ORACLE

MySQL 8.0 : InnoDB Dynamic Redo Logs

Online Redo Log Capacity

Frédéric Descamps
Community Manager
Oracle MySQL
preFOSDEM MySQL Days - February 2023

Who am I ?

about.me/lefred

@lefred

Frédéric Descamps

- *@lefred*
- *MySQL Evangelist*
- *using MySQL since version 3.20*
- *devops believer*
- *living in* 🇧🇪
- *https://lefred.be*

InnoDB Redo Logs

What is it ?

@lefred

InnoDB Redo Logs

During data modification, InnoDB caches the changes in memory (inside InnoDB Buffer Pool) to achieve better read and write performance.

The modifications are also writen to disk in a sequential way (remember the old disks?) on specific files called ***Redo Logs*** *(you can also encounter the name Transaction Logs).*

Those logs are used only in case of a crash and InnoDB needs to perform a recovery of all transactions that have been committed.

This process guarantees the durability, the ***D*** *in* ***ACID****.*

InnoDB Checkpointing - Part I

Flushing to Tablespaces

InnoDB Checkpointing

However at some point, InnoDB will also write the changed pages to disk in the tablespaces (data files). The process of writing the dirty pages (pages that have been modified) to the tablespaces is known as ***flushing*** *or* ***checkpointing****.*

The checkpoint represent the LSN value of the latest changes written to the data files.

InnoDB flushes small batches of those dirty pages from the buffer pool, this is why it's called fuzzy checkpointing.

MySQL does not flush them all at once to avoid heavy process that could disrupt the normal usage of MySQL.

Old Days

Before MySQL 8.0.30

@lefred

InnoDB Redo Logs before MySQL 8.0.30

Before MySQL 8.0.30, the InnoDB Redo Logs were configured using these variables:

- `innodb_log_file_size`*: the size of the files, the default was 48MB and the maximum could not be bigger than* `512GB / innodb_log_files_in_group`
- `innodb_log_files_in_group`*: the number of log files, default and minimum of 2 with a maximum of 100.*

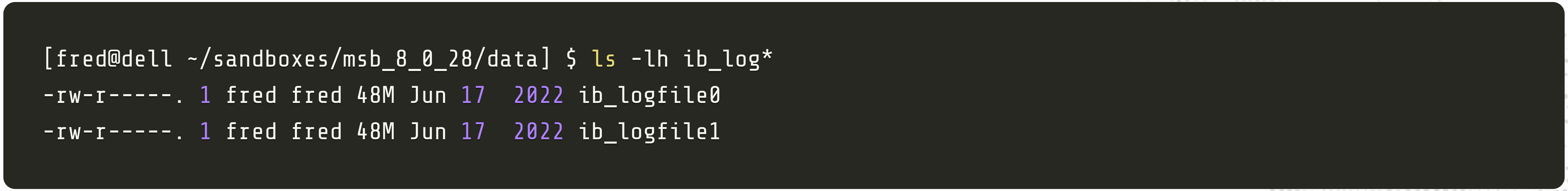
Those variables were not dynamic and required a restart of MySQL Server to modify them.

InnoDB Redo Logs before MySQL 8.0.30

Before MySQL 8.0.30, the InnoDB Redo Logs were configured using these variables:

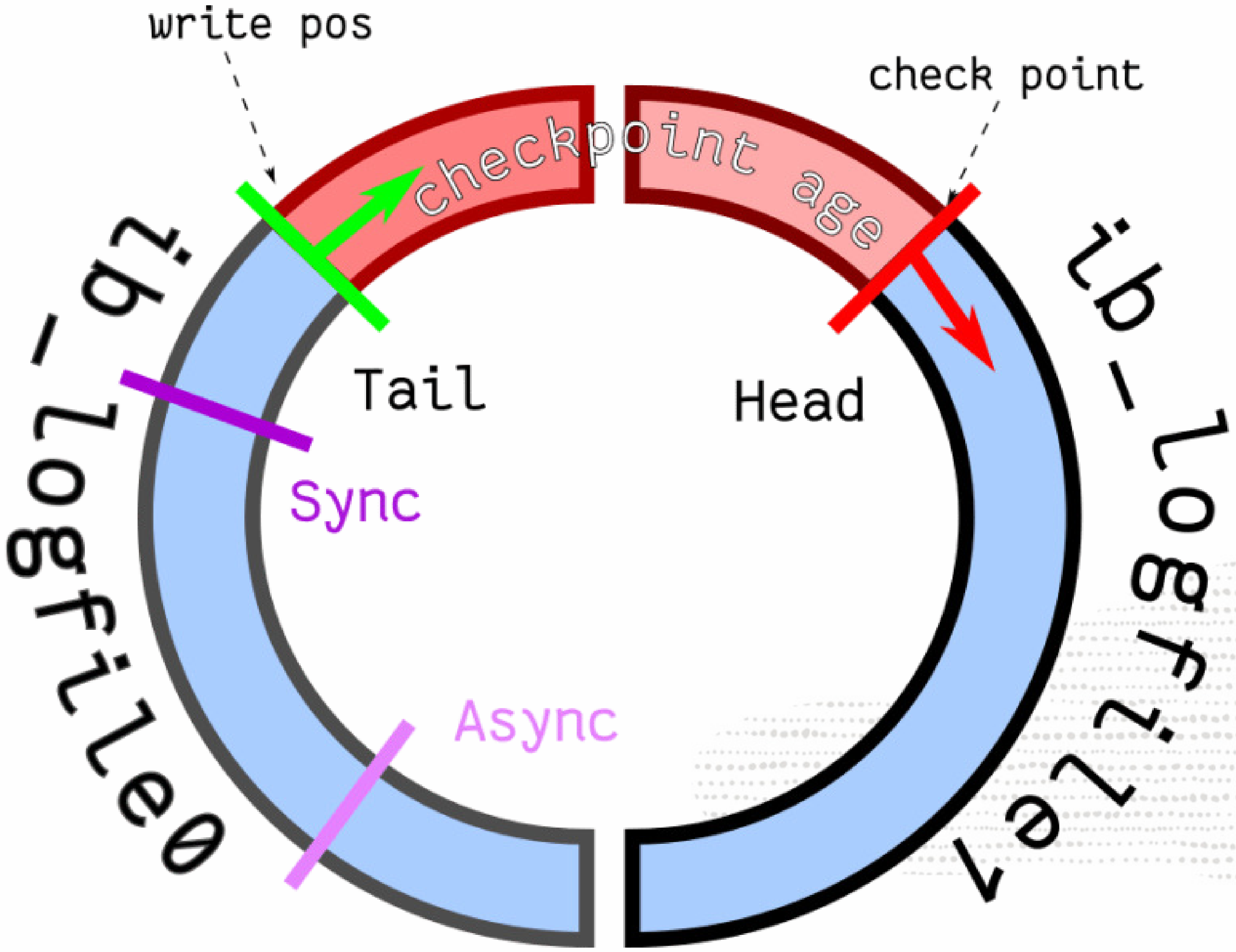
- `innodb_log_file_size`*: the size of the files, the default was 48MB and the maximum could not be bigger than* `512GB / innodb_log_files_in_group`
- `innodb_log_files_in_group`*: the number of log files, default and minimum of 2 with a maximum of 100.*

Those variables were not dynamic and required a restart of MySQL Server to modify them.



```
[fred@dell ~/sandboxes/msb_8_0_28/data] $ ls -lh ib_log*
-rw-r-----. 1 fred fred 48M Jun 17  2022 ib_logfile0
-rw-r-----. 1 fred fred 48M Jun 17  2022 ib_logfile1
```

InnoDB Redo Logs before MySQL 8.0.30 (2)



New Redo Log Architecture

Since MySQL 8.0.30

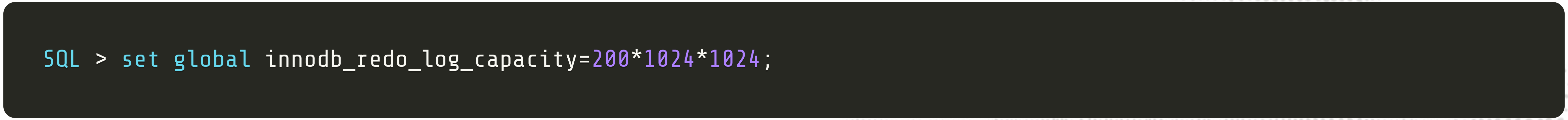
New InnoDB Redo Log Architecture

*Since MySQL 8.0.30, we don't talk about Redo Log Size anymore, but we talk about **capacity** !*

The capacity is defined in a unique variable: `innodb_redo_log_capacity` *(in bytes).*

The default is 100MB.

The variable is dynamic, it can be changed at runtime, to set it to 200MB:



```
SQL > set global innodb_redo_log_capacity=200*1024*1024;
```

InnoDB Redo Log Home Dir

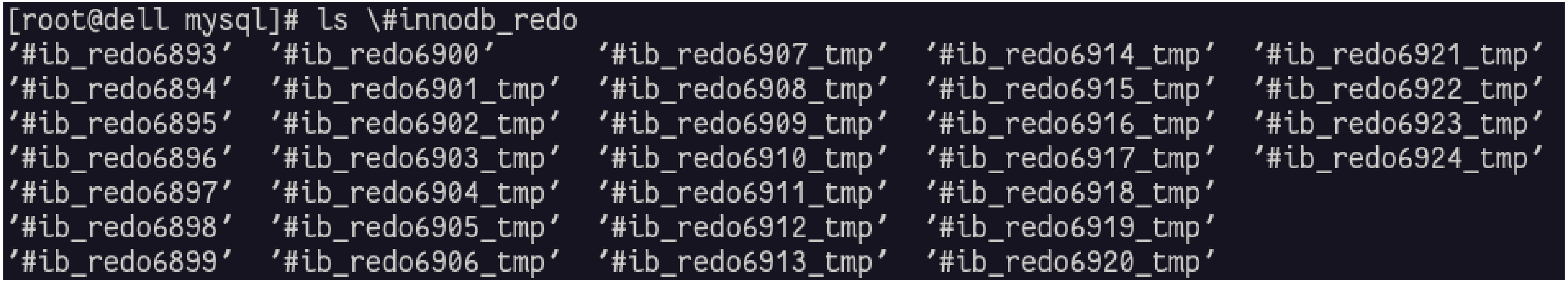
InnoDB will create ***32*** *redo log files in MySQL's datadir inside the new dedicated folder* **`#innodb_redo`** *by default.*

You can also specify another destination by mofiying (not dynamic) the variable `innodb_log_group_home_dir`*.*

Inside that directory, you will be able to find two types of files:

- `#ib_redoXXX` *(where* `XXX` *is the* `file_id`*, a sequence number): those are the active redo log files*
- `#ib_redoXXX_tmp`*: those are spare redo log files*

