本文是作者在ACMUG 2016 MySQL年会上的演讲内容，版权归作者所有。

中国MySQL用户组（China MySQL User Group）简称ACMUG。
ACMUG是覆盖中国MySQL技术爱好者的一个技术社区，是Oracle User Group Community和MairaDB Foundation共同认可的MySQL技术社区。

我们关注MySQL，MariaDB，以及其他一切周边的开源数据库和开源工具，我们交流使用经验，推广开源技术，为开源贡献力量。

我们是开放社区，欢迎任何关注MySQL及其相关技术的人加入，我愿意跟其他任何技术组织和团体保持沟通和展开合作。

我们期望在我们的活动中大家都能以开心的、轻松的姿态交流技术，分享技术，形成一个良性循环，从而每个人都可以有一份收获。

ACMUG的口号：开源，开放，开心

关注ACMUG公众号，参与社区活动，交流开源技术，分享学习心得，一起共同进步。

InnoDB Architecture and Tuning

Bin Su
Oracle, MySQL

Dec 2016

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

MySQL

Agenda

1. Indexes
2. Storage layout
3. Data Dictionary
4. Threads
5. Multi Versioning
6. Locking and Latching
7. Buffer Pool
8. Recovery
9. Other features
0. Outside InnoDB

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Indexes

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Overview

- Currently three types of Indexes in InnoDB
  - PK and Secondary Indexes with B-Tree index
  - Fulltext indexes (using AUX tables)
  - Spatial Index with R-tree
- InnoDB Internal Index
  - Adaptive Hash Index
  - Change buffering (organized as B-tree)

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

B-Tree Index

- **All data are organized in Clustered Index by PRIMARY KEY**
  - Sequential and ordered insertion (according to PK) is preferred
- **Secondary index refer to rows by PK**
  - PK keys are appended to secondary keys
  - Update PK also update Secondary indexes
  - Long PK are expensive
  - The appended PK could be significant when comparing to actual indexed column size
- **PK range scan is efficient while Secondary search probably requires two key searches (for MVCC)**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Btree Index(cont.)

- **Fast index build with bulk load**
  - New feature in 5.7
  - innodb_fill_factor to reserve space for future
  - Redo logging is disabled, pages have to be flushed
  - innodb_sort_buffer_size(1M)
- **Index Condition Pushdown(ICP)**
  - Fetaure available in 5.6
  - WHERE conditions can be evaluated immediately
  - Less accessing to Clustered index and InnoDB
- **Covering index scan**
  - Leverage the stored PRIMARY KEY on Secondary index

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Full-Text Index

- **Supported with 5.6 release**
- **An “Inverted Index” design**
  - FTS_DOC_ID column may be added during creating Full-text Index
  - Full-text search can only see the committed data
  - innodb_ft_total_cache_size / innodb_ft_result_cache_limit / innodb_ft_sort_pll_degree
- **Mecab parser plugin for Japanese**
  - Feature in 5.7
- **Ngram parser for CJK**
  - Feature in 5.7
  - Tokenizes text into sequence of n characters(ngram_token_size)

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Spatial Index

- New Feature in 5.7 releases
- Index on spatial data, which is stored as WKB(Well-Known Binary) in PK, but only MBR(minimal Bounding Rectangle) in the spatial index.
- Index is organized as R-tree, tree structure similar to B-tree as balanced tree.
- No composite Rtree index, can only index on one spatial column
- Every index entry is an MBR(Minimum Bounding Rectangle) of raw WKB data

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

InnoDB Internal Index - Adaptive Hash Index

- **Hash indexes on hot secondary index pages to speed up lookups**
  - It can be built on a prefix of the key defined
  - Not all workloads can take advantage of it, so benchmark with it first
  - Configure variable – innodb_adaptive_hash_index(on/off)
  - innodb_adaptive_hash_index_parts(8) to partition the search system into several parts to reduce contention

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Change Buffer

- **To reduce disk IO for updates to non-unique secondary index**
- **To buffer combinations of INSERT, DELETE and PURGE**
- **The buffer itself is a Btree, residing in system tablespace only**
- **Merge will be done when**
  - The related leaf page is read into buffer pool
  - In some background threads and slow shutdown
  - Thus merge may cause slowdown because of random IO
- **innodb_change_buffering / innodb_change_buffer_max_size**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Virtual Column

