ACMUG
中国MySQL用户组

版权声明

本次分享来源自中国MySQL用户组（ACMUG）

南京站活动，版权归演讲嘉宾所有。

ACMUG
中国MySQL用户组

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。

公众号ID：A_CMUG

MySQL 5.6 - 5.7

Detail of InnoDB features

赖铮- zheng.lai@oracle.com
Principle Software Developer

ORACLE

Program Agenda

- InnoDB Memcached
- GIS
- New Compression
- Transparent Encryption
- Download & Blogs

Feature 1: InnoDB Memcached Plugin

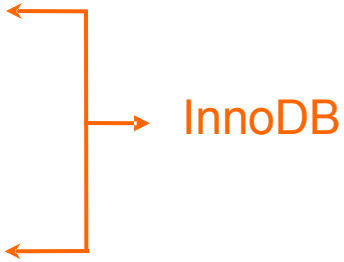
InnoDB Memcached

- InnoDB Memcached is a MySQL plugin that allows user to use Memcached APIs to directly fetch/store data from/to InnoDB Storage Engine. It bypasses the MySQL optimizer and SQL processing layer, operates on InnoDB directly with the InnoDB APIs.
- It mainly suitable for fast fetching/storing simple data
  - Ex. the user table in a social network site
  - relational model is not important to these simple data
  - Where NoSQL database are being used

InnoDB Memcached: Why?

ACMUG 中国MySQL用户组

- What would RDBMS do to process a query
  - Accepting SQL query packets
  - Parsing SQL statement
  - Optimizer generates a query plan
  - Query Processing process the plans
  - Get the metadata of the table, and open the table
  - Acquire Metadata Lock
  - Lock table with intentional lock
  - Row access/Index access find the row, lock rows
  - Extract column values and package return to QP
  - QP decides any further actions needed (order by etc.)
  - Unlocking tables
  - Closing tables
  - Returning results

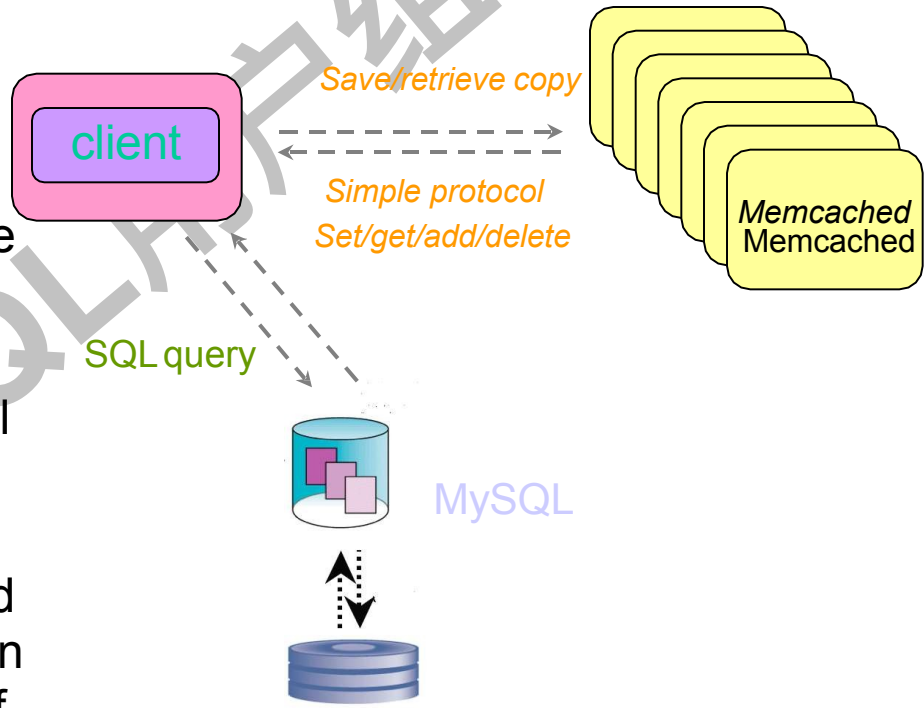


InnoDB Memcached: Why?