InnoDB Redo Log Home Dir (2)



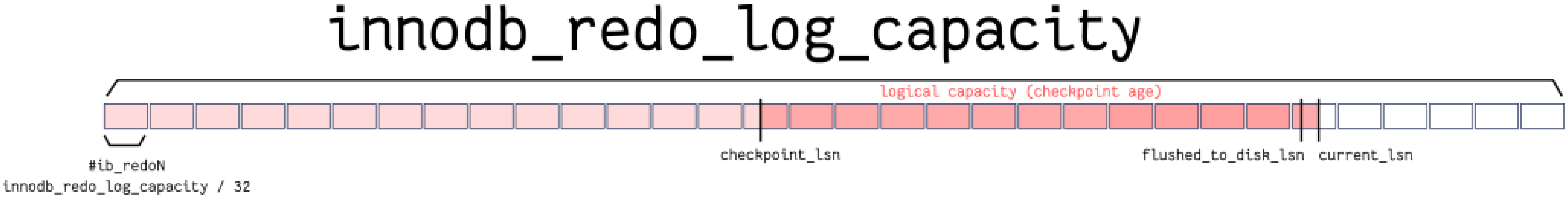
```
[root@dell mysql]# ls \#innodb_redo
'#ib_redo6893'  '#ib_redo6900'      '#ib_redo6907_tmp'  '#ib_redo6914_tmp'  '#ib_redo6921_tmp'
'#ib_redo6894'  '#ib_redo6901_tmp'  '#ib_redo6908_tmp'  '#ib_redo6915_tmp'  '#ib_redo6922_tmp'
'#ib_redo6895'  '#ib_redo6902_tmp'  '#ib_redo6909_tmp'  '#ib_redo6916_tmp'  '#ib_redo6923_tmp'
'#ib_redo6896'  '#ib_redo6903_tmp'  '#ib_redo6910_tmp'  '#ib_redo6917_tmp'  '#ib_redo6924_tmp'
'#ib_redo6897'  '#ib_redo6904_tmp'  '#ib_redo6911_tmp'  '#ib_redo6918_tmp'
'#ib_redo6898'  '#ib_redo6905_tmp'  '#ib_redo6912_tmp'  '#ib_redo6919_tmp'
'#ib_redo6899'  '#ib_redo6906_tmp'  '#ib_redo6913_tmp'  '#ib_redo6920_tmp'
```

InnoDB tries to maintain approximately 32 files here, so that it doesn't need to wait long before one of them becomes no longer needed as it would if you had just 2 big files.

This way it can reclaim them one by one when you want to resize them.

InnoDB Redo Log Capacity

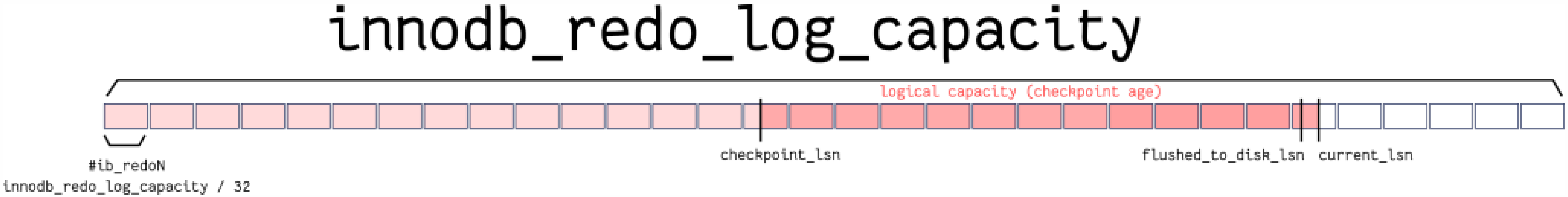
The InnoDB Redo Log Capacity can be represented like this:



- ***checkpoint_lsn*** *(*`Innodb_redo_log_checkpoint_lsn`*): an LSN point up to which all changes to the pages are guaranteed to have already been written and fsynced back to the tablespace files - basically, the sill needed portion of redo log starts here.*
- ***flushed_to_disk_lsn*** *(*`Innodb_redo_log_flushed_to_disk_lsn`*): the last position in the redo log that InnoDB has been flushed to disk.*

InnoDB Redo Log Capacity (2)

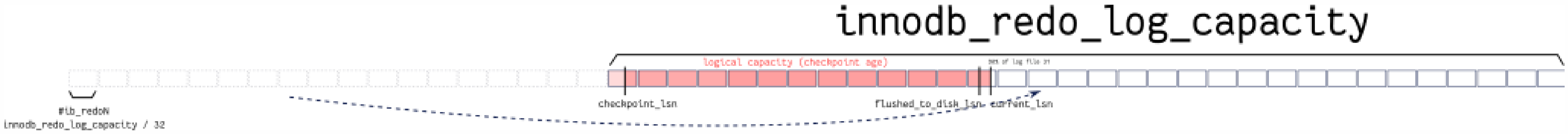
The InnoDB Redo Log Capacity can be represented like this:



- ***current_lsn*** *(`Innodb_redo_log_current_lsn`): the last written position in the redo log. That write could still be buffered inside MySQL processes buffer.*

InnoDB Redo Log Capacity (3)

When InnoDB reaches the end of the ***31st*** *file (90%), the log files governor will perform some cleanup and some active files that are not needed anymore will become the new spare ones:*

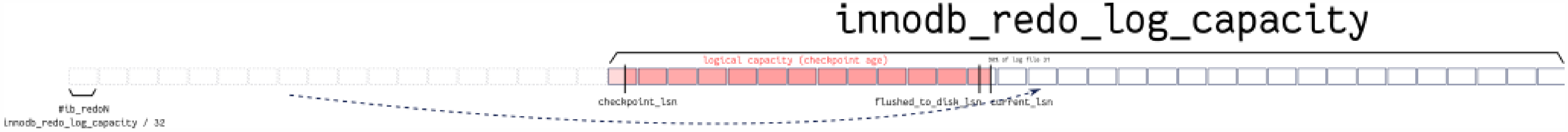


When the background thread is not able to remove a log file from the left to put it to the right, the user transaction will get stuck waiting for REDO buffers to be written to disk.

DBAs get warning in the error log notifying them to increase the InnoDB Redo Log Capacity

InnoDB Redo Log Capacity (3)

*When InnoDB reaches the end of the **31st** file (90%), the log files governor will perform some cleanup and some active files that are not needed anymore will become the new spare ones:*



```
[Warning] [MY-013865] [InnoDB] Redo log writer is waiting for a new redo log file.
                                Consider increasing innodb_redo_log_capacity.
```

right, the user transaction will get stuck waiting for REDO buffers to be written to disk.

DBAs get warning in the error log notifying them to increase the InnoDB Redo Log Capacity

InnoDB Checkpointing - part II

Details

@lefred

InnoDB Checkpointing

So we know that each time data is changed in InnoDB, the page(s) containing the data is modified in memory (in the InnoDB Buffer Pool). The page(s) is (are) noted as dirty.

In case of a sudden crash, we cannot loose all those changes… but the data in memory is gone !

*This is the reason why **diff data** of the pages are also written (and by default flushed to disk) on the redo logs. The data in those logs will be only read in case of InnoDB Recovery.*

During that process the modified pages will be reconstructed with the modified data.

InnoDB Fuzzy Checkpointing

*InnoDB flushes those dirty pages from the Buffer Pool (memory) to the table spaces (disk) in small batches, step by step. This operation is called **Fuzzy Checkpointing**.*

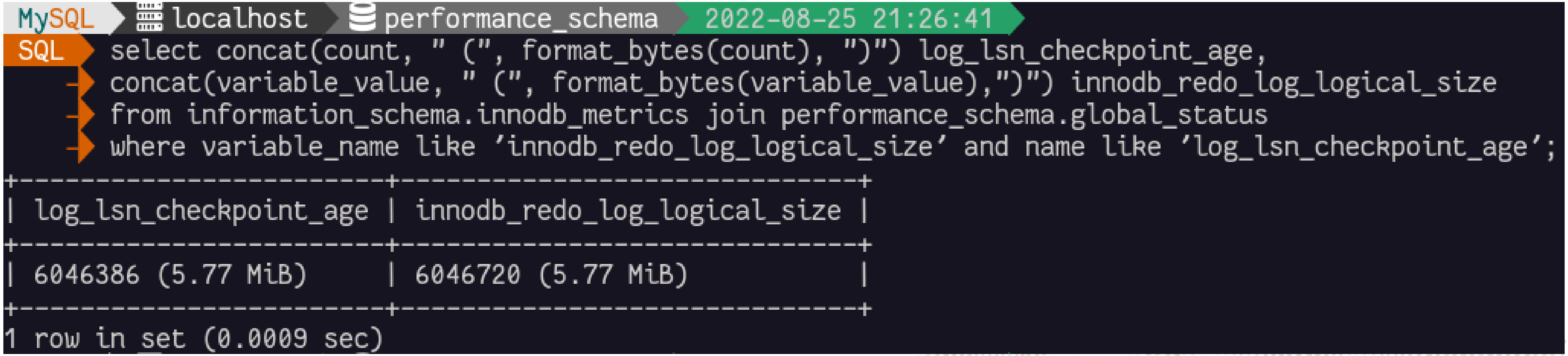
Once the pages are written to the data files on disk (InnoDB tablespaces), the corresponding entries in the Redo Log are not required anymore.