- **New feature in 5.7**
- **Generated Column**
  - b INT GENERATED ALWAYS AS(a + 1) VIRTUAL
  - AS(minute(now()) is not allowed
  - Can refer to other generated columns
- **Two types of Generated Column**
  - STORED(like normal column) or VIRTUAL
- **No need to store any Virtual Column in Clustered Index**
  - Virtual Columns can't be on PK
  - ADD or DROP VIRTUAL Columns would be very efficient, which can be done instantly without rebuild

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Virtual Index

- **Secondary indexes can be built on Virtual Columns**
  - One or more virtual columns
  - Combination of virtual or non-virtual columns
- **Virtual columns have to be materialized on secondary indexes**
- **Changes to base columns should be updated to virtual columns on secondary indexes**
  - MVCC, Rollback and Recovery supported
- **ADD or DROP virtual index is in-place operation**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Storage layout

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Tablespace

- **All data(pages) stored in tablespaces**
  - Either general or innodb-file-per-table tablespace
  - Currently, the clustered index and all secondary indexes of the same table should reside in the same tablespace
  - Logs are stored in redo/undo logs
- **Every tablespace maps to one or more data files**
- **System tablespace is a special one**
  - It includes change buffer, double write buffer, data dictionary tables, default undo log and even user tables etc.

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Tablespace Types

- **Innodb-file-per-table(default)**
  - CREATE/DROP table will create/drop the tablespace automatically
  - Different tables can be on different storage devices
  - More file handles would be used when there are lots of tables
  - Free spaces could only be reused by current table
- **General tablespace**
  - New Feature in 5.7
  - Multiple tables can reside in the same tablespace
  - It can be created in an independent directory
  - Less tablespace metadata in memory
  - Free space could not be released back to the OS

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Temporary Tablespace

- New feature in 5.7
- A dedicated tablespace(ibtmp1) created for all temporary tables, not on raw device
- It will be dropped at shutdown, and always be (re)created on startup unless it's RO
- No redo logs for temporary tables
- This tablespace doesn't support COMPRESSED row format.
- innodb_temp_data_file_path

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Undo Tablespace

- **InnoDB has undo log(rollback segment) to store the copies of data, set by innodb_undo_logs(128)**
  - Starting 5.6.3, InnoDB support multiple undo tablespaes
- **System tablespace always owns one undo log**
- **Temporary tablespace always own 32 undo logs, new in 5.7**
- **Undo tablespace is designed to reduce mutex contention to system tablespace only**
  - To hold undo logs from 33 to 128, in round robin way
- **Undo tablespaces can be assigned to SSD for better performance**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Truncate Undo Tablespace

- **New feature in 5.7**
- **Truncate suitable undo tablespace periodically if its size exceeds some threshold**
  - All pages in the tablespace should be free
  - There should be some undo logs available, at least 2 undo tablespaces and 35 undo logs
  - innodb_undo_log_truncate(off)/innodb_max_undo_log_size(1G)
- **The truncate is crash-safe, by writing DDL log file**
- **innodb_purge_rseg_truncate_frequency(128)**
  - The bigger, the slower to free undo logs

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

File Format

- **Two data file formats**
  - Antelope and Barracuda
  - innodb_file_format, which doesn't apply to general tablespaces
  - Antelope stores up to the first 768 bytes of variable-length column within Btree node, remainder are on the overflow pages
  - Barracuda will store all of the long value off-page, only have a 20-byte pointer to the overflow page; if it's <= 40 bytes, it's stored in-line
- **Note that character set will affect the final column length**
  - CHAR(255) CHARACTER SET utf8mb4: 255 * 4

ORACLE
Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

File Space

- **Each tablespace consists of “files”, which are called segments**
  - Different indexes have their own segments, two for each index
  - Segment can grow and shrink
  - A segment is number of extents. An extent, which could be of 1 / 2 / 4 MB, consists of consecutive pages
- **The smallest unit is page**
  - Same size in a single tablespace
  - 4K, 8K, 16K(default), 32K, 64K

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Space Allocation

- **Initial size could be rather small, for example, 7 or 8 pages**
  - No matter it's in innodb_file_per_table or general tablespace
  - Then it grows to one extent
  - Then 1 extent every time, if too big, 4 extents every time
- **Free pages are recycled within same segment**
- **Extent can be used by other segments if all pages within it are free**
- **Tablespace never shrinks itself**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Redo Logs

- **Log files to record changes to data, ib_logfile?**
  - innodb_log_files_in_group(2), used in a circular fashion
  - Log is organized by records, which is aligned 512 bytes
  - Log record is physical + logical
  - Nearly every record consists of (space, page_no) and the operation to do on the page
  - Log header has the last checkpoint information
- **Larger log files can ease disk IO on running, but may slow down recovery**
  - innodb_log_file_size(48MB), can be 1G etc.