samples	%	app name	symbol name
259130	4.5199	mysqld	MYSQLparse(void*)
196841	3.4334	mysqld	my_pthread_fastmutex_lock
106439	1.8566	libc-2.5.so	_int_malloc
94583	1.6498	bnx2	/bnx2
84550	1.4748	ha_innodb_plugin.so.0.0.0	ut_delay
67945	1.1851	mysqld	_ZL20make_join_statistics P4JOINP10TABLE_LISTP4ItemP16st_dynamic_array
63435	1.1065	mysqld	JOIN::optimize()
55825	0.9737	vmlinux	wakeup_stack_begin
55054	0.9603	mysqld	MYSQLlex(void*, void*)
50833	0.8867	libpthread-2.5.so	pthread_mutex_trylock
49602	0.8652	ha_innodb_plugin.so.0.0.0	**row_search_for_mysql**
47518	0.8288	libc-2.5.so	memcpy
46957	0.8190	vmlinux	.text.elf_core_dump
46499	0.8111	libc-2.5.so	malloc

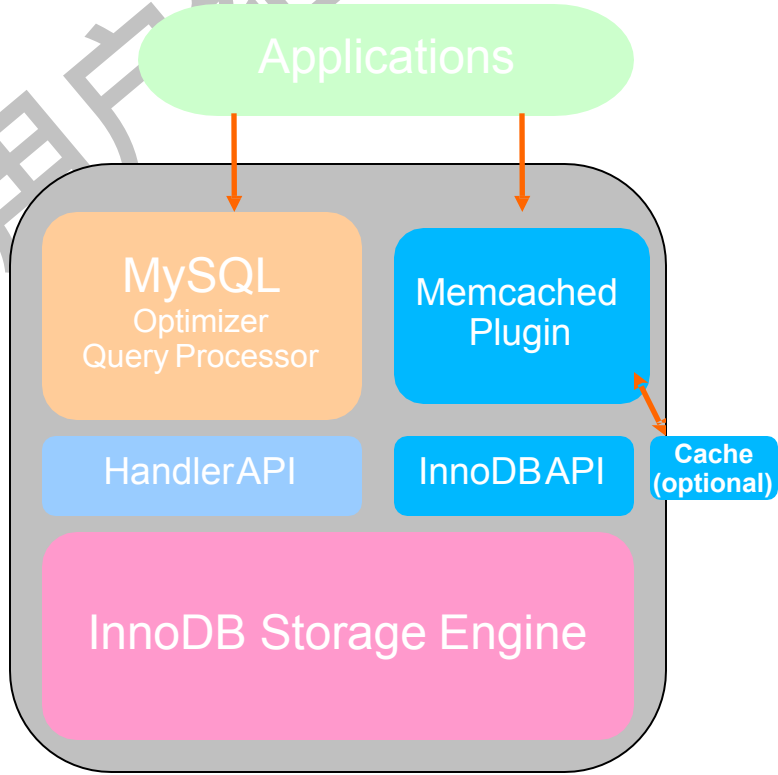
How Memcached is used with MySQL separately

- Memcached is in-memory key-value store for small data
- It is one of the most widely used In-Memory cache implementations for social network websites
- Memcached has a simple and open protocol as opposed to a rich client bound to a specific language and implementation makes it portable across a wide variety of languages and environments



InnoDB Memcached

- Access data/table with Memcached protocols as well as SQL
- Memcached access skips MySQL Optimizer and QP module as well as Handler API, but uses InnoDB internal API directly
- Skips opening/closing tables via MySQL function calls
- Can't access non-InnoDB storage engines



InnoDB Memcached: How to use?

