PERCONA

www.percona.com

Percona Server Documentation

Release 5.7.11-4

March 14, 2016

I Introduction **2**

1 The *Percona XtraDB* Storage Engine **3**

2 List of features available in *Percona Server* releases **4**
2.1 Other Reading . . . 6

3 *Percona Server* Feature Comparison **7**

4 Changed in Percona Server 5.7 **9**
4.1 Features removed from *Percona Server* 5.7 that were available in *Percona Server* 5.6 . . . 9
4.2 Changes in Percona Server 5.6 features . . . 9
4.3 Features available in *Percona Server* 5.6 that have been replaced with *MySQL* 5.7 features . . . 10
4.4 List of status variables that are no longer available in *Percona Server* 5.7 . . . 11
4.5 List of system variables that are no longer available in *Percona Server* 5.7 . . . 13
4.6 Features ported from *Percona Server* 5.6 to *Percona Server* 5.7 . . . 13

II Installation **15**

5 Installing *Percona Server* 5.7 **16**
5.1 Installing *Percona Server* from Repositories . . . 16
5.2 Installing *Percona Server* from a Binary Tarball . . . 23
5.3 Installing *Percona Server* from a Source Tarball . . . 23
5.4 Installing *Percona Server* from the Git Source Tree . . . 24
5.5 Compiling *Percona Server* from Source . . . 24
5.6 Building *Percona Server* Debian/Ubuntu packages . . . 24

6 Percona Server In-Place Upgrading Guide: From 5.6 to 5.7 **26**
6.1 Upgrading using the Percona repositories . . . 26
6.2 Upgrading using Standalone Packages . . . 28

III Scalability Improvements **32**

7 Improved Buffer Pool Scalability **33**
7.1 Version Specific Information . . . 33
7.2 Other Information . . . 33

8 Improved *InnoDB* I/O Scalability **35**
8.1 Version Specific Information . . . 35
8.2 System Variables . . . 35

IV Performance Improvements 37

9 Query Cache Enhancements 38
9.1 Diagnosing contention more easily . . . 38
9.2 Ignoring comments . . . 38
9.3 System Variables . . . 39

10 Improved `NUMA` support 40
10.1 Version Specific Information . . . 40
10.2 Command-line Options for mysqld_safe . . . 40
10.3 Other Reading . . . 40

11 Thread Pool 41
11.1 Priority connection scheduling . . . 41
11.2 Handling of Long Network Waits . . . 42
11.3 Version Specific Information . . . 42
11.4 System Variables . . . 43
11.5 Status Variables . . . 45
11.6 Other Reading . . . 46

12 XtraDB Performance Improvements for I/O-Bound Highly-Concurrent Workloads 47
12.1 Priority refill for the buffer pool free list . . . 47
12.2 Multi-threaded LRU flusher . . . 48
12.3 Parallel doublewrite buffer . . . 48
12.4 Version Specific Information . . . 49
12.5 Other Reading . . . 49

V Flexibility Improvements 50

13 Suppress Warning Messages 51
13.1 Version Specific Information . . . 51
13.2 System Variables . . . 51
13.3 Related Reading . . . 51

14 `Improved MEMORY` Storage Engine 53
14.1 Caveats . . . 53
14.2 Restrictions . . . 54
14.3 Setting Row Format . . . 54
14.4 Examples . . . 55
14.5 Implementation Details . . . 55
14.6 See Also . . . 57

15 Restricting the number of binlog files 58
15.1 Version Specific Information . . . 58
15.2 System Variables . . . 58
15.3 Example . . . 58

16 Extended `SELECT INTO OUTFILE/DUMPFILE` 60
16.1 Version Specific Information . . . 60
16.2 Other Reading . . . 60

17 Per-query variable statement 61
17.1 Examples . . . 61
17.2 Version Specific Information . . . 61

17.3 Other Reading . . . 61

18 Extended `mysqlbinlog` 62
18.1 Version Specific Information . . . 62

19 Slow Query Log Rotation and Expiration 63
19.1 Version Specific Information . . . 63
19.2 System Variables . . . 63

20 CSV engine mode for standard-compliant quote and comma parsing 65
20.1 Example . . . 65
20.2 Version Specific Information . . . 66
20.3 System Variables . . . 66
20.4 Related Reading . . . 66

21 Support for PROXY protocol 67
21.1 Version Specific Information . . . 67
21.2 System Variables . . . 67
21.3 Related Reading . . . 68

22 Per-session server-id 69
22.1 Version Specific Information . . . 69
22.2 System Variables . . . 69
22.3 Other Reading . . . 70

VI Reliability Improvements 71

23 Too Many Connections Warning 72
23.1 Version-Specific Information . . . 72

24 Handle Corrupted Tables 73
24.1 Version Specific Information . . . 73
24.2 System Variables . . . 73

VII Management Improvements 74

25 *Percona Toolkit* UDFs 75
25.1 Version Specific Information . . . 75
25.2 Other Information . . . 75
25.3 Installation . . . 75
25.4 Troubleshooting . . . 75
25.5 Other Reading . . . 76

26 Kill Idle Transactions 77
26.1 Version Specific Information . . . 77
26.2 System Variables . . . 77

27 Enforcing Storage Engine 78
27.1 Version Specific Information . . . 78
27.2 System Variables . . . 78
27.3 Example . . . 78

28 Utility user 80
28.1 Version Specific Information . . . 81

28.2 System Variables . . . 81

29 Expanded Program Option Modifiers **83**
29.1 Combining the options . . . 83
29.2 Version Specific Information . . . 84
29.3 Examples . . . 84

30 XtraDB changed page tracking **86**
30.1 User statements for handling the XtraDB changed page bitmaps . . . 86
30.2 Additional information in SHOW ENGINE INNODB STATUS . . . 87
30.3 INFORMATION_SCHEMA Tables . . . 87
30.4 Version Specific Information . . . 87
30.5 System Variables . . . 87

31 PAM Authentication Plugin **89**
31.1 Installation . . . 89
31.2 Configuration . . . 90
31.3 Creating a user . . . 90
31.4 Supplementary groups support . . . 90
31.5 Known issues . . . 90
31.6 Version Specific Information . . . 90

32 Expanded Fast Index Creation **91**
32.1 Enabling Expanded Fast Index Creation . . . 91
32.2 Version Specific Information . . . 92
32.3 System Variables . . . 92
32.4 Other Reading . . . 93

33 Backup Locks **94**
33.1 `LOCK TABLES FOR BACKUP` . . . 94
33.2 `LOCK BINLOG FOR BACKUP` . . . 94
33.3 `UNLOCK BINLOG` . . . 95
33.4 Privileges . . . 95
33.5 Interaction with other global locks . . . 95
33.6 MyISAM index and data buffering . . . 95
33.7 mysqldump . . . 96

34 Audit Log Plugin **98**
34.1 Installation . . . 99
34.2 Log Format . . . 99
34.3 Streaming the audit log to syslog . . . 100
34.4 System Variables . . . 100
34.5 Version Specific Information . . . 103

35 Start transaction with consistent snapshot **104**
35.1 Snapshot Cloning . . . 104
35.2 mysqldump . . . 104
35.3 System Variables . . . 105
35.4 Status Variables . . . 105
35.5 Other Reading . . . 105

36 Extended `SHOW GRANTS` **106**
36.1 Example . . . 106

VIII Diagnostics Improvements 108

37 User Statistics 109
37.1 Version Specific Information . . . 109
37.2 Other Information . . . 109
37.3 System Variables . . . 109
37.4 INFORMATION_SCHEMA Tables . . . 110
37.5 Commands Provided . . . 114
37.6 Status Variables . . . 115

38 Slow Query Log 116
38.1 Version Specific Information . . . 116
38.2 System Variables . . . 116
38.3 Other Information . . . 120
38.4 Related Reading . . . 122

39 Extended Show Engine *InnoDB* Status 123
39.1 Version Specific Information . . . 123
39.2 Other Information . . . 123
39.3 System Variables . . . 123
39.4 Status Variables . . . 124
39.5 INFORMATION_SCHEMA Tables . . . 128
39.6 Other reading . . . 129

40 Show Storage Engines 130
40.1 Version-Specific Information . . . 130

41 Process List 131
41.1 Version Specific Information . . . 131
41.2 INFORMATION_SCHEMA Tables . . . 131
41.3 Example Output . . . 132

42 Misc. INFORMATION_SCHEMA Tables 133
42.1 Temporary tables . . . 133
42.2 Multiple Rollback Segments . . . 134
42.3 `INFORMATION_SCHEMA` Tables . . . 134

43 Thread Based Profiling 136
43.1 Version Specific Information . . . 136

44 Metrics for scalability measurement 137
44.1 Installation . . . 137
44.2 System Variables . . . 137
44.3 Status Variables . . . 138
44.4 Version Specific Information . . . 138
44.5 Other Reading . . . 138

45 Response Time Distribution 139
45.1 Logging the queries in separate `READ` and `WRITE` tables . . . 140
45.2 Installing the plugins . . . 140
45.3 Usage . . . 141
45.4 Version Specific Information . . . 142
45.5 System Variables . . . 142
45.6 INFORMATION_SCHEMA Tables . . . 143

IX TokuDB 145

46 TokuDB Introduction 146
46.1 TokuDB Installation . . . 146
46.2 Using TokuDB . . . 149
46.3 Getting Started with TokuDB . . . 156
46.4 TokuDB Variables . . . 159
46.5 Percona TokuBackup . . . 168
46.6 TokuDB Troubleshooting . . . 172
46.7 Frequently Asked Questions . . . 196
46.8 Removing TokuDB storage engine . . . 202
46.9 Getting the Most from TokuDB . . . 203

47 TokuDB Installation 205
47.1 Prerequisites . . . 205
47.2 Installation . . . 206
47.3 Enabling the TokuDB Storage Engine . . . 206
47.4 Enabling the TokuDB Storage Engine Manually . . . 207
47.5 TokuDB Version . . . 208
47.6 Upgrade . . . 208

48 Using TokuDB 209
48.1 Fast Insertions and Richer Indexes . . . 209
48.2 Clustering Secondary Indexes . . . 209
48.3 Hot Index Creation . . . 210
48.4 Hot Column Add, Delete, Expand, and Rename (HCADER) . . . 211
48.5 Compression Details . . . 212
48.6 Changing Compression of a Table . . . 213
48.7 Read Free Replication . . . 213
48.8 Transactions and ACID-compliant Recovery . . . 213
48.9 Managing Log Size . . . 214
48.10 Recovery . . . 214
48.11 Disabling the Write Cache . . . 214
48.12 Progress Tracking . . . 215
48.13 Migrating to TokuDB . . . 215

49 TokuDB Background ANALYZE TABLE 216
49.1 Background Jobs . . . 216
49.2 Auto analysis . . . 216
49.3 System Variables . . . 217
49.4 INFORMATION_SCHEMA Tables . . . 219
49.5 Version Specific Information . . . 219

50 TokuDB Variables 221
50.1 Client Session Variables . . . 221
50.2 MySQL Server Variables . . . 225

51 TokuDB Troubleshooting 231
51.1 Known Issues . . . 231
51.2 Lock Visualization in TokuDB . . . 231
51.3 Engine Status . . . 235
51.4 Global Status . . . 247

52 Percona TokuBackup 255
52.1 Installing From Binaries . . . 255

52.2 Making a Backup . . . 256
52.3 Restoring From Backup . . . 256
52.4 Advanced Configuration . . . 257
52.5 Limitations and known issues . . . 258

53 Frequently Asked Questions **260**
53.1 Transactional Operations . . . 260
53.2 TokuDB and the File System . . . 260
53.3 Full Disks . . . 261
53.4 Backup . . . 262
53.5 Missing Log Files . . . 264
53.6 Isolation Levels . . . 264
53.7 Lock Wait Timeout Exceeded . . . 264
53.8 Query Cache . . . 264
53.9 Row Size . . . 264
53.10 NFS & CIFS . . . 265
53.11 Using Other Storage Engines . . . 265
53.12 Using MySQL Patches with TokuDB . . . 265
53.13 Truncate Table vs Delete from Table . . . 265
53.14 Foreign Keys . . . 265
53.15 Dropping Indexes . . . 266

54 Removing TokuDB storage engine **267**
54.1 Change the tables from TokuDB to InnoDB . . . 267
54.2 Removing the plugins . . . 267

X Reference **269**

55 List of upstream *MySQL* bugs fixed in *Percona Server* 5.7 **270**

56 List of variables introduced in Percona Server 5.7 **275**
56.1 System Variables . . . 275
56.2 Status Variables . . . 277

57 Known issues and limitations **285**

58 Development of Percona Server **286**
58.1 Submitting Changes . . . 286
58.2 Making a release . . . 288
58.3 Jenkins . . . 288

59 Trademark Policy **291**

60 Index of `INFORMATION_SCHEMA` Tables **292**

61 Frequently Asked Questions **293**
61.1 Q: Will *Percona Server* with *XtraDB* invalidate our *MySQL* support? . . . 293
61.2 Q: Will we have to *GPL* our whole application if we use *Percona Server* with *XtraDB*? . . . 293
61.3 Q: Do I need to install *Percona* client libraries? . . . 293
61.4 Q: When using the *Percona XtraBackup* to setup a replication slave on Debian based systems I'm getting: "ERROR 1045 (28000): Access denied for user 'debian-sys-maint'@'localhost' (using password: YES)" . . . 293

62 Copyright and Licensing Information **294**
62.1 Documentation Licensing . . . 294

62.2 Software License 294

63 *Percona Server* 5.7 Release notes **295**
63.1 *Percona Server* 5.7.11-4 295
63.2 *Percona Server* 5.7.10-3 296
63.3 *Percona Server* 5.7.10-2 296
63.4 *Percona Server* 5.7.10-1 298

64 Glossary **300**

Index **302**

Percona Server is an enhanced drop-in replacement for *MySQL*. With *Percona Server*,

- Your queries will run faster and more consistently.
- You will consolidate servers on powerful hardware.
- You will delay sharding, or avoid it entirely.
- You will save money on hosting fees and power.
- You will spend less time tuning and administering.
- You will achieve higher uptime.
- You will troubleshoot without guesswork.

Does this sound too good to be true? It's not. *Percona Server* offers breakthrough performance, scalability, features, and instrumentation. Its self-tuning algorithms and support for extremely high-performance hardware make it the clear choice for companies who demand the utmost performance and reliability from their database server.

Part I

Introduction

CHAPTER

ONE

THE *PERCONA XTRADB* STORAGE ENGINE

Percona XtraDB is an enhanced version of the *InnoDB* storage engine, designed to better scale on modern hardware, and including a variety of other features useful in high performance environments. It is fully backwards compatible, and so can be used as a drop-in replacement for standard *InnoDB*.

Percona XtraDB includes all of *InnoDB* 's robust, reliable `ACID`-compliant design and advanced `MVCC` architecture, and builds on that solid foundation with more features, more tunability, more metrics, and more scalability. In particular, it is designed to scale better on many cores, to use memory more efficiently, and to be more convenient and useful. The new features are especially designed to alleviate some of *InnoDB* 's limitations. We choose features and fixes based on customer requests and on our best judgment of real-world needs as a high-performance consulting company.

Percona XtraDB engine will not have further binary releases, it is distributed as part of *Percona Server* and *MariaDB*.

CHAPTER

TWO

LIST OF FEATURES AVAILABLE IN *PERCONA SERVER* RELEASES

Percona Server 5.1	*Percona Server* 5.5	*Percona Server* 5.6
Improved Buffer Pool Scalability	*Improved Buffer Pool Scalability*	*Improved Buffer Pool Scalability*
Configurable Insert Buffer	*Configurable Insert Buffer*	Feature not implemented
Improved InnoDB I/O Scalability	*Improved InnoDB I/O Scalability*	*Improved InnoDB I/O Scalability*
More Concurrent Transactions Available	*More Concurrent Transactions Available*	Replaced by the upstream implementation [1]
Feature not implemented	*Multiple Adaptive Hash Search Partitions*	*Multiple Adaptive Hash Search Partitions*
Dedicated Purge Thread	Replaced by the upstream implementation [2]	Replaced by the upstream implementation [2]
Drop table performance	*Drop table performance*	Replaced by the upstream fix [3]
Feature not implemented	*Atomic write support for Fusion-io devices*	*Atomic write support for Fusion-io devices*
Configuration of the Doublewrite Buffer	*Configuration of the Doublewrite Buffer*	Feature not implemented
Query Cache Enhancements	*Query Cache Enhancements*	*Query Cache Enhancements*
Fast InnoDB Checksum [4]	*Fast InnoDB Checksum* [4]	Replaced by the upstream implementation [4]
Reduced Buffer Pool Mutex Contention	*Reduced Buffer Pool Mutex Contention*	Feature not implemented
InnoDB timer-based Concurrency Throttling	*InnoDB timer-based Concurrency Throttling*	Replaced by the upstream implementation [5]
HandlerSocket	*HandlerSocket*	Feature not implemented [6]
Feature not implemented	*Improved NUMA support*	*Improved NUMA support*
Feature not implemented	*Thread Pool*	*Thread Pool*
Feature not implemented	*Binary Log Group Commit*	Replaced by the upstream implementation [7]
Support of Multiple Page Sizes [8]	*Support of Multiple Page Sizes* [8]	Replaced by the upstream implementation [8]

Continued on next page

[1] Feature has been deprecated after *Percona Server* 5.5.11-20.2. It has replaced by the upstream implementation of innodb_undo_logs in *MySQL* 5.6.3.

[2] Feature has not been ported from *Percona Server* 5.1 version. It has been replaced by the upstream Improved Purge Scheduling implementation.

[3] Feature has been has been removed and its controlling variable `innodb_lazy_drop_table` has been deprecated from *Percona Server* 5.5.30-30.2. Feature has been removed because the upstream `DROP TABLE` implementation has been improved when bugs #56332 and #51325 were fixed.

[4] Feature has been deprecated after *Percona Server* 5.1.66-14.2 and *Percona Server* 5.5.28-29.2. It has been replaced by the upstream innodb_checksum_algorithm implementation released in *MySQL* 5.6.3.

[5] Feature has been replaced by the upstream implementation innodb-performance-thread_concurrency in *MySQL* 5.6

[6] Feature will be implemented in one of the future *Percona Server* 5.6 releases.

[7] Binary Log Group Commit feature has been replaced with the *MySQL* 5.6 implementation.

[8] Feature has been deprecated in the *Percona Server* 5.1.68-14.6 and *Percona Server* 5.5.30-30.2. It has been replaced by the upstream inn-

Table 2.1 – continued from previous page

Percona Server 5.1	*Percona Server* 5.5	*Percona Server* 5.6
Suppress Warning Messages	*Suppress Warning Messages*	*Suppress Warning Messages*
Handle BLOB End of Line	*Handle BLOB End of Line*	Replaced by the upstream implementation [9]
Ability to change database for mysqlbinlog	*Ability to change database for mysqlbinlog*	*Ability to change database for mysqlbinlog*
Replication Stop Recovery	Feature not implemented	Feature not implemented
Fixed Size for the Read Ahead Area	*Fixed Size for the Read Ahead Area*	*Fixed Size for the Read Ahead Area*
Fast Shutdown	Feature not implemented	Feature not implemented
Feature not implemented	*Improved MEMORY Storage Engine*	*Improved MEMORY Storage Engine*
Feature not implemented	*Restricting the number of binlog files*	*Restricting the number of binlog files*
Ignoring missing tables in mysqldump	*Ignoring missing tables in mysqldump*	*Ignoring missing tables in mysqldump*
Too Many Connections Warning	*Too Many Connections Warning*	*Too Many Connections Warning*
Error Code Compatibility	*Error Code Compatibility*	*Error Code Compatibility*
Handle Corrupted Tables	*Handle Corrupted Tables*	*Handle Corrupted Tables*
Crash-Resistant Replication	*Crash-Resistant Replication*	Replaced by the upstream implementation [10]
Lock-Free SHOW SLAVE STATUS	*Lock-Free SHOW SLAVE STATUS*	*Lock-Free SHOW SLAVE STATUS*
Fast InnoDB Recovery Process	*Fast InnoDB Recovery Stats*	Feature not implemented
InnoDB Data Dictionary Size Limit	*InnoDB Data Dictionary Size Limit*	Replaced by the upstream implementation [11]
Expand Table Import	*Expand Table Import*	Replaced by the upstream implementation [12]
Dump/Restore of the Buffer Pool	*Dump/Restore of the Buffer Pool*	Replaced by the upstream implementation [13]
Fast Index Creation	*Fast Index Creation*	Replaced by the upstream implementation [14]
Expanded Fast Index Creation	*Expanded Fast Index Creation*	*Expanded Fast Index Creation*
Prevent Caching to FlashCache	*Prevent Caching to FlashCache*	Feature not implemented
Percona Toolkit UDFs	*Percona Toolkit UDFs*	*Percona Toolkit UDFs*
Support for Fake Changes	*Support for Fake Changes*	*Support for Fake Changes*
Kill Idle Transactions	*Kill Idle Transactions*	*Kill Idle Transactions*
XtraDB changed page tracking	*XtraDB changed page tracking*	*XtraDB changed page tracking*
Feature not implemented	*Enforcing Storage Engine*	*Enforcing Storage Engine*
Feature not implemented	*Utility user*	*Utility user*
Feature not implemented	*Extending the secure-file-priv server option*	*Extending the secure-file-priv server option*
Feature not implemented	*Expanded Program Option Modifiers*	*Expanded Program Option Modifiers*
Feature not implemented	*PAM Authentication Plugin*	*PAM Authentication Plugin*
Feature not implemented	Feature not implemented	*Log Archiving for XtraDB*

Continued on next page

odb_page_size version released in *MySQL* 5.6.4.

