All about MySQL Table Cache

Dmitry Lenev

Principal Software Engineer, Percona

About me

- Been working in MySQL Ecosystem since 2003
- Former MySQL Server Runtime developer in Oracle
  - Trigger support
  - Metadata locking subsystem
  - Part of New Data-Dictionary team
- Principal Software Engineer in Percona

Why Table Cache is needed?

A statement which uses a table needs to:

- Load table metadata from the data-dictionary (tables or .FRM)
- Open table in Storage Engine
  - Open files
  - Read headers
  - Read statistics
  - …

Expensive for each statement ⇒ caching!

What Table Cache contains?

- TABLE objects kept in cache represent:
  - open table
  - also cursor
  - record buffers
  - list of triggers

- Each connection and each use in statement need their own object:
  ⇒ Multiple TABLE objects for the same table can be needed at the same time
  ⇒ Lots of cache entries

- Cache entries can be large!

How it works?

- Keep open table entries in cache after statement end
- Next statements will re-use them if possible

- There is soft upper limit on cache size
  - If exceeded unused entries are exceeded according to LRU

- FLUSH TABLES and its variants wipe the cache

- Different connections/threads use different partitions of the Table Cache

Knobs

The main control knob: --table_open_cache option/variable

- Dynamic
- Default: 4000 (8.0)
- Affected by open_files_limit and max_connections

For bigger systems: --open_table_cache_instances option

- Start-up only
- Default: 16 (8.0)

⚠️ For MariaDB they work differently ⚠️

Gauges for cache effectiveness

Status variables:

- Open_tables
- Opened_tables
- Table_open_cache_hits
- Table_open_cache_misses
- Table_open_cache_overflows

Problems are often visible in P_S tables:

- ‘Opening tables’ state in performance_schema.processlist
- ‘stage/sql/Opening tables’ in events_stages_* tables

Gauges for memory consumption

Memory usage for TABLE objects reflected as ‘memory/sql/TABLE’ event in P_S memory usage tables:

```
SELECT * FROM
performance_schema.memory_summary_global_by_event_name
WHERE event_name = ’memory/sql/TABLE’;
```

Memory usage by triggers stored along with Table Cache entries is reflected as part of ‘memory/sql/sp_head::main_mem_root’ event:

```
SELECT * FROM
performance_schema.memory_summary_global_by_event_name WHERE
event_name = ’memory/sql/sp_head::main_mem_root’;
```

Gauges for cache contention

Contention on Table Cache partitions is visible as increased waits for LOCK_table_cache mutexes in P_S waits tables:

```
SELECT * FROM
performance_schema.events_waits_summary_global_by_event_name
WHERE event_name = 'wait/synch/mutex/sql/LOCK_table_cache';
```

Also often visible in P_S process and stages tables (look for 'Opening tables' state and stage).

New knobs and gauges in Percona Server 8.0.31

- On-demand loading of triggers for TABLE objects (allows to avoid loading them for read-only statements)
- --table_open_cache_triggers option/variable to limit the number of TABLE objects with triggers in cache:
  - Dynamic
  - Default: 524288 (i.e. disabled)
- Status variables:
  - Table_open_cache_triggers_hits
  - Table_open_cache_triggers_misses
  - Table_open_cache_triggers_overflows

Table Definition Cache

Introduced to avoid reading from .FRM file (now from the data-dictionary) each time a table is opened for Table Cache:

- Stores TABLE_SHARE objects representing table definitions
- Only one TABLE_SHARE object for each table

- There is soft upper limit on cache size
  - TABLE_SHARE objects considered as used if there are corresponding TABLE objects in the Table Cache
  - If exceeded unused entries are exceeded according to LRU

- FLUSH TABLES and its variants wipe out the cache

- Not partitioned, protected by single lock (infamous LOCK_open)

TDC - Knobs and Gauges

The only knob: --table_definition_cache option/variable

- Dynamic
- Default: 2000 (8.0)
- Affected by --table_open_cache value

Gauges:

- Status variables:
  - Open_table_definitions
  - Opened_table_definitions
- Memory consumption by TABLE_SHARE objects:

```
SELECT * FROM
performance_schema.memory_summary_global_by_event_name
WHERE event_name = 'memory/sql/TABLE_SHARE::mem_root';
```

Dictionary Cache

- Introduced in 8.0 as part of New Data-Dictionary project

- Stores data-dictionary objects of various types
- Has different partitions for different types

- For tables serves as a caching layer below Table and Table Definition Caches
- Used directly by some subsystems (e.g. InnoDB SE)

- For Table partition max capacity controlled by --max_connections
- Uses LRU strategy for eviction

- No opportunities for tuning, no observability (except profilers)!

