ISUCON 本ゆる輪読会 #4

Chapter5 データベースのチューニング

aoshima

2022 年 9 月 8 日

今日の資料

演習で使うから clone しておいてね！
https://github.com/aoshimash/techresi-isucon-workshop

準備

今日の演習は multipass で private-isu 環境を作成してそこで行います。

”../enshu/private-isu/README.md” の手順に沿って multipass で VM を作成してください。
（こっちからでも見えます。
https://github.com/aoshimash/techresi-isucon-workshop/ch5/enshu/private-isu/README.md）

データベースの種類

RDBMS SQL による問い合わせや、強い一貫性が特徴。
（例: MySQL, MariaDB, PostgreSQL, SQLite, RDS）

NoSQL リレーショナルデータベース以外のデータベースシステムの総称。スケールさせやすいものが多い。
（例: memcached, Redis, MongoDB, DynamoDB）

NewSQL RDBMS の特徴を持ったまま、スケールさせることができるらしい。使ったことないからよく知らない。
（例: Cloud Spanner, TiDB, Cockroach DB）

今日は RDMBS の話。

OS から負荷を観察する

htop で OS の負荷を観測しつつベンチマーカーを走らせてみる。

まずは、VM にログインして htop を起動しておく。

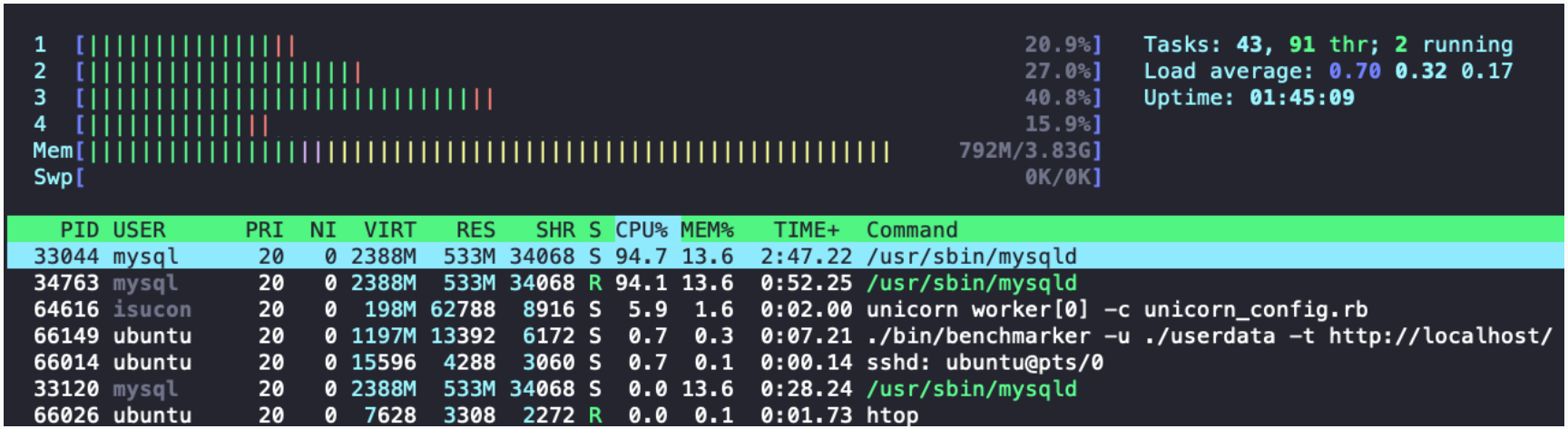
```
multipass shell private-isu
htop
```

ターミナルで別のタブを開いてベンチマーカーを実行。

```
multipass shell private-isu
cd /home/isucon/private_isu.git/benchmarker
./bin/benchmarker -u ./userdata -t http://localhost/
```

htop を起動しているタブに戻って OS の負荷を観測してみる

mysql の負荷が高いことがわかる



MySQL のプロセスリスト

今度は MySQL 上でどんなプロセスが動いているのか確認する。

またターミナルに新たなタブを作成し、mysql にログインする。

```
multipass shell private-isu
sudo mysql -u root -proot
mysql> SHOW PROCESSLIST;
```

ただし、MySQL へのアクセスがない状態だとプロセスリストを見てもあんまり面白くないので、ベンチマーカーを実行しながら、プロセスリストを確認してみよう。

```
mysql> SHOW PROCESSLIST;
+----+-----------------+-----------+---------+---------+------+------------------------+--------------------------------------------------------------------------------+
| Id | User            | Host      | db      | Command | Time | State                  | Info                                                                           |
+----+-----------------+-----------+---------+---------+------+------------------------+--------------------------------------------------------------------------------+
|  5 | event_scheduler | localhost | NULL    | Daemon  | 7190 | Waiting on empty queue | NULL                                                                           |
| 10 | isuconp         | localhost | isuconp | Execute |    0 | executing              | SELECT * FROM `comments` WHERE `post_id` = 9981 ORDER BY `created_at` DESC LIMIT 3 |
| 11 | root            | localhost | NULL    | Query   |    0 | init                   | SHOW PROCESSLIST                                                               |
+----+-----------------+-----------+---------+---------+------+------------------------+--------------------------------------------------------------------------------+
3 rows in set (0.00 sec)
```