[9] Feature has been replaced by the *MySQL* 5.6 binary-mode configuration option.

[10] Feature has been replaced by the *MySQL* 5.6 relay-log-recovery implementation.

[11] Feature has been replaced by the *MySQL* 5.6 table_definition_cache implementation.

[12] Feature has been replaced by the *MySQL* 5.6 Improved Tablespace Management implementation.

[13] Feature has been replaced by the *MySQL* 5.6 preloading the InnoDB buffer pool at startup implementation.

[14] Feature has been replaced by the 5.6' ALGORITHM= option implementation.

Table 2.1 – continued from previous page

Percona Server 5.1	*Percona Server* 5.5	*Percona Server* 5.6
InnoDB Statistics	*InnoDB Statistics*	Replaced by the upstream implementation [15]
User Statistics	*User Statistics*	*User Statistics*
Slow Query Log	*Slow Query Log*	*Slow Query Log*
Count InnoDB Deadlocks	*Count InnoDB Deadlocks*	*Count InnoDB Deadlocks*
Log All Client Commands (syslog)	*Log All Client Commands (syslog)*	*Log All Client Commands (syslog)*
Response Time Distribution	*Response Time Distribution*	*Response Time Distribution*
Show Storage Engines	*Show Storage Engines*	*Show Storage Engines*
Show Lock Names	*Show Lock Names*	*Show Lock Names*
Process List	*Process List*	*Process List*
Misc. INFORMATION_SCHEMA Tables	*Misc. INFORMATION_SCHEMA Tables*	*Misc. INFORMATION_SCHEMA Tables*
Feature not implemented	*Extended Show Engine InnoDB Status*	*Extended Show Engine InnoDB Status*
Feature not implemented	*Thread Based Profiling*	*Thread Based Profiling*
Feature not implemented	Feature not implemented	*XtraDB Performance Improvements for I/O-Bound Highly-Concurrent Workloads*
Feature not implemented	Feature not implemented	*Page cleaner thread tuning*
Feature not implemented	Feature not implemented	*Statement Timeout*
Feature not implemented	*Extended SELECT INTO OUTFILE/DUMPFILE*	*Extended SELECT INTO OUTFILE/DUMPFILE*
Feature not implemented	Feature not implemented	*Per-query variable statement*
Feature not implemented	*Extended mysqlbinlog*	*Extended mysqlbinlog*
Feature not implemented	*Slow Query Log Rotation and Expiration*	*Slow Query Log Rotation and Expiration*
Feature not implemented	*Metrics for scalability measurement*	*Metrics for scalability measurement*
Feature not implemented	*Audit Log*	*Audit Log*
Feature not implemented	Feature not implemented	*Backup Locks*
Feature not implemented	Feature not implemented	*CSV engine mode for standard-compliant quote and comma parsing*
Feature not implemented	Feature not implemented	*Super read-only*

2.1 Other Reading

- *Changed in Percona Server 5.6*
- *Percona Server In-Place Upgrading Guide: From 5.6 to 5.7*
- Upgrading from MySQL 5.1 to 5.5
- What Is New in MySQL 5.5
- What Is New in MySQL 5.6

[15] Feature has been replaced by the *MySQL* 5.6 Persistent Optimizer Statistics for InnoDB Tables implementation.

CHAPTER

THREE

PERCONA SERVER FEATURE COMPARISON

Percona Server is an enhanced drop-in replacement for *MySQL*. With *Percona Server*,

- Your queries will run faster and more consistently.
- You will consolidate servers on powerful hardware.
- You will delay sharding, or avoid it entirely.
- You will save money on hosting fees and power.
- You will spend less time tuning and administering.
- You will achieve higher uptime.
- You will troubleshoot without guesswork.

We provide these benefits by significantly enhancing *Percona Server* as compared to the standard *MySQL* database server:

Features	Percona Server 5.7.10	MySQL 5.7.10
Open source	Yes	Yes
ACID Compliance	Yes	Yes
Multi-Version Concurrency Control	Yes	Yes
Row-Level Locking	Yes	Yes
Automatic Crash Recovery	Yes	Yes
Table Partitioning	Yes	Yes
Views	Yes	Yes
Subqueries	Yes	Yes
Triggers	Yes	Yes
Stored Procedures	Yes	Yes
Foreign Keys	Yes	Yes
GTID Replication	Yes	Yes

Extra Features for Developers	Percona Server 5.7.10	MySQL 5.7.10
NoSQL Socket-Level Interface	Yes	Yes
Extra Hash/Digest Functions	Yes	

Extra Diagnostic Features	Percona Server 5.7.10	MySQL 5.7.10
INFORMATION_SCHEMA Tables	75	61
Global Performance and Status Counters	385	355
Per-Table Performance Counters	Yes	
Per-Index Performance Counters	Yes	
Per-User Performance Counters	Yes	
Per-Client Performance Counters	Yes	
Per-Thread Performance Counters	Yes	
Global Query Response Time Statistics	Yes	
Enhanced SHOW ENGINE INNODB STATUS	Yes	
Undo Segment Information	Yes	
Temporary tables Information	Yes	

Performance & Scalability Enhancements	Percona Server 5.7.10	MySQL 5.7.10
Improved scalability by splitting mutexes	Yes	
Improved MEMORY Storage Engine	Yes	
Improved Flushing	Yes	

Extra Features for DBA/Operations Staff	Percona Server 5.7.10	MySQL 5.7.10
Configurable Page Sizes	Yes	Yes
Configurable Fast Index Creation	Yes	
Changed Page Tracking	Yes	
PAM Authentication	Yes	Yes [1]
Threadpool	Yes	Yes [1]
Backup Locks	Yes	
Extended `SHOW GRANTS`	Yes	
Improved Handling of Corrupted Tables	Yes	
Ability to Kill Idle Transactions	Yes	
START TRANSACTION WITH CONSISTENT SNAPSHOT improvements	Yes	

Running Database as a Service	Percona Server 5.7.10	MySQL 5.7.10
Special Utility User	Yes	
Expanded Program Option Modifiers	Yes	
Enforcing the Specific Storage Engine	Yes	

[1] Feature available in Enterprise version only

CHAPTER

FOUR

CHANGED IN PERCONA SERVER 5.7

Percona Server 5.7 is based on *MySQL* 5.7 and incorporates many of the improvements found in *Percona Server* 5.6.

4.1 Features removed from *Percona Server* 5.7 that were available in *Percona Server* 5.6

Some features that were present in *Percona Server* 5.6 have been removed in *Percona Server* 5.7. These are:

- Handlersocket - This feature might be included in a future release if HandlerSocket starts supporting 5.7.
- Support for Fake Changes - Instead of slave prefetching using the fake changes, a 5.7 intra-schema parallel replication slave should be used.
- `SHOW ENGINE INNODB STATUS` no longer prints the count of active Read-Only transactions.
- InnoDB redo log archiving has been removed due to lack of user uptake of the feature.

4.2 Changes in Percona Server 5.6 features

- The minor *Percona Server* version number ("y" in "5.a.b-x.y") has been dropped to simplify *Percona Server* versioning.
- Performance Schema memory instrumentation support has been added to the *Audit Log Plugin*, *Metrics for scalability measurement*, and *PAM Authentication Plugin*, and to the core server to track memory used by *User Statistics*, *Per-query variable statement*, *XtraDB changed page tracking*, and *Thread Pool* features.
- *Audit Log Plugin* now produces diagnostics in a format consistent with the rest of the server.
- The performance_schema.metadata_locks table now displays `backup` and `binlog` lock information too. The `object_type` column has two new valid values: `backup`, and `binlog`.
- `XTRADB_RSEG` table schema has been changed to support new possible *InnoDB* page sizes. The `zip_size` column has been removed and replaced by new columns `physical_page_size`, `logical_page_size`, and `is_compressed`.
- `XTRADB_READ_VIEW` table no longer contains the `READ_VIEW_UNDO_NUMBER` column, which was associated with unused code and always contained zero.
- Interaction between `--hidden-` option modifier and session_track_system_variables has been implemented as follows: any variables with `--hidden-` modifier become hidden from the latter variable too. Thus, they should not be present there. Even if you never set `session_track_system_variables`, care must be taken if a variable contained in its default value (i.e. autocommit) is hidden.

- Nested `SET STATEMENT ... FOR SET STATEMENT ... FOR ...` statements will have different effect in the innermost clause in case the nested clauses set the same variables: in 5.6 the innermost assignment had effect whereas in 5.7 the outermost assignment is effective.
- *Utility user* is treated as a `SUPER` user for the purposes of offline mode: utility user connections are not dropped if server switches to offline mode and new utility user connections can be established to such server.
- The server will abort startup with an error message if conflicting `enforce_storage_engine` and `disabled_storage_engines` option values are specified, that is, if the enforced storage engine is in the list of disabled storage engines.

4.3 Features available in *Percona Server* 5.6 that have been replaced with *MySQL* 5.7 features

Some *Percona Server* 5.6 features have been replaced by similar or equivalent *MySQL* 5.7 features, so we now keep the *MySQL* 5.7 implementations in *Percona Server* 5.7. These are:

- Lock-Free SHOW SLAVE STATUS NONBLOCKING has been replaced by a regular `SHOW SLAVE STATUS` implementation. Oracle implementation forbids calling it from a stored function.
- Behavior corresponding to `slow_query_log_timestamp_precision` set to `microsecond` is now the default, the variable itself and the behavior corresponding to the variable's `second` value is removed.
- Behavior corresponding to `slow_query_log_timestamp_always` set to `TRUE` is now the default, the variable itself and the behavior corresponding to the variable's `FALSE` value is removed.
- Statement timeout feature has been replaced by Oracle Server-side SELECT statement timeouts implementation. Differences: the Oracle variable is named max_execution_time instead of `max_statement_time`; variable `have_statement_timeout` variable has been removed removed; the timeouts only apply for read-only SELECTs.
- Atomic write support on fusionIO devices with NVMFS has been replaced by Oracle implementation. It is no longer required to enable `innodb_use_atomic_writes` variable, and this variable has been removed. The atomic write support will be enabled, and the doublewrite buffer disabled, on supporting devices automatically. The Oracle implementation does not silently adjust `innodb_flush_method` to `O_DIRECT` if it has a different value. The user must set it to `O_DIRECT` explicitly, or atomic writes will not be enabled.
- Online GTID migration patch has been replaced by an upstream variable gtid_mode made dynamic.
- The Error Code Compatibility has been replaced by the multiple start-error-number directive in `sql/share/errmsg-utf8.txt` support.
- Ignoring missing tables in mysqldump with `--ignore-create-error` option has been replaced by the more general upstream option –ignore-error option.
- innodb_log_block_size has been replaced by innodb_log_write_ahead_size variable. To avoid read on write when the storage block size is not equal to 512 bytes, the latter should be set to the same value the former was. If `innodb_log_block_size` was set to non-default values, new log files must be created during the upgrade.
- Extended secure-file-priv server option, which was used to disable `LOAD DATA INFILE`, `SELECT INTO OUTFILE` statements, and `LOAD_FILE()` function completely, has been replaced by upstream introducing `NULL` as a possible value to this variable. To migrate, any value-less settings must be replaced by `NULL`.
- innodb_sched_priority_cleaner variable has been removed, as the effect of setting it to 39 (corresponding to nice value of -20), is now enabled by default.
- innodb_adaptive_hash_index_partitions has been replaced by innodb_adaptive_hash_index_parts.

- In the default server setup (with *InnoDB* being the only one XA-capable storage engine), `--tc-heuristic-recover=COMMIT` is silently converted to `ROLLBACK`. If *TokuDB* or another XA-supporting 3rd party storage engine is installed, `--tc-heuristic-recover=ROLLBACK` option is unavailable. The default value of `tc-heuristic-recover` option in *Percona Server* 5.6 but not in *MySQL* 5.6 was `NONE` as a result of fix for upstream bug #70860. Since Oracle fixed the same bug in 5.7, the default value is `OFF` now.
- innodb_log_checksum_algorithm feature has been replaced by innodb_log_checksums option. In particular, to get the effect of setting the `innodb_log_checksum_algorithm` to `crc32`, `innodb_log_checksums` should be set to `ON`, which is a default setting for this variable.
- innodb_buffer_pool_populate server option and numa_interleave `mysql_safe.sh` option have been replaced by innodb_numa_interleave server option. Note that `flush_caches` option still remains.
- Ability to change database for mysqlbinlog implementation has been replaced from *MariaDB* one with *MySQL* rewrite-db one. The feature is mostly identical with two differences: 1) multiple rewrite rules must be given as separate options, and the ability to list them in a single rule, separated by commas, is lost. That is, any `--rewrite-db='a->b,c->d'` occurrences must be replaced with `--rewrite-db='a->b' --rewrite-db='c->d'`. 2) Whitespace around database names is not ignored.
- INFORMATION_SCHEMA.PROCESSLIST.TID column has been has been replaced by PERFORMANCE_SCHEMA.THREADS.THREAD_OS_ID column. If running under thread pool, `THREAD_OS_ID` column will always be `NULL`, whereas in the 5.6 implementation `TID` column showed either `NULL` or the assigned worker thread id at the moment.
- innodb_foreground_preflush server variable has been removed as the upstream implemented a similar feature without a controlling option.
- Log All Client Commands (syslog) feature has been replaced by Oracle mysql Logging implementation.
- Support for Multiple user level locks per connection has been replaced by Oracle implementation, which is based on the same contributed patch by *Kostja Osipov*.
- super-read-only option has been replaced by Oracle super_read_only variable implementation.
- Mutex names in `SHOW ENGINE INNODB MUTEX` have been replaced by Oracle mutex name implementation.
- *Percona Server* now uses packaging similar to the upstream *MySQL* version. Most important change is that for *Debian*/*Ubuntu* upgrades you now need to run `mysql_upgrade` manually.

4.4 List of status variables that are no longer available in *Percona Server* 5.7

Following status variables available in *Percona Server* 5.6 are no longer present in *Percona Server* 5.7:

Status Variables	Replaced by
Com_purge_archived	InnoDB redo log archiving has been removed due to lack of user uptake of the feature.
Com_purge_archived_before_date	InnoDB redo log archiving has been removed due to lack of user uptake of the feature.
read_views_memory	transaction descriptors replaced by the upstream implementation
descriptors_memory	transaction descriptors replaced by the upstream implementation
innodb_mem_total	This variable was always zero in 5.6 with the default innodb_use_sys_malloc setting
innodb_deadlocks	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (lock_deadlocks)
Innodb_ibuf_merges	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (ibuf_merges)
Innodb_ibuf_merged_deletes	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (ibuf_merges_delete)
Innodb_ibuf_merged_delete_marks	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (ibuf_merges_delete_mark)
Innodb_ibuf_discarded_deletes	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (ibuf_merges_discard_delete)
Innodb_ibuf_discarded_delete_marks	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (ibuf_merges_discard_delete_mark)
Innodb_ibuf_discarded_inserts	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (ibuf_merges_discard_insert)
Innodb_ibuf_merged_inserts	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (ibuf_merges_insert)
Innodb_ibuf_size	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (ibuf_size)
Innodb_s_lock_os_waits	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (innodb_rwlock_s_os_waits)
Innodb_s_lock_spin_rounds	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (innodb_rwlock_s_spin_rounds)
Innodb_s_lock_spin_waits	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (innodb_rwlock_s_spin_waits)
Innodb_x_lock_os_waits	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (innodb_rwlock_x_os_waits)
Innodb_x_lock_spin_rounds	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (innodb_rwlock_x_spin_rounds)
Innodb_x_lock_spin_waits	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (innodb_rwlock_x_spin_waits)
Innodb_current_row_locks	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (lock_row_lock_current_waits)
Innodb_history_list_length	Information now available in

4.5 List of system variables that are no longer available in *Percona Server* 5.7

Following system variables available in *Percona Server* 5.6 are no longer present in *Percona Server* 5.7:

Warning: *Percona Server* 5.7 won't be able to start if some of these variables are set in the server's configuration file.

System Variables	Feature Comment
`gtid_deployment_step`	Replaced by an upstream variable gtid_mode made dynamic.
`innodb_fake_changes`	Instead of slave prefetching using the fake changes, a 5.7 intra-schema parallel replication slave should be used.
`innodb_locking_fake_changes`	Instead of slave prefetching using the fake changes, a 5.7 intra-schema parallel replication slave should be used.
`innodb_log_archive`	InnoDB redo log archiving has been removed due to lack of user uptake of the feature.
`innodb_log_arch_dir`	InnoDB redo log archiving has been removed due to lack of user uptake of the feature.
`innodb_log_arch_expire_sec`	InnoDB redo log archiving has been removed due to lack of user uptake of the feature.
`max_statement_time`	Replaced by upstream max_execution_time variable in Server-side SELECT statement timeouts implementation.
`have_statement_timeout`	Variable has been removed due to upstream feature implementation
`have_statement_timeout`	Variable has been removed due to upstream feature implementation
`innodb_use_atomic_writes`	Variable has been removed due to upstream feature implementation
`innodb_adaptive_hash_index_partitions`	Replaced by upstream variable innodb_adaptive_hash_index_parts

4.6 Features ported from *Percona Server* 5.6 to *Percona Server* 5.7

Following features were ported from *Percona Server* 5.6 to *Percona Server* 5.7:

Feature Ported	Version
Improved Buffer Pool Scalability	`5.7.10-1`
Improved InnoDB I/O Scalability	`5.7.10-1`
Query Cache Enhancements	`5.7.10-1`
Improved NUMA support	`5.7.10-1`
Thread Pool	`5.7.10-1`
XtraDB Performance Improvements for I/O-Bound Highly-Concurrent Workloads	`5.7.10-1`
Suppress Warning Messages	`5.7.10-1`
Improved MEMORY Storage Engine	`5.7.10-1`
Restricting the number of binlog files	`5.7.10-1`

Continued on next page

Table 4.1 – continued from previous page

Feature Ported	Version
Extended SELECT INTO OUTFILE/DUMPFILE	5.7.10-1
Per-query variable statement	5.7.10-1
Extended mysqlbinlog	5.7.10-1
Slow Query Log Rotation and Expiration	5.7.10-1
CSV engine mode for standard-compliant quote and comma parsing	5.7.10-1
Support for PROXY protocol	5.7.10-1
Per-session server-id	5.7.10-1
Too Many Connections Warning	5.7.10-1
Handle Corrupted Tables	5.7.10-1
Percona Toolkit UDFs	5.7.10-1
Kill Idle Transactions	5.7.10-1
Enforcing Storage Engine	5.7.10-1
Utility user	5.7.10-1
Expanded Program Option Modifiers	5.7.10-1
XtraDB changed page tracking	5.7.10-1
PAM Authentication Plugin	5.7.10-1
Expanded Fast Index Creation	5.7.10-1
Backup Locks	5.7.10-1
Audit Log Plugin	5.7.10-1
Start transaction with consistent snapshot	5.7.10-1
Extended SHOW GRANTS	5.7.10-1
User Statistics	5.7.10-1
Slow Query Log	5.7.10-1
Extended Show Engine InnoDB Status	5.7.10-1
Show Storage Engines	5.7.10-1
Process List	5.7.10-1
Misc. INFORMATION_SCHEMA Tables	5.7.10-1
Thread Based Profiling	5.7.10-1
Metrics for scalability measurement	5.7.10-1
Response Time Distribution	5.7.10-1

Part II

Installation

CHAPTER

FIVE

INSTALLING *PERCONA SERVER* 5.7

This page provides the information on how to you can install *Percona Server*. Following options are available:

- *Installing Percona Server from Repositories* (recommended)
- Installing *Percona Server* from Downloaded *rpm* or *apt* Packages
- *Installing Percona Server from a Binary Tarball*
- *Installing Percona Server from a Source Tarball*
- *Installing Percona Server from the Git Source Tree*
- *Compiling Percona Server from Source*

Before installing, you might want to read the *Percona Server 5.7 Release notes*.