The position up to which InnoDB has written the data to the disk is the value of `Innodb_redo_log_checkpoint_lsn`*.*

*InnoDB Checkpointing is **adaptive**. This means that considering the checkpoint age (*`log_lsn_checkpoint_age`*) InnoDB will decide to flush less or more aggressively.*

InnoDB Fuzzy Checkpointing (2)

For info, `log_lsn_checkpoint_age` *and* `inndob_redo_log_logical_size` *are almost equivalent:*



```
MySQL  localhost  performance_schema  2022-08-25 21:26:41
SQL  select concat(count, " (", format_bytes(count), ")") log_lsn_checkpoint_age,
  -> concat(variable_value, " (", format_bytes(variable_value),")") innodb_redo_log_logical_size
  -> from information_schema.innodb_metrics join performance_schema.global_status
  -> where variable_name like ’innodb_redo_log_logical_size’ and name like ’log_lsn_checkpoint_age’;
+------------------------+------------------------------+
| log_lsn_checkpoint_age | innodb_redo_log_logical_size |
+------------------------+------------------------------+
| 6046386 (5.77 MiB)     | 6046720 (5.77 MiB)           |
+------------------------+------------------------------+
1 row in set (0.0009 sec)
```

LSN Checkpoint Age and Redo Log Capacity

MySQL performs this adaptive flushing considering these thresholds:

- ***soft limit for logical capacity**: to avoid deadlocks InnoDB doesn't let the user transactions to use up the whole* `innodb_redo_log_capacity`*. Instead it keeps them below **soft logical capacity** which is roughly **30/32** of it. When this limitation is exceeded, all user threads are paused and a message is sent to the error_log.*

- ***hard limit for logical capacity**: this limitation is never exceeded. If space isn't reclaimed after 1 second wait when the limit is reached, logs are written as much as possible or crash InnoDB !*

LSN Checkpoint Age and Redo Log Capacity (2)

- ***async flush point*** *(`log_max_modified_age_async`): writes are allowed but page flushing will be gradually increased to reach the next threshold. This will lead to a drop of performance. In the code, async flush point can be called `adaptive_flush_min_age`. This is **7/8** of the soft logical capacity. However, in practice, it seems that the adaptive flushing already starts at `innodb_adaptive_flushing_lwm` (by default 10% of soft logical capacity), and reaches maximum allowed IO capacity already at 82% of the async flush point.*

- ***sync flush point*** *(`log_max_modified_age_sync`): at this point the checkpointer will request page cleaners to flush as much of dirty pages to get the checkpoint age below this threshold and will wait for it synchronously. **Terrible performance**. This is also called `adaptive_flush_max_age`. This is **15/16** of the soft logical capacity.*

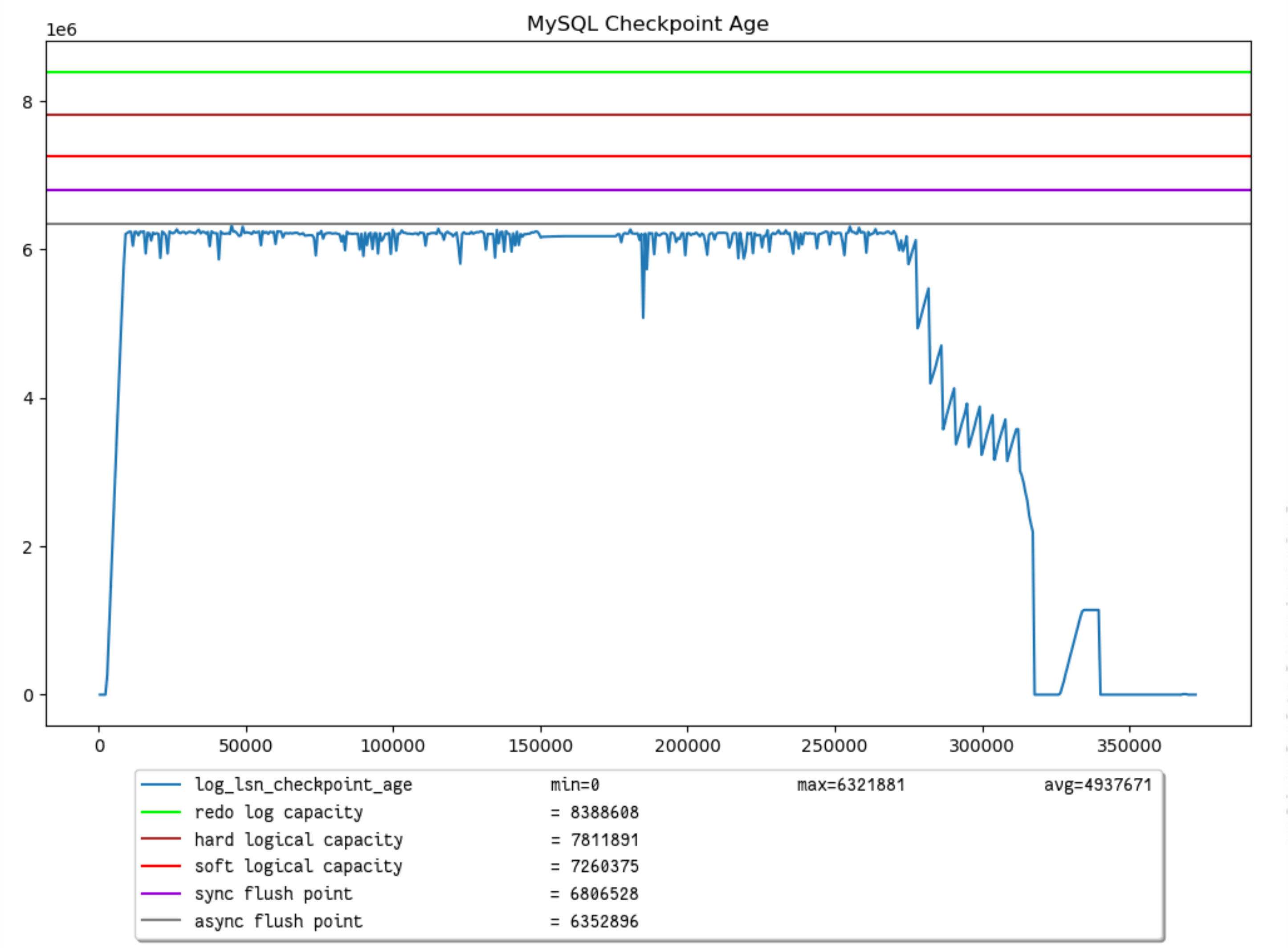
LSN Checkpoint Age and Redo Log Capacity (3)

- ***aggressive_checkpoint_min_age***: *this represents* ***31/32*** *of soft logical capacity. When this point is reached, MySQL already asked to InnoDB to flush dirty pages from the Buffer Pool at full speed.*

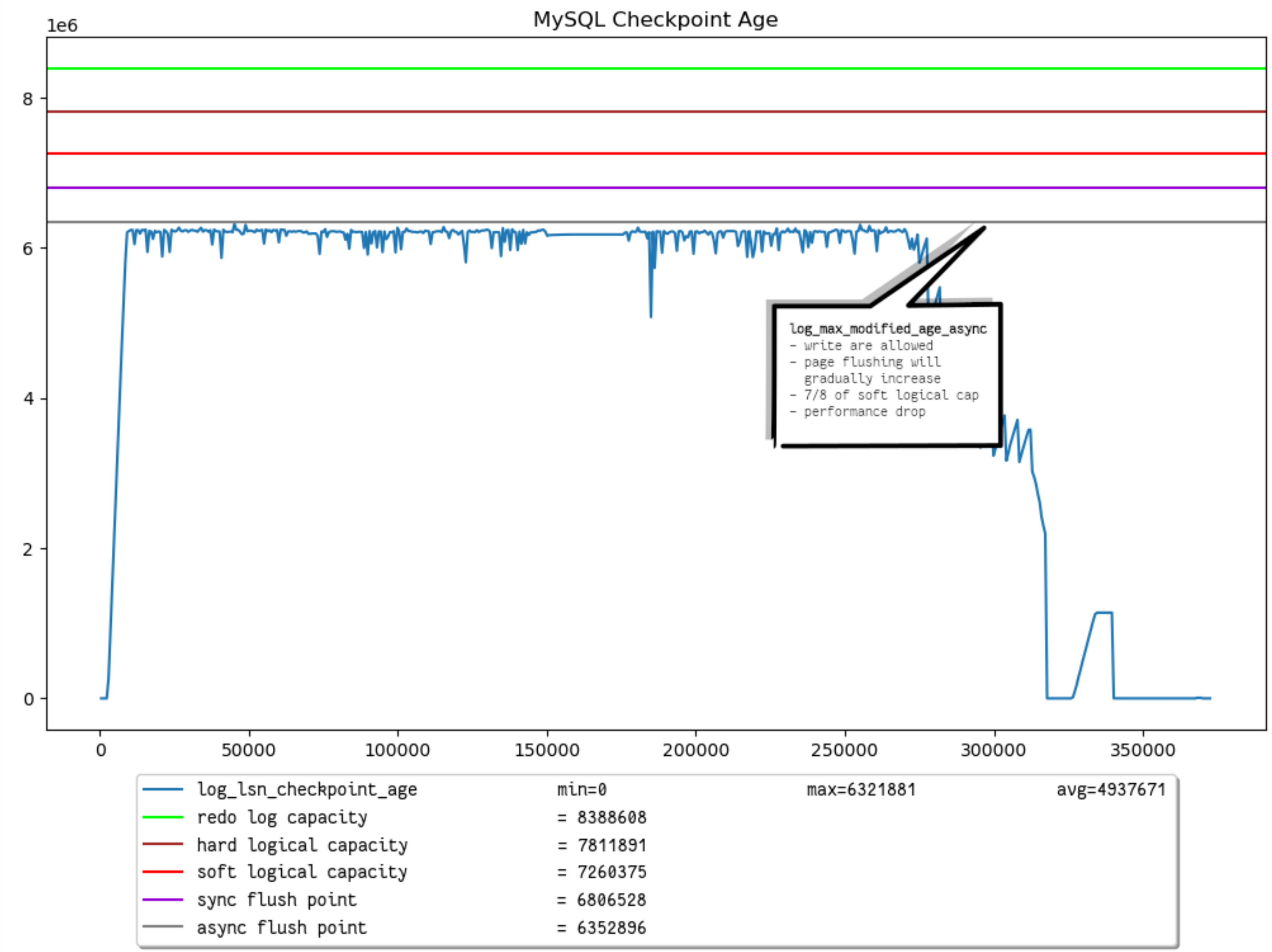
The checkpointer will not sleep for 1 second between attempting updating checkpoint lsn. Instead it will request a sync checkpoint as often as possible and will also update `checkpoint_lsn` *value to the redo log header as soon as possible afterwards.*

This is performed to be able to reclaim the space faster. As we are already at the top speed, this doesn't add any more pressure to the page cleaners.

