How large should be mysql innodb_buffer_pool_size?

I have a busy database with solely InnoDB tables which is about 5GB in size. The database runs on a Debian server using SSD disks and I've set max connections = 800 which sometimes saturate and grind the server to halt. The average query per second is about 2.5K. So I need to optimize memory usage to make room for maximum possible connections.

I've seen suggestions that innodb_buffer_pool_size should be up to %80 of the total memory. On the other hand I get this warning from tuning-primer script:

```
Max Memory Ever Allocated : 91.97 G
Configured Max Per-thread Buffers : 72.02 G
Configured Max Global Buffers : 19.86 G
Configured Max Memory Limit : 91.88 G
Physical Memory : 94.58 G
```

Here are my current innodb variables:

```
| innodb_adaptive_flushing                          | ON
|
| innodb_adaptive_hash_index                        | ON
|
| innodb_additional_mem_pool_size                   | 20971520
|
| innodb_autoextend_increment                       | 8
|
| innodb_autoinc_lock_mode                          | 1
|
| innodb_buffer_pool_instances                      | 1
|
| innodb_buffer_pool_size                           | 20971520000
|
| innodb_change_buffering                           | all
|
| innodb_checksums                                  | ON
|
| innodb_commit_concurrency                         | 0
|
| innodb_concurrency_tickets                        | 500
|
| innodb_data_file_path                             | ibdata1:10M:autoextend
|
| innodb_data_home_dir                              |
|
| innodb_doublewrite                                | ON
|
| innodb_fast_shutdown                              | 1
|
| innodb_file_format                                | Antelope
|
| innodb_file_format_check                          | ON
|
| innodb_file_format_max                            | Antelope
|
| innodb_file_per_table                             | ON
|
| innodb_flush_log_at_trx_commit                    | 2
|
| innodb_flush_method                               | O_DIRECT
|
| innodb_force_load_corrupted                       | OFF
|
| innodb_force_recovery                             | 0
|
| innodb_io_capacity                                | 200
|
| innodb_large_prefix                               | OFF
|
| innodb_lock_wait_timeout                          | 50
|
| innodb_locks_unsafe_for_binlog                    | OFF
```

```
|
| innodb_log_buffer_size                              | 4194304
|
| innodb_log_file_size                                | 524288000
|
| innodb_log_files_in_group                           | 2
|
| innodb_log_group_home_dir                           | ./
|
| innodb_max_dirty_pages_pct                          | 75
|
| innodb_max_purge_lag                                | 0
|
| innodb_mirrored_log_groups                          | 1
|
| innodb_old_blocks_pct                               | 37
|
| innodb_old_blocks_time                              | 0
|
| innodb_open_files                                   | 300
|
| innodb_purge_batch_size                             | 20
|
| innodb_purge_threads                                | 0
|
| innodb_random_read_ahead                            | OFF
|
| innodb_read_ahead_threshold                         | 56
|
| innodb_read_io_threads                              | 4
|
| innodb_replication_delay                            | 0
|
| innodb_rollback_on_timeout                          | OFF
|
| innodb_rollback_segments                            | 128
|
| innodb_spin_wait_delay                              | 6
|
| innodb_stats_method                                 | nulls_equal
|
| innodb_stats_on_metadata                            | ON
|
| innodb_stats_sample_pages                           | 8
|
| innodb_strict_mode                                  | OFF
|
| innodb_support_xa                                   | ON
|
| innodb_sync_spin_loops                              | 30
|
| innodb_table_locks                                  | ON
|
| innodb_thread_concurrency                           | 4
|
| innodb_thread_sleep_delay                           | 10000
|
| innodb_use_native_aio                               | ON
|
| innodb_use_sys_malloc                               | ON
|
| innodb_version                                      | 1.1.8
|
| innodb_write_io_threads                             | 4
|
```

A side note that might be relevant: I see that when I try to insert a large post (say over 10KB) from Drupal (which sits on a separate web server) to database, it lasts forever and the page does not return correctly.