5.1 Installing *Percona Server* from Repositories

Percona provides repositories for **yum** (`RPM` packages for *Red Hat*, *CentOS* and *Amazon Linux AMI*) and **apt** (`.deb` packages for *Ubuntu* and *Debian*) for software such as *Percona Server*, *Percona XtraBackup*, and *Percona Toolkit*. This makes it easy to install and update your software and its dependencies through your operating system's package manager. This is the recommend way of installing where possible.

Following guides describe the installation process for using the official Percona repositories for `.deb` and `.rpm` packages.

5.1.1 Installing *Percona Server* on *Debian* and *Ubuntu*

Ready-to-use packages are available from the *Percona Server* software repositories and the download page.

Supported Releases:

- Debian:
- 7.0 (wheezy)
- 8.0 (jessie)
- Ubuntu:
- 12.04LTS (precise)
- 14.04LTS (trusty)
- 15.04 (vivid)
- 15.10 (wily)

Supported Platforms:

- x86
- x86_64 (also known as `amd64`)

What's in each DEB package?

The `percona-server-server-5.7` package contains the database server itself, the `mysqld` binary and associated files.

The `percona-server-common-5.7` package contains files common to the server and client.

The `percona-server-client-5.7` package contains the command line client.

The `percona-server-5.7-dbg` package contains debug symbols for the server.

The `percona-server-test-5.7` package contains the database test suite.

The `percona-server-source-5.7` package contains the server source.

The `libperconaserverclient20-dev` package contains header files needed to compile software to use the client library.

The `libperconaserverclient20` package contains the client shared library. The `18.1` is a reference to the version of the shared library. The version is incremented when there is a ABI change that requires software using the client library to be recompiled or its source code modified.

Installing *Percona Server* from Percona `apt` repository

1. Fetch the repository packages from Percona web:

   ```
   wget https://repo.percona.com/apt/percona-release_0.1-3.$(lsb_release -sc)_all.deb
   ```

2. Install the downloaded package with **dpkg**. To do that, run the following commands as root or with **sudo**:

   ```
   dpkg -i percona-release_0.1-3.$(lsb_release -sc)_all.deb
   ```

   Once you install this package the Percona repositories should be added. You can check the repository setup in the `/etc/apt/sources.list.d/percona-release.list` file.

3. Remember to update the local cache:

   ```
   $ sudo apt-get update
   ```

4. After that you can install the server package:

   ```
   $ sudo apt-get install percona-server-server-5.7
   ```

Note: *Percona Server* 5.7 comes with the *TokuDB storage engine*. You can find more information on how to install and enable the *TokuDB* storage in the *TokuDB Installation* guide.

Percona `apt` Testing repository

Percona offers pre-release builds from the testing repository. To enable it add the just uncomment the testing repository lines in the Percona repository definition in your repository file (default `/etc/apt/sources.list.d/percona-release.list`). It should looks like this (in this example `VERSION` is the name of your distribution):

```
# Testing & pre-release packages
#
deb http://repo.percona.com/apt VERSION testing
deb-src http://repo.percona.com/apt VERSION testing
```

Apt-Pinning the packages

In some cases you might need to "pin" the selected packages to avoid the upgrades from the distribution repositories. You'll need to make a new file `/etc/apt/preferences.d/00percona.pref` and add the following lines in it:

```
Package: *
Pin: release o=Percona Development Team
Pin-Priority: 1001
```

For more information about the pinning you can check the official debian wiki.

Installing *Percona Server* using downloaded deb packages

Download the packages of the desired series for your architecture from the download page. The easiest way is to download bundle which contains all the packages. Following example will download *Percona Server* `5.7.10-3` release packages for *Debian* 8.0:

```
$ wget https://www.percona.com/downloads/Percona-Server-5.7/Percona-Server-5.7.10-3/binary/debia
```

You should then unpack the bundle to get the packages:

```
$ tar xvf Percona-Server-5.7.10-3-r63dafaf-jessie-x86_64-bundle.tar
```

After you unpack the bundle you should see the following packages:

```
$ ls *.deb
libperconaserverclient20-dev_5.7.10-3-1.jessie_amd64.deb
libperconaserverclient20_5.7.10-3-1.jessie_amd64.deb
percona-server-5.7-dbg_5.7.10-3-1.jessie_amd64.deb
percona-server-client-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-common-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-server-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-source-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-test-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-tokudb-5.7_5.7.10-3-1.jessie_amd64.deb
```

Now you can install *Percona Server* by running:

```
$ sudo dpkg -i *.deb
```

This will install all the packages from the bundle. Another option is to download/specify only the packages you need for running *Percona Server* installation (`libperconaserverclient20_5.7.10-3-1.jessie_amd64.deb`, `percona-server-client-5.7_5.7.10-3-1.jessie_amd64.deb`, `percona-server-common-5.7_5.7.10-3-1.` and `percona-server-server-5.7_5.7.10-3-1.jessie_amd64.deb`. Optionally you can install `percona-server-tokudb-5.7_5.7.10-3-1.jessie_amd64.deb` if you want *TokuDB* storage engine).

Note: *Percona Server* 5.7 comes with the *TokuDB storage engine*. You can find more information on how to install and enable the *TokuDB* storage in the *TokuDB Installation* guide.

Warning: When installing packages manually like this, you'll need to make sure to resolve all the dependencies and install missing packages yourself.

Running *Percona Server*

Percona Server stores the data files in `/var/lib/mysql/` by default. You can find the configuration file that is used to manage *Percona Server* in `/etc/mysql/my.cnf`. *Debian* and *Ubuntu* installation automatically creates a special `debian-sys-maint` user which is used by the control scripts to control the *Percona Server* `mysqld` and `mysqld_safe` services. Login details for that user can be found in `/etc/mysql/debian.cnf`.

1. Starting the service

   Percona Server is started automatically after it gets installed unless it encounters errors during the installation process. You can also manually start it by running:

   ```
   $ sudo service mysql start
   ```

2. Confirming that service is running

   You can check the service status by running:

   ```
   $ service mysql status
   ```

3. Stopping the service

   You can stop the service by running:

   ```
   $ sudo service mysql stop
   ```

4. Restarting the service

   You can restart the service by running:

   ```
   $ sudo service mysql restart
   ```

Note: *Debian* 8.0 (jessie) and *Ubuntu* 15.04 (vivid) come with systemd as the default system and service manager so you can invoke all the above commands with `sytemctl` instead of `service`. Currently both are supported.

Uninstalling *Percona Server*

To uninstall *Percona Server* you'll need to remove all the installed packages. Removing packages with **apt-get remove** will leave the configuration and data files. Removing the packages with **apt-get purge** will remove all the packages with configuration files and data files (all the databases). Depending on your needs you can choose which command better suits you.

1. Stop the *Percona Server* service

   ```
   $ sudo service mysql stop
   ```

2. Remove the packages

   (a) Remove the packages. This will leave the data files (databases, tables, logs, configuration, etc.) behind. In case you don't need them you'll need to remove them manually.

   ```
   $ sudo apt-get remove percona-server*
   ```

(a) Purge the packages. **NOTE**: This will remove all the packages and delete all the data files (databases, tables, logs, etc.)

```
$ sudo apt-get purge percona-server*
```

5.1.2 Installing *Percona Server* on Red Hat Enterprise Linux and CentOS

Ready-to-use packages are available from the *Percona Server* software repositories and the download page. The *Percona* **yum** repository supports popular *RPM*-based operating systems, including the *Amazon Linux AMI*.

The easiest way to install the *Percona Yum* repository is to install an *RPM* that configures **yum** and installs the Percona GPG key.

Supported Releases:

- *CentOS* 6 and *RHEL* 6 (Current Stable) [1]
- *CentOS* 7 and *RHEL* 7
- *Amazon Linux AMI* (works the same as *CentOS* 6)

The *CentOS* repositories should work well with *Red Hat Enterprise Linux* too, provided that **yum** is installed on the server.

Supported Platforms:

- x86
- x86_64 (also known as `amd64`)

What's in each RPM package?

Each of the *Percona Server* RPM packages have a particular purpose.

The `Percona-Server-server-57` package contains the server itself (the `mysqld` binary).

The `Percona-Server-57-debuginfo` package contains debug symbols for the server.

The `Percona-Server-client-57` package contains the command line client.

The `Percona-Server-devel-57` package contains the header files needed to compile software using the client library.

The `Percona-Server-shared-57` package includes the client shared library.

The `Percona-Server-shared-compat` package includes shared libraries for software compiled against old versions of the client library. Following libraries are included in this package: `libmysqlclient.so.12`, `libmysqlclient.so.14`, `libmysqlclient.so.15`, `libmysqlclient.so.16`, and `libmysqlclient.so.18`.

The `Percona-Server-test-57` package includes the test suite for *Percona Server*.

[1] "Current Stable": We support only the current stable RHEL6/CentOS6 release, because there is no official (i.e. RedHat provided) method to support or download the latest OpenSSL on RHEL/CentOS versions prior to 6.5. Similarly, and also as a result thereof, there is no official Percona way to support the latest Percona Server builds on RHEL/CentOS versions prior to 6.5. Additionally, many users will need to upgrade to OpenSSL 1.0.1g or later (due to the Heartbleed vulnerability), and this OpenSSL version is not available for download from any official RHEL/Centos repository for versions 6.4 and prior. For any officially unsupported system, src.rpm packages may be used to rebuild Percona Server for any environment. Please contact our support service if you require further information on this.

Installing *Percona Server* from Percona `yum` repository

1. Install the Percona repository

   You can install Percona yum repository by running the following command as a `root` user or with sudo:

   ```
   yum install http://www.percona.com/downloads/percona-release/redhat/0.1-3/percona-release-0.1-3.
   ```

   You should see some output such as the following:

   ```
   Retrieving http://www.percona.com/downloads/percona-release/redhat/0.1-3/percona-release-0.1-3.n
   Preparing...                ########################################### [100%]
      1:percona-release        ########################################### [100%]
   ```

2. Testing the repository

   Make sure packages are now available from the repository, by executing the following command:

   ```
   yum list | grep percona
   ```

   You should see output similar to the following:

   ```
   ...
   Percona-Server-57-debuginfo.x86_64      5.7.10-3.1.el7                 @percona-release-x86_64
   Percona-Server-client-57.x86_64         5.7.10-3.1.el7                 @percona-release-x86_64
   Percona-Server-devel-57.x86_64          5.7.10-3.1.el7                 @percona-release-x86_64
   Percona-Server-server-57.x86_64         5.7.10-3.1.el7                 @percona-release-x86_64
   Percona-Server-shared-57.x86_64         5.7.10-3.1.el7                 @percona-release-x86_64
   Percona-Server-shared-compat-57.x86_64  5.7.10-3.1.el7                 @percona-release-x86_64
   Percona-Server-test-57.x86_64           5.7.10-3.1.el7                 @percona-release-x86_64
   Percona-Server-tokudb-57.x86_64         5.7.10-3.1.el7                 @percona-release-x86_64
   ...
   ```

3. Install the packages

   You can now install *Percona Server* by running:

   ```
   yum install Percona-Server-server-57
   ```

Note: *Percona Server* 5.7 comes with the *TokuDB storage engine*. You can find more information on how to install and enable the *TokuDB* storage in the *TokuDB Installation* guide.

Percona *yum* Testing repository

Percona offers pre-release builds from our testing repository. To subscribe to the testing repository, you'll need to enable the testing repository in `/etc/yum.repos.d/percona-release.repo`. To do so, set both `percona-testing-$basearch` and `percona-testing-noarch` to `enabled = 1` (Note that there are 3 sections in this file: release, testing and experimental - in this case it is the second section that requires updating). **NOTE:** You'll need to install the Percona repository first (ref above) if this hasn't been done already.

Installing *Percona Server* using downloaded rpm packages

1. Download the packages of the desired series for your architecture from the download page. The easiest way is to download bundle which contains all the packages. Following example will download *Percona Server* 5.7.10-3 release packages for *CentOS* 7:

```
$ wget https://www.percona.com/downloads/Percona-Server-5.7/Percona-Server-5.7.10-3/binary/redha
```

2. You should then unpack the bundle to get the packages:

```
$ tar xvf Percona-Server-5.7.10-3-r63dafaf-el7-x86_64-bundle.tar
```

After you unpack the bundle you should see the following packages:

```
$ ls *.rpm

Percona-Server-57-debuginfo-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-client-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-devel-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-server-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-shared-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-shared-compat-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-test-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-tokudb-57-5.7.10-3.1.el7.x86_64.rpm
```

3. Now you can install *Percona Server* 5.7 by running:

```
rpm -ivh Percona-Server-server-57-5.7.10-3.1.el7.x86_64.rpm \
Percona-Server-client-57-5.7.10-3.1.el7.x86_64.rpm \
Percona-Server-shared-57-5.7.10-3.1.el7.x86_64.rpm
```

This will install only packages required to run the *Percona Server* 5.7. Optionally you can install *TokuDB* storage engine by adding the `Percona-Server-tokudb-57-5.7.10-3.1.el7.x86_64.rpm` to the command above. You can find more information on how to install and enable the *TokuDB* storage in the *TokuDB Installation* guide.

To install all the packages (for debugging, testing, etc.) you should run:

```
$ rpm -ivh *.rpm
```

Note: When installing packages manually like this, you'll need to make sure to resolve all the dependencies and install missing packages yourself.

Running *Percona Server*

Percona Server stores the data files in `/var/lib/mysql/` by default. You can find the configuration file that is used to manage *Percona Server* in `/etc/my.cnf`.

1. Starting the service

   Percona Server isn't started automatically on *RHEL* and *CentOS* after it gets installed. You should start it by running:

   ```
   service mysql start
   ```

2. Confirming that service is running

   You can check the service status by running:

   ```
   service mysql status
   ```

3. Stopping the service

   You can stop the service by running:

```
service mysql stop
```

4. Restarting the service

   You can restart the service by running:

   ```
   service mysql restart
   ```

Note: *RHEL* 7 and *CentOS* 7 come with systemd as the default system and service manager so you can invoke all the above commands with `sytemctl` instead of `service`. Currently both are supported.

Uninstalling *Percona Server*

To completely uninstall *Percona Server* you'll need to remove all the installed packages and data files.

1. Stop the *Percona Server* service

   ```
   service mysql stop
   ```

2. Remove the packages

   ```
   yum remove Percona-Server*
   ```

3. Remove the data and configuration files

   ```
   rm -rf /var/lib/mysql
   rm -f /etc/my.cnf
   ```

Warning: This will remove all the packages and delete all the data files (databases, tables, logs, etc.), you might want to take a backup before doing this in case you need the data.

5.2 Installing *Percona Server* from a Binary Tarball

Percona Server offers multiple tarballs depending on the *OpenSSL* library available in the distribution:

- ssl100 - for all *Debian/Ubuntu* versions (`libssl.so.1.0.0 => /usr/lib/x86_64-linux-gnu/libssl.so.1.0.0 (0x00007f2e389a5000)`);
- ssl101 - for *CentOS* 6 and *CentOS* 7 (`libssl.so.10 => /usr/lib64/libssl.so.10 (0x00007facbe8c4000)`);

You can download the binary tarballs from the `Linux - Generic` section on the download page.

Fetch and extract the correct binary tarball. For example for *Debian Wheezy*:

```
$ wget https://www.percona.com/downloads/Percona-Server-5.7/Percona-Server-5.7.10-3/binary/tarball/Pe
```

5.3 Installing *Percona Server* from a Source Tarball

Fetch and extract the source tarball. For example:

```
$ wget https://www.percona.com/downloads/Percona-Server-5.7/Percona-Server-5.7.10-3/binary/tarball/Pe
$ tar xfz Percona-Server-5.7.10-3-Linux.x86_64.ssl100.tar.gz
```

Next, follow the instructions in *Compiling Percona Server from Source* below.

5.4 Installing *Percona Server* from the Git Source Tree

Percona uses the Github revision control system for development. To build the latest *Percona Server* from the source tree you will need `git` installed on your system.

You can now fetch the latest *Percona Server* 5.7 sources.

```
$ git clone https://github.com/percona/percona-server.git
$ cd percona-server
$ git checkout 5.7
$ git submodule init
$ git submodule update
```

If you are going to be making changes to *Percona Server* 5.7 and wanting to distribute the resulting work, you can generate a new source tarball (exactly the same way as we do for release):

```
$ cmake .
$ make dist
```

Next, follow the instructions in *Compiling Percona Server from Source* below.

5.5 Compiling *Percona Server* from Source

After either fetching the source repository or extracting a source tarball (from Percona or one you generated yourself), you will now need to configure and build Percona Server.

First, run cmake to configure the build. Here you can specify all the normal build options as you do for a normal *MySQL* build. Depending on what options you wish to compile Percona Server with, you may need other libraries installed on your system. Here is an example using a configure line similar to the options that Percona uses to produce binaries:

```
$ cmake . -DCMAKE_BUILD_TYPE=RelWithDebInfo -DBUILD_CONFIG=mysql_release -DFEATURE_SET=community -DW
```

Now, compile using make

```
$ make
```

Install:

```
$ make install
```

Percona Server 5.7 will now be installed on your system.

5.6 Building *Percona Server* Debian/Ubuntu packages

If you wish to build your own Percona Server Debian/Ubuntu (dpkg) packages, you first need to start with a source tarball, either from the Percona website or by generating your own by following the instructions above(*Installing Percona Server from the Git Source Tree*).

Extract the source tarball:

```
$ tar xfz percona-server-5.7.10-3.tar.gz
$ cd percona-server-5.7.10-3
```

Put the debian packaging in the directory that Debian expects it to be in:

```
$ cp -ap build-ps/debian debian
```

Update the changelog for your distribution (here we update for the unstable distribution - sid), setting the version number appropriately. The trailing one in the version number is the revision of the Debian packaging.

```
$ dch -D unstable --force-distribution -v "5.7.10-3-1" "Update to 5.7.10-3"
```

Build the Debian source package:

```
$ dpkg-buildpackage -S
```

Use sbuild to build the binary package in a chroot:

```
$ sbuild -d sid percona-server-5.7_5.7.10_3-1.dsc
```

You can give different distribution options to dch and sbuild to build binary packages for all Debian and Ubuntu releases.

Note: *PAM Authentication Plugin* is not built with the server by default. In order to build the Percona Server with PAM plugin, additional option `-DWITH_PAM=ON` should be used.

CHAPTER

SIX

PERCONA SERVER IN-PLACE UPGRADING GUIDE: FROM 5.6 TO 5.7

In-place upgrades are those which are done using the existing data in the server. Generally speaking, this is stopping the server, installing the new server and starting it with the same data files. While they may not be suitable for high-complexity environments, they may be adequate for many scenarios.

The following is a summary of the more relevant changes in the 5.7 series. It's strongly recommended to that you read the following guides as they contain the list of incompatible changes that could cause automatic upgrade to fail:

- *Changed in Percona Server 5.7*
- Upgrading MySQL
- Upgrading from MySQL 5.6 to 5.7

Warning: Upgrade from 5.6 to 5.7 on a crashed instance is not recommended. If the server instance has crashed, crash recovery should be run before proceeding with the upgrade.

6.1 Upgrading using the Percona repositories

The easiest and recommended way of installing - where possible - is by using the *Percona* repositories.

Instructions for enabling the repositories in a system can be found in:

- *Percona APT Repository*
- *Percona YUM Repository*

6.1.1 DEB-based distributions

Note: Following commands will need to be run either as a root user or with **sudo**.

Having done the full backup (or dump if possible), stop the server:

```
$ service mysql stop
```

and proceed to do the modifications needed in your configuration file, as explained at the beginning of this guide.

Note: If you're running *Debian*/*Ubuntu* system with systemd as the default system and service manager you can invoke the above command with **systemctl** instead of **service**. Currently both are supported.