- No issues in common use-cases known.

Typical Problems - Affected Workloads

Scenarios in which performance/scalability issues with Table Cache typically show up:

- Quick, small queries
  - PK lookups
  - Range lookups
  - Simple, well-indexed joins
  - …
- Lots of them
- Concurrency

Memory consumption issues can be observed for more generic workloads.

Problem 1: Cache is too small

What does user see?

- System doesn't perform/is slower than expected

Why?

- Working set doesn't fit into table cache
  - workload bigger than default cache size
  - misconfiguration
  - effect of open_files_limit

Indicators:

- table_open_cache_misses/overflows counters are high and growing
- open_tables value is close to table_open_cache

Solution: increase table_open_cache value!

Problem 2: Cache is wiped out

What does user see?

- Occasional drops in performance

Why?

- Table Cache is wiped out and has to be repopulated again
  - Due to FLUSH TABLES and variants (backup!)
  - I_S queries that need to open lots of tables (5.7)

Indicators:

- Spikes and drops in table_open_cache_misses/hits
- Occasional drops in open_tables value

Solution:

- Avoid FLUSH TABLES (xtrabackup, improved mysqldump –single-transaction support in PS)
- Upgrade to 8.0 (in case of I_S problem)

Problem 3: Cache is too big

What does user see?

- Server consumes too much memory (OOM, swapping)

Why?

- Table Cache occupies too much memory
  - Lots of connections/tables
  - Wide records
  - Triggers (many or big)

Indicators:

- ‘memory/sql/TABLE’ event in P_S.memory_summary_global_by_event_name
- ‘memory/sql/sp_head::main_mem_root’ (for triggers, includes routines!)

Solutions:

- consider decreasing cache size
- --table_open_cache_triggers in Percona 8.0.31
- tricks (change column types, move triggers bodies to routines)

Problem 4: Contention on Table Cache

What does user see?

- System doesn't perform well enough under concurrent load

Why?

- Lock protecting Table Cache partition becomes point of contention when
  - Size of table cache is sufficient
  - Workload consisting of short statements
  - Lots of connections (1K+)
  - Bigger machines (multi-CPU/cores)

Indicators:

- table_open_cache_misses is low
- 'wait/synch/mutex/sql/LOCK_table_cache' event in P_S.events_waits_summary_global_by_event_name

Solution: Increase --open_cache_table_instances (⚠️ MariaDB)

Table Definition Cache Problems

1. Cache is too small
   - Might occur (lots of tables, misconfiguration)
   - In practice hard to notice because of Table Cache presence
   - Increase --table_definition_cache value if really necessary
2. Cache is wiped out
   - Affects TDC similarly to Table Cache
   - Same advice as for Table Cache (avoid FLUSH TABLES, upgrade to 8.0)
3. Cache is too big
   - Mostly irrelevant, since there is only one object for each table
4. Contention on Table Definition Cache
   - Doesn't happen if Table Cache properly configured

Final word of caution

- Normally defaults of modern versions work well!

- Do not fiddle with Table Cache options unless you see a problem or have pretty special case (fast queries/ lots of connections/big working set, memory constraints)

Percona is a world-class open source database software, support, and services company focused on helping you scale and innovate with speed as you grow.

83M+
Software Downloads
TTM as of November 2022

100K+
Blog Views Per Month

800+
Customers

40%
YoY MRR Growth
As of Q4 2021

Trusted by...

More than a third of the Fortune 50

4 of the top 6 Retailers

3 of the top 5 Healthcare Companies

9 of the top 10 Tech Companies

6 of the top 10 Gaming Companies

4 of the top 5 Manufacturing Companies

Thank You!

Dmitry Lenev

Principal Software Engineer, Percona

dmitry.lenev@percona.com

percona.com