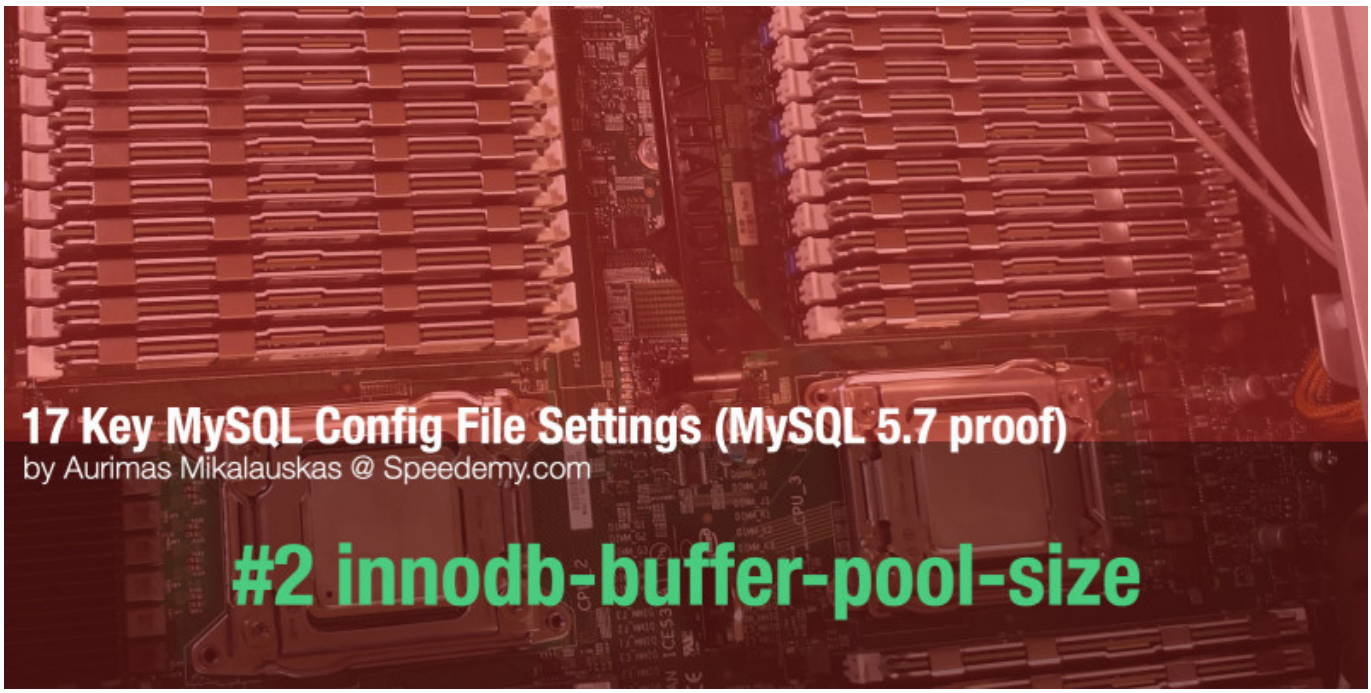
INNODB_BUFFER_POOL_SIZE: GET THE BEST OF YOUR MEMORY



Now that we're settled with the storage engine (and if you're still in doubt, hit me up in comments), let's talk about the most important InnoDB variable: innodb_buffer_pool_size.

WHAT IS INNODB BUFFER POOL?

Computers use most of their memory to improve access to most commonly used data. This is known as caching and it is a very important part of computing, because accessing data on a disk can be 100 to 100,000 times slower, depending on the amount of data being accessed (http://highscalability.com/numbers-everyone-should-know).

Just think about it

A report that takes 1 second to generate when all data is in memory could take **over a day** to generate if all data had to be read from disk every single time (assuming also random I/O).

MyISAM is using OS file system cache to cache the data that queries are reading over and over again. Whereas InnoDB uses a very different approach.