Then install the new server with:

```
$ apt-get install percona-server-server-5.7
```

If you're using *Percona Server* 5.6 with *TokuDB* you'll need to specify the *TokuDB* package as well:

```
$ apt-get install percona-server-server-5.7 percona-server-tokudb-5.7
```

The installation script will *NOT* run automatically **mysql_upgrade** as it was the case in previous versions. You'll need to run the command manually and restart the service after it's finished.

```
$ mysql_upgrade

Checking if update is needed.
Checking server version.
Running queries to upgrade MySQL server.
Checking system database.
mysql.columns_priv                                OK
mysql.db                                          OK
mysql.engine_cost                                 OK
...
Upgrade process completed successfully.
Checking if update is needed.

$ service mysql restart
```

Note that this procedure is the same for upgrading from *MySQL* 5.6 or 5.7 to *Percona Server* 5.7.

6.1.2 RPM-based distributions

Note: Following commands will need to be run either as a root user or with **sudo**.

Having done the full backup (and dump if possible), stop the server:

```
$ service mysql stop
```

Note: If you're running *RHEL/CentOS* system with systemd as the default system and service manager you can invoke the above command with **systemctl** instead of **service**. Currently both are supported.

and check your installed packages with:

```
$ rpm -qa | grep Percona-Server
Percona-Server-shared-56-5.6.28-rel76.1.el7.x86_64
Percona-Server-server-56-5.6.28-rel76.1.el7.x86_64
Percona-Server-devel-56-5.6.28-rel76.1.el7.x86_64
Percona-Server-client-56-5.6.28-rel76.1.el7.x86_64
Percona-Server-test-56-5.6.28-rel76.1.el7.x86_64
Percona-Server-56-debuginfo-5.6.28-rel76.1.el7.x86_64
```

After checking, proceed to remove them without dependencies:

```
$ rpm -qa | grep Percona-Server | xargs rpm -e --nodeps
```

It is important that you remove it without dependencies as many packages may depend on these (as they replace `mysql`) and will be removed if omitted.

Note that this procedure is the same for upgrading from *MySQL* 5.6 or 5.7 to *Percona Server* 5.7: just grep `'^mysql-'` instead of `Percona-Server` and remove them.

You will have to install the following package:

- `Percona-Server-server-57`

```
$ yum install Percona-Server-server-57
```

If you're using *Percona Server* 5.6 with *TokuDB* you'll need to specify the *TokuDB* package as when doing the upgrade:

```
$ yum install Percona-Server-server-57 Percona-Server-tokudb-57
```

Once installed, proceed to modify your configuration file - `my.cnf` - and reinstall the plugins if necessary.

Note: If you're using *TokuDB* storage engine you'll need to comment out all the *TokuDB* specific variables in your configuration file(s) before starting the server, otherwise server won't be able to start. *RHEL/CentOS* 7 automatically backs up the previous configuration file to `/etc/my.cnf.rpmsave` and installs the default `my.cnf`. After upgrade/install process completes you can move the old configuration file back (after you remove all the unsupported system variables).

You can now start the `mysql` service:

```
$ service mysql start
```

and use `mysql_upgrade` to migrate to the new grant tables, it will rebuild the indexes needed and do the modifications needed:

Note: If you're using *TokuDB* storage engine you'll need re-enable the storage engine plugin by running the: `ps_tokudb_admin --enable` before running `mysql_upgrade` otherwise you'll get errors.

```
$ mysql_upgrade
Checking if update is needed.
Checking server version.
Running queries to upgrade MySQL server.
Checking system database.
mysql.columns_priv                                OK
mysql.db                                          OK
...
pgrade process completed successfully.
Checking if update is needed.
```

Once this is done, just restart the server as usual:

```
$ service mysql restart
```

After the service has been successfully restarted you can use the new *Percona Server* 5.7.

6.2 Upgrading using Standalone Packages

6.2.1 DEB-based distributions

Having done the full backup (and dump if possible), stop the server:

```
$ sudo /etc/init.d/mysqld stop
```

and remove the installed packages with their dependencies:

```
$ sudo apt-get autoremove percona-server-server-56 percona-server-client-56
```

Once removed, proceed to do the modifications needed in your configuration file, as explained at the beginning of this guide.

Then, download the following packages for your architecture:

- `percona-server-server-5.7`
- `percona-server-client-5.7`
- `percona-server-common-5.7`
- `libperconaserverclient20`

Following example will download *Percona Server* `5.7.10-3` release packages for *Debian* 8.0:

```
$ wget https://www.percona.com/downloads/Percona-Server-5.7/Percona-Server-5.7.10-3/binary/debian/jes
```

You should then unpack the bundle to get the packages:

```
$ tar xvf Percona-Server-5.7.10-3-r63dafaf-jessie-x86_64-bundle.tar
```

After you unpack the bundle you should see the following packages:

```
$ ls *.deb
libperconaserverclient20-dev_5.7.10-3-1.jessie_amd64.deb
libperconaserverclient20_5.7.10-3-1.jessie_amd64.deb
percona-server-5.7-dbg_5.7.10-3-1.jessie_amd64.deb
percona-server-client-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-common-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-server-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-source-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-test-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-tokudb-5.7_5.7.10-3-1.jessie_amd64.deb
```

Now you can install *Percona Server* by running:

```
$ sudo dpkg -i *.deb
```

This will install all the packages from the bundle. Another option is to download/specify only the packages you need for running *Percona Server* installation (`libperconaserverclient20_5.7.10-3-1.jessie_amd64.deb`, `percona-server-client-5.7_5.7.10-3-1.jessie_amd64.deb`, `percona-server-common-5.7_5.7.10-3-1.` and `percona-server-server-5.7_5.7.10-3-1.jessie_amd64.deb`. Optionally you can install `percona-server-tokudb-5.7_5.7.10-3-1.jessie_amd64.deb` if you want *TokuDB* storage engine).

Note: *Percona Server* 5.7 comes with the *TokuDB storage engine*. You can find more information on how to install and enable the *TokuDB* storage in the *TokuDB Installation* guide.

Warning: When installing packages manually like this, you'll need to make sure to resolve all the dependencies and install missing packages yourself.

The installation script will not run automatically **mysql_upgrade**, so you'll need to run it yourself and restart the service afterwards.

6.2.2 RPM-based distributions

Having done the full backup (and dump if possible), stop the server:

```
$ service mysql stop
```

and check your installed packages:

```
$ rpm -qa | grep Percona-Server

Percona-Server-shared-56-5.6.28-rel76.1.el6.x86_64
Percona-Server-server-56-5.6.28-rel76.1.el6.x86_64
Percona-Server-client-56-5.6.28-rel76.1.el6.x86_64
Percona-Server-tokudb-56-5.6.28-rel76.1.el6.x86_64
```

You may have a forth, `shared-compat`, which is for compatibility purposes.

After checked that, proceed to remove them without dependencies:

```
$ rpm -qa | grep Percona-Server | xargs rpm -e --nodeps
```

It is important that you remove it without dependencies as many packages may depend on these (as they replace `mysql`) and will be removed if ommited.

Note that this procedure is the same for upgrading from *MySQL* 5.6 to *Percona Server* 5.7, just grep `'^mysql-'` instead of `Percona-Server` and remove them.

Download the packages of the desired series for your architecture from the download page. The easiest way is to download bundle which contains all the packages. Following example will download *Percona Server* 5.7.10-3 release packages for *CentOS* 7:

```
$ wget https://www.percona.com/downloads/Percona-Server-5.7/Percona-Server-5.7.10-3/binary/redhat/7/x
```

You should then unpack the bundle to get the packages:

```
$ tar xvf Percona-Server-5.7.10-3-r63dafaf-el7-x86_64-bundle.tar
```

After you unpack the bundle you should see the following packages:

```
$ ls *.rpm
Percona-Server-57-debuginfo-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-client-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-devel-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-server-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-shared-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-shared-compat-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-test-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-tokudb-57-5.7.10-3.1.el7.x86_64.rpm
```

Now you can install *Percona Server* 5.7 by running:

```
rpm -ivh Percona-Server-server-57-5.7.10-3.1.el7.x86_64.rpm \
Percona-Server-client-57-5.7.10-3.1.el7.x86_64.rpm \
Percona-Server-shared-57-5.7.10-3.1.el7.x86_64.rpm
```

This will install only packages required to run the *Percona Server* 5.7. Optionally you can install *TokuDB* storage engine by adding the `Percona-Server-tokudb-57-5.7.10-3.1.el7.x86_64.rpm` to the command above. You can find more information on how to install and enable the *TokuDB* storage in the *TokuDB Installation* guide.

To install all the packages (for debugging, testing, etc.) you should run:

```
$ rpm -ivh *.rpm
```

Note: When installing packages manually like this, you'll need to make sure to resolve all the dependencies and install missing packages yourself.

Once installed, proceed to modify your configuration file - `my.cnf` - and install the plugins if necessary. If you're using *TokuDB* storage engine you'll need to comment out all the *TokuDB* specific variables in your configuration file(s) before starting the server, otherwise server won't be able to start. *RHEL/CentOS* 7 automatically backs up the previous configuration file to `/etc/my.cnf.rpmsave` and installs the default `my.cnf`. After upgrade/install process completes you can move the old configuration file back (after you remove all the unsupported system variables).

As the schema of the grant table has changed, the server must be started without reading them:

```
$ service mysql start
```

and use `mysql_upgrade` to migrate to the new grant tables, it will rebuild the indexes needed and do the modifications needed:

Note: If you're using *TokuDB* storage engine you'll need re-enable the storage engine plugin by running the: `ps_tokudb_admin --enable` before running `mysql_upgrade` otherwise you'll get errors.

```
$ mysql_upgrade
```

After this is done, just restart the server as usual:

```
$ service mysql restart
```

Part III

Scalability Improvements

CHAPTER

SEVEN

IMPROVED BUFFER POOL SCALABILITY

The *InnoDB* buffer pool is a well known point of contention when many queries are executed concurrently. In *XtraDB*, the global mutex protecting the buffer pool has been split into several mutexes to decrease contention.

This feature splits the single global InnoDB buffer pool mutex into several mutexes:

Name	Protects
flush_state_mutex	flushing state of dirty blocks
LRU_list_mutex	LRU lists of blocks in buffer pool
flush_list_mutex	flush list of dirty blocks to flush
free_list_mutex	list of free blocks in buffer pool
zip_free_mutex	lists of free area to treat compressed pages
zip_hash_mutex	hash table to search compressed pages

The goal of this change is to reduce mutex contention, which can be very impacting when the working set does not fit in memory.

7.1 Version Specific Information

- `5.7.10-1` - Feature ported from *Percona Server* 5.6

7.2 Other Information

7.2.1 Detecting Mutex Contention

You can detect when you suffer from mutex contention in the buffer pool by reading the information provided in the SEMAPHORES section of the output of SHOW ENGINE INNODB STATUS:

Under normal circumstances this section should look like this:

```
SEMAPHORES
----------
OS WAIT ARRAY INFO: reservation count 50238, signal count 17465
Mutex spin waits 0, rounds 628280, OS waits 31338
RW-shared spins 38074, OS waits 18900; RW-excl spins 0, OS waits 0
```

If you have a high-concurrency workload this section may look like this:

```
1 ----------
2 SEMAPHORES
3 ----------
4 OS WAIT ARRAY INFO: reservation count 36255, signal count 12675
```

```
5 --Thread 10607472 has waited at buf/buf0rea.c line 420 for 0.00 seconds the semaphore:
6 Mutex at 0x358068 created file buf/buf0buf.c line 597, lock var 0
7 waiters flag 0
8 --Thread 3488624 has waited at buf/buf0buf.c line 1177 for 0.00 seconds the semaphore:
9 Mutex at 0x358068 created file buf/buf0buf.c line 597, lock var 0
10 waiters flag 0
11 --Thread 6896496 has waited at btr/btr0cur.c line 442 for 0.00 seconds the semaphore:
12 S-lock on RW-latch at 0x8800244 created in file buf/buf0buf.c line 547
13 a writer (thread id 14879600) has reserved it in mode  exclusive
14 number of readers 0, waiters flag 1
15 Last time read locked in file btr/btr0cur.c line 442
16 Last time write locked in file buf/buf0buf.c line 1797
[...]
17 Mutex spin waits 0, rounds 452650, OS waits 22573
18 RW-shared spins 27550, OS waits 13682; RW-excl spins 0, OS waits 0
```

Note that in the second case you will see indications that threads are waiting for a mutex created in the file buf/buf0buf.c (lines 5 to 7 or 8 to 10). Such an indication is a sign of buffer pool contention.

CHAPTER

EIGHT

IMPROVED *INNODB* I/O SCALABILITY

Because *InnoDB* is a complex storage engine it must be configured properly in order to perform at its best. Some points are not configurable in standard *InnoDB*. The goal of this feature is to provide a more exhaustive set of options for *XtraDB*, like ability to change the log block size.

8.1 Version Specific Information

- `5.7.10-1`
  - Feature ported from *Percona Server* 5.6

8.2 System Variables

variable `innodb_use_global_flush_log_at_trx_commit`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Type Boolean

Default Value True

Range True/False

This variable is used to control the ability of the user to set the value of the global *MySQL* variable `innodb_flush_log_at_trx_commit`.

If `innodb_use_global_flush_log_at_trx_commit=0` (False), the client can set the global *MySQL* variable, using:

```
SET innodb_use_global_flush_log_at_trx_commit=N
```

If `innodb_use_global_flush_log_at_trx_commit=1` (True), the user session will use the current value of `innodb_flush_log_at_trx_commit`, and the user cannot reset the value of the global variable using a `SET` command.

variable `innodb_flush_method`

Version Info

- `5.7.10-3` - Ported from *Percona Server* 5.6

Command Line Yes

Config File Yes

Scope Global

Dyn No

Variable Type Enumeration

Default Value `fdatasync`

Allowed Values `fdatasync`, `O_DSYNC`, `O_DIRECT`, `O_DIRECT_NO_FSYNC`, `ALL_O_DIRECT`

This is an existing *MySQL* 5.7 system variable that has a new allowed value `ALL_O_DIRECT`. It determines the method *InnoDB* uses to flush its data and log files. (See `innodb_flush_method` in the *MySQL* 5.7 Reference Manual).

The following values are allowed:

- `fdatasync`: use `fsync()` to flush data, log, and parallel doublewrite files.
- `O_SYNC`: use `O_SYNC` to open and flush the log and parallel doublewrite files; use `fsync()` to flush the data files. Do not use `fsync()` to flush the parallel doublewrite file.
- `O_DIRECT`: use O_DIRECT to open the data files and `fsync()` system call to flush data, log, and parallel doublewrite files.
- `O_DIRECT_NO_FSYNC`: use `O_DIRECT` to open the data files, but don't use `fsync()` system call to flush data, log, and parallel doublewrite files.
- `ALL_O_DIRECT`: use `O_DIRECT` to open both data and log files, and use `fsync()` to flush the data files but not the log or parallel doublewrite files. This option is recommended when *InnoDB* log files are big (more than 8GB), otherwise there might be even a performance degradation. **Note**: When using this option on *ext4* filesystem variable innodb_log_write_ahead_size should be set to 4096 (default log-block-size in *ext4*) in order to avoid the `unaligned AIO/DIO` warnings.

8.2.1 Status Variables

The following information has been added to `SHOW ENGINE INNODB STATUS` to confirm the checkpointing activity:

```
The max checkpoint age
The current checkpoint age target
The current age of the oldest page modification which has not been flushed to disk yet.
The current age of the last checkpoint
...
---
LOG
---
Log sequence number 0 1059494372
Log flushed up to   0 1059494372
Last checkpoint at  0 1055251010
Max checkpoint age  162361775
Checkpoint age target 104630090
Modified age        4092465
Checkpoint age      4243362
0 pending log writes, 0 pending chkp writes
...
```

Part IV

Performance Improvements

CHAPTER

NINE

QUERY CACHE ENHANCEMENTS

This page describes the enhancements for the query cache. At the moment three features are available:

- Disabling the cache completely
- Diagnosing contention more easily
- Ignoring comments

9.1 Diagnosing contention more easily

This features provides a new thread state - `Waiting on query cache mutex`. It has always been difficult to spot query cache bottlenecks because these bottlenecks usually happen intermittently and are not directly reported by the server. This new thread state appear in the output of SHOW PROCESSLIST, easing diagnostics.

Imagine that we run three queries simultaneously (each one in a separate thread):

```
> SELECT number from t where id > 0;
> SELECT number from t where id > 0;
> SELECT number from t where id > 0;
```

If we experience query cache contention, the output of `SHOW PROCESSLIST` will look like this:

```
> SHOW PROCESSLIST;
Id      User    Host            db      Command Time    State                           Info
2       root    localhost       test    Sleep   2       NULL
3       root    localhost       test    Query   2       Waiting on query cache mutex  SELECT number f
4       root    localhost       test    Query   1       Waiting on query cache mutex  SELECT number f
5       root    localhost       test    Query   0       NULL
```

9.2 Ignoring comments

This feature adds an option to make the server ignore comments when checking for a query cache hit. For example, consider these two queries:

```
/* first query  */ select name from users where users.name like 'Bob%';
/* retry search */ select name from users where users.name like 'Bob%';
```

By default (option off), the queries are considered different, so the server will execute them both and cache them both.

If the option is enabled, the queries are considered identical, so the server will execute and cache the first one and will serve the second one directly from the query cache.

9.3 System Variables

variable `query_cache_strip_comments`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

Default Value Off

Makes the server ignore comments when checking for a query cache hit.

9.3.1 Other Reading

- MySQL general thread states
- Query cache freezes

CHAPTER

TEN

IMPROVED NUMA SUPPORT

In cases where the buffer pool memory allocation was bigger than size of the node, system would start swapping already allocated memory even if there is available memory on other node. This is would happen if the default *NUMA* memory allocation policy was selected. In that case system would favor one node more than other which caused the node to run out of memory. Changing the allocation policy to interleaving, memory will be allocated in round-robin fashion over the available node. This can be done by using the upstream innodb_numa_interleave. This feature extends the upstream implementation by implementing the `flush_caches` variable.

It is generally recommended to enable all of the options together to maximize the performance effects on the NUMA architecture.

10.1 Version Specific Information

- **`5.7.10-1`:** Feature ported from *Percona Server* 5.6

10.2 Command-line Options for mysqld_safe

variable `flush_caches`

Command Line Yes

Config File Yes

Location mysqld_safe

Dynamic No

Variable Type Boolean

Default Value OFF

Range ON/OFF

When enabled this will flush and purge buffers/caches before starting the server to help ensure NUMA allocation fairness across nodes. This option is useful for establishing a consistent and predictable behavior for normal usage and/or benchmarking.

10.3 Other Reading

- The MySQL "swap insanity" problem and the effects of the NUMA architecture
- A brief update on NUMA and MySQL

CHAPTER

ELEVEN

THREAD POOL

MySQL executes statements using one thread per client connection. Once the number of connections increases past a certain point performance will degrade.

This feature enables the server to keep the top performance even with large number of client connections by introducing a dynamic thread pool. By using the thread pool server would decrease the number of threads, which will then reduce the context switching and hot locks contentions. Using the thread pool will have the most effect with `OLTP` workloads (relatively short CPU-bound queries).

In order to enable the thread pool variable `thread_handling` should be set up to `pool-of-threads` value. This can be done by adding:

```
thread_handling=pool-of-threads
```

to the *MySQL* configuration file `my.cnf`.

Although the default values for the thread pool should provide good performance, additional tuning can be performed with the dynamic system variables described below.

Note: Current implementation of the thread pool is built in the server, unlike the upstream version which is implemented as a plugin. Another significant implementation difference is that this implementation doesn't try to minimize the number of concurrent transactions like the `MySQL Enterprise Threadpool`. Because of these things this implementation isn't compatible with the upstream one.

11.1 Priority connection scheduling

In *Percona Server* `5.6.11-60.3` priority connection scheduling for thread pool has been implemented. Even though thread pool puts a limit on the number of concurrently running queries, the number of open transactions may remain high, because connections with already started transactions are put to the end of the queue. Higher number of open transactions has a number of implications on the currently running queries. To improve the performance new `thread_pool_high_prio_tickets` variable has been introduced.

This variable controls the high priority queue policy. Each new connection is assigned this many tickets to enter the high priority queue. Whenever a query has to be queued to be executed later because no threads are available, the thread pool puts the connection into the high priority queue if the following conditions apply:

1. The connection has an open transaction in the server.
2. The number of high priority tickets of this connection is non-zero.