LSN Checkpoint Age and Redo Log Capacity (4)

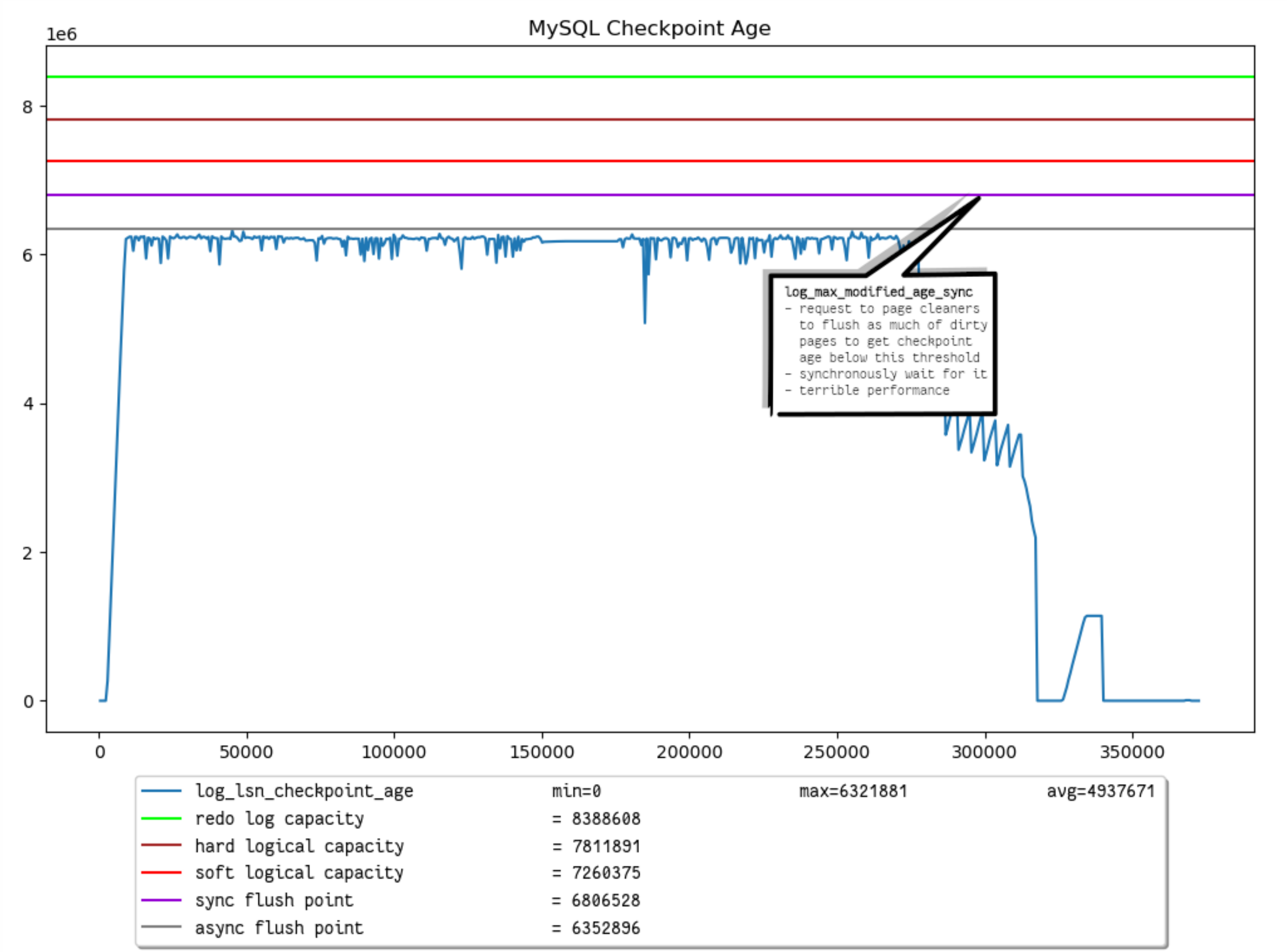


LSN Checkpoint Age and Redo Log Capacity (4)



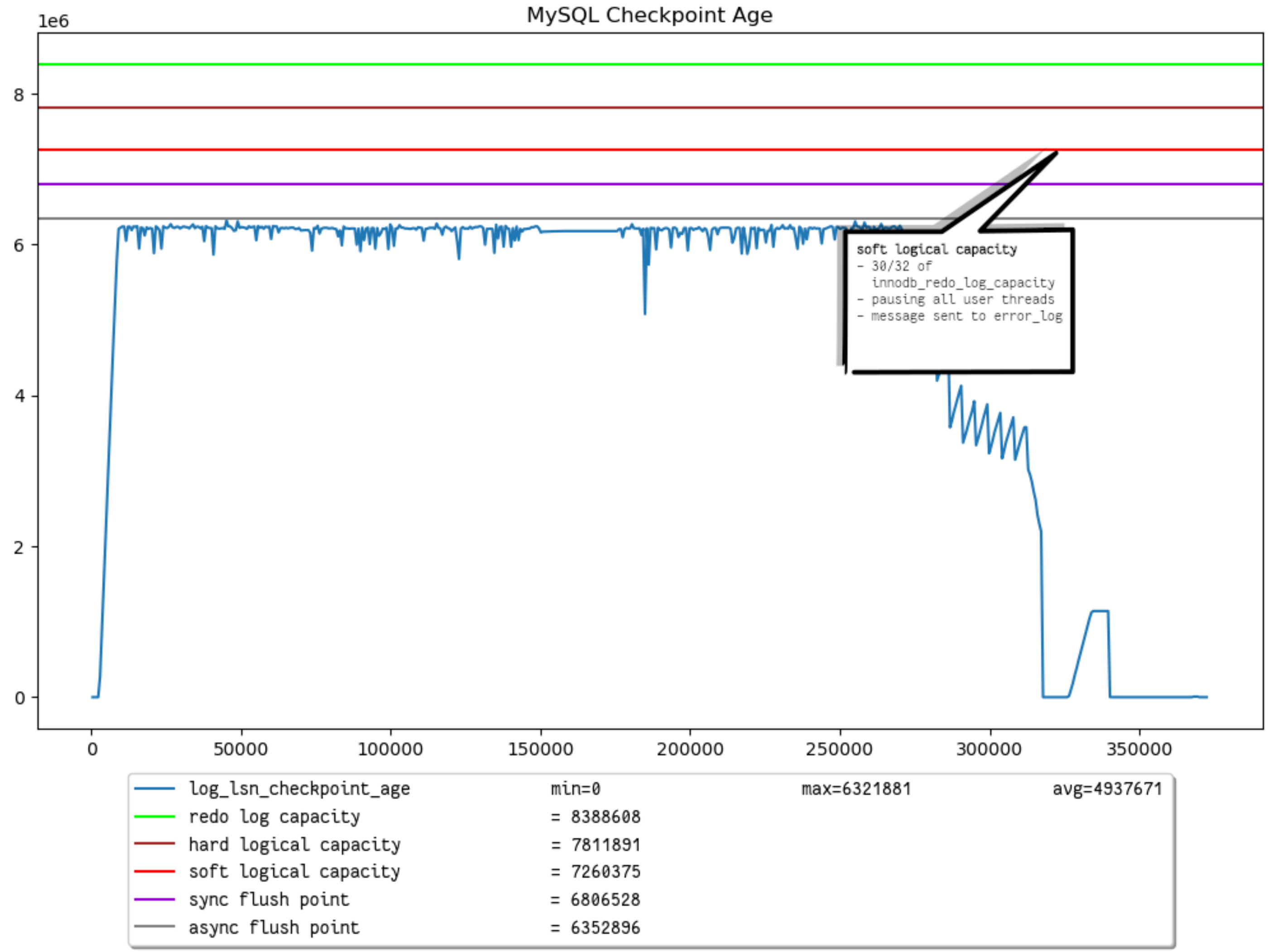
log_max_modified_age_async
- write are allowed
- page flushing will gradually increase
- 7/8 of soft logical cap
- performance drop

LSN Checkpoint Age and Redo Log Capacity (4)



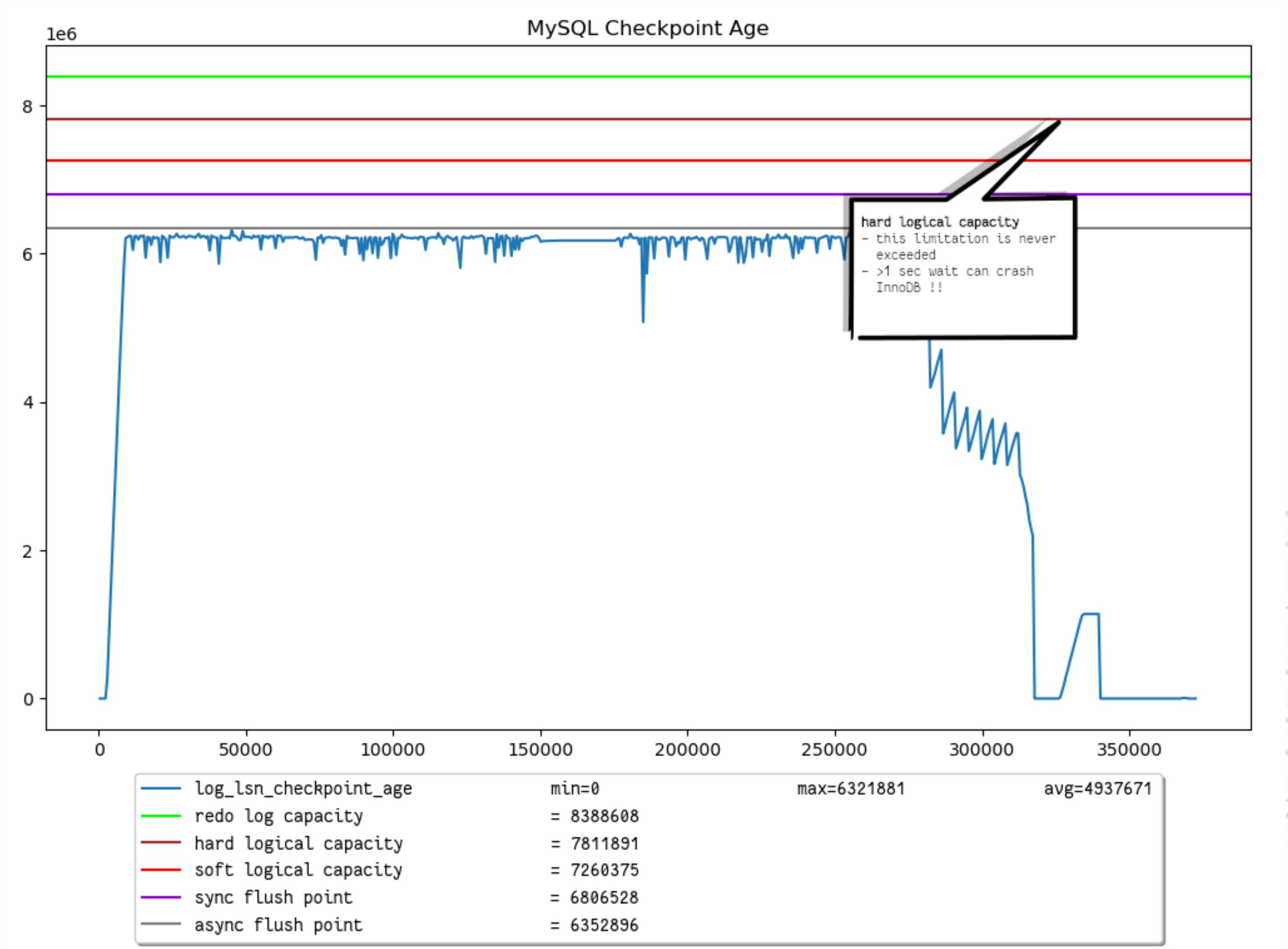
log_max_modified_age_sync
- request to page cleaners to flush as much of dirty pages to get checkpoint age below this threshold
- synchronously wait for it
- terrible performance

LSN Checkpoint Age and Redo Log Capacity (4)



soft logical capacity
- 30/32 of innodb_redo_log_capacity
- pausing all user threads
- message sent to error_log

LSN Checkpoint Age and Redo Log Capacity (4)



hard logical capacity
- this limitation is never exceeded
- >1 sec wait can crash InnoDB !!