- Configure the InnoDB memcached with “InnoDB Memcached System Tables”
- Install Memcached
  - *mysql> install plugin daemon_memcached soname “libmemcached.so”;*
- Using the Memcached

```
telnet 127.0.0.1 11211
  > set a11 10 0 9
  > 123456789
  > STORED
  > get a11
  > VALUE a11 0 9
  > 123456789
```

Operate with the InnoDB Memcached

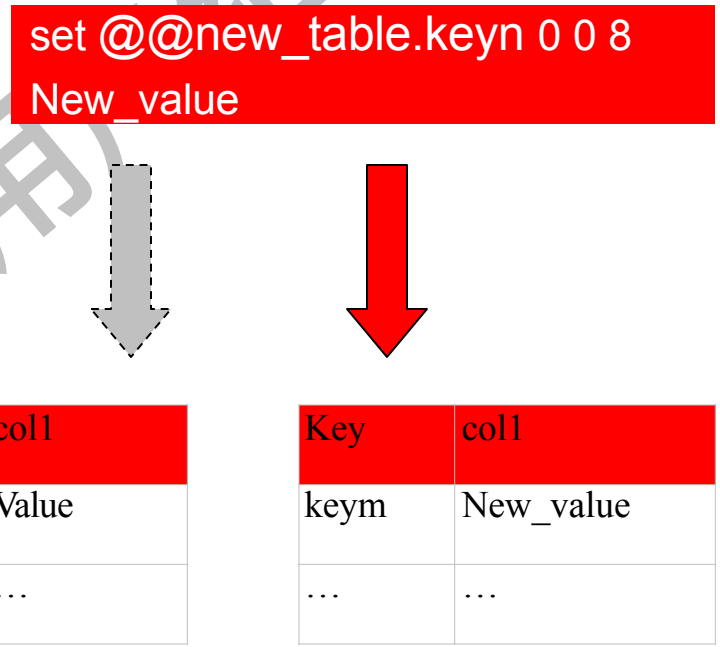
ACMUG 中国MySQL用户组

- All memcached commands are supported (mapped to InnoDB's select, insert, update, delete and truncate table commands

command	Description	Example	InnoDB action
get	*Reads a value*	*get mykey*	*Search (select)*
set	*Set a key unconditionally*	*set mykey 0 60 5*	*insert*
add	*Add a new key*	*add newkey 0 60 5*	*update*
replace	*Overwrite existing key*	*replace key 0 60 5*	*update*
append	*Append data to existing key*	*append key 0 60 15*	*update*
prepend	*Prepend data to existing key*	*prepend key 0 60 15*	*update*
incr	*Increments numerical key value by given number*	*incr mykey 2*	*update*
decr	*Decrements numerical key value by given number*	*decr mykey 5*	*update*
delete	*Deletes an existing key*	*delete mykey*	*delete*
Flush_all	*Invalidate specific items immediately*	*flush_all*	*truncate*
stats	*statistics*	*stats*	