If both the above conditions hold, the connection is put into the high priority queue and its tickets value is decremented. Otherwise the connection is put into the common queue with the initial tickets value specified with this option.

Each time the thread pool looks for a new connection to process, first it checks the high priority queue, and picks connections from the common queue only when the high priority one is empty.

The goal is to minimize the number of open transactions in the server. In many cases it is beneficial to give short-running transactions a chance to commit faster and thus deallocate server resources and locks without waiting in the same queue with other connections that are about to start a new transaction, or those that have run out of their high priority tickets.

The default thread pool behavior is to always put events from already started transactions into the high priority queue, as we believe that results in better performance in vast majority of cases.

With the value of `0`, all connections are always put into the common queue, i.e. no priority scheduling is used as in the original implementation in *MariaDB*. The higher is the value, the more chances each transaction gets to enter the high priority queue and commit before it is put in the common queue.

In some cases it is required to prioritize all statements for a specific connection regardless of whether they are executed as a part of a multi-statement transaction or in the autocommit mode. Or vice versa, some connections may require using the low priority queue for all statements unconditionally. To implement this new `thread_pool_high_prio_mode` variable has been introduced in *Percona Server*.

11.1.1 Low priority queue throttling

One case that can limit thread pool performance and even lead to deadlocks under high concurrency is a situation when thread groups are oversubscribed due to active threads reaching the oversubscribe limit, but all/most worker threads are actually waiting on locks currently held by a transaction from another connection that is not currently in the thread pool.

What happens in this case is that those threads in the pool that have marked themselves inactive are not accounted to the oversubscribe limit. As a result, the number of threads (both active and waiting) in the pool grows until it hits `thread_pool_max_threads` value. If the connection executing the transaction which is holding the lock has managed to enter the thread pool by then, we get a large (depending on the `thread_pool_max_threads` value) number of concurrently running threads, and thus, suboptimal performance as a result. Otherwise, we get a deadlock as no more threads can be created to process those transaction(s) and release the lock(s).

Such situations are prevented by throttling the low priority queue when the total number of worker threads (both active and waiting ones) reaches the oversubscribe limit. That is, if there are too many worker threads, do not start new transactions and create new threads until queued events from the already started transactions are processed.

11.2 Handling of Long Network Waits

Certain types of workloads (large result sets, BLOBs, slow clients) can have longer waits on network I/O (socket reads and writes). Whenever server waits, this should be communicated to the Thread Pool, so it can start new query by either waking a waiting thread or sometimes creating a new one. This implementation has been ported from *MariaDB* patch MDEV-156 in *Percona Server* `5.6.15-63.0`.

11.3 Version Specific Information

- **`5.6.10-60.2`** `Thread Pool` feature ported from *Percona Server* 5.6.

11.4 System Variables

variable `thread_pool_idle_timeout`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 60 (seconds)

This variable can be used to limit the time an idle thread should wait before exiting.

variable `thread_pool_high_prio_mode`

Command Line Yes

Config File Yes

Scope Global, Session

Dynamic Yes

Variable Type String

Default Value `transactions`

Allowed Values `transactions`, `statements`, `none`

This variable is used to provide more fine-grained control over high priority scheduling either globally or per connection.

The following values are allowed:

- `transactions` (the default). In this mode only statements from already started transactions may go into the high priority queue depending on the number of high priority tickets currently available in a connection (see `thread_pool_high_prio_tickets`).
- `statements`. In this mode all individual statements go into the high priority queue, regardless of connection's transactional state and the number of available high priority tickets. This value can be used to prioritize `AUTOCOMMIT` transactions or other kinds of statements such as administrative ones for specific connections. Note that setting this value globally essentially disables high priority scheduling, since in this case all statements from all connections will use a single queue (the high priority one)
- `none`. This mode disables high priority queue for a connection. Some connections (e.g. monitoring) may be insensitive to execution latency and/or never allocate any server resources that would otherwise impact performance in other connections and thus, do not really require high priority scheduling. Note that setting `thread_pool_high_prio_mode` to `none` globally has essentially the same effect as setting it to `statements` globally: all connections will always use a single queue (the low priority one in this case).

variable `thread_pool_high_prio_tickets`

Command Line Yes

Config File Yes

Scope Global, Session

Dynamic Yes

Variable Type Numeric

Default Value 4294967295

This variable controls the high priority queue policy. Each new connection is assigned this many tickets to enter the high priority queue. Setting this variable to `0` will disable the high priority queue.

variable `thread_pool_max_threads`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 100000

This variable can be used to limit the maximum number of threads in the pool. Once this number is reached no new threads will be created.

variable `thread_pool_oversubscribe`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 3

The higher the value of this parameter the more threads can be run at the same time, if the values is lower than `3` it could lead to more sleeps and wake-ups.

variable `thread_pool_size`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value Number of processors

This variable can be used to define the number of threads that can use the CPU at the same time.

variable `thread_pool_stall_limit`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Numeric

Default Value 500 (ms)

The number of milliseconds before a running thread is considered stalled. When this limit is reached thread pool will wake up or create another thread. This is being used to prevent a long-running query from monopolizing the pool.

variable `extra_port`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Numeric

Default Value 0

This variable can be used to specify additional port *Percona Server* will listen on. This can be used in case no new connections can be established due to all worker threads being busy or being locked when `pool-of-threads` feature is enabled. To connect to the extra port following command can be used:

```
mysql --port='extra-port-number' --protocol=tcp
```

variable `extra_max_connections`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 1

This variable can be used to specify the maximum allowed number of connections plus one extra `SUPER` users connection on the `extra_port`. This can be used with the `extra_port` variable to access the server in case no new connections can be established due to all worker threads being busy or being locked when `pool-of-threads` feature is enabled.

11.5 Status Variables

variable `Threadpool_idle_threads`

Variable Type Numeric

Scope Global

This status variable shows the number of idle threads in the pool.

variable `Threadpool_threads`

Variable Type Numeric

Scope Global

This status variable shows the number of threads in the pool.

11.6 Other Reading

- Thread pool in MariaDB 5.5
- Thread pool implementation in Oracle MySQL

CHAPTER

TWELVE

XTRADB PERFORMANCE IMPROVEMENTS FOR I/O-BOUND HIGHLY-CONCURRENT WORKLOADS

12.1 Priority refill for the buffer pool free list

In highly-concurrent I/O-bound workloads the following situation may happen:

1. Buffer pool free lists are used faster than they are refilled by the LRU cleaner thread.
2. Buffer pool free lists become empty and more and more query and utility (i.e. purge) threads stall, checking whether a buffer pool free list has became non-empty, sleeping, performing single-page LRU flushes.
3. The number of buffer pool free list mutex waiters increases.
4. When the LRU manager thread (or a single page LRU flush by a query thread) finally produces a free page, it is starved from putting it on the buffer pool free list as it must acquire the buffer pool free list mutex too. However, being one thread in up to hundreds, the chances of a prompt acquisition are low.

To avoid this *Percona Server* has implemented priority refill for the buffer pool buffer pool free list. This implementation adjusts the buffer pool free list producer to always acquire the mutex with high priority and buffer pool free list consumer to always acquire the mutex with low priority. The implementation makes use of *thread priority lock framework*.

Even the above implementation does not fully resolve the buffer pool free list mutex contention, as the mutex is still being acquired needlessly whenever the buffer pool free list is empty. This is addressed by delegating all the LRU flushes to the to the LRU manager thread, never attempting to evict a page or perform a LRU single page flush by a query thread, and introducing a backoff algorithm to reduce buffer pool free list mutex pressure on empty buffer pool free lists. This is controlled through a new system variable `innodb_empty_free_list_algorithm`.

variable `innodb_empty_free_list_algorithm`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Values legacy, backoff

Default Value backoff

When `legacy` option is set, server will used the upstream algorithm and when the `backoff` is selected, *Percona* implementation will be used.

12.2 Multi-threaded LRU flusher

Percona Server `5.7.10-3` has introduced a true multi-threaded LRU flushing. In this scheme, each buffer pool instance has its own dedicated LRU manager thread that is tasked with performing LRU flushes and evictions to refill the free list of that buffer pool instance. Existing multi-threaded flusher no longer does any LRU flushing and is tasked with flush list flushing only.

This has been done to address the shortcomings of the existing MySQL 5.7 multi-threaded flusher:

- All threads still synchronize on each coordinator thread iteration. If a particular flushing job is stuck on one of the worker threads, the rest will idle until the stuck one completes.
- The coordinator thread heuristics focus on flush list adaptive flushing without considering the state of free lists, which might be in need of urgent refill for a subset of buffer pool instances on a loaded server.
- LRU flushing is serialized with flush list flushing for each buffer pool instance, introducing the risk that the right flushing mode will not happen for a particular instance because it is being flushed in the other mode.

The following *InnoDB* metrics are no longer accounted, as their semantics do not make sense under the current LRU flushing design: `buffer_LRU_batch_flush_avg_time_slot`, `buffer_LRU_batch_flush_avg_pass`, `buffer_LRU_batch_flush_avg_time_thread`, `buffer_LRU_batch_flush_avg_time_est`.

The need for *InnoDB* recovery thread writer threads is also removed, consequently all associated code is deleted.

12.3 Parallel doublewrite buffer

The legacy doublewrite buffer is shared between all the buffer pool instances and all the flusher threads. It collects all the page write requests into a single buffer, and, when the buffer fills, writes it out to disk twice, blocking any new write requests until the writes complete. This becomes a bottleneck with increased flusher parallelism, limiting the effect of extra cleaner threads. In addition, single page flushes, if they are performed, are subject to above and also contend on the doublewrite mutex.

To address these issues *Percona Server* `5.7.11-4` has introduced private doublewrite buffers for each buffer pool instance, for each batch flushing mode (LRU or flush list). For example, with four buffer pool instances, there will eight doublewrite shards. Only one flusher thread can access any shard at a time, and each shard is added to and flushed completely independently from the rest. This does away with the mutex and the event wait does not block other threads from proceeding anymore, it only waits for the asynchronous I/O to complete. The only inter-thread synchronization is between the flusher thread and I/O completion threads.

The new doublewrite buffer is contained in a new file, where all the shards are contained, at different offsets. This file is created on startup, and removed on a clean shutdown. If it's found on a crashed instance startup, its contents are read and any torn pages are restored. If it's found on a clean instance startup, the server startup is aborted with an error message.

The location of the doublewrite file is governed by a new `innodb_parallel_doublewrite_path` global, read-only system variable. It defaults to `xb_doublewrite` in the data directory. The variable accepts both absolute and relative paths. In the latter case they are treated as relative to the data directory. The doublewrite file is not a tablespace from *InnoDB* internals point of view.

The legacy *InnoDB* doublewrite buffer in the system tablespace continues to address doublewrite needs of single page flushes, and they are free to use the whole of that buffer (128 pages by default) instead of the last eight pages as currently used. Note that single page flushes will not happen in *Percona Server* unless `innodb_empty_free_list_algorithm` is set to `legacy` value.

The existing system tablespace is not touched in any way for this feature implementation, ensuring that cleanly-shutdown instances may be freely moved between different server flavors.

12.3.1 Interaction with `innodb_flush_method`

Regardless of `innodb_flush_method` setting, the parallel doublewrite file is opened with `O_DIRECT` flag to remove OS caching, then its access is further governed by the exact value set: if it's set to `O_DSYNC`, the parallel doublewrite is opened with `O_SYNC` flag too. Further, if it's one of `O_DSYNC`, `O_DIRECT_NO_FSYNC`, or `ALL_O_DIRECT`, then the doublewrite file is not flushed after a batch of writes to it is completed.

variable `innodb_parallel_doublewrite_path`

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value `xb_doublewrite`

This variable is used to specify the location of the parallel doublewrite file. It accepts both absolute and relative paths. In the latter case they are treated as relative to the data directory.

Percona Server has introduced several options, only available in builds compiled with `UNIV_PERF_DEBUG` C preprocessor define.

variable `innodb_sched_priority_master`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

12.4 Version Specific Information

- `5.7.10-1`
  - Feature partially ported from *Percona Server* 5.6
- `5.7.10-3`
  - Implemented support for multi-threaded LRU
- `5.7.11-4`
  - Implemented support for parallel doublewrite buffer

12.5 Other Reading

- *Page cleaner thread tuning*
- Bug #74637 - make dirty page flushing more adaptive
- Bug #67808 - in innodb engine, double write and multi-buffer pool instance reduce concurrency
- Bug #69232 - buf_dblwr->mutex can be splited into two

Part V

Flexibility Improvements

CHAPTER

THIRTEEN

SUPPRESS WARNING MESSAGES

This feature is intended to provide a general mechanism (using `log_warnings_silence`) to disable certain warning messages to the log file. Currently, it is only implemented for disabling message #1592 warnings. This feature does not influence warnings delivered to a client.

13.1 Version Specific Information

- `5.7.10-1`: Variable `log_warnings_suppress` ported from *Percona Server* 5.6.
- `5.7.11-4`: Feature has been removed from *Percona Server* 5.7 because *MySQL* 5.7.11 has implemented a new system variable, log_statements_unsafe_for_binlog, which implements the same effect.

13.2 System Variables

variable `log_warnings_suppress`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type SET

Default Value `(empty string)`

Range `(empty string),1592`

It is intended to provide a more general mechanism for disabling warnings than existed previously with variable `suppress_log_warning_1592`. When set to the empty string, no warnings are disabled. When set to `1592`, warning #1592 messages (unsafe statement for binary logging) are suppressed. In the future, the ability to optionally disable additional warnings may also be added.

13.3 Related Reading

- MySQL bug 42851
- MySQL InnoDB replication
- InnoDB Startup Options and System Variables

- InnoDB Error Handling

CHAPTER

FOURTEEN

IMPROVED MEMORY STORAGE ENGINE

As of MySQL 5.5.15, a *Fixed Row Format* (FRF) is still being used in the MEMORY storage engine. The fixed row format imposes restrictions on the type of columns as it assigns on advance a limited amount of memory per row. This renders a VARCHAR field in a CHAR field in practice and makes impossible to have a TEXT or BLOB field with that engine implementation.

To overcome this limitation, the *Improved MEMORY Storage Engine* is introduced in this release for supporting **true** VARCHAR, VARBINARY, TEXT and BLOB fields in MEMORY tables.

This implementation is based on the *Dynamic Row Format* (DFR) introduced by the mysql-heap-dynamic-rows patch.

DFR is used to store column values in a variable-length form, thus helping to decrease memory footprint of those columns and making possible BLOB and TEXT fields and real VARCHAR and VARBINARY.

Unlike the fixed implementation, each column value in DRF only uses as much space as required. This is, for variable-length values, up to 4 bytes is used to store the actual value length, and then only the necessary number of blocks is used to store the value.

Rows in DFR are represented internally by multiple memory blocks, which means that a single row can consist of multiple blocks organized into one set. Each row occupies at least one block, there can not be multiple rows within a single block. Block size can be configured when creating a table (see below).

This DFR implementation has two caveats regarding to ordering and indexes.

14.1 Caveats

14.1.1 Ordering of Rows

In the absence of ORDER BY, records may be returned in a different order than the previous MEMORY implementation.

This is not a bug. Any application relying on a specific order without an ORDER BY clause may deliver unexpected results. A specific order without ORDER BY is a side effect of a storage engine and query optimizer implementation which may and will change between minor *MySQL* releases.

14.1.2 Indexing

It is currently impossible to use indexes on BLOB columns due to some limitations of the *Dynamic Row Format*. Trying to create such an index will fail with the following error:

```
BLOB column '<name>' can't be used in key specification with the used table type.
```

14.2 Restrictions

For performance reasons, a mixed solution is implemented: the fixed format is used at the beginning of the row, while the dynamic one is used for the rest of it.

The size of the fixed-format portion of the record is chosen automatically on `CREATE TABLE` and cannot be changed later. This, in particular, means that no indexes can be created later with `CREATE INDEX` or `ALTER TABLE` when the dynamic row format is used.

All values for columns used in indexes are stored in fixed format at the first block of the row, then the following columns are handled with `DRF`.

This sets two restrictions to tables:

- the order of the fields and therefore,
- the minimum size of the block used in the table.

14.2.1 Ordering of Columns

The columns used in fixed format must be defined before the dynamic ones in the `CREATE TABLE` statement. If this requirement is not met, the engine will not be able to add blocks to the set for these fields and they will be treated as fixed.

14.2.2 Minimum Block Size

The block size has to be big enough to store all fixed-length information in the first block. If not, the `CREATE TABLE` or `ALTER TABLE` statements will fail (see below).

14.3 Setting Row Format

Taking the restrictions into account, the *Improved MEMORY Storage Engine* will choose `DRF` over `FRF` at the moment of creating the table according to following criteria:

- There is an implicit request of the user in the column types **OR**
- There is an explicit request of the user **AND** the overhead incurred by `DFR` is beneficial.

14.3.1 Implicit Request

The implicit request by the user is taken when there is at least one `BLOB` or `TEXT` column in the table definition. If there are none of these columns and no relevant option is given, the engine will choose `FRF`.

For example, this will yield the use of the dynamic format:

```
mysql> CREATE TABLE t1 (f1 VARCHAR(32), f2 TEXT, PRIMARY KEY (f1)) ENGINE=HEAP;
```

While this will not:

```
mysql> CREATE TABLE t1 (f1 VARCHAR(16), f2 VARCHAR(16), PRIMARY KEY (f1)) ENGINE=HEAP;
```

14.3.2 Explicit Request

The explicit request is set with one of the following options in the `CREATE TABLE` statement:

- `KEY_BLOCK_SIZE = <value>`
  - Requests the DFR with the specified block size (in bytes)
- `ROW_FORMAT = DYNAMIC`
  - Requests the dynamic format with the default block size (256 bytes)

Despite its name, the `KEY_BLOCK_SIZE` option refers to a block size used to store data rather then indexes. The reason for this is that an existing `CREATE TABLE` option is reused to avoid introducing new ones.

The Improved MEMORY Engine checks whether the specified block size is large enough to keep all key column values. If it is too small, table creation will abort with an error.

After `DRF` is requested explicitly and there are no `BLOB` or `TEXT` columns in the table definition, the *Improved MEMORY Engine* will check if using the dynamic format provides any space saving benefits as compared to the fixed one:

- if the fixed row length is less than the dynamic block size (plus the dynamic row overhead - platform dependent) **OR**
- there isn't any variable-length columns in the table or `VARCHAR` fields are declared with length 31 or less,

the engine will revert to the fixed format as it is more space efficient in such case. The row format being used by the engine can be checked using `SHOW TABLE STATUS`.

14.4 Examples

On a 32-bit platform:

```
mysql> CREATE TABLE t1 (f1 VARCHAR(32), f2 VARCHAR(32), f3 VARCHAR(32), f4 VARCHAR(32),
                        PRIMARY KEY (f1)) KEY_BLOCK_SIZE=124 ENGINE=HEAP ROW_FORMAT=DYNAMIC;

mysql> SHOW TABLE STATUS LIKE 't1';
Name  Engine  Version Row_format      Rows    Avg_row_length  Data_length     Max_data_length Index_l
t1    MEMORY  10      Dynamic 0       X       0       X       0       0       NULL    NULL    NULL
```

On a 64-bit platform:

```
mysql> CREATE TABLE t1 (f1 VARCHAR(32), f2 VARCHAR(32), f3 VARCHAR(32), f4 VARCHAR(32),
                        PRIMARY KEY (f1)) KEY_BLOCK_SIZE=124 ENGINE=HEAP ROW_FORMAT=DYNAMIC;

mysql> SHOW TABLE STATUS LIKE 't1';
Name  Engine  Version Row_format      Rows    Avg_row_length  Data_length     Max_data_length Index_l
t1    MEMORY  10      Fixed   0       X       0       X       0       0       NULL    NULL    NULL
```

14.5 Implementation Details

MySQL MEMORY tables keep data in arrays of fixed-size chunks. These chunks are organized into two groups of `HP_BLOCK` structures:

- `group1` contains indexes, with one `HP_BLOCK` per key (part of `HP_KEYDEF`),
- `group2` contains record data, with a single `HP_BLOCK` for all records.

While columns used in indexes are usually small, other columns in the table may need to accommodate larger data. Typically, larger data is placed into `VARCHAR` or `BLOB` columns.

The Improved MEMORY Engine implements the concept of dataspace, `HP_DATASPACE`, which incorporates the `HP_BLOCK` structures for the record data, adding more information for managing variable-sized records.