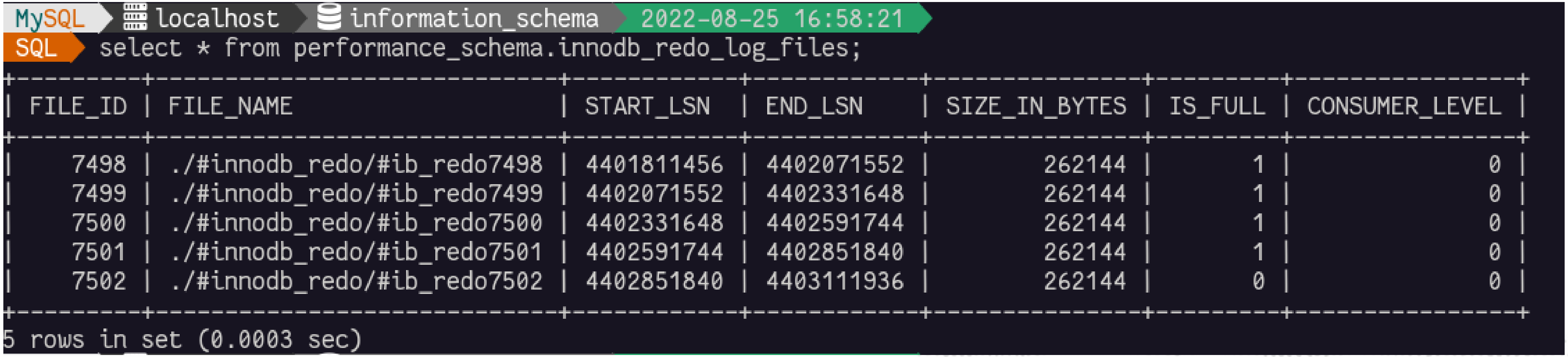
InnoDB Redo Log

Instrumentation

@lefred

Instrumentation - Performance_Schema

The new Redo Log is instrumented in `Performance_Schema` *in the table* `innodb_redo_log_files`*:*

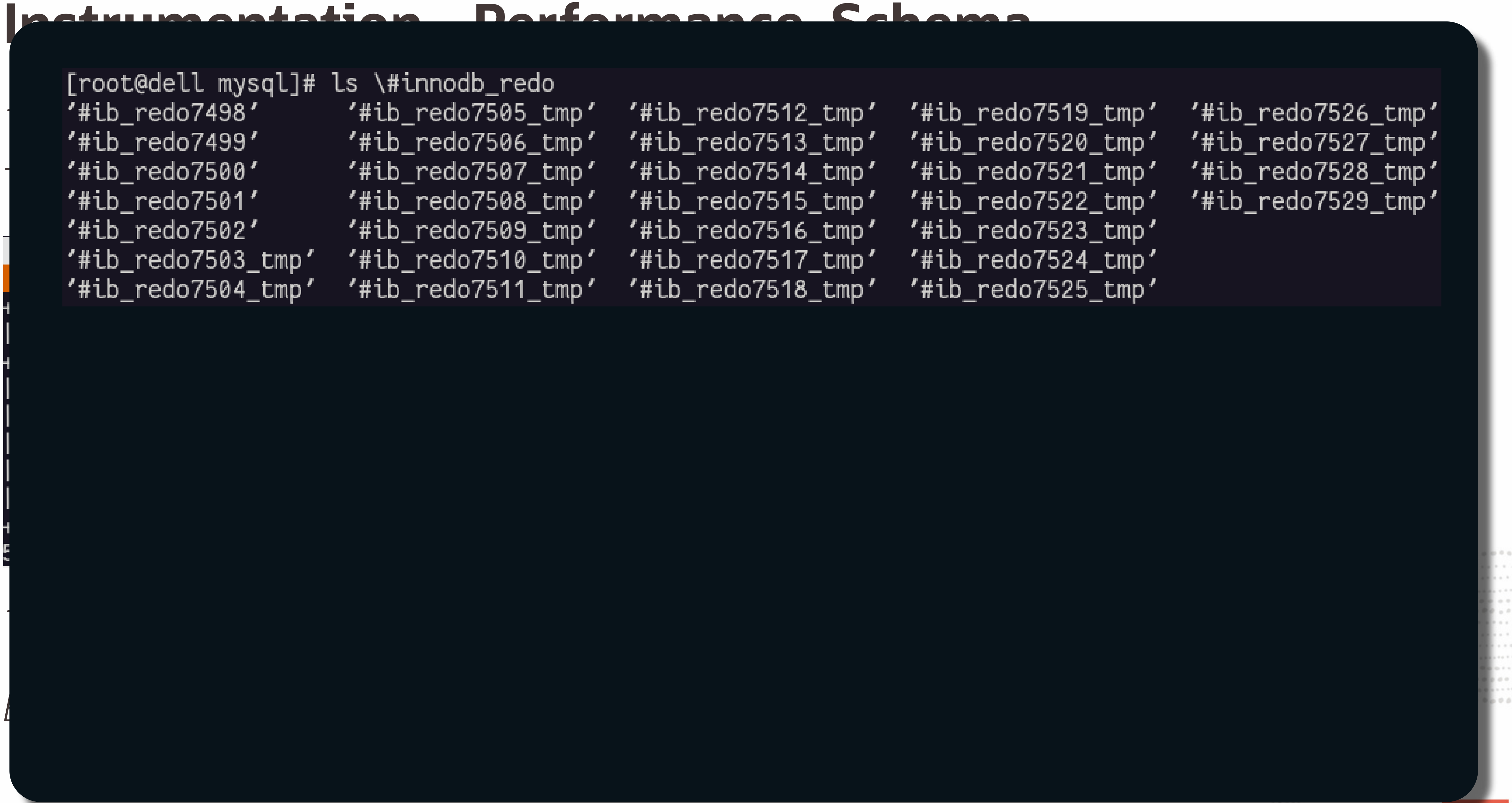


```
MySQL  localhost  information_schema  2022-08-25 16:58:21
SQL  select * from performance_schema.innodb_redo_log_files;
+---------+-----------------------------+------------+------------+---------------+---------+----------------+
| FILE_ID | FILE_NAME                   | START_LSN  | END_LSN    | SIZE_IN_BYTES | IS_FULL | CONSUMER_LEVEL |
+---------+-----------------------------+------------+------------+---------------+---------+----------------+
|    7498 | ./#innodb_redo/#ib_redo7498 | 4401811456 | 4402071552 |        262144 |       1 |              0 |
|    7499 | ./#innodb_redo/#ib_redo7499 | 4402071552 | 4402331648 |        262144 |       1 |              0 |
|    7500 | ./#innodb_redo/#ib_redo7500 | 4402331648 | 4402591744 |        262144 |       1 |              0 |
|    7501 | ./#innodb_redo/#ib_redo7501 | 4402591744 | 4402851840 |        262144 |       1 |              0 |
|    7502 | ./#innodb_redo/#ib_redo7502 | 4402851840 | 4403111936 |        262144 |       0 |              0 |
+---------+-----------------------------+------------+------------+---------------+---------+----------------+
5 rows in set (0.0003 sec)
```

This means there are 5 active redo log files and 27 (32-5) spare ones (`_tmp`*)*

Each active redo log file is associated with a particular range of LSN values.

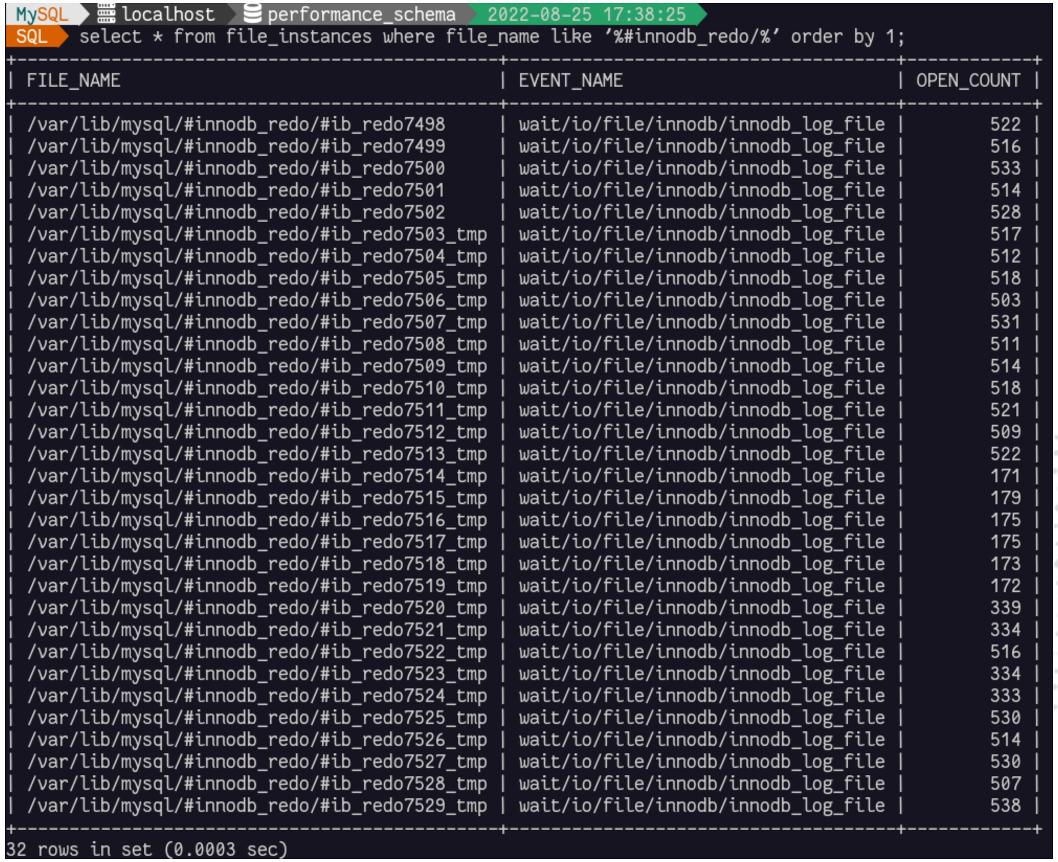
Instrumentation - Performance Schema



```
[root@dell mysql]# ls \#innodb_redo
'#ib_redo7498'      '#ib_redo7505_tmp'  '#ib_redo7512_tmp'  '#ib_redo7519_tmp'  '#ib_redo7526_tmp'
'#ib_redo7499'      '#ib_redo7506_tmp'  '#ib_redo7513_tmp'  '#ib_redo7520_tmp'  '#ib_redo7527_tmp'
'#ib_redo7500'      '#ib_redo7507_tmp'  '#ib_redo7514_tmp'  '#ib_redo7521_tmp'  '#ib_redo7528_tmp'
'#ib_redo7501'      '#ib_redo7508_tmp'  '#ib_redo7515_tmp'  '#ib_redo7522_tmp'  '#ib_redo7529_tmp'
'#ib_redo7502'      '#ib_redo7509_tmp'  '#ib_redo7516_tmp'  '#ib_redo7523_tmp'
'#ib_redo7503_tmp'  '#ib_redo7510_tmp'  '#ib_redo7517_tmp'  '#ib_redo7524_tmp'
'#ib_redo7504_tmp'  '#ib_redo7511_tmp'  '#ib_redo7518_tmp'  '#ib_redo7525_tmp'
```

