MySQL (MariaDB и так далее) - тюним innodb_buffer_pool_size

Источники:

- Решение 1: http://dba.stackexchange.com/a/27341
- Решение 2: http://dba.stackexchange.com/a/19181
- Решение 3: https://www.percona.com/blog/2014/01/28/10-mysql-performance-tuning-settings-after-installation/

Решение 1

The biggest table you have makes up 16.47% (28/170) of the total data. Even if the table was highly written and highly read, not all 28G of the table is loaded in the buffer pool at one given moment. What you need to calculate is how much of the InnoDB Buffer Pool is loaded at any given moment on the current DB Server.

Here is a more granular way to determine innodb_buffer_pool_size for a new DB Server given the dataset currently loaded in the current DB Server's InnoDB Buffer Pool.

Run the following on your current MySQL Instance (server you are migrating from)

```
SHOW GLOBAL STATUS LIKE 'Innodb_buffer_pool_pages_data';  -- IBPDataPages
SHOW GLOBAL STATUS LIKE 'Innodb_buffer_pool_pages_total'; -- IBPTotalPages
SHOW GLOBAL STATUS LIKE 'Innodb_page_size'; -- IPS
```

Run the formula IBPPctFull = IBPDataPages * 100.0 / IBPTotalPages.

```
SET @IBPDataPages = (SELECT VARIABLE_VALUE FROM
information_schema.global_status WHERE VARIABLE_NAME =
'Innodb_buffer_pool_pages_data'); -- SELECT @IBPDataPages;
SET @IBPTotalPages = (SELECT VARIABLE_VALUE FROM
information_schema.global_status WHERE VARIABLE_NAME =
'Innodb_buffer_pool_pages_total'); -- SELECT @IBPTotalPages;
SET @IBPPctFull = CAST(@IBPDataPages * 100.0 / @IBPTotalPages AS
DECIMAL(5,2));
SELECT @IBPPctFull;
```

If IBPPctFull is 95% or more, you should set innodb_buffer_pool_size to 75% of the DB Server's RAM.

If IBPPctFull is less than 95%, run this formula : IBPSize = IPS X IBPDataPages / (1024*1024*1024) X 1.05.

```
SET @IBPSize = (SELECT VARIABLE_VALUE FROM information_schema.global_status
WHERE VARIABLE_NAME = 'Innodb_page_size'); -- SELECT @IBPSize;
SET @IBPDataPages = (SELECT VARIABLE_VALUE FROM
```

```
information_schema.global_status WHERE VARIABLE_NAME =
'Innodb_buffer_pool_pages_data'); -- SELECT @IBPDataPages;
SET @IBPSize = concat(ROUND(@IBPSize * @IBPDataPages / (1024*1024*1024) *
1.05, 2), ' GB' );
SELECT @IBPSize;
```

The number for IBPSize (in GB) is the number that more closely fits your actual working dataset.

Now, if IBPSize is still too big for the biggest Amazon EC2 RAM Config, use 75% of the RAM for the Amazon EC2 DB Server.

Решение 2

Here is what you should do. First run this query

```
SELECT CEILING(Total_InnoDB_Bytes*1.6/POWER(1024,3)) RIBPS FROM (SELECT
SUM(data_length+index_length) Total_InnoDB_Bytes
FROM information_schema.tables WHERE engine='InnoDB') A;
```

This will give you the RIBPS, Recommended InnoDB Buffer Pool Size based on all InnoDB Data and Indexes with an additional 60%.

For Example

```
mysql>     SELECT CEILING(Total_InnoDB_Bytes*1.6/POWER(1024,3)) RIBPS FROM
    ->     (SELECT SUM(data_length+index_length) Total_InnoDB_Bytes
    ->     FROM information_schema.tables WHERE engine='InnoDB') A;
+-------+
| RIBPS |
+-------+
|     8 |
+-------+
1 row in set (4.31 sec)

mysql>
```

With this output, you would set the following in /etc/my.cnf

```
[mysqld]
innodb_buffer_pool_size=8G
```

Next, service mysql restart

After the restart, run mysql for a week or two. Then, run this query:

```
SELECT (PagesData*PageSize)/POWER(1024,3) DataGB FROM (SELECT variable_value
PagesData FROM information_schema.global_status
WHERE variable_name='Innodb_buffer_pool_pages_data') A, (SELECT
```

```
variable_value PageSize FROM information_schema.global_status
WHERE variable_name='Innodb_page_size') B;
```

This will give you how many actual GB of memory is in use by InnoDB Data in the InnoDB Buffer Pool at this moment.

Решение 3

Here are 3 MySQL performance tuning settings that you should always look at. If you do not, you are very likely to run into problems very quickly.

innodb_buffer_pool_size: this is the #1 setting to look at for any installation using InnoDB. The buffer pool is where data and indexes are cached: having it as large as possible will ensure you use memory and not disks for most read operations. Typical values are 5-6GB (8GB RAM), 20-25GB (32GB RAM), 100-120GB (128GB RAM).

innodb_log_file_size: this is the size of the redo logs. The redo logs are used to make sure writes are fast and durable and also during crash recovery. Up to MySQL 5.1, it was hard to adjust, as you wanted both large redo logs for good performance and small redo logs for fast crash recovery. Fortunately crash recovery performance has improved a lot since MySQL 5.5 so you can now have good write performance and fast crash recovery. Until MySQL 5.5 the total redo log size was limited to 4GB (the default is to have 2 log files). This has been lifted in MySQL 5.6.

Starting with innodb_log_file_size = 512M (giving 1GB of redo logs) should give you plenty of room for writes. If you know your application is write-intensive and you are using MySQL 5.6, you can start with innodb_log_file_size = 4G.

max_connections: if you are often facing the ‘Too many connections’ error, max_connections is too low. It is very frequent that because the application does not close connections to the database correctly, you need much more than the default 151 connections. The main drawback of high values for max_connections (like 1000 or more) is that the server will become unresponsive if for any reason it has to run 1000 or more active transactions. Using a connection pool at the application level or a thread pool at the MySQL level can help here.

mysql, mariadb, производительность, innodb buffer pool size