Regarding these, I'm wondering what should be my innodb_buffer_pool_size for optimal performance. I appreciate your suggestions to set this and other parameters optimally for this scenario.

mysql innodb

edited Oct 21 '12 at 15:21

asked Oct 21 '12 at 15:03
alfish
769 5 11 17

4 Answers

Your **innodb_buffer_pool_size** is enormous. You have it set at `20971520000` . That's 19.5135 GB. If you only 5GB of InnoDB data and indexes, then you should only have about 8GB. Even this may too high.

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.

I have written about this before : What to set innodb_buffer_pool and why..?

You could just run this `DataGB` query right now rather than reconfiguring, restarting and waiting a week.

This value `DataGB` more closely resembles how big the InnoDB Buffer Pool should be + (percentage specified in innodb_change_buffer_max_size). I am sure this will be far less than the 20000M you have reserved right now. The savings in RAM can be used for tuning other things like

- join_buffer_size
- sort_buffer_size
- read_buffer_size
- read_rnd_buffer_size
- max_connection

CAVEAT #1

This is very important to note : At times, InnoDB may require an additional 10% over the value for the **innodb_buffer_pool_size**. Here is what the MySQL Documentation says on this:

> The larger you set this value, the less disk I/O is needed to access data in tables. On a dedicated database server, you may set this to up to 80% of the machine physical memory size. Be prepared to scale back this value if these other issues occur:
>
> Competition for physical memory might cause paging in the operating system.
>
> InnoDB reserves additional memory for buffers and control structures, so that the total allocated space is approximately 10% greater than the specified size.
>
> The address space must be contiguous, which can be an issue on Windows systems with DLLs that load at specific addresses.
>
> The time to initialize the buffer pool is roughly proportional to its size. On large installations, this initialization time may be significant. For example, on a modern Linux x86_64 server, initialization of a 10GB buffer pool takes approximately 6 seconds. See Section 8.9.1, "The InnoDB Buffer Pool".

CAVEAT #2

I See the following values in your `my.cnf`

```
| innodb_io_capacity                              | 200 |
| innodb_read_io_threads                          | 4   |
| innodb_thread_concurrency                       | 4   |
| innodb_write_io_threads                         | 4   |
```

These number will impede InnoDB from accessing multiple cores

Please set the following:

```
[mysqld]
innodb_io_capacity = 2000
innodb_read_io_threads = 64
innodb_thread_concurrency = 0
innodb_write_io_threads = 64
```

I have written about this before in the DBA StackExchange

- `May 26, 2011` : About single threaded versus multithreaded databases performance
- `Sep 12, 2011` : Possible to make MySQL use more than one core?
- `Sep 20, 2011` : Multi cores and MySQL Performance

I just *answered a question like this in ServerFault using a more concise formula*:

```
SELECT CONCAT(CEILING(RIBPS/POWER(1024,pw)),SUBSTR(' KMGT',pw+1,1))
Recommended_InnoDB_Buffer_Pool_Size FROM
(
    SELECT RIBPS,FLOOR(LOG(RIBPS)/LOG(1024)) pw
    FROM
    (
        SELECT SUM(data_length+index_length)*1.1*growth RIBPS
        FROM information_schema.tables AAA,
        (SELECT 1.25 growth) BBB
        WHERE ENGINE='InnoDB'
    ) AA
) A;
```

edited Apr 13 at 12:43
Community ♦
1

answered Oct 22 '12 at 0:30
RolandoMySQLDBA
122k 17 172 319

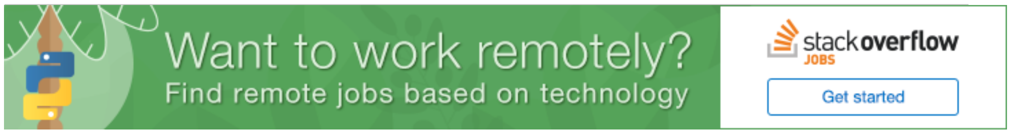
Thanks for this great post! Your formula starting with `SELECT (PagesData*PageSize)/POWER(1024,3) DataGB FROM...` generates the following error on MySQL 5.7: "*The 'INFORMATION_SCHEMA.GLOBAL_STATUS' feature is disabled; see the documentation for 'show_compatibility_56'*". Would you have an updated version by any chance? – Benjamin Jun 27 '16 at 11:31