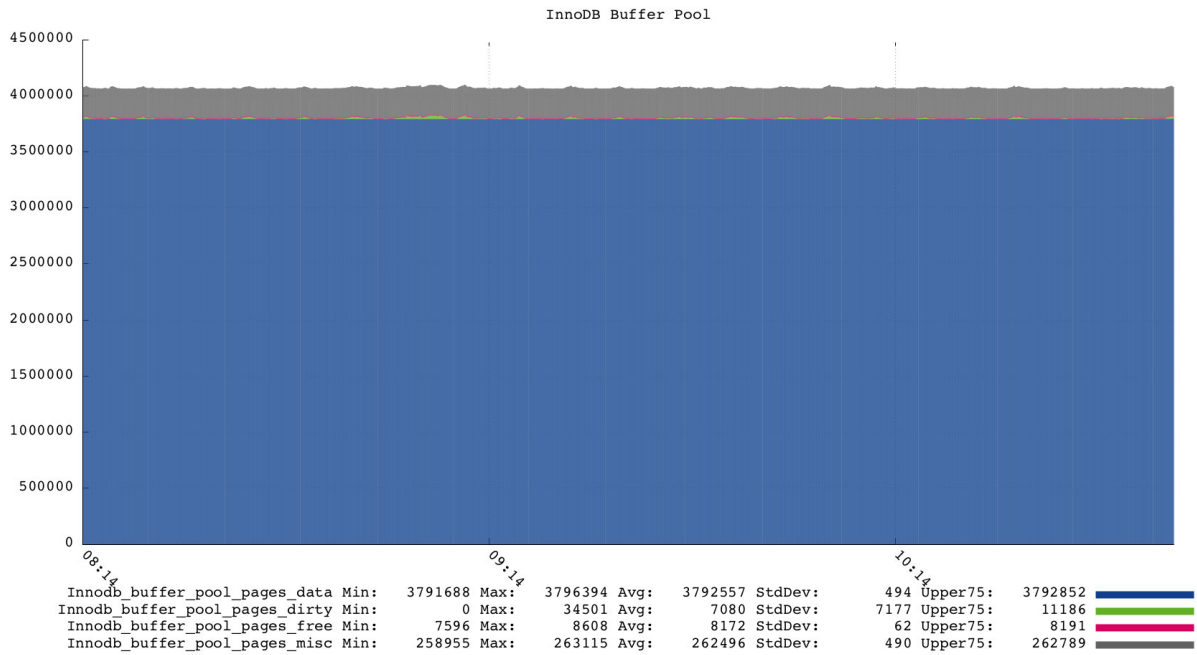
Instead of relying on OS to do the “right thing”, InnoDB handles caching itself – within the InnoDB Buffer Pool – and throughout this course you will learn how it works and why it was a good idea to implement it that way.

INNODB BUFFER POOL IS MORE THAN JUST A CACHE

InnoDB buffer pool actually serves multiple purposes. It’s used for:

- Data caching – this is definitely a big part of it
- Indices caching – yes, these share the same buffer pool
- Buffering – modified data (often called “dirty data”) lives here before it’s flushed
- Storing internal structures – some structures such as Adaptive Hash Index (we’ll get to it) or row locks are also stored in the InnoDB Buffer Pool

Here’s a very typical InnoDB Buffer Pool page distrubition from a customer machine with innodb-buffer-pool-size set to 62G:



(http://www.speedemy.com/wp-content/uploads/2015/09/2-ibp-example.jpg)

As you can see, Buffer Pool is mostly filled with regular InnoDB pages, but about 10% of it is used for other purposes.

Oh and in case you’re wondering what units are used in this graph, that’s InnoDB pages. A single page is typically 16 kilobytes in size, so you can multiply these numbers by 16,384 to get a sense of usage in bytes.