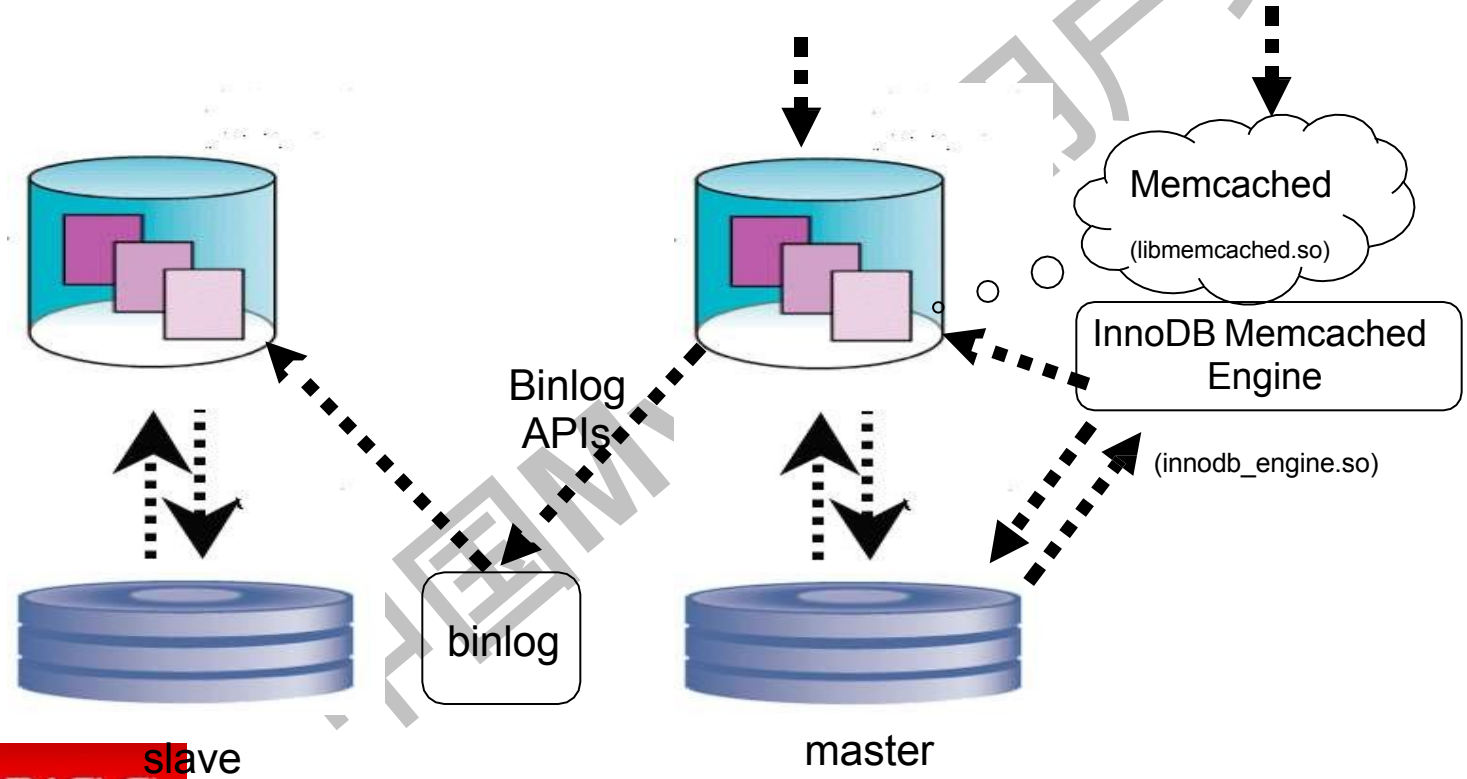
How to use InnoDB Memcached

- table must be registered in the “containers” table
- Call the “get” or “set/add” command with the new table mapping name, prefixed with “@@” symbol:
  - get @@new_table.key
  - Set @@new_table.key
- The connection’s table mapping with switch to “new_table”