Instrumentation - Performance_Schema (2)

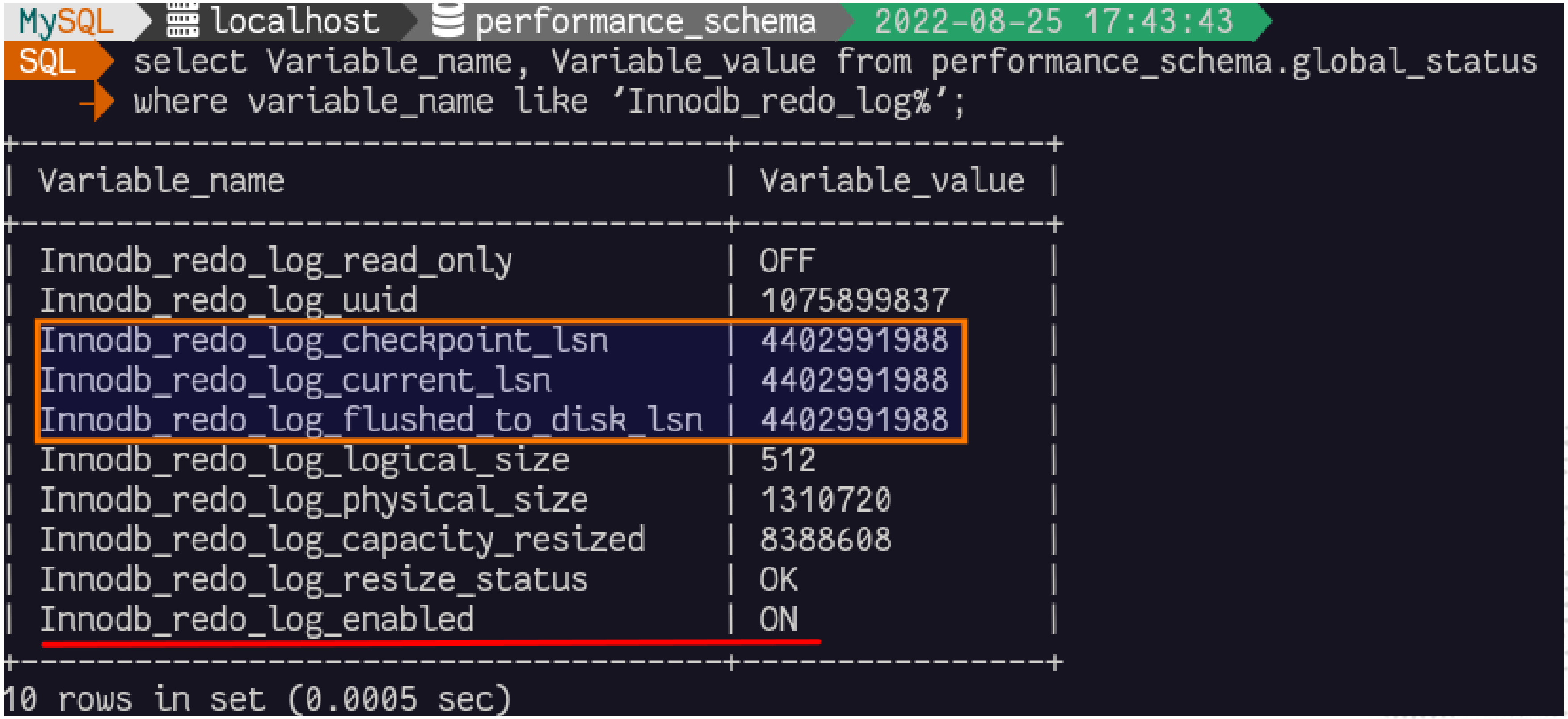
All the files are also instrumented in `Performance_Schema`*'s file instance tables (*`file_instances and file_summary_by_instance`*):*



```
MySQL  localhost  performance_schema  2022-08-25 17:38:25
SQL  select * from file_instances where file_name like ’%#innodb_redo/%’ order by 1;
+----------------------------------------------+-------------------------------------+------------+
| FILE_NAME                                    | EVENT_NAME                          | OPEN_COUNT |
+----------------------------------------------+-------------------------------------+------------+
| /var/lib/mysql/#innodb_redo/#ib_redo7498     | wait/io/file/innodb/innodb_log_file |        522 |
| /var/lib/mysql/#innodb_redo/#ib_redo7499     | wait/io/file/innodb/innodb_log_file |        516 |
| /var/lib/mysql/#innodb_redo/#ib_redo7500     | wait/io/file/innodb/innodb_log_file |        533 |
| /var/lib/mysql/#innodb_redo/#ib_redo7501     | wait/io/file/innodb/innodb_log_file |        514 |
| /var/lib/mysql/#innodb_redo/#ib_redo7502     | wait/io/file/innodb/innodb_log_file |        528 |
| /var/lib/mysql/#innodb_redo/#ib_redo7503_tmp | wait/io/file/innodb/innodb_log_file |        517 |
| /var/lib/mysql/#innodb_redo/#ib_redo7504_tmp | wait/io/file/innodb/innodb_log_file |        512 |
| /var/lib/mysql/#innodb_redo/#ib_redo7505_tmp | wait/io/file/innodb/innodb_log_file |        518 |
| /var/lib/mysql/#innodb_redo/#ib_redo7506_tmp | wait/io/file/innodb/innodb_log_file |        503 |
| /var/lib/mysql/#innodb_redo/#ib_redo7507_tmp | wait/io/file/innodb/innodb_log_file |        531 |
| /var/lib/mysql/#innodb_redo/#ib_redo7508_tmp | wait/io/file/innodb/innodb_log_file |        511 |
| /var/lib/mysql/#innodb_redo/#ib_redo7509_tmp | wait/io/file/innodb/innodb_log_file |        514 |
| /var/lib/mysql/#innodb_redo/#ib_redo7510_tmp | wait/io/file/innodb/innodb_log_file |        518 |
| /var/lib/mysql/#innodb_redo/#ib_redo7511_tmp | wait/io/file/innodb/innodb_log_file |        521 |
| /var/lib/mysql/#innodb_redo/#ib_redo7512_tmp | wait/io/file/innodb/innodb_log_file |        509 |
| /var/lib/mysql/#innodb_redo/#ib_redo7513_tmp | wait/io/file/innodb/innodb_log_file |        522 |
| /var/lib/mysql/#innodb_redo/#ib_redo7514_tmp | wait/io/file/innodb/innodb_log_file |        171 |
| /var/lib/mysql/#innodb_redo/#ib_redo7515_tmp | wait/io/file/innodb/innodb_log_file |        179 |
| /var/lib/mysql/#innodb_redo/#ib_redo7516_tmp | wait/io/file/innodb/innodb_log_file |        175 |
| /var/lib/mysql/#innodb_redo/#ib_redo7517_tmp | wait/io/file/innodb/innodb_log_file |        175 |
| /var/lib/mysql/#innodb_redo/#ib_redo7518_tmp | wait/io/file/innodb/innodb_log_file |        173 |
| /var/lib/mysql/#innodb_redo/#ib_redo7519_tmp | wait/io/file/innodb/innodb_log_file |        172 |
| /var/lib/mysql/#innodb_redo/#ib_redo7520_tmp | wait/io/file/innodb/innodb_log_file |        339 |
| /var/lib/mysql/#innodb_redo/#ib_redo7521_tmp | wait/io/file/innodb/innodb_log_file |        334 |
| /var/lib/mysql/#innodb_redo/#ib_redo7522_tmp | wait/io/file/innodb/innodb_log_file |        516 |
| /var/lib/mysql/#innodb_redo/#ib_redo7523_tmp | wait/io/file/innodb/innodb_log_file |        334 |
| /var/lib/mysql/#innodb_redo/#ib_redo7524_tmp | wait/io/file/innodb/innodb_log_file |        333 |
| /var/lib/mysql/#innodb_redo/#ib_redo7525_tmp | wait/io/file/innodb/innodb_log_file |        530 |
| /var/lib/mysql/#innodb_redo/#ib_redo7526_tmp | wait/io/file/innodb/innodb_log_file |        514 |
| /var/lib/mysql/#innodb_redo/#ib_redo7527_tmp | wait/io/file/innodb/innodb_log_file |        530 |
| /var/lib/mysql/#innodb_redo/#ib_redo7528_tmp | wait/io/file/innodb/innodb_log_file |        507 |
| /var/lib/mysql/#innodb_redo/#ib_redo7529_tmp | wait/io/file/innodb/innodb_log_file |        538 |
+----------------------------------------------+-------------------------------------+------------+
32 rows in set (0.0003 sec)
```