SIZING INNODB BUFFER POOL

So what should innodb-buffer-pool-size be set to? I'm glad you asked! I was just about to get there.

DEDICATED SERVER

On a dedicated MySQL server running all InnoDB, as a rule of thumb, recommendation is to set the innodb-buffer-pool-size to 80% of the total available memory on the server.

Why not 90% or 100%?

Because other things need memory too:

- Every query needs at least few kilobytes of memory (and sometimes – few megabytes!)
- There's various other internal MySQL structures and caches
- InnoDB has a number of structures using memory beyond the buffer pool (Dictionary cache, File system, Lock system and Page hash tables, etc.
- There's also some MySQL files that must be in OS cache (binary logs, relay logs, innodb transaction logs).
- Plus, you want to leave some room for the operating system memory structures.

By the way, this number is *NOT* pulled out of the hat – I've seen hundreds of systems large and small, and even on the servers with 512GB of RAM, we found 80% to be about the right size for sustainable operation.

If you see a lot of free memory, sure you can bump it up a bit (especially as MySQL 5.7 makes this much easier), but do not let MySQL consume all memory, or you will face problems. BIG problems. Namely, swapping.

Don't let your database server swap!

Swapping is the worst thing that can happen to a database server – it's much worse than having buffer pool size that's not large enough. One example how this may go wrong is InnoDB using a lock that it typically uses when accessing a page in memory (100ns access time) to access the page that is swapped out (10ms access time). This would cause all currently running queries to stall and as these things usually don't walk alone, you can guess where this is going...

SHARED SERVER

If your MySQL server shares resources with application, rules of thumb no longer work.

In such an environment it's much harder to find the *right* number. On the other hand, it's usually less so important to find the *right* size and a *good enough* number is often good enough in a shared environment.

In any case, first I check the actual size of the InnoDB tables. Chances are, you don't really need much memory. Remember this query from one of the earlier posts:

```
SELECT engine,
  count(*) as TABLES,
  concat(round(sum(table_rows)/1000000,2),'M') rows,
  concat(round(sum(data_length)/(1024*1024*1024),2),'G') DATA,
  concat(round(sum(index_length)/(1024*1024*1024),2),'G') idx,
  concat(round(sum(data_length+index_length)/(1024*1024*1024),2),'G') total_size,
  round(sum(index_length)/sum(data_length),2) idxfrac
FROM information_schema.TABLES
WHERE table_schema not in ('mysql', 'performance_schema', 'information_schema')
GROUP BY engine
ORDER BY sum(data_length+index_length) DESC LIMIT 10;
```

This will give you an idea how much memory you'd need for InnoDB buffer pool if you wanted to cache the entire dataset. And note that in many cases you don't need that, you only want to cache your working set (actively used data).

If it all fits in, say, half the memory on the server, great – set it to the size of all InnoDB tables combined and forget it (for some time). If not, let me teach you a very simple trick to determine if the InnoDB buffer pool is well sized.

Using server command line, run the following:

```
$ mysqladmin ext -ri1 | grep Innodb_buffer_pool_reads
| Innodb_buffer_pool_reads                 | 1832098003     |
| Innodb_buffer_pool_reads                 | 595            |
| Innodb_buffer_pool_reads                 | 915            |
| Innodb_buffer_pool_reads                 | 734            |
| Innodb_buffer_pool_reads                 | 622            |
| Innodb_buffer_pool_reads                 | 710            |
| Innodb_buffer_pool_reads                 | 664            |
| Innodb_buffer_pool_reads                 | 987            |
| Innodb_buffer_pool_reads                 | 1287           |
| Innodb_buffer_pool_reads                 | 967            |
| Innodb_buffer_pool_reads                 | 1181           |
| Innodb_buffer_pool_reads                 | 949            |
```

What you see here is the number of reads from disk into the buffer pool (per second). These numbers above are pretty darn high (luckily, this server has an IO device that can handle 4000 IO operations per second) and if this was an OLTP system, I would highly recommend to increase the innodb buffer pool size and add more memory to the server if needed.