SHOW PROCESSLIST が実行された瞬間の情報しかわからないので、大量のクエリを網羅的に解析するのには向いていない。

スロークエリログの出力設定の確認

まずは、現在のスロークエリログの出力設定を確認する。

```
mysql> SHOW GLOBAL VARIABLES LIKE 'slow_query_log';
+----------------+-------+
| Variable_name  | Value |
+----------------+-------+
| slow_query_log | OFF   |
+----------------+-------+
1 row in set (0.00 sec)
```

スロークエリログの出力設定

スロークエリログの出力設定をする。

”/etc/mysql/my.cnf” をエディタで開いて、次の内容を末尾に追記する。

```
[mysqld]
# スロークエリログを有効にする
slow_query_log = 1

# スロークエリログの出力先
slow_query_log_file = /var/log/mysql/mysql-slow.log

# 指定した秒数以上かかったクエリのみログに出力する(0なので全クエリをログに出力)
long_query_time = 0
```

ファイルを編集したら mysql を再起動して設定を反映する。

```
sudo systemctl restart mysql
```

スロークエリログの出力設定の再確認

スロークエリログの出力設定がちゃんと反映されているか確認する。

```
mysql> SHOW GLOBAL VARIABLES LIKE 'slow_query_log';
+----------------+-------+
| Variable_name  | Value |
+----------------+-------+
| slow_query_log | ON    |
+----------------+-------+
1 row in set (0.01 sec)

mysql> SHOW GLOBAL VARIABLES LIKE 'slow_query_log_file';
+---------------------+----------------------------------+
| Variable_name       | Value                            |
+---------------------+----------------------------------+
| slow_query_log_file | /var/log/mysql/mysql-slow.log    |
+---------------------+----------------------------------+
1 row in set (0.01 sec)

mysql> SHOW GLOBAL VARIABLES LIKE 'long_query_time';
+-----------------+----------+
| Variable_name   | Value    |
+-----------------+----------+
| long_query_time | 0.000000 |
+-----------------+----------+
1 row in set (0.00 sec)
```

スロークエリログを取得する

ベンチマーカーを回す前にログファイルをローテートして、

```
sudo mv /var/log/mysql/mysql-slow.log /var/log/mysql/mysql-slow.log.bak1
sudo systemctl restart mysql
```

”/home/isucon/private_isu.git/benchmarker” ディレクトリで以下を実行。

```
./bin/benchmarker -u ./userdata -t http://localhost/
```

そして、スロウクエリログファイルを分析した結果をファイルに出力する

```
sudo pt-query-digest /var/log/mysql/mysql-slow.log > ~/pt-query-digest.log2
less ~/pt-query-digest.log2
```

pt-query-log の結果の見方

pt-query-log の結果は大きく分けて 3 部構成になっており、上から全体的な統計、ランキング、各クエリの詳細になっている。まずはランキングに注目してみる。

```
# Profile
# Rank Query ID                           Response time Calls R/Call V/M   Item
# ==== ================================== ============= ===== ====== ===== ====
#    1 0x624863D30DAC59FA16849282... 48.1336 71.5%  1978 0.0243  0.00 SELECT comments
#    2 0x422390B42D4DD86C7539A5F4... 15.7837 23.5%  2008 0.0079  0.00 SELECT comments
#    3 0x1CD48AE21E9C97BE44D0B069...  0.7857  1.2%    79 0.0099  0.00 SELECT posts
# MISC 0xMISC                          2.5968  3.9% 32050 0.0001   0.0 <22 ITEMS>
```

”Response time” が実行時間の合計と全体に占める割合。”Calls” が実行された回数。”R/Call” が 1 回あたりの時間。次はランク 1 位のクエリの詳細を見ていく。

pt-query-log の結果の詳細

```
# Query 1: 27.47 QPS, 0.67x concurrency, ID 0x624863D30DAC59FA16849282195BE09F at byte 3059304
# This item is included in the report because it matches --limit.
# Scores: V/M = 0.00
# Time range: 2022-09-07T13:16:46 to 2022-09-07T13:17:58
# Attribute    pct   total     min     max     avg     95%  stddev  median
# ============ === ======= ======= ======= ======= ======= ======= =======
# Count          5    1978
# Exec time     71     48s    23ms    44ms    24ms    24ms   921us    23ms
# Lock time     21     3ms       0    25us     1us     1us       0     1us
# Rows sent      0   4.72k       0       3    2.44    2.90    1.13    2.90
# Rows examine  48 188.65M  97.66k  97.67k  97.66k  97.04k       0  97.04k
# Query size     2 158.81k      80      83   82.21   80.10    0.12   80.10
# String:
# Databases    isuconp
# Hosts        localhost
# Users        isuconp
# Query_time distribution
#   1us
#  10us
# 100us
#   1ms
#  10ms  ################################################################
# 100ms
#    1s
#  10s+
# Tables
#    SHOW TABLE STATUS FROM `isuconp` LIKE 'comments'\G
#    SHOW CREATE TABLE `isuconp`.`comments`\G
# EXPLAIN /*!50100 PARTITIONS*/
SELECT * FROM `comments` WHERE `post_id` = 10000 ORDER BY `created_at` DESC LIMIT 3\G
```