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Redo Logs(Cont.)

- **The size of in-memory log buffer affects large transactions, the larger, the less disk IO**
  - innodb_log_buffer_size(16M), can be tens to hundreds
  - Can extend automatically if one redo log is too long(BLOBs)
- **innodb_flush_log_at_trx_commit(1)**
  - Could improve insert rate significantly
  - 0: Write out and flush approximately once per second
  - 1: Write out and flush at each commit
  - 2: Write out at each commit and flush approximately once per second
- **Resize redo logs offline**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Data Dictionary

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Data Dictionary

- **InnoDB now keeps its own data dictionary**
  - In INNODB_SYS_* tables
  - Could be out-of-sync with Server's data dictionary, such as after a crash of DDL
- **We are aiming to implement an universal data dictionary**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Threads

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Foreground Threads

- **User threads**
  - Using MySQL threads for execution, one thread per connection
- **Limit the user threads running concurrently**
  - To reduce contention in InnoDB
  - innodb_thread_concurrency, default 0, no checking
  - innodb_concurrency_tickets(5000), trade off between small and large transactions
  - Exceeded threads will wait in a FIFO queue

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Background Threads

- **Master thread**
  - Flush logs, change buffer merge, table cache cleanup, checkpoint, etc.
- **IO threads, max 130**
  - Read threads: innodb_read_io_threads(4)
  - Write threads: innodb_write_io_threads(4)
  - Change buffer thread for merging
  - Log thread for flushing logs
- **Purge, Cleaner, Lock timeout, Monitor threads, etc.**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Multi Versioning (MVCC)

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Overview

- **InnoDB keeps information/pointers about old versions of changed rows in its PK**
  - All information is stored in undo logs(rollback segment)
  - Only changed columns have to be logged
  - Reading or transaction rollback will read old versions from undo logs
- **InnoDB supports all four transaction isolation**
  - Serializable, Repeatable Read, Read Committed, Read Uncommitted

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Two Types of Reading

- **Consistent Read**
  - Use MV to present a snapshot at a point in time
  - No locking, no conflict with write, fast
  - See changes committed before the point of time
- **Locking Read(SELECT … FOR UPDATE/LOCK IN SHARE MODE)**
  - Lock the currently existing row
  - All locks will be released when commit or rollback
  - Slower due to locking, UPDATE has larger overhead
- **Results of consistent read and locking read are different**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Multi-Versioning

- **Every record in Clustered index has system fields**
  - DB_TRX_ID: the last transaction that modifies the row
  - DB_ROLL_PTR: points to the undo log record in undo logs
- **Records in Clustered index are updated in-place**
- **Records in Secondary index are delete-marked and new records are inserted directly**
- **To verify if a row is readable, DB_TRX_ID is checked first, and then proper version can be read from undo logs by referencing DB_ROLL_PTR**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Multi Versioning(Cont.)

- **Two types of undo logs**
  - Insert undo logs, needed by rollback, can be discarded on commit
  - Update undo logs, used by consistent read, can be discarded by purge
- **All undo logs for the same row are linked**
  - No limit on number of old versions → large undo logs
  - Intermediate versions have to be kept
  - Also records on indexes could not be purged
- **Prevent long running transactions, commit regularly**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Purge

- **Index records and old versions need to be removed**
  - When they are not needed for any active transaction
- **Purge threads will do the job automatically**
  - innodb_purge_threads(4)
  - Undo logs from one table would be purged by same purge thread
- **Slow down DMLs when purge threads are lagging**
  - innodb_max_purge_lag(0)
  - innodb_max_purge_lag_delay(0)

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Locking and Latching

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Lock Types

- **Intention Locks**
  - Table-level locks indicating how to lock rows in the table
- **Record Locks**
  - Lock the index record only
- **Gap Locks**
  - Lock the gap between index records, set after record moved
- **Next-key Locks**
  - Combination of record lock and a gap lock before the record
- **Insert intention Locks**
  - A type of GAP lock set by INSERT before insertion, this works only with GAP lock to prevent “Phantom reads”

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Gap Locks

- **The purpose is to block insertion, to prevent phantom rows, used in RR, serializable mode**
- **The whole range on a page is defined by “infimum” and “supremum”**
  - It's possible to lock the supremum and gap before it
- **A gap can span one or more possible index records, or none**
- **Not all queries have to place gap locks**
  - SELECT * FROM t1 WHERE building = 12 AND room = 3;
  - No gap lock for select by (building, room) if it's unique index
  - Scan using only building in condition requires gap lock

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Gap Locks(Cont.)