InnoDB Memcached with Binlog

Handler APIs have to be used for Binlog



InnoDB Memcached: Performance

Leverages the read-only transaction optimizations in 5.7

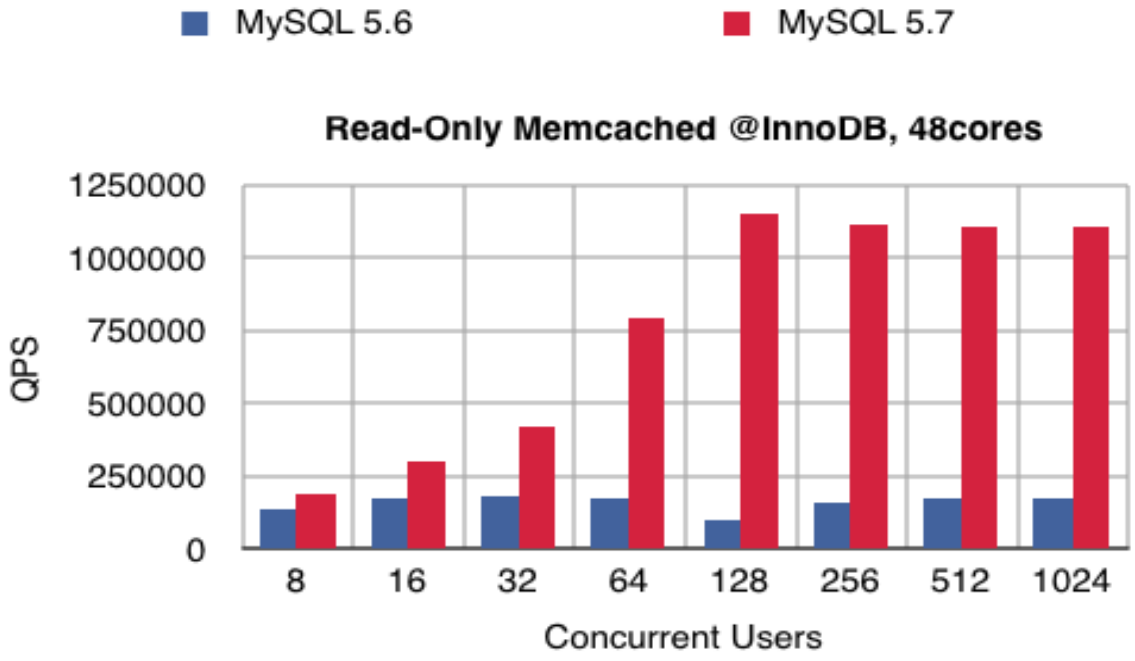
Fixed several bottlenecks in the Memcache and the plugin code

1.1 Million GET/s

Limiting factors were:

- The network
- Memcached client

InnoDB Memcached: Performance



ACMUG 中国MySQL用户组

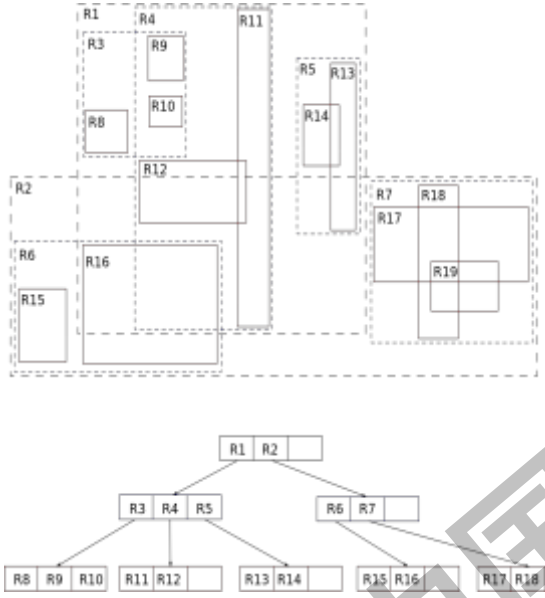
Feature 2: GIS

GIS

InnoDB supports Spatial Index from 5.7.5:

1. The spatial index is implemented as an R-tree, which is a special tree structure for spatial data
2. Can be indexed on all geometric type supported by MySQL
3. Supports transactions and MVCC just like other indexes in InnoDB
4. Isolation levels are also observed, using predicate lock preventing phantom at serializable reads

GIS: R-Tree



R-Tree

- A special data structure that used specifically for multi-dimension spatial data search.
- Queries more like: find object “within”, “intersects”, “touches” another spatial object
- In InnoDB, the tree is build on MySQL geometrics datatypes: POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, GEOMETRY

GIS: InnoDB Spatial Index as an R-tree

- R-tree in InnoDB
  - For InnoDB spatial Index, the index itself does not contain the actual data. The data is stored in the Primary Index of Table
  - The InnoDB spatial index consists of MBR(Minimum Bounding Box) of the object and its Primary Key.
  - Once the search locates the correct MBR for the queries, the real data is then retrieved

GIS: Predicate Lock

- No ordering for Geometry data
- GAP lock relies on ordering. So GAP lock does not work for R-tree
- Predicate Lock is used
  - Disadvantage on Generic Predicate lock
    - “Generic” Predicate locks are less efficient to set and check.

