



# INNODB<sup>®</sup>

Transactional Storage for MySQL  
FAST. RELIABLE. PROVEN.

## Crash Recovery and Media Recovery in InnoDB

MySQL Conference, April 2009

Heikki Tuuri, CEO, Innobase Oy

**INNOBASE**

# Topics

- Why Does MySQL/InnoDB Crash?
- Redo Logging
- Checkpointing
- Rolling Back Incomplete Transactions
- Warming Up the Buffer Pool
- Recovery Time, Benchmarks
- Media Recovery =  
Recovering From a Backup

# Topics (continued)

- You Must Have Binlogging Turned On!
- Statement-Based and Row-Based Binlogging
- Use a Backup of the InnoDB TableSpaces and Roll Forward Using the Binlogs
- Cold Backup
- InnoDB Hot Backup

# Why Does MySQL/InnoDB Crash?



- Most crashes are from InnoDB or MySQL bugs
- Hardware failure
- Operating system bugs
- Power blackout
- Operating system crashes and the disk 'fakes the cache flush'
- Broken disk, corrupted files
- The last 2 types require 'media recovery'

# Redo Logging

- The redo log remembers EVERY operation on any page in the database
- Redo log record format: <space id, page no, operation code, data>
- An example of a redo log record:  
    <0, 1234, 'insert',  
        'after record at offset 5444',  
        '(25, heikki, ...)'>
- 'Physiological' logging, per-page

# Redo Logging (continued)

- Physiological means that the log record is per page and it codes the page operation in a concise way:
  - 'Reorganize page 1234'
  - 'Delete all records on page 1234 after position 6543'
  - - ...

# Checkpointing

Checkpoint is a 'log sequence number' LSN such that

- The data pages in the files contain all changes to the database earlier than LSN
- InnoDB keeps buffer pool pages in a 'flush list', ordered by oldest not-yet-flushed modification

# Checkpointing (continued)

- InnoDB's redo log files have a fixed capacity
- The 'age' of the latest checkpoint must not exceed this capacity
- If the checkpoint age would become too old, InnoDB writes the oldest pages from the flush list to the data files

# Doublewrite Buffer

- What if a 16 kB data page write fails in the middle of the disk write?
- To recover from that, InnoDB has a 2 MB 'doublewrite buffer'
- First write the flushed pages sequentially to the doublewrite buffer
- After that, write the pages to their correct positions
- Crash recovery checks page consistency

# Doublewrite Buffer (continued)

- In theory, the overhead of the doublewrite relative to the actual disk writes is small, because the write to the doublewrite buffer is sequential
- In practice, the overhead can matter
  - If `innodb_doublewrite` is enabled (the default), InnoDB stores all data twice, first to the doublewrite buffer, and then to the actual data files
  - Disable doublewrite with `--skip-innodb_doublewrite` for benchmarks or cases when top performance is needed rather than concern for data integrity or possible failures

# Rollback of Incomplete Transactions

- Incomplete transactions are rolled back in crash recovery
- InnoDB has a list of 'undo logs' where incomplete transactions are listed
- In crash recovery, those transactions are rolled back in background
- Incomplete transactions hold locks which can disturb normal processing

# Rollback of Incomplete Transactions (continued)

- XA standard for distributed transactions introduces 'PREPARED' transactions
- PREPARED = 'ready to commit'
- The XA coordinator decides whether to roll back or commit a PREPARED transaction
- Normally, XA is hidden from the user: MySQL acts as the coordinator behind the scenes

# Warming up the Buffer Pool

- Resuming normal operation after a crash is slow because data must be loaded in the buffer pool
- With a 16 GB buffer pool this can easily the warmup can easily take 60 minutes
- Jeremy Cole and Kevin Burton have suggested batched warmup methods to speed up the warmup phase
- InnoDB could dump a list of pages in the buffer pool to help the warmup

# Log File Size: Performance Versus Fast Crash Recovery

- Big InnoDB redo log files improve database performance: less flushing at checkpoints is needed
- The downside: big redo log files make crash recovery take longer
- InnoDB allows total redo log file size in range 2 MB - 4 GB
- For a big installation, 2 redo log files, 1 GB each is often an optimal choice: crash recovery time stays tolerable