If you don't have access to the command line, I highly suggest you get one as you're a lot more flexible there. But if you want a GUI alternative, MySQL Workbench (https://www.mysql.com/products/workbench/) is your friend. Under **PERFORMANCE**, open the **Dashboard** and you will see both "InnoDB buffer pool disk reads" and also "InnoDB Disk Reads" (most of the time they go hand in hand).

CHANGING INNODB BUFFER POOL

Finally, here's how you actually change the innodb-buffer-pool-size.

If you're already on MySQL 5.7, you are extremely lucky, because that means you can change it online! Just run the following command as a SUPER user (i.e. root) and you're done:

```
mysql> SET GLOBAL innodb_buffer_pool_size=size_in_bytes;
```

Well not exactly done – you still need to change it in my.cnf file too, but at least you will not need to restart the server as the innodb buffer pool will be resized automatically with something along these lines in the error log:

```
[Note] InnoDB: Resizing buffer pool from 134217728 to 21474836480. (unit=134217728)
[Note] InnoDB: disabled adaptive hash index.
[Note] InnoDB: buffer pool 0 : 159 chunks (1302369 blocks) were added.
[Note] InnoDB: buffer pool 0 : hash tables were resized.
[Note] InnoDB: Resized hash tables at lock_sys, adaptive hash index, dictionary.
[Note] InnoDB: Completed to resize buffer pool from 134217728 to 21474836480.
[Note] InnoDB: Re-enabled adaptive hash index.
```

All earlier versions of MySQL do require a restart, so:

1. set an appropriate innodb_buffer_pool_size in my.cnf
2. restart mysql server
3. celebrate improved MySQL performance

P.S. If you have stumbled on this post by accident and you're not sure what other posts I am talking about, start here (/17-key-mysql-config-file-settings-mysql-5-7-proof).

SHARE THIS POST

(mailto:?subject=innodb_buffer_p thought you might enjoy this! Check it out when you have a chance: https://www.speedemy.c key-mysql-config-file-settings/innodb_buffer_

ABOUT THE AUTHOR

AURIMAS MIKALAUSKAS

Facebook (https://www.facebook.com/aurimas.mikalauskas)

Twitter (https://twitter.com/aurimas_m)

Google+ (https://plus.google.com/+AurimasMikalauskasPlus/posts)

Linux hacker since the age of 0xE, deeply passionate about high performance and scalability. Former architect of scalable systems at Percona - largest independent MySQL and MongoDB consultancy company (for 9 years). Audiobook addict, piano & guitar virtuoso, scuba diver, wind surfer and a wannabe glider pilot. Learn more about the author (http://www.speedemy.com/about/).

20 Comments SPEEDEMY 1 Login

Recommend Tweet Share Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Miguel • 2 years ago

From this Percona forum thread:

https://www.percona.com/for...

also mentions to check Innodb_buffer_pool_read_requests to check the ratio of requests and reads. From that thread they say the ratio at least should be around 100 times to check most data is being cached and not actually read from this.

My current situation is that I run a web server with Percona Server 5.6.37. All are Wordpress databases. I have converted them into innodb and using your query to see the data size is 1.67 Gb. My buffer pool is currently 512Mb.

I checked the buffer_pool_reads and requests and requests are 200 times the number of reads.

I am wondering if I should increase the buffer pool to 1 Gb or even bigger. Currently I assume since the buffer pool is less than 1 Gb, the buffer pool instances are only 1 (this guy says so: https://www.saotn.org/mysql...)

https://www.oauth.org/mysql...).

Should I increase the buffer pool size and the number of instances to improve concurrency?

Reply • Share ›

Aurimas Mikalauskas Mod ➚ Miguel • 2 years ago

The key is looking at the relative numbers rather than absolute ones. Absolute numbers are meaningless in day to day life as they could be showing high counts due to some nightly jobs (e.g. mysqldump reading full dataset).