Variable-size records are stored in multiple "chunks", which means that a single record of data (a database "row") can consist of multiple chunks organized into one "set", contained in `HP_BLOCK` structures.

In variable-size format, one record is represented as one or many chunks depending on the actual data, while in fixed-size mode, one record is always represented as one chunk. The index structures would always point to the first chunk in the chunkset.

Variable-size records are necessary only in the presence of variable-size columns. The *Improved Memory Engine* will be looking for `BLOB` or `VARCHAR` columns with a declared length of 32 or more. If no such columns are found, the table will be switched to the fixed-size format. You should always put such columns at the end of the table definition in order to use the variable-size format.

Whenever data is being inserted or updated in the table, the *Improved Memory Engine* will calculate how many chunks are necessary.

For `INSERT` operations, the engine only allocates new chunksets in the recordspace. For `UPDATE` operations it will modify the length of the existing chunkset if necessary, unlinking unnecessary chunks at the end, or allocating and adding more if a larger length is needed.

When writing data to chunks or copying data back to a record, fixed-size columns are copied in their full format, while `VARCHAR` and `BLOB` columns are copied based on their actual length, skipping any `NULL` values.

When allocating a new chunkset of N chunks, the engine will try to allocate chunks one-by-one, linking them as they become allocated. For allocating a single chunk, it will attempt to reuse a deleted (freed) chunk. If no free chunks are available, it will try to allocate a new area inside a `HP_BLOCK`.

When freeing chunks, the engine will place them at the front of a free list in the dataspace, each one containing a reference to the previously freed chunk.

The allocation and contents of the actual chunks varies between fixed and variable-size modes:

- Format of a fixed-size chunk:
  - `uchar[]`
    - With `sizeof=chunk_dataspace_length`, but at least `sizeof(uchar*)` bytes. It keeps actual data or pointer to the next deleted chunk, where `chunk_dataspace_length` equals to full record length
  - `uchar`
    - Status field (1 means "in use", 0 means "deleted")
- Format of a variable-size chunk:
  - `uchar[]`
    - With `sizeof=chunk_dataspace_length`, but at least `sizeof(uchar*)` bytes. It keeps actual data or pointer to the next deleted chunk, where `chunk_dataspace_length` is set according to table's `key_block_size`
  - `uchar*`
    - Pointer to the next chunk in this chunkset, or NULL for the last chunk
  - `uchar`
    - Status field (1 means "first", 0 means "deleted", 2 means "linked")

Total chunk length is always aligned to the next `sizeof(uchar*)`.

14.6 See Also

- Dynamic row format for MEMORY tables

CHAPTER

FIFTEEN

RESTRICTING THE NUMBER OF BINLOG FILES

Maximum number of binlog files can now be restricted in *Percona Server* with `max_binlog_files`. When variable `max_binlog_files` is set to non-zero value, the server will remove the oldest binlog file(s) whenever their number exceeds the value of the variable.

This variable can be used with the existing `max_binlog_size` variable to limit the disk usage of the binlog files. If `max_binlog_size` is set to 1G and `max_binlog_files` to 20 this will limit the maximum size of the binlogs on disk to 20G. The actual size limit is not necessarily `max_binlog_size` * `max_binlog_files`. Server restart or `FLUSH LOGS` will make the server start a new log file and thus resulting in log files that are not fully written in these cases limit will be lower.

15.1 Version Specific Information

- 5.7.10-1: Variable `max_binlog_files` ported from *Percona Server* 5.6.

15.2 System Variables

variable `max_binlog_files`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type ULONG

Default Value 0 (unlimited)

Range 0-102400

15.3 Example

Number of the binlog files before setting this variable

```
$ ls -l mysql-bin.0* | wc -l
26
```

Variable `max_binlog_files` is set to 20:

```
max_binlog_files = 20
```

In order for new value to take effect `FLUSH LOGS` needs to be run. After that the number of binlog files is 20

```
$ ls -l mysql-bin.0* | wc -l
20
```

CHAPTER

SIXTEEN

EXTENDED `SELECT INTO OUTFILE/DUMPFILE`

Percona Server has extended the `SELECT INTO ... OUTFILE` and `SELECT INTO DUMPFILE` commands to add the support for UNIX sockets and named pipes. Before this was implemented the database would return an error for such files.

This feature allows using `LOAD DATA LOCAL INFILE` in combination with `SELECT INTO OUTFILE` to quickly load multiple partitions across the network or in other setups, without having to use an intermediate file which wastes space and I/O.

16.1 Version Specific Information

- `5.7.10-1` - Feature ported from *Percona Server* 5.6

16.2 Other Reading

- *MySQL* bug: #44835

CHAPTER

SEVENTEEN

PER-QUERY VARIABLE STATEMENT

Percona Server has implemented per-query variable statement support. This feature provides the ability to set variable values only for a certain query, after execution of which the previous values will be restored. Per-query variable values can be set up with the following command:

```
mysql> SET STATEMENT <variable=value> FOR <statement>;
```

17.1 Examples

If we want to increase the `sort_buffer_size` value just for one specific sort query we can do it like this:

```
mysql> SET STATEMENT sort_buffer_size=100000 FOR SELECT name FROM name ORDER BY name;
```

This feature can also be used with *Statement Timeout* to limit the execution time for a specific query:

```
mysql> SET STATEMENT max_statement_time=1000 FOR SELECT name FROM name ORDER BY name;
```

We can provide more than one variable we want to set up:

```
mysql> SET STATEMENT sort_buffer_size=100000, max_statement_time=1000 FOR SELECT name FROM name ORDE
```

17.2 Version Specific Information

- `5.7.10-1` Feature ported from *Percona Server* 5.6

17.3 Other Reading

- WL#681: Per query variable settings

CHAPTER

EIGHTEEN

EXTENDED `MYSQLBINLOG`

Percona Server has implemented compression support for **mysqlbinlog**. This is similar to support that both `mysql` and `mysqldump` programs include (the `-C`, `--compress` options "Use compression in server/client protocol"). Using the compressed protocol helps reduce the bandwidth use and speed up transfers.

Percona Server has also implemented support for `SSL`. **mysqlbinlog** now accepts the `SSL` connection options as all the other client programs. This feature can be useful with `--read-from-remote-server` option. Following `SSL` options are now available:

- `--ssl` - Enable SSL for connection (automatically enabled with other flags).
- `--ssl-ca=name` - CA file in PEM format (check OpenSSL docs, implies –ssl).
- `--ssl-capath=name` - CA directory (check OpenSSL docs, implies –ssl).
- `--ssl-cert=name` - X509 cert in PEM format (implies –ssl).
- `--ssl-cipher=name` - SSL cipher to use (implies –ssl).
- `--ssl-key=name` - X509 key in PEM format (implies –ssl).
- `--ssl-verify-server-cert` - Verify server's "Common Name" in its cert against hostname used when connecting. This option is disabled by default.

18.1 Version Specific Information

- `5.7.10-1` Feature ported from *Percona Server* 5.6

CHAPTER

NINETEEN

SLOW QUERY LOG ROTATION AND EXPIRATION

Note: This feature is currently considered BETA quality.

Percona has implemented two new variables, `max_slowlog_size` and `max_slowlog_files` to provide users with ability to control the slow query log disk usage. These variables have the same behavior as upstream variable max_binlog_size and `max_binlog_files` variable used for controlling the binary log.

Warning: For this feature to work variable `slow_query_log_file` needs to be set up manually and without the `.log` sufix. The slow query log files will be named using `slow_query_log_file` as a stem, to which a dot and a sequence number will be appended.

19.1 Version Specific Information

- `5.7.10-1`: Feature ported from *Percona Server* 5.6

19.2 System Variables

variable `max_slowlog_size`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type numeric

Default Value 0 (unlimited)

Range 4096 - 1073741824

Slow query log will be rotated automatically when its size exceeds this value. The default is `0`, don't limit the size. When this feature is enabled slow query log file will be renamed to `slow_query_log_file`.000001.

variable `max_slowlog_files`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type numeric

Default Value 0 (unlimited)

Range 0 - 102400

Maximum number of slow query log files. Used with `max_slowlog_size` this can be used to limit the total amount of slow query log files. When this number is reached server will create a new slow query log file with increased sequence number. Log file with the lowest sequence number will be deleted.

CHAPTER

TWENTY

CSV ENGINE MODE FOR STANDARD-COMPLIANT QUOTE AND COMMA PARSING

MySQL CSV Storage Engine is non-standard with respect to embedded " and , character parsing. Fixing this issue unconditionally would break *MySQL* CSV format compatibility for any pre-existing user tables and for data exchange with other *MySQL* instances, but it would improve compatibility with other CSV producing/consuming tools.

To keep both *MySQL* and other tool compatibility, a new dynamic, global/session server variable `csv_mode` has been implemented. This variable allows an empty value (the default), and `IETF_QUOTES`.

If `IETF_QUOTES` is set, then embedded commas are accepted in quoted fields as-is, and a quote character is quoted by doubling it. In legacy mode embedded commas terminate the field, and quotes are quoted with a backslash.

20.1 Example

Table:

```
mysql> CREATE TABLE albums (
  `artist` text NOT NULL,
  `album` text NOT NULL
  ) ENGINE=CSV DEFAULT CHARSET=utf8
  ;
```

Following example shows the difference in parsing for default and `IETF_QUOTES` `csv_quotes`.

```
mysql> INSERT INTO albums VALUES ("Great Artist", "Old Album"),
 ("Great Artist", "Old Album \"Limited Edition\"");
```

If the variable `csv_mode` is set to empty value (default) parsed data will look like:

```
"Great Artist","Old Album"
"Great Artist","\"Limited Edition\",Old Album"
```

If the variable `csv_mode` is set to `IETF_QUOTES` parsed data will look like as described in CSV rules:

```
"Great Artist","Old Album"
"Great Artist","""Limited Edition"",Old Album"
```

Parsing the CSV file which has the proper quotes (shown in the previous example) can show different results:

With `csv_mode` set to empty value, parsed data will look like:

```
mysql> select * from albums;
+--------------+--------------------+
| artist       | album              |
+--------------+--------------------+
```

```
| Great Artist | Old Album          |
| Great Artist | ""Limited Edition" |
+--------------+--------------------+
2 rows in set (0.02 sec)
```

With `csv_mode` set to `IETF_QUOTES` parsed data will look like:

```
mysql> set csv_mode='IETF_QUOTES';
Query OK, 0 rows affected (0.00 sec)

mysql> select * from albums;
+--------------+----------------------------+
| artist       | album                      |
+--------------+----------------------------+
| Great Artist | Old Album                  |
| Great Artist | "Limited Edition",Old Album |
+--------------+----------------------------+
```

20.2 Version Specific Information

- `5.7.10-1`: Feature ported from *Percona Server* 5.6

20.3 System Variables

variable `csv_mode`

Command Line Yes

Config File Yes

Scope Global, Session

Dynamic Yes

Variable Type SET

Default Value `(empty string)`

Range `(empty string), IETF_QUOTES`

Setting this variable is to `IETF_QUOTES` will enable the standard-compliant quote parsing: commas are accepted in quoted fields as-is, and quoting of `"` is changed from `\"` to `""`. If the variable is set to empty value (the default), then the old parsing behavior is kept.

20.4 Related Reading

- MySQL bug #71091

CHAPTER

TWENTYONE

SUPPORT FOR PROXY PROTOCOL

The proxy protocol allows an intermediate proxying server speaking proxy protocol (ie. HAProxy) between the server and the ultimate client (i.e. mysql client etc) to provide the source client address to the server, which normally would only see the proxying server address instead.

As the proxy protocol amounts to spoofing the client address, it is disabled by default, and can be enabled on per-host or per-network basis for the trusted source addresses where trusted proxy servers are known to run. Unproxied connections are not allowed from these source addresses.

Note: You need to ensure proper firewall ACL's in place when this feature is enabled.

Proxying is supported for TCP over IPv4 and IPv6 connections only. UNIX socket connections can not be proxied and do not fall under the effect of proxy-protocol-networks='*'.

As a special exception, it is forbidden for the proxied IP address to be `127.0.0.1` or `::1`.

21.1 Version Specific Information

- `5.7.10-1`: Feature ported from *Percona Server* 5.6

21.2 System Variables

variable `proxy_protocol_networks`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value `(empty string)`

This variable is a global-only, read-only variable, which is either a `*` (to enable proxying globally, a non-recommended setting), or a list of comma-separated IPv4 and IPv6 network and host addresses, for which proxying is enabled. Network addresses are specified in CIDR notation, i.e. `192.168.0.0/24`. To prevent source host spoofing, the setting of this variable must be as restrictive as possible to include only trusted proxy hosts.

21.3 Related Reading

- PROXY protocol specification

CHAPTER

TWENTYTWO

PER-SESSION SERVER-ID

Variable `server_id` is a global variable. In multi-master replication setups or for external replication, `server_id` can be useful as a session variable. In that case a session replaying a binary log from another server would set it to that server's id. That way binary log has the ultimate source server id attached to it no matter how many hosts it passes, and it would provide loop detection for multi-master replication.

This was implemented by introducing the new `pseudo_server_id` variable. This variable, when set to non-zero value, will cause all binary log events in that session to have that `server_id` value. A new variable was introduced instead of converting `server_id` to have both global and session scope in order to preserve compatibility.

You should use this option at your own risk because it is very easy to break replication when using `pseudo_server_id`. One special case is circular replication which definitely will be broken if you set `pseudo_server_id` to a value not assigned to any participating server (ie., setup is `1->2->3->4->1`, and `pseudo_server_id` is set to `5`). It is also possible to create a temporary table `foo`, then change `pseudo_server_id` and from now `foo` will not be visible by this session until `pseudo_server_id` gets restored.

22.1 Version Specific Information

- `5.7.10-1` Feature ported from *Percona Server* 5.6

22.2 System Variables

variable `pseudo_server_id`

Command Line Yes

Config File No

Scope Session

Dynamic Yes

Default Value 0

When this variable is set to `0` (default), it will use the global `server_id` value. **Note:** this is different from the setting the global `server_id` to `0` which disables replication. Setting this variable to non-zero value will cause binary log events in that session to have it as `server_id` value. Setting this variable requires `SUPER` privileges.

22.3 Other Reading

- MDEV-500 - Session variable for server_id
- Upstream bug #35125 - allow the ability to set the server_id for a connection for logging to binary log

Part VI

Reliability Improvements

CHAPTER

TWENTYTHREE

TOO MANY CONNECTIONS WARNING

This feature issues the warning `Too many connections` to the log, if `log_error_verbosity` is set to `2` or higher.

23.1 Version-Specific Information

- `5.7.10-1`: Feature ported from *Percona Server* 5.6.

CHAPTER

TWENTYFOUR

HANDLE CORRUPTED TABLES

Instead of crashing the server as they used to do, corrupted *InnoDB* tables are simply disabled, so that the database remains available while the corruption is being fixed.

This feature adds a new system variable.

24.1 Version Specific Information

- `5.7.10-1`: Feature ported from *Percona Server* 5.6

24.2 System Variables

variable `innodb_corrupt_table_action`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type ULONG

Range `assert`, `warn`, `salvage`

- With the default value *XtraDB* will intentionally crash the server with an assertion failure as it would normally do when detecting corrupted data in a single-table tablespace.
- If the `warn` value is used it will pass corruption of the table as `corrupt table` instead of crashing itself. For this to work `innodb_file_per_table` should be enabled. All file I/O for the datafile after detected as corrupt is disabled, except for the deletion.
- When the option value is `salvage`, *XtraDB* allows read access to a corrupted tablespace, but ignores corrupted pages".

Part VII

Management Improvements

CHAPTER

TWENTYFIVE

PERCONA TOOLKIT UDFS

Three *Percona Toolkit* UDFs that provide faster checksums are provided:

- `libfnv1a_udf`
- `libfnv_udf`
- `libmurmur_udf`

25.1 Version Specific Information

- `5.7.10-1`: Feature ported from *Percona Server* 5.6

25.2 Other Information

- Author / Origin: Baron Schwartz

25.3 Installation

These UDFs are part of the *Percona Server* packages. To install one of the UDFs into the server, execute one of the following commands, depending on which UDF you want to install:

```
mysql -e "CREATE FUNCTION fnv1a_64 RETURNS INTEGER SONAME 'libfnv1a_udf.so'"
mysql -e "CREATE FUNCTION fnv_64 RETURNS INTEGER SONAME 'libfnv_udf.so'"
mysql -e "CREATE FUNCTION murmur_hash RETURNS INTEGER SONAME 'libmurmur_udf.so'"
```

Executing each of these commands will install its respective UDF into the server.

25.4 Troubleshooting

If you get the error:

```
ERROR 1126 (HY000): Can't open shared library 'fnv_udf.so' (errno: 22 fnv_udf.so: cannot open shared
```

Then you may need to copy the .so file to another location in your system. Try both `/lib` and `/usr/lib`. Look at your environment's `$LD_LIBRARY_PATH` variable for clues. If none is set, and neither `/lib` nor `/usr/lib` works, you may need to set `LD_LIBRARY_PATH` to `/lib` or `/usr/lib`.

25.5 Other Reading

- *Percona Toolkit* documentation

CHAPTER

TWENTYSIX

KILL IDLE TRANSACTIONS

This feature limits the age of idle *XtraDB* transactions. If a transaction is idle for more seconds than the threshold specified, it will be killed. This prevents users from blocking purge by mistake.

26.1 Version Specific Information

- **`5.7.10-1:`** Feature ported from *Percona Server* 5.6

26.2 System Variables

variable `innodb_kill_idle_transaction`

Scope `GLOBAL`

Config `YES`

Dynamic `YES`

Variable Type `INTEGER`

Default Value 0 (disabled)

Units Seconds

To enable this feature, set this variable to the desired seconds wait until the transaction is killed.

CHAPTER

TWENTYSEVEN

ENFORCING STORAGE ENGINE

Percona Server has implemented variable which can be used for enforcing the use of a specific storage engine.

When this variable is specified and a user tries to create a table using an explicit storage engine that is not the specified enforced engine, he will get either an error if the `NO_ENGINE_SUBSTITUTION` SQL mode is enabled or a warning if `NO_ENGINE_SUBSTITUTION` is disabled and the table will be created anyway using the enforced engine (this is consistent with the default *MySQL* way of creating the default storage engine if other engines aren't available unless `NO_ENGINE_SUBSTITUTION` is set).

In case user tries to enable `enforce_storage_engine` with engine that isn't available, system will not start.

Note: If you're using `enforce_storage_engine`, you must either disable it before doing `mysql_upgrade` or perform `mysql_upgrade` with server started with `--skip-grants-tables`.

27.1 Version Specific Information

- `5.7.10-1` Feature ported from *Percona Server* 5.6

27.2 System Variables

variable `enforce_storage_engine`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type String

Default Value NULL

Note: This variable is not case sensitive.

27.3 Example

Adding following option to *my.cnf* will start the server with InnoDB as enforced storage engine.

```
enforce_storage_engine=InnoDB
```

CHAPTER

TWENTYEIGHT

UTILITY USER

Percona Server has implemented ability to have a *MySQL* user who has system access to do administrative tasks but limited access to user schema. This feature is especially useful to those operating *MySQL* As A Service.

This user has a mixed and special scope of abilities and protection:

- Utility user will not appear in the mysql.user table and can not be modified by any other user, including root.
- Utility user will not appear in `USER_STATISTICS`, `CLIENT_STATISTICS` or `THREAD_STATISTICS` tables.
- Utility user's queries may appear in the general and slow logs.
- Utility user doesn't have the ability create, modify, delete or see any schemas or data not specified (except for information_schema).
- Utility user may modify all visible, non read-only system variables (see *Expanded Program Option Modifiers* functionality).
- Utility user may see, create, modify and delete other system users only if given access to the mysql schema.
- Regular users may be granted proxy rights to the utility user but any attempt to impersonate the utility user will fail. The utility user may not be granted proxy rights on any regular user. For example running: GRANT PROXY ON utility_user TO regular_user; will not fail, but any actual attempt to impersonate as the utility user will fail. Running: *GRANT PROXY ON regular_user TO utility_user;* will fail when utility_user is an exact match or is more specific than than the utility user specified.

When the server starts, it will note in the log output that the utility user exists and the schemas that it has access to.

In order to have the ability for a special type of MySQL user, which will have a very limited and special amount of control over the system and can not be see or modified by any other user including the root user, three new options have been added.

Option `utility_user` specifies the user which the system will create and recognize as the utility user. The host in the utility user specification follows conventions described in the MySQL manual, i.e. it allows wildcards and IP masks. Anonymous user names are not permitted to be used for the utility user name.