# Crash Recovery Benchmark

- Michael Izioumtchenko from the InnoDB QA team has run some crash recovery test times on a DBT2 workload
  - Test: start dbt2 run, abort in 10m, recover
  - dbt2 in a 16 GB buffer pool, 4 GB total redo log file size, 50 warehouses
  - Latest InnoDB Plugin development version:  
1546304262 redo, 7m scan, 30.5m apply -> 37.5m

# Why Does an InnoDB Database Get Corrupt?



- Sorting order changes make InnoDB B-trees look corrupt
- Bugs in the purge can make secondary indexes inconsistent
- Serious corruption from OS file system bugs

# Why Does an InnoDB Database Get Corrupt? (continued)

- If the disk subsystem fakes the flush of files to persistent storage, then an operating system crash or a power blackout can cause serious corruption
- Disk failure causes serious corruption or loss of files

# Media Recovery

- What do you need:
  - A backup of the database files: cold or hot
  - MySQL's archived binlog files

# Statement-based and row-based binlog

- You need the binlog to repeat the database operations after the backup was taken
- Statement-based binlog logs whole SQL statements:  
`INSERT INTO t SELECT * FROM s`
- Row-based binlog logs operation per row:  
`INSERT INTO t VALUES ('some_row_from_table_s')`

# Statement-based and row-based binlog (continued)

- If you use the READ COMMITTED isolation level or `innodb_locks_unsafe_for_binlog` in `my.cnf`, you need 'row-based binlog'
- Otherwise, MySQL's default, compact 'statement-based binlog' suffices
- Why? Because if there is less locking (READ COMMITTED), then you need more binlog information to repeat accurately  
`INSERT INTO t SELECT * FROM s`

# What Is an InnoDB Backup?

- Copies of InnoDB data files in a 'consistent' state
- Copies of `ib_logfiles` that match the data files
- Copies of MySQL's `.frm` files for each InnoDB table

# Cold Backup



- Shut down mysqld
- Copy ibdata files and possible .ibd files
- Copy ib\_logfiles
- Copy the .frm files for InnoDB tables

# Hot Backup



- Buy the InnoDB Hot Backup program from <http://www.innodb.com> NOW!
- Or use a file system snapshot mechanism like LVM to make a snapshot of relevant InnoDB/MySQL files (this is not tested by Innobase)

# InnoDB Hot Backup

- Backs up a RUNNING database without disturbing it
- Copies the data files
- Archives the InnoDB redo log that accumulated during the backup run
- Perl script innobackup backs up also .frm files

# InnoDB Hot Backup (continued)

- `ibbackup --apply-log` applies the archived InnoDB redo log to the backup to make it consistent

# File System Snapshots

- The principle is that a snapshot is like an InnoDB database that crashed: InnoDB's crash recovery will bring it to a consistent state
- Problems:
  - File system snapshots may not be consistent over disk partitions
  - File system snapshots may cause excessive file system log generation

# The Last Resort - Table Dumps

- Hot and cold InnoDB backups are 'binary backups' - they are complex binary files with complex data structures
- There might be some hidden corruption in a binary backup
- To play safe, best to take SQL dumps of important tables from time to time
- Table dumps are less likely to contain hidden corruption - they are human-readable

# Point-In-Time Recovery

- Suppose that you have a consistent backup
- Start mysqld on it
- Use the program 'mysqlbinlog' to pipe in the SQL in binlog files to the backup
- Voila! You have the database recovered to a specific point of time

# Point-In-Time Recovery (continued)

- Problem: re-running the SQL in the binlog can take a LOOONG time
- Since MySQL reruns the SQL in the binlog in a sequential fashion, it can take even longer than the original SQL execution did!



Q  
QUESTIONS  
&  
ANSWERS  
A

**INNOBASE**

# INNOBASE

an **ORACLE**® company

## INNODB

### INNODB Plugin

### INNODB Hot Backup

### Embedded INNODB