Instrumentation - Status

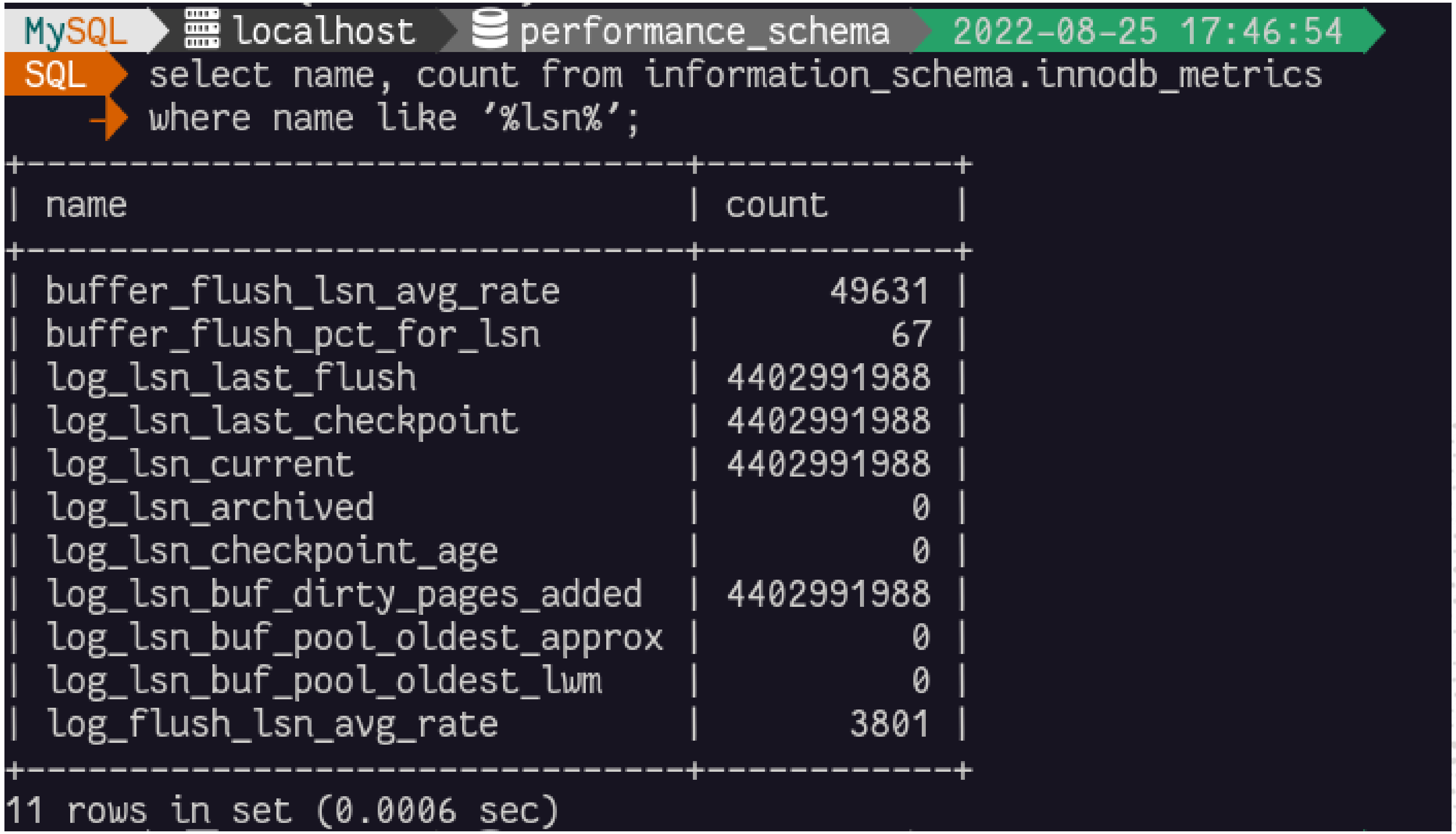
There are status variables providing information about the "flushpointing" operations:



```
MySQL  localhost  performance_schema  2022-08-25 17:43:43
SQL  select Variable_name, Variable_value from performance_schema.global_status
  -> where variable_name like 'Innodb_redo_log%';
+-----------------------------------+----------------+
| Variable_name                     | Variable_value |
+-----------------------------------+----------------+
| Innodb_redo_log_read_only         | OFF            |
| Innodb_redo_log_uuid              | 1075899837     |
| Innodb_redo_log_checkpoint_lsn    | 4402991988     |
| Innodb_redo_log_current_lsn       | 4402991988     |
| Innodb_redo_log_flushed_to_disk_lsn | 4402991988   |
| Innodb_redo_log_logical_size      | 512            |
| Innodb_redo_log_physical_size     | 1310720        |
| Innodb_redo_log_capacity_resized  | 8388608        |
| Innodb_redo_log_resize_status     | OK             |
| Innodb_redo_log_enabled           | ON             |
+-----------------------------------+----------------+
10 rows in set (0.0005 sec)
```

Instrumentation - InnoDB Metrics

Information is also available in InnoDB Metrics:



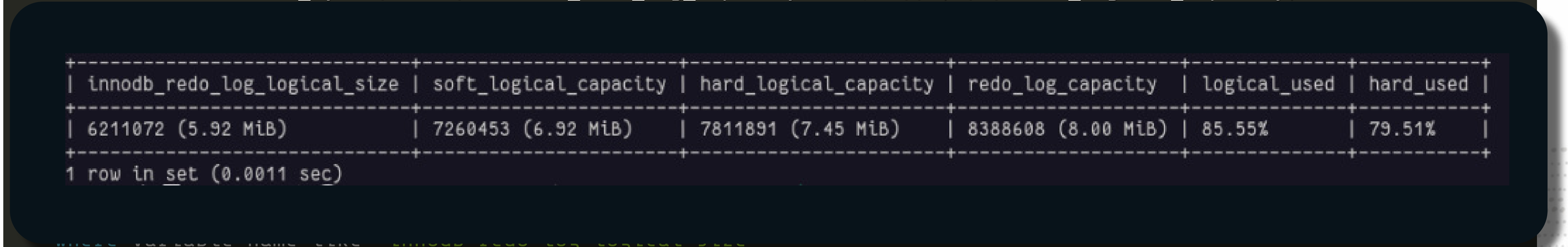
```
MySQL  localhost  performance_schema  2022-08-25 17:46:54
SQL  select name, count from information_schema.innodb_metrics
  -> where name like '%lsn%';
+-------------------------------+------------+
| name                          | count      |
+-------------------------------+------------+
| buffer_flush_lsn_avg_rate     |      49631 |
| buffer_flush_pct_for_lsn      |         67 |
| log_lsn_last_flush            | 4402991988 |
| log_lsn_last_checkpoint       | 4402991988 |
| log_lsn_current               | 4402991988 |
| log_lsn_archived              |          0 |
| log_lsn_checkpoint_age        |          0 |
| log_lsn_buf_dirty_pages_added | 4402991988 |
| log_lsn_buf_pool_oldest_approx |         0 |
| log_lsn_buf_pool_oldest_lwm   |          0 |
| log_flush_lsn_avg_rate        |       3801 |
+-------------------------------+------------+
11 rows in set (0.0006 sec)
```

When the appropriate InnoDB Metrics are enabled, it's also possible to get an overview of the Redo Log's usage and see where we are in relation to the soft and hard redo log logical capacity:

```
select concat(variable_value, " (",
              format_bytes(variable_value),")") innodb_redo_log_logical_size,
       concat(round(count*8/7), " (",
              format_bytes(round(count*8/7)), ")") soft_logical_capacity,
       concat(round(@@innodb_redo_log_capacity*29.8/32), " (",
              format_bytes(round(@@innodb_redo_log_capacity*29.8/32)) ,")") hard_logical_capacity,
       concat(@@innodb_redo_log_capacity, " (",
              format_bytes(@@innodb_redo_log_capacity) ,")") redo_log_capacity,
       concat(round(variable_value / (count*8/7)*100,2), "%") logical_used,
       concat(round(variable_value / (@@innodb_redo_log_capacity*29.8/32)*100,2), "%") hard_used
  from performance_schema.global_status
  join information_schema.innodb_metrics
 where variable_name like 'innodb_redo_log_logical_size'
   and name like 'log_max_modified_age_async';
```

When the appropriate InnoDB Metrics are enabled, it's also possible to get an overview of the Redo Log's usage and see where we are in relation to the soft and hard redo log logical capacity:

```
select concat(variable_value, " (",
              format_bytes(variable_value),")") innodb_redo_log_logical_size,
       concat(round(count*8/7), " (",
              format_bytes(round(count*8/7)), ")") soft_logical_capacity,
       concat(round(@@innodb_redo_log_capacity*29.8/32), " (",
              format_bytes(round(@@innodb_redo_log_capacity*29.8/32)) ,")") hard_logical_capacity,
```



```
+------------------------------+-------------------------+-------------------------+-------------------+---------------+-----------+
| innodb_redo_log_logical_size | soft_logical_capacity   | hard_logical_capacity   | redo_log_capacity | logical_used  | hard_used |
+------------------------------+-------------------------+-------------------------+-------------------+---------------+-----------+
| 6211072 (5.92 MiB)           | 7260453 (6.92 MiB)      | 7811891 (7.45 MiB)      | 8388608 (8.00 MiB) | 85.55%       | 79.51%    |
+------------------------------+-------------------------+-------------------------+-------------------+---------------+-----------+
1 row in set (0.0011 sec)
```

```
 where variable_name like 'innodb_redo_log_logical_size'
   and name like 'log_max_modified_age_async';
```

Recommendations

Not too small, not too big

Recommendations

It's not recommended to oversize the Redo Log Capacity.

Redo Log files consume disk space and increases the recovery time in case of a restart (`innodb_fast_shutdown=1`*) or a sudden crash.*

And it also slows down shutdown when `innodb_fast_shutdown=0`*.*

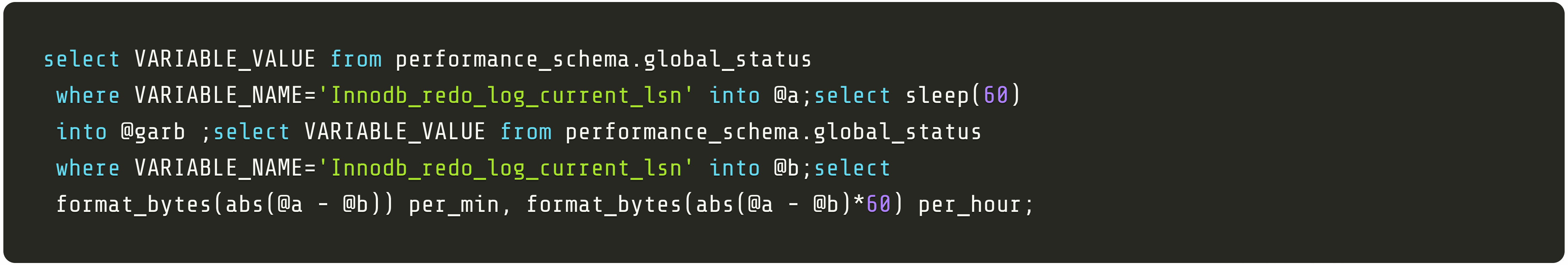
Recommendations (2)

During peak traffic time, you can get an estimation of the required amount for the Redo Log Capacity by running the query below (all in one single line):