GIS: Predicate Lock

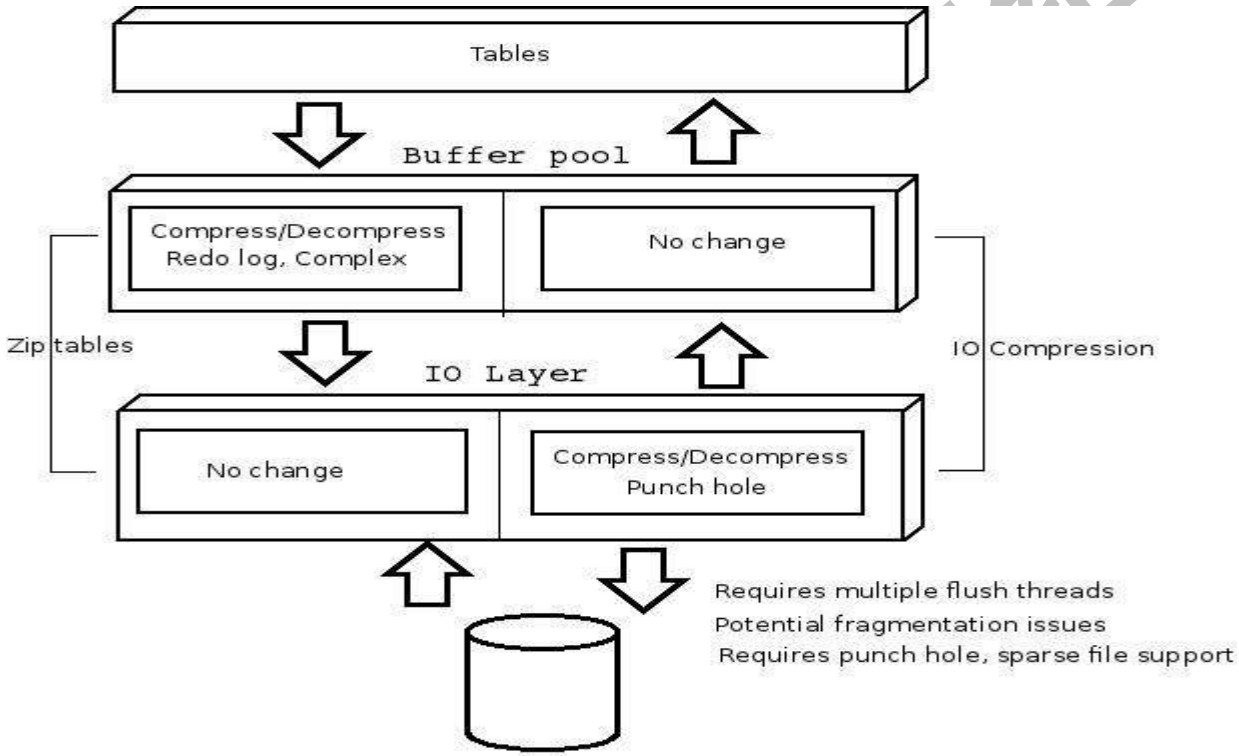
- Place a predicate lock by search
  - If a search operation's predicate is consistent with a node's
    BP, the predicate must be attached to the node.
- Test on predicate lock by insertion
  - An insert operation can therefore limit itself to checking only the predicates attached to its target leaf.
- The predicates and their node attachments are only removed when the