This user must not be an exact match to any other user that exists in the mysql.user table. If the server detects that the user specified with this option exactly matches any user within the mysql.user table on start up, the server will report an error and shut down gracefully. If host name wildcards are used and a more specific user specification is identified on start up, the server will report a warning and continue.

> Example: `--utility_user` =frank@% and frank@localhost exists within the mysql.user table.

If a client attempts to create a MySQL user that matches this user specification exactly or if host name wildcards are used for the utility user and the user being created has the same name and a more specific host, the creation attempt will fail with an error.

> Example: `--utility_user` =frank@% and CREATE USER 'frank@localhost';

As a result of these requirements, it is strongly recommended that a very unique user name and reasonably specific host be used and that any script or tools test that they are running within the correct user by executing 'SELECT CURRENT_USER()' and comparing the result against the known utility user.

Option `utility_user_password` specifies the password for the utility user and MUST be specified or the server will shut down gracefully with an error.

Example: `--utility_user_password` ='Passw0rD';

Option `utility_user_schema_access` specifies the name(s) of the schema(s) that the utility user will have access to read write and modify. If a particular schema named here does not exist on start up it will be ignored. If a schema by the name of any of those listed in this option is created after the server is started, the utility user will have full access to it.

Example: `--utility_user_schema_access` =schema1,schema2,schema3 ;

Option `utility_user_privileges` allows a comma-separated list of extra access privileges to grant to the utility user.

Example: `--utility-user-privileges` ="CREATE, DROP, LOCK TABLES"

28.1 Version Specific Information

- **`5.7.10-1`** Feature ported from *Percona Server* 5.6

28.2 System Variables

variable `utility_user`

Command Line Yes

Config File utility_user=<user@host>

Scope Global

Dynamic No

Variable Type String

Default Value NULL

Specifies a MySQL user that will be added to the internal list of users and recognized as the utility user.

variable `utility_user_password`

Command Line Yes

Config File utility_user_password=<password>

Scope Global

Dynamic No

Variable Type String

Default Value NULL

Specifies the password required for the utility user.

variable `utility_user_schema_access`

Command Line Yes

Config File utility_user_schema_access=<schema>,<schema>,<schema>

Scope Global

Dynamic No

Variable Type String

Default Value NULL

Specifies the schemas that the utility user has access to in a comma delimited list.

variable `utility_user_privileges`

Command Line Yes

Config File utility_user_privileges=<privilege1>,<privilege2>,<privilege3>

Scope Global

Dynamic No

Variable Type String

Default Value NULL

This variable can be used to specify a comma-separated list of extra access privileges to grant to the utility user. Supported values for the privileges list are: `SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, SHUTDOWN, PROCESS,FILE, GRANT, REFERENCES, INDEX, ALTER, SHOW DATABASES, SUPER, CREATE TEMPORARY TABLES, LOCK TABLES,EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, CREATE TABLESPACE`

CHAPTER

TWENTYNINE

EXPANDED PROGRAM OPTION MODIFIERS

MySQL has the concept of options modifiers which is a simple way to modify either the way that *MySQL* interprets an option or the way the option behaves. Option modifiers are used by simply prepending the name of the modifier and a dash "-" before the actual configuration option name. For example specifying –maximum-query_cache_size=4M on the mysqld commad line or specifying maximum-query_cache_size=4M in the `my.cnf` will prevent any client from setting the `query_cache_size` value larger than 4MB.

Currently MySQL supports five existing option modifiers:

- disable [disable-<option_name>] disables or ignores option_name.
- enable [enable-<option_name>] enables option_name.
- loose [loose-<option_name>] - mysqld will not exit with an error if it does not recognize option_name, but instead it will issue only a warning.
- maximum [maximum-<option_name>=<value>] indicates that a client can not set the value of option_name greater than the limit specified. If the client does attempt to set the value of option_name greater than the limit, the option_name will simply be set to the defined limit.
- skip [skip-<option_name>] skips or ignores option_name.

In order to offer more control over option visibility, access and range limits, the following new option modifiers have been added

- minimum [minimum-<option_name>=<value>] indicates that clients can not set the value of option_name to less than the limit specified. If the client does attempt to set the value of option_name lesser than the limit, the option_name will simply be set to the defined limit.
- hidden [hidden-<option_name>=<TRUE/FALSE>] indicates that clients can not see or modify the value of option_name.
- readonly [readonly-<option_name>=<TRUE/FALSE>] indicates that clients can see the value of option_name but can not modify the value.

29.1 Combining the options

Some of the option modifiers may be used together in the same option specification, example:

```
--skip-loose-<option_name>
--loose-readonly-<option_name>=<T/F>
--readonly-<option_name>=<T/F>
--hidden-<option_name>=<T/F>
```

29.2 Version Specific Information

- `5.7.10-1` Feature ported from *Percona Server* 5.6

29.3 Examples

Adding the following option to the `my.cnf` will set the minimum limit on `query_cache_size`

```
minimum-query_cache_size = 4M
```

Trying to set up bigger value will work correctly, but if we try to set it up with smaller than the limit, defined minimum limit will be used and warning (1292) will be issued:

Initial `query_cache_size` size:

```
mysql> show variables like 'query_cache_size';
+------------------+---------+
| Variable_name    | Value   |
+------------------+---------+
| query_cache_size | 8388608 |
+------------------+---------+
1 row in set (0.00 sec)
```

Setting up bigger value:

```
mysql> set global query_cache_size=16777216;
Query OK, 0 rows affected (0.00 sec)

mysql> show variables like 'query_cache_size';
+------------------+----------+
| Variable_name    | Value    |
+------------------+----------+
| query_cache_size | 16777216 |
+------------------+----------+
1 row in set (0.00 sec)
```

Setting up smaller value:

```
mysql> set global query_cache_size=1048576;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> show warnings;
+---------+------+-----------------------------------------------------------+
| Level   | Code | Message                                                   |
+---------+------+-----------------------------------------------------------+
| Warning | 1292 | Truncated incorrect query_cache_size value: '1048576' |
+---------+------+-----------------------------------------------------------+
1 row in set (0.00 sec)

mysql> show variables like 'query_cache_size';
+------------------+---------+
| Variable_name    | Value   |
+------------------+---------+
| query_cache_size | 4194304 |
+------------------+---------+
1 row in set (0.00 sec)
```

Adding following option to `my.cnf` will make `query_cache_size` hidden.

```
hidden-query_cache_size=1
```

```
mysql> show variables like 'query_cache%';
+------------------------------+---------+
| Variable_name                | Value   |
+------------------------------+---------+
| query_cache_limit            | 1048576 |
| query_cache_min_res_unit     | 4096    |
| query_cache_strip_comments   | OFF     |
| query_cache_type             | ON      |
| query_cache_wlock_invalidate | OFF     |
+------------------------------+---------+
5 rows in set (0.00 sec)
```

Adding following option to `my.cnf` will make `query_cache_size` read-only

```
readonly-query_cache_size=1
```

Trying to change the variable value will result in error:

```
mysql> show variables like 'query_cache%';
+------------------------------+---------+
| Variable_name                | Value   |
+------------------------------+---------+
| query_cache_limit            | 1048576 |
| query_cache_min_res_unit     | 4096    |
| query_cache_size             | 8388608 |
| query_cache_strip_comments   | OFF     |
| query_cache_type             | ON      |
| query_cache_wlock_invalidate | OFF     |
+------------------------------+---------+
6 rows in set (0.00 sec)

mysql> set global query_cache_size=16777216;
ERROR 1238 (HY000): Variable 'query_cache_size' is a read only variable
```

CHAPTER

THIRTY

XTRADB CHANGED PAGE TRACKING

XtraDB now tracks the pages that have changes written to them according to the redo log. This information is written out in special changed page bitmap files. This information can be used to speed up incremental backups using Percona XtraBackup by removing the need to scan whole data files to find the changed pages. Changed page tracking is done by a new *XtraDB* worker thread that reads and parses log records between checkpoints. The tracking is controlled by a new read-only server variable `innodb_track_changed_pages`.

Bitmap filename format used for changed page tracking is `ib_modified_log_<seq>_<startlsn>.xdb`. The first number is the sequence number of the bitmap log file and the *startlsn* number is the starting LSN number of data tracked in that file. Example of the bitmap log files should look like this:

```
ib_modified_log_1_0.xdb
ib_modified_log_2_1603391.xdb
```

Sequence number can be used to easily check if all the required bitmap files are present. Start LSN number will be used in *XtraBackup* and `INFORMATION_SCHEMA` queries to determine which files have to be opened and read for the required LSN interval data. The bitmap file is rotated on each server restart and whenever the current file size reaches the predefined maximum. This maximum is controlled by a new `innodb_max_bitmap_file_size` variable.

Old bitmap files may be safely removed after a corresponding incremental backup is taken. For that there are server *User statements for handling the XtraDB changed page bitmaps*. Removing the bitmap files from the filesystem directly is safe too, as long as care is taken not to delete data for not-yet-backuped LSN range.

This feature will be used for implementing faster incremental backups that use this information to avoid full data scans in *Percona XtraBackup*.

30.1 User statements for handling the XtraDB changed page bitmaps

New statements have been introduced for handling the changed page bitmap tracking. All of these statements require `SUPER` privilege.

- `FLUSH CHANGED_PAGE_BITMAPS` - this statement can be used for synchronous bitmap write for immediate catch-up with the log checkpoint. This is used by innobackupex to make sure that XtraBackup indeed has all the required data it needs.
- `RESET CHANGED_PAGE_BITMAPS` - this statement will delete all the bitmap log files and restart the bitmap log file sequence.
- `PURGE CHANGED_PAGE_BITMAPS BEFORE <lsn>` - this statement will delete all the change page bitmap files up to the specified log sequence number.

30.2 Additional information in SHOW ENGINE INNODB STATUS

When log tracking is enabled, the following additional fields are displayed in the LOG section of the `SHOW ENGINE INNODB STATUS` output:

- "Log tracked up to:" displays the LSN up to which all the changes have been parsed and stored as a bitmap on disk by the log tracking thread
- "Max tracked LSN age:" displays the maximum limit on how far behind the log tracking thread may be.

30.3 INFORMATION_SCHEMA Tables

This table contains a list of modified pages from the bitmap file data. As these files are generated by the log tracking thread parsing the log whenever the checkpoint is made, it is not real-time data.

table `INFORMATION_SCHEMA.INNODB_CHANGED_PAGES`

Columns

- **space_id** (*INT(11)*) – space id of modified page
- **page_id** (*INT(11)*) – id of modified page
- **start_lsn** (*BIGINT(21)*) – start of the interval
- **end_lsn** (*BIGINT(21)*) – end of the interval

The `start_lsn` and the `end_lsn` columns denote between which two checkpoints this page was changed at least once. They are also equal to checkpoint LSNs.

Number of records in this table can be limited by using the variable `innodb_max_changed_pages`.

30.4 Version Specific Information

- **`5.7.10-1`** Feature ported from *Percona Server* 5.6

30.5 System Variables

variable `innodb_max_changed_pages`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 1000000

Range 1 - 0 (unlimited)

This variable is used to limit the result row count for the queries from `INNODB_CHANGED_PAGES` table.

variable `innodb_track_changed_pages`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Boolean

Default Value 0 - False

Range 0-1

This variable is used to enable/disable *XtraDB changed page tracking* feature.

variable `innodb_max_bitmap_file_size`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 104857600 (100 MB)

Range 4096 (4KB) - 18446744073709551615 (16EB)

This variable is used to control maximum bitmap size after which the file will be rotated.

CHAPTER

THIRTYONE

PAM AUTHENTICATION PLUGIN

Percona PAM Authentication Plugin is a free and Open Source implementation of the *MySQL*'s authentication plugin. This plugin acts as a mediator between the *MySQL* server, the *MySQL* client, and the PAM stack. The server plugin requests authentication from the PAM stack, forwards any requests and messages from the PAM stack over the wire to the client (in cleartext) and reads back any replies for the PAM stack.

> PAM plugin uses dialog as its client side plugin. Dialog plugin can be loaded to any client application that uses `libperconaserverclient/libperconaserverclient` library.

Here are some of the benefits that Percona dialog plugin offers over the default one:

- It correctly recognizes whether PAM wants input to be echoed or not, while the default one always echoes the input on the user's console.
- It can use the password which is passed to *MySQL* client via "-p" parameter.
- Dialog client installation bug has been fixed.
- This plugin works on *MySQL* and *Percona Server*.

Percona offers two versions of this plugin:

- Full PAM plugin called *auth_pam*. This plugin uses *dialog.so*. It fully supports the PAM protocol with arbitrary communication between client and server.
- Oracle-compatible PAM called *auth_pam_compat*. This plugin uses *mysql_clear_password* which is a part of Oracle MySQL client. It also has some limitations, such as, it supports only one password input. You must use `-p` option in order to pass the password to *auth_pam_compat*.

These two versions of plugins are physically different. To choose which one you want used, you must use *IDENTIFIED WITH 'auth_pam'* for auth_pam, and *IDENTIFIED WITH 'auth_pam_compat'* for auth_pam_compat.

31.1 Installation

This plugin requires manual installation because it isn't installed by default.

```
mysql> INSTALL PLUGIN auth_pam SONAME 'auth_pam.so';
```

After the plugin has been installed it should be present in the plugins list. To check if the plugin has been correctly installed and active

```
mysql> SHOW PLUGINS;
...
...
| auth_pam                      | ACTIVE   | AUTHENTICATION     | auth_pam.so | GPL     |
```

31.2 Configuration

In order to use the plugin, authentication method should be configured. Simple setup can be to use the standard UNIX authentication method (`pam_unix`).

Note: To use `pam_unix`, mysql will need to be added to the shadow group in order to have enough privileges to read the /etc/shadow.

A sample */etc/pam.d/mysqld* file:

```
auth       required     pam_unix.so
account    required     pam_unix.so
```

For added information in the system log, you can expand it to be:

```
auth       required     pam_warn.so
auth       required     pam_unix.so audit
account    required     pam_unix.so audit
```

31.3 Creating a user

After the PAM plugin has been configured, users can be created with the PAM plugin as authentication method

```
mysql> CREATE USER 'newuser'@'localhost' IDENTIFIED WITH auth_pam;
```

This will create a user `newuser` that can connect from `localhost` who will be authenticated using the PAM plugin. If the `pam_unix` method is being used user will need to exist on the system.

31.4 Supplementary groups support

Percona Server has implemented PAM plugin support for supplementary groups. Supplementary or secondary groups are extra groups a specific user is member of. For example user `joe` might be a member of groups: `joe` (his primary group) and secondary groups `developers` and `dba`. A complete list of groups and users belonging to them can be checked with `cat /etc/group` command.

This feature enables using secondary groups in the mapping part of the authentication string, like "`mysql, developers=joe, dba=mark`". Previously only primary groups could have been specified there. If user is a member of both `developers` and `dba`, PAM plugin will map it to the `joe` because `developers` matches first.

31.5 Known issues

Default mysql stack size is not enough to handle `pam_ecryptfs` module. Workaround is to increase the *MySQL* stack size by setting the thread-stack variable to at least `512KB` or by increasing the old value by `256KB`.

31.6 Version Specific Information

- `5.7.10-1` Feature ported from *Percona Server* 5.6

CHAPTER

THIRTYTWO

EXPANDED FAST INDEX CREATION

Note: This feature implementation is considered BETA quality.

Percona has implemented several changes related to *MySQL*'s fast index creation feature. This feature expands the `ALTER TABLE` command by adding a new clause that provides online index renaming capability, that is renaming indexes without rebuilding the whole table.

32.1 Enabling Expanded Fast Index Creation

Fast index creation was implemented in *MySQL* as a way to speed up the process of adding or dropping indexes on tables with many rows. However, cases have been found in which fast index creation creates an inconsistency between *MySQL* and *InnoDB* data dictionaries.

This feature implements a session variable that enables extended fast index creation. Besides optimizing DDL directly, `expand_fast_index_creation` may also optimize index access for subsequent DML statements because using it results in much less fragmented indexes.

32.1.1 mysqldump

A new option, `--innodb-optimize-keys`, was implemented in **mysqldump**. It changes the way *InnoDB* tables are dumped, so that secondary and foreign keys are created after loading the data, thus taking advantage of fast index creation. More specifically:

- `KEY`, `UNIQUE KEY`, and `CONSTRAINT` clauses are omitted from `CREATE TABLE` statements corresponding to *InnoDB* tables.
- An additional `ALTER TABLE` is issued after dumping the data, in order to create the previously omitted keys.

32.1.2 `ALTER TABLE`

When `ALTER TABLE` requires a table copy, secondary keys are now dropped and recreated later, after copying the data. The following restrictions apply:

- Only non-unique keys can be involved in this optimization.
- If the table contains foreign keys, or a foreign key is being added as a part of the current `ALTER TABLE` statement, the optimization is disabled for all keys.

32.1.3 OPTIMIZE TABLE

Internally, OPTIMIZE TABLE is mapped to ALTER TABLE ... ENGINE=innodb for *InnoDB* tables. As a consequence, it now also benefits from fast index creation, with the same restrictions as for ALTER TABLE.

32.1.4 Caveats

InnoDB fast index creation uses temporary files in tmpdir for all indexes being created. So make sure you have enough tmpdir space when using expand_fast_index_creation. It is a session variable, so you can temporarily switch it off if you are short on tmpdir space and/or don't want this optimization to be used for a specific table.

There's also a number of cases when this optimization is not applicable:

- UNIQUE indexes in ALTER TABLE are ignored to enforce uniqueness where necessary when copying the data to a temporary table;
- ALTER TABLE and OPTIMIZE TABLE always process tables containing foreign keys as if expand_fast_index_creation is OFF to avoid dropping keys that are part of a FOREIGN KEY constraint;
- **mysqldump –innodb-optimize-keys** ignores foreign keys because *InnoDB* requires a full table rebuild on foreign key changes. So adding them back with a separate ALTER TABLE after restoring the data from a dump would actually make the restore slower;
- **mysqldump –innodb-optimize-keys** ignores indexes on AUTO_INCREMENT columns, because they must be indexed, so it is impossible to temporarily drop the corresponding index;
- **mysqldump –innodb-optimize-keys** ignores the first UNIQUE index on non-nullable columns when the table has no PRIMARY KEY defined, because in this case *InnoDB* picks such an index as the clustered one.

32.2 Version Specific Information

- 5.7.10-1 Feature ported from *Percona Server* 5.6

32.3 System Variables

variable expand_fast_index_creation

Command Line Yes

Config File No

Scope Local/Global

Dynamic Yes

Variable Type Boolean

Default Value OFF

Range ON/OFF

32.4 Other Reading

- Improved InnoDB fast index creation
- Thinking about running OPTIMIZE on your InnoDB Table? Stop!

CHAPTER

THIRTYTHREE

BACKUP LOCKS

Percona Server has implemented this feature to be a lightweight alternative to `FLUSH TABLES WITH READ LOCK` for both physical and logical backups. Three new statements are now available: `LOCK TABLES FOR BACKUP`, `LOCK BINLOG FOR BACKUP` and `UNLOCK BINLOG`.

33.1 LOCK TABLES FOR BACKUP

`LOCK TABLES FOR BACKUP` uses a new MDL lock type to block updates to non-transactional tables and DDL statements for all tables. More specifically, if there's an active `LOCK TABLES FOR BACKUP` lock, all DDL statements and all updates to MyISAM, CSV, MEMORY and ARCHIVE tables will be blocked in the `Waiting for backup lock` status as visible in `PERFORMANCE_SCHEMA` or `PROCESSLIST`. `SELECT` queries for all tables and `INSERT/REPLACE/UPDATE/DELETE` against *InnoDB*, Blackhole and Federated tables are not affected by `LOCK TABLES FOR BACKUP`. Blackhole tables obviously have no relevance for backups, and Federated tables are ignored by both logical and physical backup tools.

Unlike `FLUSH TABLES WITH READ LOCK`, `LOCK TABLES FOR BACKUP` does not flush tables, i.e. storage engines are not forced to close tables and tables are not expelled from the table cache. As a result, `LOCK TABLES FOR BACKUP` only waits for conflicting statements to complete (i.e. DDL and updates to non-transactional tables). It never waits for SELECTs, or UPDATEs to *InnoDB* tables to complete, for example.