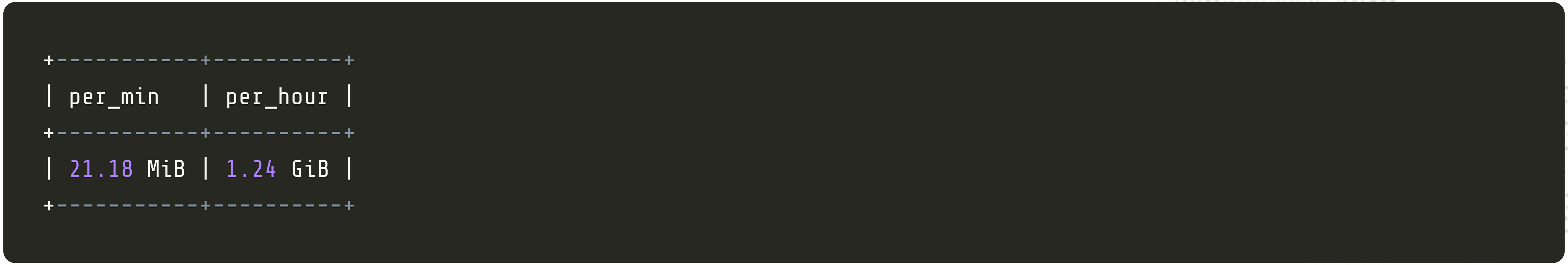
```
select VARIABLE_VALUE from performance_schema.global_status
 where VARIABLE_NAME='Innodb_redo_log_current_lsn' into @a;select sleep(60)
 into @garb ;select VARIABLE_VALUE from performance_schema.global_status
 where VARIABLE_NAME='Innodb_redo_log_current_lsn' into @b;select
 format_bytes(abs(@a - @b)) per_min, format_bytes(abs(@a - @b)*60) per_hour;
```

Recommendations (2)

During peak traffic time, you can get an estimation of the required amount for the Redo Log Capacity by running the query below (all in one single line):



```
select VARIABLE_VALUE from performance_schema.global_status
 where VARIABLE_NAME='Innodb_redo_log_current_lsn' into @a;select sleep(60)
 into @garb ;select VARIABLE_VALUE from performance_schema.global_status
 where VARIABLE_NAME='Innodb_redo_log_current_lsn' into @b;select
 format_bytes(abs(@a - @b)) per_min, format_bytes(abs(@a - @b)*60) per_hour;
```



```
+-----------+----------+
| per_min   | per_hour |
+-----------+----------+
| 21.18 MiB | 1.24 GiB |
+-----------+----------+
```

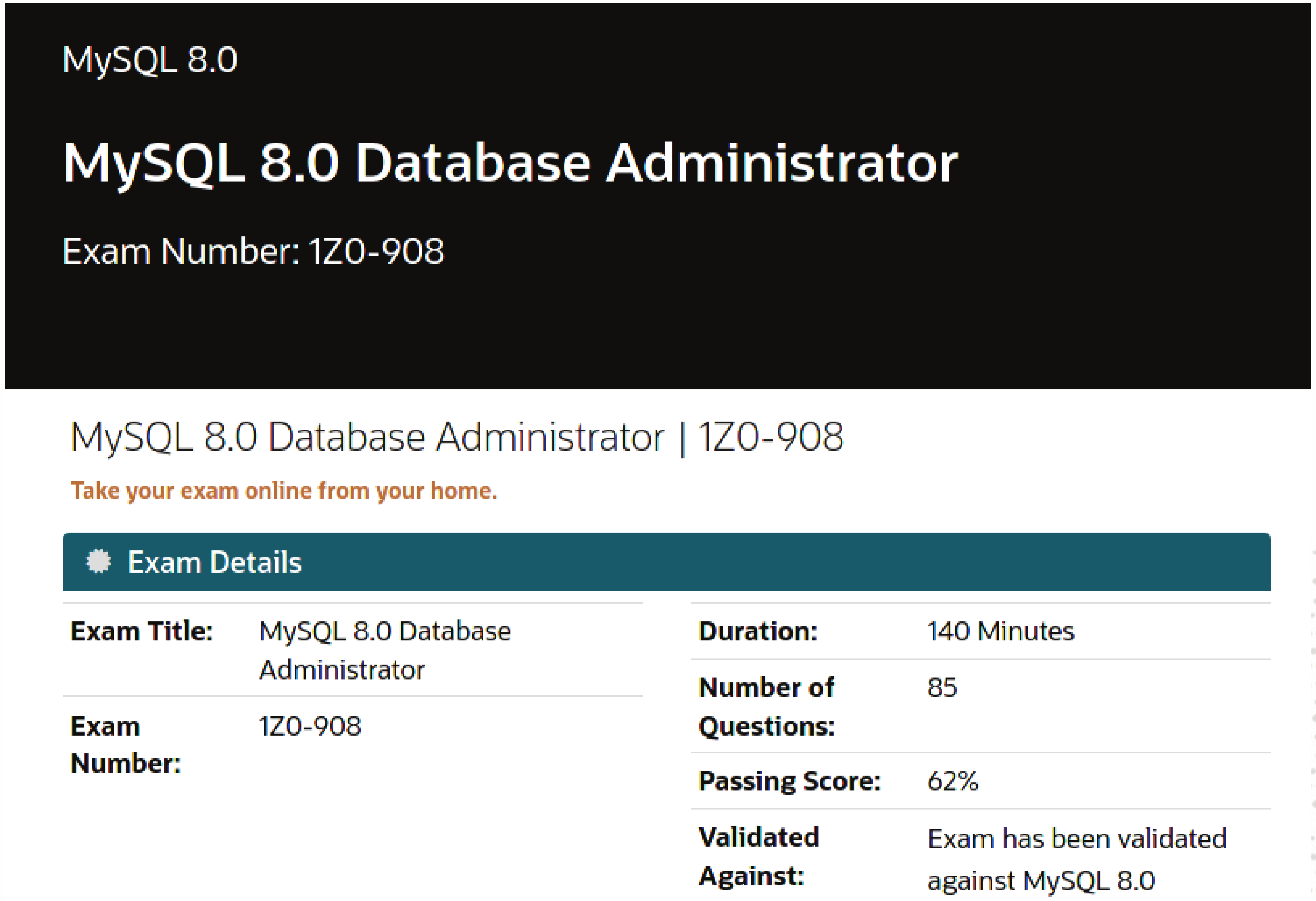
Share your ❤️ to MySQL

#mysql

Join our slack channel!

bit.ly/mysql-slack

MySQL 8.0 DBA Certification



MySQL 8.0

MySQL 8.0 Database Administrator

Exam Number: 1Z0-908

MySQL 8.0 Database Administrator | 1Z0-908

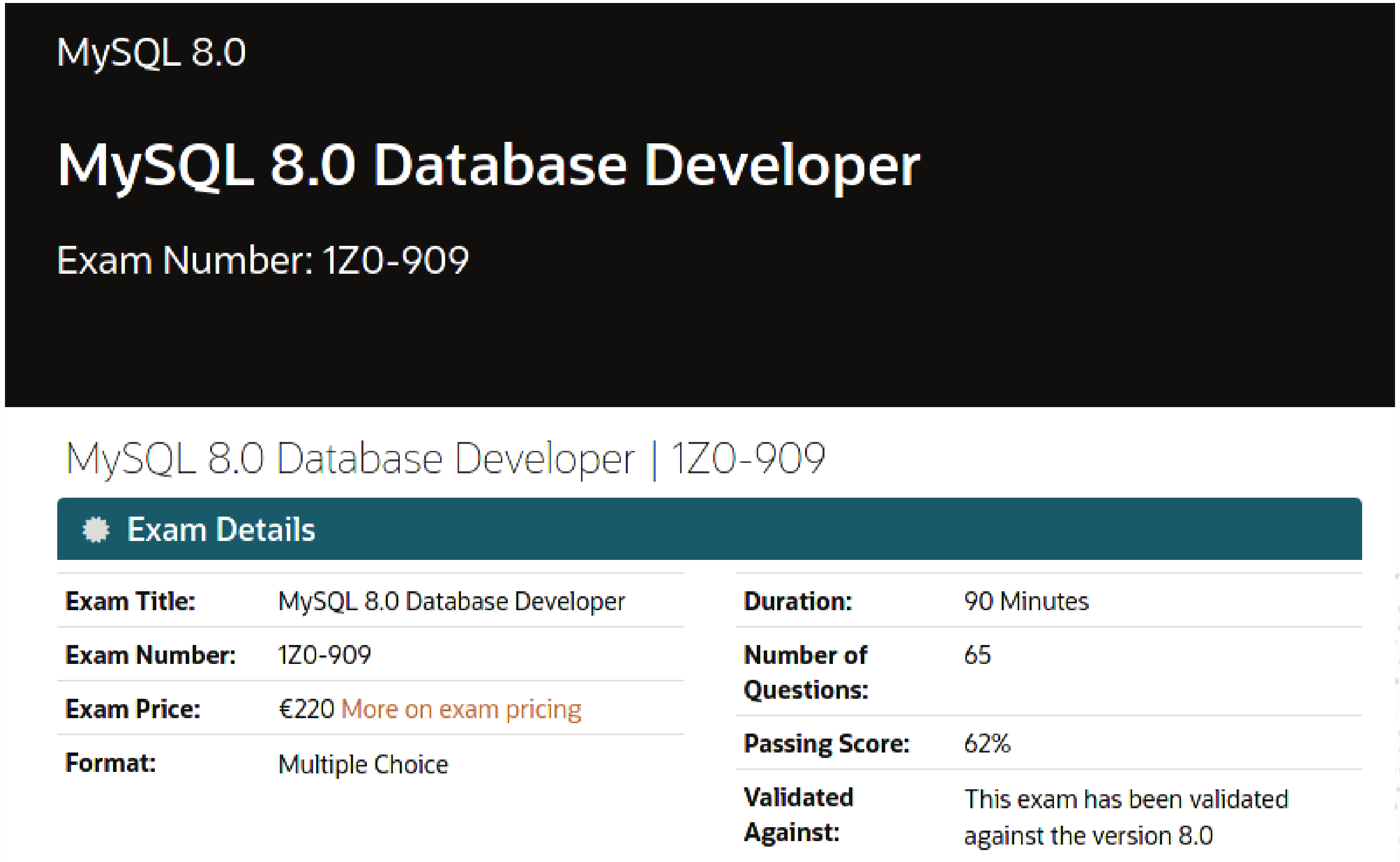
Take your exam online from your home.

Exam Details

Exam Title:	MySQL 8.0 Database Administrator
Exam Number:	1Z0-908

Duration:	140 Minutes
Number of Questions:	85
Passing Score:	62%
Validated Against:	Exam has been validated against MySQL 8.0

MySQL 8.0 Developer Certification



MySQL 8.0

MySQL 8.0 Database Developer

Exam Number: 1Z0-909

MySQL 8.0 Database Developer | 1Z0-909

Exam Details

Exam Title:	MySQL 8.0 Database Developer	**Duration:**	90 Minutes
Exam Number:	1Z0-909	**Number of Questions:**	65
Exam Price:	€220 More on exam pricing	**Passing Score:**	62%
Format:	Multiple Choice	**Validated Against:**	This exam has been validated against the version 8.0

Questions ?