Feature 3: New Compression

Transparent PageIO Compression

Proof of concept patch originally from FusionIO
Currently Linux/Windows only
Requires sparse file support : NVMFS, XFS, EXT4, ZFS & NTFS
Linux 2.6.39+ added PUNCH HOLE support
Can co-exist with current Zip tables
Only works on tablespaces that are not shared
Doesn't work on the system tablespace

Transparent PageIO Compression



Zip vs Page IO compression

Zip compression

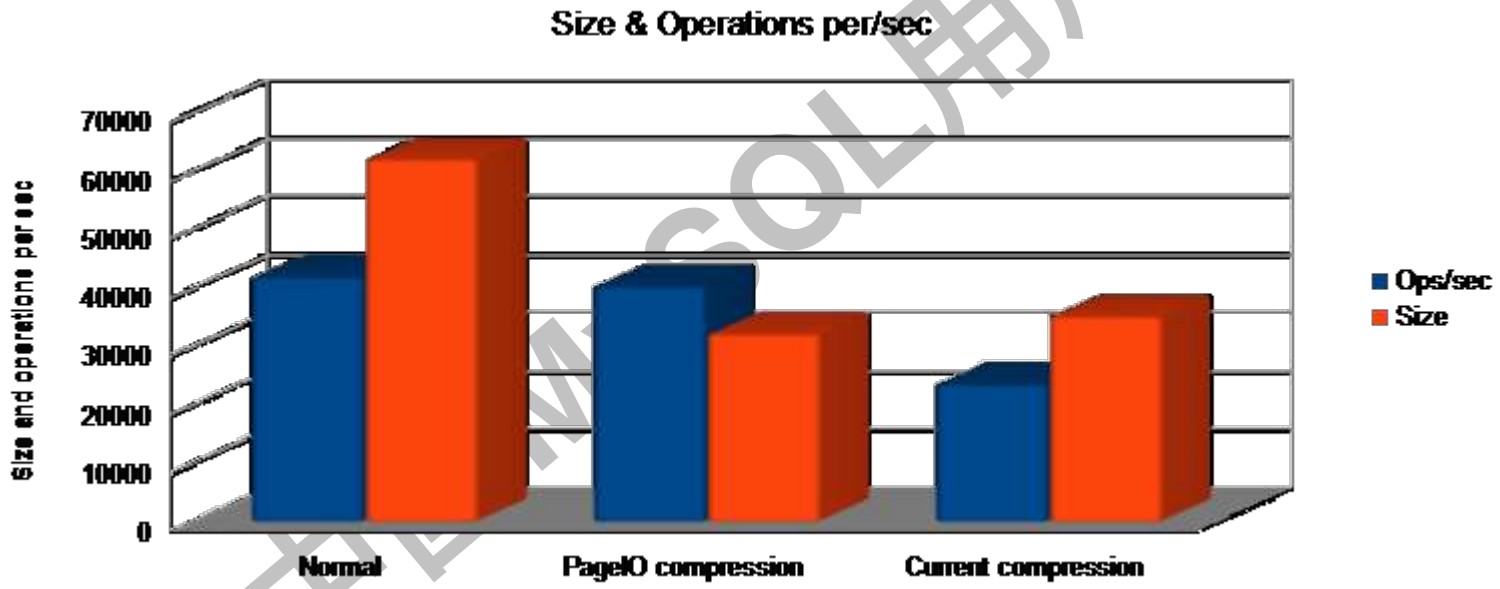
- Tested and tried, works well enough
- Complicates buffer pool code
- Special page format required
- No IO layer changes
- Algorithm supported - Zlib
- Can't compress system tablespace
- Can't compress UNDO tablespace

PageIO compression

- Requires OS/FS support
- Simple
- Works with all file types, system tablespaces
- Potential fragmentation issues
- NVMFS doesn't suffer from fragmentation
- Adds to the cost of IO
- Current algorithms are tuned to existing assumptions
- Requires multi-threaded flushing
- Easy to add new algorithms.