pt-query-log の結果の詳細

Count	解析対象期間中に実行されたクエリ数
Exec time	クエリの実行にかかった時間
Lock time	クエリの実行までにかかった時間。他のスレッドによるロックの待ち時間。
Rows sent	クエリを実行し、クライアントに返した行数。
Rows examine	クエリの実行に走査した行数
Query size	実行したクエリの長さ（文字数）

pt-query-log の結果の詳細

```
# Query 1: 27.47 QPS, 0.67x concurrency, ID 0x624863D30DAC59FA16849282195BE0
# This item is included in the report because it matches --limit.
# Scores: V/M = 0.00
# Time range: 2022-09-07T13:16:46 to 2022-09-07T13:17:58
# Attribute    pct   total     min     max     avg     95%  stddev  median
# ============ === ======= ======= ======= ======= ======= ======= =======
# Count          5    1978
# Exec time     71     48s    23ms    44ms    24ms    24ms   921us    23ms
# Lock time     21     3ms       0    25us     1us     1us       0     1us
# Rows sent      0   4.72k       0       3    2.44    2.90    1.13    2.90
# Rows examine  48 188.65M  97.66k  97.67k  97.66k  97.04k       0  97.04k
# Query size     2 158.81k      80      83   82.21   80.10    0.12   80.10
```

つまり、このクエリは 1978 回呼び出されており、合計で 48 秒かかっているが、Lock time は 3ms ほどで小さいということがわかります。そして、今回注目するところは、Rows sent と Rows examine の差です。クエリには”LIMIT 3” がついているので、1 クエリで最大でも 3 行しか返さないのに、平均して 97.66k 行も処理していることがわかります。

インデックスを付与する

一番遅いクエリがわかったので、これを改善していく。

```
SELECT * FROM `comments` WHERE `post_id` = 10000 ORDER BY `created_at` DESC LIMIT 3\G
```

comments テーブルから post_id で検索をするのが遅いのであれば、post_id にインデックスを貼れば解決するのではないかと推測ができる。

```
mysql> use isuconp;
mysql> ALTER TABLE `comments` ADD INDEX `post_id_idx`(`post_id`);
```

これで改善されたはず。

再度ベンチマーカーを回す

まずは、ログのローテーション

```
sudo mv /var/log/mysql/mysql-slow.log /var/log/mysql/mysql-slow.log.bak2
sudo systemctl restart mysql
```

”/home/isucon/private_isu.git/benchmarker” ディレクトリで以下を実行。

```
./bin/benchmarker -u ./userdata -t http://localhost/
```

そして、スロウクエリログファイルを分析した結果をファイルに出力する

```
sudo pt-query-digest /var/log/mysql/mysql-slow.log > ~/pt-query-digest.log3
less ~/pt-query-digest.log3
```

結果が改善されているか確認

```
# Profile
# Rank Query ID                           Response time Calls R/Call V/M   Item
# ==== ================================== ============= ===== ====== ===== ====
#    1 0x1CD48AE21E9C97BE44D0B069...  7.8391 38.6%   800 0.0098  0.00 SELECT posts
#    2 0xDA556F9115773A1A99AA0165...  2.1853 10.8% 89023 0.0000  0.00 ADMIN PREPARE
#    3 0xCDEB1AFF2AE2BE51B2ED5CF0...  1.4697  7.2%   180 0.0082  0.00 SELECT comments
#    4 0x19759A5557089FD5B718D440...  1.4651  7.2%  5421 0.0003  0.00 SELECT posts
#    5 0x396201721CD58410E070DA94...  1.3285  6.5% 41340 0.0000  0.00 SELECT users
#    6 0x009A61E5EEBD5A5E4097914B...  1.1869  5.9%    95 0.0125  0.00 INSERT posts
#    7 0x624863D30DAC59FA16849282...  0.8850  4.4% 19572 0.0000  0.00 SELECT comments
#    8 0x7A12D0C8F433684C3027353C...  0.8766  4.3%    78 0.0112  0.00 SELECT posts
#    9 0x422390B42D4DD86C7539A5F4...  0.8442  4.2% 20062 0.0000  0.00 SELECT comments
#   10 0x9F2038550F51B0A3AB05CA52...  0.6291  3.1%   156 0.0040  0.01 INSERT comments
#   11 0xE83DA93257C7B787C67B1B05...  0.5008  2.5%   180 0.0028  0.00 SELECT posts
#   12 0x26489ECBE26887E480CA8067...  0.4893  2.4%   155 0.0032  0.00 INSERT users
# MISC 0xMISC                         0.5891  2.9% 90262 0.0000   0.0 <15 ITEMS>
```

最初にランク 1 位だったクエリが 7 位まで落ちた！