I've given some examples of monitoring relative counters here:

http://www.speedemy.com/ins...

For the Innodb_buffer_pool_reads, you want to know capacity of your disks and then see how high is the number. Most of the time, I am guessing it will be zero or close to zero. However, say your disks can handle 1000 iops – in such case even 100 innodb_buffer_pool_reads per scond could have you worried, especially if you're seeing an increase over time (e.g. comparing the numbers from day 1 and day 7), because once things don't fit in the innodb buffer pool anymore, situation can get out of hands quite easily – i.e. the growth in the number of disk io requests is usually not linear (except when your workingset growth is slowing down)

Reply • Share ›

Miguel ➚ Aurimas Mikalauskas • 2 years ago

What about my last question? Should I increase the buffer pool size? And the number of instances?

Reply • Share ›

Aurimas Mikalauskas Mod ➚ Miguel • 2 years ago

That's the point, you can't know until you measure Innodb_buffer_pool_reads relative numbers. As for number of buffer pool instances, generic benchmarks show 5.6 works better with 2-4 instances, not more. YMMV.

Reply • Share ›

Miguel ➚ Aurimas Mikalauskas • 2 years ago

I know, but from the link i have sent you if you want to have 2-4 instances your pool size must be 1 Gb at least. Is that true? If it is true is wise to start with 1 Gb and 2 instances?

I am using munin to monitor my server performance but i dont find the key information. What do you suggest to track stats apart of pt-query-digest?

Reply • Share ›

Aurimas Mikalauskas Mod ➚ Miguel
• 2 years ago

Sorry I had disappeared, had some important issues to take care of. Regarding buffer pool size - yes that's correct. Sorry have not mentioned it - haven't worked with small sized buffer pools in a while, so I just forgot. 1GB and 2 instances makes sense, although I wouldn't expect that to be much of an improvement unless you have contention on the buffer pool and also you see a decent number of buffer pool misses.

Regarding stats tracking, Percona has some nice tools:

https://www.percona.com/doc...

Reply • Share ›

Miguel ➚ Aurimas Mikalauskas
• 2 years ago

No worries! I really apreciate you answer so quickly!

Well, I have set buffer pool in 2 Gb and 4 instances and from the numbers I get I see a ratio of 200x so most data doesn´t need to be read from disk.

i have set the slow query log with 0 time to record all queries and as Wordpress databases, the most problematic table is wp_options, specially updates and inserts. I am cleaning expired transients with wp-cli but any other hint to improve the performance? Maybe add an index? There is much controversy about adding indexes on autoload column.

Profile

see more

Reply • Share ›

Nprkurnool Ecilhderabady • 3 years ago

Hi Aurimas,

Lets say , we are inserting 1 GB data into innodb table through

Lets say...We are inserting 1 GB data into innodb table through text file using " load data infile '/db/metrics.txt' into table ap_metrics;" . my doubt is , what role innodb buffer pool plays here and how data in inserted (complete flow)and written to disk (ap_metrics.ibd) , ,,innodb file pertable is enabled.

• Reply • Share ›

Aurimas Mikalauskas Mod ➚ Nprkurnool Ecilhderabady • 3 years ago

Hi, Nprkurnool,

Let me give you a short answer here. For a long answer, check out page 23 of my ebook where I talk about Redo logging:

http://www.speedemy.com/boo...

The short (and obviously simplified) answer is that your data will first be loaded to the transaction log and memory (hence, the innodb buffer pool), then a separate thread will be flushing the data to disk (additionally, if you're NOT using O_DIRECT flushing method, also a copy of data will hit OS write buffers) - that flushing will mostly be done from the innodb buffer pool. And when the dirty data is flushed, it will still remain in the innodb buffer pool until it's eventually evicted by LRU policy.

