The Wayback Machine - https://web.archive.org/web/20160417094246/https://blogs.oracl...

Introducing page_cleaner thread in InnoDB

By Calvin Sun on Apr 11, 2011

Note: *this article was originally published on http://blogs.innodb.com on April 11, 2011 by Inaam Rana.*

In MySQL 5.6.2 we have introduced a new background thread named the page_cleaner in InnoDB. Adaptive flushing of modified buffer pool pages in the main background thread, async flushing by a foreground query thread if the log files are near to wrapping around, idle flushing and shutdown flushing are now moved to this thread, leaving only the last resort sync flushing in foreground query threads. We've also added counters for these activities.

As page_cleaner is all about the flushing of dirty pages to disk it'll do you a world of good if you can go through this post where I have explained different types of flushing that happen inside InnoDB and the conditions that trigger flushing. The page_cleaner thread is only concerned with flush_list flushing (this may change in future releases). So let us dig a bit deeper into why flush_list flushing happens and why it would make sense to do this flushing in a separate thread. As is usually the case I have to skip some details to keep this note simple.

On a busy server flush_list flushing (which I'll simply call flushing from this point onwards) happens under four conditions. In order to understand these conditions let us familiarize ourselves with the concept of checkpoint_age. The checkpoint_age is the

difference between the current_lsn (the latest change to the database) and the last checkpoint_lsn (the lsn when last checkpoint happened). We obviously don't want to let this difference grow beyond the log file size because if that happens then we end up overwriting redo log entries before the corresponding dirty pages are flushed to the disk, losing the ability to recover them. In order to avoid the above situation we maintain two high water marks to indicate if we are nearing the end of reusable redo log space. Lets call these water marks async_water_mark and sync_water_mark, where the later represents a more urgent situation than the former. Now that we have clarified the checkpoint_age concept let us get back to the four conditions under which flushing of dirty pages happens:

1. `checkpoint_age < async_water_mark`
    - This condition means that we have enough reusable redo space. As such there is no hurry to flush dirty pages to the disk. This is the condition where we'd like our server to be most of the time.
    - Based on adaptive_flushing heuristics we flush some dirty pages in this state. This flushing happens in the background master thread.
    - During the flushing no other threads are blocked, so queries continue normally.

2. `async_water_mark < checkpoint_age < sync_water_mark`
    - As we move past the first water mark we try to bring some more urgency to our flushing. The query thread noticing this condition will trigger a flush and will wait for that flushing to end. This type of flushing does not happen in background.
    - Other query threads are allowed to proceed. Therefore we call it async flushing because only the query thread that is doing the flushing is blocked.
3. `checkpoint_age > sync_water_mark`
    - This is like a panic button. We have very little reusable redo log space available. The query thread detecting this condition immediately starts flushing.
    - Other query threads are blocked. The idea is to stop the advance of checkpoint_age.
    - This type of flushing not only happens in foreground it actually tends to bring the whole system to a stall.
4. `%n_dirty_pages > innodb_max_dirty_page_pct`
    - %age of dirty pages in the buffer pool exceeds the user settable value of innodb_max_dirty_page_pct.
    - The flushing happens in the background master thread and is non-blocking for query threads.

The page_cleaner thread:

As explained above flushing can happen in the query thread e.g.: the async and sync flushing. It can also happen in the background master thread e.g.: the adaptive flushing and the max_dirty_page_pct flushing. There are two issues with this scheme. First, the

master thread is also tasked to do other background activities. It has to do change buffer merges and possibly purge (though starting with 5.5 we have an option to use a dedicated thread for purge and in 5.6.2 we can even have multi-threaded purge). Under very heavy workload it is possible that the master thread is unable to find enough time to flush dirty pages. The second issue is with async flushing. It should be a background task but it is executed in the query thread. While other threads are allowed to proceed, the unfortunate thread which detects the condition is blocked on a huge dirty page flush.

To address these two issues we came up with the idea of having a dedicated background thread and named it the page_cleaner thread. All background flushing activity previously done in the master thread is off loaded to the page_cleaner. Also, the async flushing is now a background task performed in the page_cleaner thread. Query threads are only ever blocked for flushing if we cross the sync flushing water mark. The page_cleaner thread wakes up every second, checks the state of the system and performs the flushing activity if required. The flushing that happens when the server is idle or at shutdown is now also done by the page_cleaner thread.

Finally we have added some counters to the innodb_metrics table related to the above four types of flushing to give you a picture of how your system is behaving.

```
mysql> select name, comment from information_schema.innodb_metrics where name like
'buffer_flush_%';
+-----------------------------------+----------------------------------+
| name | comment |
+-----------------------------------+----------------------------------+
| buffer_flush_adaptive_flushes | Occurrences of adaptive flush |
| buffer_flush_adaptive_pages | Number of pages flushed as part of adaptive
flushing |
| buffer_flush_async_flushes | Occurrences of async flush |
| buffer_flush_async_pages | Number of pages flushed as part of async flushing |
| buffer_flush_sync_flushes | Number of sync flushes |
| buffer_flush_sync_pages | Number of pages flushed as part of sync flushing |
| buffer_flush_max_dirty_flushes | Number of flushes as part of max dirty page
flush |
| buffer_flush_max_dirty_pages | Number of pages flushed as part of max dirty
flushing |
```

Category: Features

Tags: none

Permanent link to this entry

« MySQL 5.6: Data... | Main | Information Schema... »
Comments:

Post a Comment:
Comments are closed for this entry.

About

This is the InnoDB team blog.

Search

Enter search term:

Search only this blog

Recent Posts

- Transaction life cycle improvements in 5.7.3
- MySQL 5.7.3: Deep dive into 1mil QPS with InnoDB & Memcached
- InnoDB Temporary Tables just got faster
- InnoDB 5.7 performance improvements
- InnoDB Redundant Row Format
- Redo Logging in InnoDB
- Introduction to Transaction Locks in InnoDB Storage Engine
- Data Organization in InnoDB
- Repeatable Read Isolation Level in InnoDB - How Consistent Read View Works
- Online ALTER TABLE in MySQL 5.6

Top Tags

- alter
- column
- compression
- consistent
- create
- data
- ddl
- drop
- extents
- failure
- format
- fts
- index
- innodb
- isolation
- level
- locks
- logging

- memcached
- mvcc
- online
- performance
- plugin
- read
- redo
- repeatable
- row
- segments
- snapshot
- stats
- table
- tables
- tablespace
- temporary
- transaction
- view
- wal

Categories

- Features
- InnoDB
- MySQL
- News
- Performance
- Private
- Tips

Archives

« April 2016

Sun	Mon	Tue	Wed	Thu	Fri	Sat
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Today

Bookmarks

- Events

- MySQL Support Blogs
- Oracle Technology Network
- Oracle Technology Network Blog
- Planet MySQL
- Recent Articles
- Technology Newsletters
- The State of the Dolphin

Menu

- Blogs Home
- Weblog
- Login

Feeds

RSS

- All
- /Features
- /InnoDB
- /MySQL
- /News
- /Performance
- /Private
- /Tips
- Comments

Atom

- All
- /Features
- /InnoDB
- /MySQL
- /News
- /Performance
- /Private
- /Tips
- Comments

The views expressed on this blog are those of the author and do not necessarily reflect the views of Oracle. Terms of Use | Your Privacy Rights | Cookie Preferences