- S-lock and X-lock can be placed on the same gap by different transactions
- Gap locks can be merged when index records get deleted
- Gap locks could result in complex deadlock
- Can be disabled for index scans by using isolation level <= Read Commit

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

How to prevent INSERT

- **In RR or Serializable isolation level, next-key lock is used for searches and index scans**
  - Thus the gap before the selected records are locked too
- **INSERT sets Insert Intention Lock before insertion, which is a type of gap lock, and record lock on inserted row**
- **Insert Intention Locks within the same gaps don't conflicts with each other**
  - No conflicts on the gap and also no conflicts on different values

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

What locks to be set?

- **SELECT … FROM reads snapshot and sets no locks**
  - In SERIALIZABLE, shared next-key locks would be set
- **SELECT … FOR UPDATE/LOCK IN SHARE MODE**
  - S/X next-key locks on all index records scanned
  - locks are expected to be released for not matched rows
- **UPDATE and DELETE set locks on every index record that is scanned**
- **If no indexes on the table, every rows of the table become locked, which happens in RR and blocks all inserts**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Predicate Locks

- **New feature in 5.7**
- **It's only used for Spatial Index (Rtree) search only now, because for multi-dimension data, it's difficult to define the “next” key**
- **Predicate lock simply lock the MBR which is used for the query**
  - ST_Contains(@poly, point), lock the MBR(@poly)
  - Other transactions could not insert or modify a row which would have matched the condition
- **Predicate lock doesn't conflict with record lock or table lock**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Locking Wait

- **innodb_lock_wait_timeout(50 seconds), which applies to row locks only**
- **The length of time before giving up a lock request**
- **If timeout, the current statement is rolled back**
  - To rollback whole transaction, enable innodb_rollback_on_timeout(off)
- **If OLTP performance is cared, decrease the value, otherwise, increase it**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Deadlock Detector

- **Deadlock can happen in one or multiple tables**
  - Mainly because of updates, even if duplicate checking
- **How to prevent**
  - Keep the transaction as small as possible
  - Do locking in the same order, from small to big, etc.
- **Disable deadlock detector, to prevent contention on lock mutex**
  - New feature in 5.7
  - innodb_deadlock_detect(on)
  - innodb_lock_wait_timeout takes effect

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Latches

- **Two kinds of latches**
  - Mutex, exclusive
  - Rw-lock, includes S, X and SX lock
  - SX lock improves concurrency of index accessing, etc. which is new feature in 5.7
- **To get a latch, first Spin Waiting, then Sleep and Wait**
- **innodb_spin_wait_delay(6), dynamic one**
  - Wait a random time between spinnings, to prevent unexpected cache invalidation
  - Set to 0 to disable it
- **Hot mutexes, log_sys->mutex, fil_system->mutex, etc.**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Buffer Pool

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Overview

- **An area in main memory for caching table and index data**
- **Ideally, the larger size the better**
  - Innodb_buffer_pool_size, which is dynamic
  - On a dedicated server, up to 80% of physical memory can be assigned to BP
- **One or more buffer pool instances, to improve concurrency**
  - Innodb_buffer_pool_instances(8), depending on number of cores
  - Pages are mapped randomly by hash function
- **Buffer Pool dump and restore for speedy warmup**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Buffer Pool Resize

- **This can be done online, new feature in 5.7**
- **Both increase and decrease buffer pool size are performed in chunks**
  - innodb_buffer_pool_chunk_size(128MB)
- **innodb_buffer_pool_size = N * (innodb_buffer_pool_instances * innodb_buffer_pool_chunk_size)**
- **Size decrement can only start if enough pages can be withdrawn, which means the pages should not be held by active transactions**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Buffer Pool Read

- **Data pages are generally read from disk into Buffer Pool by executing threads, which is synchronized read**
- **Change buffer merge thread also reads pages**
- **Read-ahead prefetches a group of pages**
  - If sequential or batch of pages are in BP
  - Linear, innodb_read_ahead_threshold(56)
  - Random, innodb_random_read_ahead(off)
  - Asynchronous read