I get 307 RIBPS and 264G. That's mean that I need 307GB of RAM? – E_Blue Sep 6 '16 at 20:50

More like 264G. But you should have enough RAM for that, else give the mentioned 80% of your RAM to mysql, depending on what else runs on the system. – sjas Sep 28 '16 at 8:23

The greatest post I've ever read! I have a ~big database around 3GB. After reading your answer/article and the links speed got up to 2x – fat_mike Dec 10 '16 at 3:45

2 @Benjamin: As of MySQL 5.7.6 the information_schema is merged into performance_schema. So just change "information_schema" to "performance_schema" in the query to make it work. Source: dev.mysql.com/doc/refman/5.7/en/status-table.html – Ralph Bolton Mar 13 at 10:59



Your title asks about innodb_buffer_pool_size, but I suspect that is not the real problem. (Rolando commented on why you have set it big enough, even too big.)

> I've set max connections = 800 which sometimes saturate and grind the server to halt.

That is unclear. 800 users in "Sleep" mode has virtually zero impact on the system. 800 active threads would be a disaster. How many threads are "running"?

Are the threads blocking each other? See SHOW ENGINE INNODB STATUS for some clues on deadlocks, etc.

Are any queries showing up in the slowlog? Let's optimize them.

What version are you using? XtraDB (a drop-in replacement for InnoDB) does a better job of using multiple cores. 5.6.7 does an even better job.

innodb_buffer_pool_instances -- change this to 8 (assuming a 20G buffer_pool); it will cut back slightly on the Mutex contention.

Are you I/O bound or are you CPU bound? The solutions are radically different, depending on your answer.

SSD -- It might be better if all the log files were on non-SSD drives.

answered Oct 22 '12 at 23:19
Rick James
27.4k 2 15 43

Something like this? Using `SHOW VARIABLES` and `SHOW GLOBAL STATUS` :

Expression: `innodb_buffer_pool_size / _ram`
Meaning: % of RAM used for InnoDB buffer_pool
Recommended range: 60~80%

Expression: `Innodb_buffer_pool_reads / Innodb_buffer_pool_read_requests`
Meaning: Read requests that had to hit disk
Recommended range: 0-2%
What to do if out of range: Increase innodb_buffer_pool_size if you have enough RAM.

Expression: `Innodb_pages_read / Innodb_buffer_pool_read_requests`
Meaning: Read requests that had to hit disk
Recommended range: 0-2%
What to do if out of range: Increase innodb_buffer_pool_size if you have enough RAM.

Expression: `Innodb_pages_written / Innodb_buffer_pool_write_requests`
Meaning: Write requests that had to hit disk
Recommended range: 0-15%
What to do if out of range: Check innodb_buffer_pool_size

Expression: `Innodb_buffer_pool_reads / Uptime`
Meaning: Reads
Recommended range: 0-100/sec.
What to do if out of range: Increase innodb_buffer_pool_size?

Expression: `(Innodb_buffer_pool_reads + Innodb_buffer_pool_pages_flushed)  / Uptime`
Meaning: InnoDB I/O
Recommended range: 0-100/sec.
What to do if out of range: Increase innodb_buffer_pool_size?

Expression: `Innodb_buffer_pool_pages_flushed / Uptime`
Meaning: Writes (flushes)
Recommended range: 0-100/sec.
What to do if out of range: Increase innodb_buffer_pool_size?

Expression: `Innodb_buffer_pool_wait_free / Uptime`
Meaning: Counter for when there are no free pages in buffer_pool. That is, all pages are dirty.
Recommended range: 0-1/sec.
What to do if out of range: First be sure innodb_buffer_pool_size is set reasonably; if still trouble, decrease innodb_max_dirty_pages_pct

answered Feb 7 '15 at 1:02
Rick James
27.4k 2 15 43

More memory is always better, but in my experience most of the times buffer pool size should not fit your data size. Many tables are inactive most of the times, like backup tables lying around, so innodb buffer pool size should rather fit you acive data size.

The time frame you specify for active pages influences the performance, but there's an optimal point, where you won't get that more performance for a bigger buffer size. You could estimate/calculate/measure that by "show engine status innodb".

answered Mar 14 '14 at 10:23
user77376

148 4