PageIO Compression Benchmark

FusionIO – 25G BP – maxid 50 Million 64 Requesters - Linkbench



Transparent PageIO Compression

Page IO Compression

Configuration options

```
CREATE TABLE T(C INT) ENGINE=InnoDB, COMPRESSION=”ZLIB”;
CREATE TABLE T(C INT) ENGINE=InnoDB, COMPRESSION=”LZ4”;
ALTER TABLE T COMPRESSION=”LZ4”
ALTER TABLE T COMPRESSION=”ZLIB”;
ALTER TABLE T COMPRESSION=”NONE”;
OPTIMIZE TABLE T;
```

Feature 4: Transparent Encryption

Transparent Data Encryption

Two-tier encryption key architecture

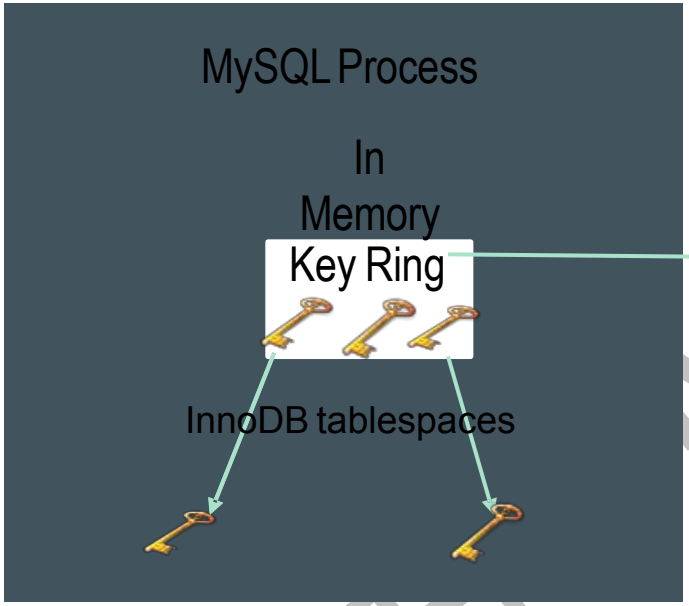
- Master Key
  - The One key that is used to control access to tablespace encryption
- Secret or Private key
  - Actual hidden keys that do the encryption of the tablespace data
  - Secret keys are never ever seen by users – only internal code.
- Innodb tablespaces will have master key to encrypt a private per tablespace encryption key
- Only the master key is Rotated
- The **keyring plugin** provides a way to retain or cache master keys

Transparent Data Encryption

- Provides easy administration of keys
- Enforces clear separation of keys from encrypted data
  - Huge data sets are too large to de/re-encrypt
- Provides assisted key rotation – without having to re-encrypt data.

Transparent Data Encryption

OKV,
KMIP Compliance Key
Key file



Put master keys
On MySQL Key Ring

Keys on the keyring are only accessible to internal Components Internal Code or Internal plugins

Key Rings are not persist – in memory and protected

Transparent Data Encryption

1: Keyring lib file is ready

2:Start server with options:
----early-plugin-load=keyring_file.so –keyring_file_data=./ring

3:Key ring plugin is loaded
Innodb is in per tablespace mode
> CREATE TABLE … ENGINE=innodb ENCRYPT=”Y”;
Success: the tablespace file for this table is now encrypted.

4:Master key rotation
> ALTER INSTANCE ROTATE INNODB MASTER KEY;
Success: master key has been rotated, all private keys of encrypted tables have been re-encrypted by new master key.

5:Import/export encrypted tablespace
On source server runs
> FLUSH TABLES t1 FOR EXPORT;
Copy t1.cfg,t1.cfp,t1.ibd to destination server
On destination server runs:
>ALTER TABLE t1 DISCARD TABLESPACE;
> ALTER TABLE t1 IMPORT TABLESPACE;

Download & Blogs

ACMUG 中国MySQL用户组

http://labs.mysql.com
http://dev.mysql.com/downloads/mysql/

http://mysqlserverteam.com/