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

LRU Algorithm

- **Buffer Pools maintains a list for all cached pages, which is a variation of LRU list, from new to old**
- **To read in a page, put it in the midpoint of the LRU list**
  - innodb_old_blocks_pct(37 or 3/8 of the list)
  - Make it as big as possible(95) to active as familiar LRU
  - innodb_old_blocks_time(1000), duration to stay in old list
- **First access to the page will move it to the new list**
- **Make scan resistance from large full table scans and read-ahead**
- **Scan the LRU tail to find free page for replacement**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Flushing

- **Dirty pages have to be flushed to disk**
  - Start if innodb_max_dirty_pages_pct_lwm(0) is set
  - Try to keep innodb_max_dirty_pages_pct(75)
- **InnoDB uses redo logs in a circular fashion, so before reusing part of the logs, all related page changes have to be flushed to disk**
  - Sharp checkpoint
  - It affects performance significantly in a write-intensive workload

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Adaptive Flushing

- **Based on the number of dirty pages and redo logs generation rate**
  - innodb_adaptive_flushing(on)
  - Decide how many dirty pages to flush per second to smooth the overall performance
- **Flushing is done by page cleaner threads**
  - innodb_page_cleaners(4), not bigger than BP instances
- **Start flushing if innodb_adaptive_flushing_lwm(10)**
  - (Current LSN – Oldest dirty pages' LSN) / LOG CAPACITY
- **How often to adjust flushing: innodb_flush_avg_loops(30)**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Adaptive Flushing(Cont.)

- **innodb_lru_scan_depth(1024)**
  - How deep page cleaner thread will examine the tail of LRU
- **innodb_io_capacity limits the pages to be flushed, evaluated about every second**
  - Should be comparable with the capable of disk IO per second
  - Set higher if disk is fast or it's a write workload
  - But don't set too high (>=20000) unless it's necessary
  - innodb_io_capacity_max could be double
- **innodb_flush_neighbors(1)**
  - Whether to flush (contiguous) dirty pages
- **innodb_flush_method, O_DIRECT prevents double buffering**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Page Checksums

- **To detect if page is corrupted**
  - Enable by innodb_checksums(ON)
- **Calculated and updated when writing pages out**
- **Checked when page is read into BP**
- **Overhead is for sure**
- **There are kinds of algorithms**
  - innodb_checksum_algorithm
  - 'Crc32' is faster than 'innodb'

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Double Write Buffer

- Pages could be partially flushed out to disk, which results in inconsistent pages
- Double write buffer resides in system tablespace
- Disk overhead and mutex contention
- Disable double write buffer if FS supports atomic writes, like Fusion IO NVMFS

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Recovery

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Checkpoint

- **Fuzzy Checkpoint**
  - Dirty data pages would not be written out explicitly
  - Just try a sync on possibly cached pages if necessary
  - Redo logs will be flushed out
- **Sharp Checkpoint**
  - Since redo log files have to be reused, so once no more rooms in the files, some redo logs should be freed
  - At the mean time, related dirty pages have to be flushed out

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Shutdown

- **Generally, dirty pages and redo logs are flushed during shutdown**
- **innodb_fast_shutdown(1)**
  - 0: slow(clean) shutdown, a full purge and change buffer merge
  - 1: fast shutdown, skip above two operations
  - 2: like a crash, just flush out the redo logs
- **Basically, the faster server shuts down, the slower it restarts**
- **Please do a slow shutdown before upgrade/downgrade**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Redo Recovery

- **Necessary after crash and fast_shutdown=2**
- **Start if redo logs found, find the latest checkpoint**
  - Find all tablespaces
  - Applied pages in double buffer to tablespaces
  - Scan and collect redo logs from latest checkpoint to the end
  - Apply logs to data pages if necessary(LSN), no more logs written
- **Factor of recovery time**
  - Number of redo logs have to be applied
  - Number of dirty pages to be read

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Undo Recovery

- To logically rollback all not committed transactions
- Undo logs in undo tablespaces are also recovered in redo recovery
- Start after redo recovery finishes, resurrect all uncommitted transactions
  - Roll back DD transactions one by one
  - Then roll back user transactions
- Change buffer merge and purge delete-marked records will go on too

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Other features

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

BLOB

- **Storage depends on the row format**
  - If it's fully on overflow pages, it won't be read unless they are touched by the query.
  - The shorter the row, the more rows in a page
- **To fetch/update partial blob could be inefficient**
  - Especially when the BLOB is compressed
- **Consider if storing blob into separate table is necessary**
- **In the same row, one large blob could be faster than several medium ones**
- **Many blobs can result in fragments and waste**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Native Partitioning

- **Feature in 5.7**
- **Data are partitioned across every partition, each partition has the same table structure**
  - In-memory objects consumption
  - More efficient if part of data are accessed only
- **Types of partition: RANGE/LIST/HASH/KEY**
- **Different partitions can reside in different TABLESPACE and DATA DIRECTORY**
- **ALTER PARTITION**
  - Pay attention to operations which will copy data row by row, like REORGANIZE/COALESCE, etc.

