS?

AIGARS KADIKIS
TECHNICAL SUPPORT ENGINEER
ZABBIX
THANK YOU!

AIGARS KADIKIS
TECHNICAL SUPPORT ENGINEER
ZABBIX