If it's actively used and innodb buffer pool is large enough, eviction will likely not happen.

Hope that answers your question.

• Reply • Share ›

David Jarosch • 4 years ago

Hello Aurimas,

great Topic :). How does the InnoDB Buffer Pool actually work with large tables i.e. 30GB. I've got a InnoDB Buffer Pool size of 36GB. Does the complete table will be pushed into the InnoDB Buffer Pool, or just a small part of data?

Cheers,

• Reply • Share ›

Aurimas Mikalauskas Mod ➚ David Jarosch • 4 years ago

Hi, David -

Thanks a lot for a wonderful question.

Every table (or any innodb tablespace file for that matter) is divided into pages, so basically a page is a single unit here. The smallest block of data. It's usually 16 kilobytes in size (although are exceptions, such as compression or custom size, but that's not used very often) and InnoDB works with these pages. So if you're only using some

works with these pages. So if you're only using some recent data from that table (and the table is indexed properly, so that old data is not touched when not needed), then only those recent data pages will be stored in the innodb buffer pool. If that's just 1MB of data (64 pages), then only 1MB of that table will be in the buffer pool.

However, if you're sometimes hitting old pages too, they too will get into the buffer pool.

I am oversimplifying a bit, of course. E.g. there will also be secondary key data plus there will be other structures stored in the buffer pool, but the basic idea is this.

Regards!

Reply • Share ›

David Jarosch ➚ Aurimas Mikalauskas • 4 years ago

Hello Aurimas,

thank you for your fast reply and detailed explanation!

Cheers,
David

Reply • Share ›

yoyo42 • 4 years ago

If I have a duplicate staging environment on a shared server (that is rarely used and only for testing) such that running the SELECT statement is double what the production database is, is it ok to set innodb_buffer_pool_size to half the output from the SELECT statement you mention above? So instead of setting it to 1G, set it to 500mb? I have 2G total memory and MySQL is on the same OS as nginx/php etc.

Reply • Share ›

Aurimas Mikalauskas Mod ➚ yoyo42 • 4 years ago

Since it's test environment, I wouldn't sweat it. In fact, having queries running slower in staging/test environment may prompt you to think better about optimizing the queries before pushing them to production. In other words, you can surely set it to 500M and just beware that for select that reads 1GB of data, there will be a significant IO involved and therefore query will be a number of times slower. What you may consider though is using innodb compression (even if only in testing) to reduce the working set size.

Reply • Share ›

yoyo42 ➚ Aurimas Mikalauskas • 4 years ago

This is Wordpress so I wouldn't be creating nor testing the queries but other aspects such as

styling, new versions etc. Also, the staging and prod environment are located on the same virtual

> FREE RESOURCES

MySQL Configuration Tuning Handbook (https://www.speedemy.com/books/mysql-configuration-tuning-ebook/)

RECENT POSTS

Best places to look for MySQL help (https://www.speedemy.com/best-places-look-mysql-help/)

How to create mysql login-path (https://www.speedemy.com/create-mysql-login-path/)

Inspecting MySQL micro-stalls with pt-stalk (https://www.speedemy.com/inspecting-mysql-micro-stalls-with-pt-stalk/)

Advanced MySQL Slow Query Logging Part 3: fine-tuning the logging process (https://www.speedemy.com/advanced-mysql-slow-query-logging-3/)

11 new features coming in MySQL 8.0 that will make your eyebrows raise (https://www.speedemy.com/new-in-mysql-8-0-dr/)

CATEGORIES

Apache (https://www.speedemy.com/category/apache/)

Architecture (https://www.speedemy.com/category/architecture/)

Elasticsearch (https://www.speedemy.com/category/elasticsearch/)

Full Text Search (https://www.speedemy.com/category/full-text-search/)

MySQL (https://www.speedemy.com/category/mysql/)

Web Servers (https://www.speedemy.com/category/web-servers/)

Search