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Table-level Compression

- **ROW_FORMAT/KEY_BLOCK_SIZE**
  - These are applied to the whole table/tablespace
- **Lossless zlib which implements LZ77**
- **To avoid recompression and Index page splits**
  - Delete-marked / modification log / padding
  - innodb_compression_failure_threshold_pct(5)
  - innodb_compression_pad_pct_max(50)
- **Buffer Pool maintains both compressed/uncompressed pages, eviction depends**
- **Compressed pages could be written to redo logs**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Transparent Page Compression

- Feature in 5.7
- It relies on sparse file and “hold punching” support
- It takes effect when sizeof(compressed page) <= innodb_page_size - file_system_block_size
- If compression fails, write the page out as is, otherwise, compress and release empty ending blocks
- ZLIB and LZ4 are now supported
- It doesn't work with table-level compression
- Currently, only innodb-file-per-table is supported

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Transparent Page Compression(Cont.)

- **In Windows, make NTFS Cluster Size smaller**
- **Trade off between large page sizes and write amplification**
- **It can be observed by innodb_sys_tablespaces**
  - FS_BLOCK_SIZE: File system block size
  - FILE_SIZE / ALLOCATED SIZE
- **A table can consist of pages with different compression settings**
  - OPTIMIZE TABLE t

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Encryption

- **Page-level encryption which only supports innodb-file-per-table for now, feature in 5.7**
- **Two encryption keys**
  - master key and tablespace key
  - Master key periodic rotation is rolled forward
  - Keyring_file and keyring_okv plugins
- **Encryption and decryption happen during IO**
- **Advanced Encryption Standard (AES) block-based encryption**
  - Electronic Codebook (ECB) / Cipher Block Chaining (CBC)
- **Altering the encryption attribute requires COPY**
- **Encryption is done after compression**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Outside InnoDB

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Environment

- **Hardware**
  - More powerful CPU
  - More memory
  - Proper malloc() lib, like jemalloc in Linux
  - SSD/Fusion IO
- **Software**
  - OS
  - File System: ZFS, EXT4, etc.
  - MySQL(different versions)

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Monitor

- **Statistics**
  - Tables in INFORMATION_SCHEMA show lots of statistics/options of all TABLES/TABLESPACES, etc.
- **Dynamic statistics**
  - SHOW ENGINE INNODB STATUS
  - SHOW STATUS LIKE '...'
  - INFORMATION_SCHEMA.inntdb_metrics table has all counters for InnoDB status and more
  - Records of tables in Performance Schema, monitor ALTER TABLE
  - MySQL Enterprise Monitor
- **Also monitor OS status**

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Benchmark

- **Do benchmark before running servers online**
- **Some generic benchmarks**
  - Sysbench
  - DBT2
  - LinkBench
  - ...

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

A sample my.cnf

innodb_file_per_table = 1
innodb_log_file_size = 1024M
innodb_log_buffer_size = 64M
innodb_log_files_in_group = 3 / 12 / ...
innodb_checksum_algorithm = none / crc32
innodb_doublewrite = 0 / 1
innodb_flush_log_at_trx_commit = 2 / 1
innodb_flush_method = O_DIRECT
innodb_use_native_aio = 1
innodb_adaptive_hash_index = 0
innodb_spin_wait_delay = 6
innodb_adaptive_flushing = 1

innodb_flush_neighbors = 0
innodb_read_io_threads = 16
innodb_write_io_threads = 16
innodb_io_capacity = 15000
innodb_max_dirty_pages_pct = 90
innodb_max_dirty_pages_pct_lwm = 10
innodb_lru_scan_depth = 4000
innodb_page_cleaners = 4

innodb_purge_threads = 4
innodb_max_purge_lag_delay = 30000000
innodb_max_purge_lag = 1000000

ORACLE

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Thanks

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.