If an "unsafe" statement is executed in the same connection that is holding a `LOCK TABLES FOR BACKUP` lock, it fails with the following error:

```
ERROR 1880 (HY000): Can't execute the query because you have a conflicting backup lock
```

`UNLOCK TABLES` releases the lock acquired by `LOCK TABLES FOR BACKUP`.

33.2 LOCK BINLOG FOR BACKUP

`LOCK BINLOG FOR BACKUP` uses another new MDL lock type to block all operations that might change either binary log position or `Exec_Master_Log_Pos` or `Exec_Gtid_Set` (i.e. master binary log coordinates corresponding to the current SQL thread state on a replication slave) as reported by `SHOW MASTER/SLAVE STATUS`. More specifically, a commit will only be blocked if the binary log is enabled (both globally, and for connection with sql_log_bin), or if commit is performed by a slave thread and would advance `Exec_Master_Log_Pos` or `Executed_Gtid_Set`. Connections that are currently blocked on the global binlog lock can be identified by the `Waiting for binlog lock` status in `PROCESSLIST`. `LOCK TABLES FOR BACKUP` flushes the current binary log coordinates to *InnoDB*. Thus, under active `LOCK TABLES FOR BACKUP`, the binary log coordinates in *InnoDB* are consistent with its redo log and any non-transactional updates (as the latter are blocked by `LOCK TABLES FOR BACKUP`). It is planned that this change will enable *Percona XtraBackup* to avoid issuing the more invasive `LOCK BINLOG FOR BACKUP` command under some circumstances.

33.3 UNLOCK BINLOG

`UNLOCK BINLOG` releases the `LOCK BINLOG FOR BACKUP` lock, if acquired by the current connection. The intended use case for *Percona XtraBackup* is:

```
LOCK TABLES FOR BACKUP
... copy .frm, MyISAM, CSV, etc. ...
LOCK BINLOG FOR BACKUP
UNLOCK TABLES
... get binlog coordinates ...
... wait for redo log copying to finish ...
UNLOCK BINLOG
```

33.4 Privileges

Both `LOCK TABLES FOR BACKUP` and `LOCK BINLOG FOR BACKUP` require the `RELOAD` privilege. The reason for that is to have the same requirements as `FLUSH TABLES WITH READ LOCK`.

33.5 Interaction with other global locks

Both `LOCK TABLES FOR BACKUP` and `LOCK BINLOG FOR BACKUP` have no effect if the current connection already owns a `FLUSH TABLES WITH READ LOCK` lock, as it's a more restrictive lock. If `FLUSH TABLES WITH READ LOCK` is executed in a connection that has acquired `LOCK TABLES FOR BACKUP` or `LOCK BINLOG FOR BACKUP`, `FLUSH TABLES WITH READ LOCK` fails with an error.

If the server is operating in the read-only mode (i.e. `read_only` set to `1`), statements that are unsafe for backups will be either blocked or fail with an error, depending on whether they are executed in the same connection that owns `LOCK TABLES FOR BACKUP` lock, or other connections.

33.6 MyISAM index and data buffering

MyISAM key buffering is normally write-through, i.e. by the time each update to a *MyISAM* table is completed, all index updates are written to disk. The only exception is delayed key writing feature which is disabled by default.

When the global system variable `delay_key_write` is set to `ALL`, key buffers for all *MyISAM* tables are not flushed between updates, so a physical backup of those tables may result in broken *MyISAM* indexes. To prevent this, `LOCK TABLES FOR BACKUP` will fail with an error if `delay_key_write` is set to `ALL`. An attempt to set `delay_key_write` to `ALL` when there's an active backup lock will also fail with an error.

Another option to involve delayed key writing is to create *MyISAM* tables with the DELAY_KEY_WRITE option and set the `delay_key_write` variable to `ON` (which is the default). In this case, `LOCK TABLES FOR BACKUP` will not be able to prevent stale index files from appearing in the backup. Users are encouraged to set `delay_key_writes` to `OFF` in the configuration file, `my.cnf`, or repair *MyISAM* indexes after restoring from a physical backup created with backup locks.

MyISAM may also cache data for bulk inserts, e.g. when executing multi-row INSERTs or `LOAD DATA` statements. Those caches, however, are flushed between statements, so have no effect on physical backups as long as all statements updating *MyISAM* tables are blocked.

33.7 mysqldump

`mysqldump` has also been extended with a new option, `lock-for-backup` (disabled by default). When used together with the `--single-transaction` option, the option makes `mysqldump` issue `LOCK TABLES FOR BACKUP` before starting the dump operation to prevent unsafe statements that would normally result in an inconsistent backup.

When used without the `single-transaction` option, `lock-for-backup` is automatically converted to `lock-all-tables`.

Option `lock-for-backup` is mutually exclusive with `lock-all-tables`, i.e. specifying both on the command line will lead to an error.

If the backup locks feature is not supported by the target server, but `lock-for-backup` is specified on the command line, `mysqldump` aborts with an error.

33.7.1 Version Specific Information

- **`5.7.10-1`** Feature ported from *Percona Server* 5.6

33.7.2 System Variables

variable `have_backup_locks`

Command Line Yes

Config File No

Scope Global

Dynamic No

Variable Type Boolean

Default Value YES

This is a server variable implemented to help other utilities decide what locking strategy can be implemented for a server. When available, the backup locks feature is supported by the server and the variable value is always `YES`.

variable `have_backup_safe_binlog_info`

Command Line Yes

Config File No

Scope Global

Dynamic No

Variable Type Boolean

Default Value YES

This is a server variable implemented to help other utilities decide if `LOCK BINLOG FOR BACKUP` can be avoided in some cases. When the necessary server-side functionality is available, this server system variable exists and its value is always `YES`.

33.7.3 Status Variables

variable `Com_lock_tables_for_backup`

Variable Type Numeric

Scope Global/Session

variable `Com_lock_binlog_for_backup`

Variable Type Numeric

Scope Global/Session

variable `Com_unlock_binlog`

Variable Type Numeric

Scope Global/Session

These status variables indicate the number of times the corresponding statements have been executed.

33.7.4 Client Command Line Parameter

option `lock-for-backup`

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value Off

When used together with the `--single-transaction` option, the option makes `mysqldump` issue `LOCK TABLES FOR BACKUP` before starting the dump operation to prevent unsafe statements that would normally result in an inconsistent backup.

CHAPTER

THIRTYFOUR

AUDIT LOG PLUGIN

Percona Audit Log Plugin provides monitoring and logging of connection and query activity that were performed on specific server. Information about the activity will be stored in the XML log file where each event will have its `NAME` field, its own unique `RECORD_ID` field and a `TIMESTAMP` field. This implementation is alternative to the MySQL Enterprise Audit Log Plugin

Audit Log plugin produces the log of following events:

- **Audit** - Audit event indicates that audit logging started or finished. `NAME` field will be `Audit` when logging started and `NoAudit` when logging finished. Audit record also includes server version and command-line arguments.

Example of the Audit event:

```
<AUDIT_RECORD
 "NAME"="Audit"
 "RECORD"="1_2014-04-29T09:29:40"
 "TIMESTAMP"="2014-04-29T09:29:40 UTC"
 "MYSQL_VERSION"="5.6.17-65.0-655.trusty"
 "STARTUP_OPTIONS"="--basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib/mysql/plugin --user
 "OS_VERSION"="x86_64-debian-linux-gnu",
 />
```

- **Connect/Disconnect** - Connect record event will have `NAME` field `Connect` when user logged in or login failed, or `Quit` when connection is closed. Additional fields for this event are `CONNECTION_ID`, `STATUS`, `USER`, `PRIV_USER`, `OS_LOGIN`, `PROXY_USER`, `HOST`, and `IP`. `STATUS` will be `0` for successful logins and non-zero for failed logins.

Example of the Disconnect event:

```
<AUDIT_RECORD
 "NAME"="Quit"
 "RECORD"="24_2014-04-29T09:29:40"
 "TIMESTAMP"="2014-04-29T10:20:13 UTC"
 "CONNECTION_ID"="49"
 "STATUS"="0"
 "USER"=""
 "PRIV_USER"=""
 "OS_LOGIN"=""
 "PROXY_USER"=""
 "HOST"=""
 "IP"=""
 "DB"=""
 />
```

- **Query** - Additional fields for this event are: `COMMAND_CLASS` (values come from the `com_status_vars` array in the `sql/mysqld.cc` ` file in a MySQL source distribution. Examples are `select`, `alter_table`,

`create_table`, etc.), `CONNECTION_ID`, `STATUS` (indicates error when non-zero), `SQLTEXT` (text of SQL-statement), `USER`, `HOST`, `OS_USER`, `IP`. Possible values for the `NAME` name field for this event are `Query`, `Prepare`, `Execute`, `Change user`, etc.

Example of the Query event:

```
<AUDIT_RECORD
 "NAME"="Query"
 "RECORD"="23_2014-04-29T09:29:40"
 "TIMESTAMP"="2014-04-29T10:20:10 UTC"
 "COMMAND_CLASS"="select"
 "CONNECTION_ID"="49"
 "STATUS"="0"
 "SQLTEXT"="SELECT * from mysql.user"
 "USER"="root[root] @ localhost []"
 "HOST"="localhost"
 "OS_USER"=""
 "IP"=""
 />
```

34.1 Installation

Audit Log plugin is shipped with *Percona Server*, but it is not installed by default. To enable the plugin you must run the following command:

```
INSTALL PLUGIN audit_log SONAME 'audit_log.so';
```

You can check if the plugin is loaded correctly by running:

```
SHOW PLUGINS;
```

Audit log should be listed in the output:

```
+-------------------------------+----------+--------------------+--------------+---------+
| Name                          | Status   | Type               | Library      | License |
+-------------------------------+----------+--------------------+--------------+---------+
...
| audit_log                     | ACTIVE   | AUDIT              | audit_log.so | GPL     |
+-------------------------------+----------+--------------------+--------------+---------+
```

34.2 Log Format

The audit log plugin supports four log formats: `OLD`, `NEW`, `JSON`, and `CSV`. `OLD` and `NEW` formats are based on XML, where the former outputs log record properties as XML attributes and the latter as XML tags. Information logged is the same in all four formats. The log format choice is controlled by `audit_log_format` variable.

Example of the `OLD` format:

```
<AUDIT_RECORD
 "NAME"="Query"
 "RECORD"="2_2014-04-28T09:29:40"
 "TIMESTAMP"="2014-04-28T09:29:40 UTC"
 "COMMAND_CLASS"="install_plugin"
 "CONNECTION_ID"="47"
 "STATUS"="0"
 "SQLTEXT"="INSTALL PLUGIN audit_log SONAME 'audit_log.so'"
```

```
 "USER"="root[root] @ localhost []"
 "HOST"="localhost"
 "OS_USER"=""
 "IP"=""
/>
```

Example of the `NEW` format:

```
<AUDIT_RECORD>
 <NAME>Quit</NAME>
 <RECORD>10902_2014-04-28T11:02:54</RECORD>
 <TIMESTAMP>2014-04-28T11:02:59 UTC</TIMESTAMP>
 <CONNECTION_ID>36</CONNECTION_ID>
 <STATUS>0</STATUS>
 <USER></USER>
 <PRIV_USER></PRIV_USER>
 <OS_LOGIN></OS_LOGIN>
 <PROXY_USER></PROXY_USER>
 <HOST></HOST>
 <IP></IP>
 <DB></DB>
</AUDIT_RECORD>
```

Example of the `JSON` format:

```
{"audit_record":{"name":"Query","record":"4707_2014-08-27T10:43:52","timestamp":"2014-08-27T10:44:19
```

Example of the `CSV` format:

```
"Query","49284_2014-08-27T10:47:11","2014-08-27T10:47:23 UTC","show_databases","37",0,"show database
```

34.3 Streaming the audit log to syslog

To stream the audit log to syslog you'll need to set `audit_log_handler` variable to `SYSLOG`. To control the syslog file handler, the following variables can be used: `audit_log_syslog_ident`, `audit_log_syslog_facility`, and `audit_log_syslog_priority` These variables have the same meaning as appropriate parameters described in the syslog(3) manual.

Note: Variables: `audit_log_strategy`, `audit_log_buffer_size`, `audit_log_rotate_on_size`, `audit_log_rotations` have effect only with `FILE` handler.

34.4 System Variables

variable `audit_log_strategy`

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value ASYNCHRONOUS

Allowed values `ASYNCHRONOUS`, `PERFORMANCE`, `SEMISYNCHRONOUS`, `SYNCHRONOUS`

This variable is used to specify the audit log strategy, possible values are:

- ASYNCHRONOUS - (default) log using memory buffer, do not drop messages if buffer is full
- PERFORMANCE - log using memory buffer, drop messages if buffer is full
- SEMISYNCHRONOUS - log directly to file, do not flush and sync every event
- SYNCHRONOUS - log directly to file, flush and sync every event

This variable has effect only when audit_log_handler is set to FILE.

variable audit_log_file

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value audit.log

This variable is used to specify the filename that's going to store the audit log. It can contain the path relative to the datadir or absolute path.

variable audit_log_flush

Command Line Yes

Scope Global

Dynamic Yes

Variable Type String

Default Value OFF

When this variable is set to ON log file will be closed and reopened. This can be used for manual log rotation.

variable audit_log_buffer_size

Command Line Yes

Scope Global

Dynamic No

Variable Type Numeric

Default Value 4096

This variable can be used to specify the size of memory buffer used for logging, used when audit_log_strategy variable is set to ASYNCHRONOUS or PERFORMANCE values. This variable has effect only when audit_log_handler is set to FILE.

variable audit_log_format

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value OLD

Allowed values OLD, NEW, CSV, JSON

This variable is used to specify the audit log format. The audit log plugin supports four log formats: `OLD`, `NEW`, `JSON`, and `CSV`. `OLD` and `NEW` formats are based on XML, where the former outputs log record properties as XML attributes and the latter as XML tags. Information logged is the same in all four formats.

variable `audit_log_policy`

Command Line Yes

Scope Global

Dynamic Yes

Variable Type String

Default Value ALL

Allowed values `ALL`, `LOGINS`, `QUERIES`, `NONE`

This variable is used to specify which events should be logged. Possible values are:

- `ALL` - all events will be logged
- `LOGINS` - only logins will be logged
- `QUERIES` - only queries will be logged
- `NONE` - no events will be logged

variable `audit_log_rotate_on_size`

Command Line Yes

Scope Global

Dynamic No

Variable Type Numeric

Default Value 0 (don't rotate the log file)

This variable is used to specify the maximum audit log file size. Upon reaching this size the log will be rotated. The rotated log files will be present in the same same directory as the current log file. A sequence number will be appended to the log file name upon rotation. This variable has effect only when `audit_log_handler` is set to `FILE`.

variable `audit_log_rotations`

Command Line Yes

Scope Global

Dynamic No

Variable Type Numeric

Default Value 0

This variable is used to specify how many log files should be kept when `audit_log_rotate_on_size` variable is set to non-zero value. This variable has effect only when `audit_log_handler` is set to `FILE`.

variable `audit_log_handler`

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value FILE

Allowed values FILE, SYSLOG

This variable is used to configure where the audit log will be written. If it is set to FILE, the log will be written into a file specified by audit_log_file variable. If it is set to SYSLOG, the audit log will be written to syslog.

variable audit_log_syslog_ident

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value percona-audit

This variable is used to specify the ident value for syslog. This variable has the same meaning as the appropriate parameter described in the syslog(3) manual.

variable audit_log_syslog_facility

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value LOG_USER

This variable is used to specify the facility value for syslog. This variable has the same meaning as the appropriate parameter described in the syslog(3) manual.

variable audit_log_syslog_priority

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value LOG_INFO

This variable is used to specify the priority value for syslog. This variable has the same meaning as the appropriate parameter described in the syslog(3) manual.

34.5 Version Specific Information

- 5.7.10-1 Feature ported from *Percona Server* 5.6

CHAPTER

THIRTYFIVE

START TRANSACTION WITH CONSISTENT SNAPSHOT

Percona Server has ported *MariaDB* enhancement for `START TRANSACTION WITH CONSISTENT SNAPSHOTS` feature to *MySQL* 5.6 group commit implementation. This enhancement makes binary log positions consistent with *InnoDB* transaction snapshots.

This feature is quite useful to obtain logical backups with correct positions without running a `FLUSH TABLES WITH READ LOCK`. Binary log position can be obtained by two newly implemented status variables: `Binlog_snapshot_file` and `Binlog_snapshot_position`. After starting a transaction using the `START TRANSACTION WITH CONSISTENT SNAPSHOT`, these two variables will provide you with the binlog position corresponding to the state of the database of the consistent snapshot so taken, irrespectively of which other transactions have been committed since the snapshot was taken.

35.1 Snapshot Cloning

The *Percona Server* implementation extends the `START TRANSACTION WITH CONSISTENT SNAPSHOT` syntax with the optional `FROM SESSION` clause:

```
START TRANSACTION WITH CONSISTENT SNAPSHOT FROM SESSION <session_id>;
```

When specified, all participating storage engines and binary log instead of creating a new snapshot of data (or binary log coordinates), create a copy of the snapshot which has been created by an active transaction in the specified session. `session_id` is the session identifier reported in the `Id` column of `SHOW PROCESSLIST`.

Currently snapshot cloning is only supported by *XtraDB* and the binary log. As with the regular `START TRANSACTION WITH CONSISTENT SNAPSHOT`, snapshot clones can only be created with the `REPEATABLE READ` isolation level.

For *XtraDB*, a transaction with a cloned snapshot will only see data visible or changed by the donor transaction. That is, the cloned transaction will see no changes committed by transactions that started after the donor transaction, not even changes made by itself. Note that in case of chained cloning the donor transaction is the first one in the chain. For example, if transaction A is cloned into transaction B, which is in turn cloned into transaction C, the latter will have read view from transaction A (i.e. the donor transaction). Therefore, it will see changes made by transaction A, but not by transaction B.

35.2 mysqldump

`mysqldump` has been updated to use new status variables automatically when they are supported by the server and both `--single-transaction` and `--master-data` are specified on the command line. Along with the `mysqldump` improvements introduced in *Backup Locks* there is now a way to generate `mysqldump` backups that are guaranteed to be consistent without using `FLUSH TABLES WITH READ LOCK` even if `--master-data` is requested.

35.3 System Variables

variable `have_snapshot_cloning`

Command Line Yes

Config File No

Scope Global

Dynamic No

Variable Type Boolean

This server variable is implemented to help other utilities detect if the server supports the `FROM SESSION` extension. When available, the snapshot cloning feature and the syntax extension to `START TRANSACTION WITH CONSISTENT SNAPSHOT` are supported by the server, and the variable value is always `YES`.

35.4 Status Variables

variable `Binlog_snapshot_file`

Variable Type String

Scope Global

variable `Binlog_snapshot_position`

Variable Type Numeric

Scope Global

These status variables are only available when the binary log is enabled globally.

35.5 Other Reading

- MariaDB Enhancements for START TRANSACTION WITH CONSISTENT SNAPSHOT

CHAPTER

THIRTYSIX

EXTENDED SHOW GRANTS

In Oracle *MySQL* `SHOW GRANTS` displays only the privileges granted explicitly to the named account. Other privileges might be available to the account, but they are not displayed. For example, if an anonymous account exists, the named account might be able to use its privileges, but `SHOW GRANTS` will not display them. In *Percona Server* `SHOW GRANTS` command was extended to display all the effectively available privileges to the account.

36.1 Example

If we create the following users:

```
mysql> CREATE USER grantee@localhost IDENTIFIED BY 'grantee1';
Query OK, 0 rows affected (0.50 sec)

mysql> CREATE USER grantee IDENTIFIED BY 'grantee2';
Query OK, 0 rows affected (0.09 sec)

mysql> CREATE DATABASE db2;
Query OK, 1 row affected (0.20 sec)

mysql> GRANT ALL PRIVILEGES ON db2.* TO grantee WITH GRANT OPTION;
Query OK, 0 rows affected (0.12 sec)
```

- `SHOW GRANTS` output before the change:

```
mysql> SHOW GRANTS;
+-----------------------------------------------------------------------------------------------------------
| Grants for grantee@localhost
+-----------------------------------------------------------------------------------------------------------
| GRANT USAGE ON *.* TO 'grantee'@'localhost' IDENTIFIED BY PASSWORD '*9823FF338D44DAF02422CF24DD1F8
+-----------------------------------------------------------------------------------------------------------
1 row in set (0.04 sec)
```

Although the grant for the `db2` database isn't shown, `grantee` user has enough privileges to create the table in that database:

```
user@trusty:~$ mysql -ugrantee -pgrantee1 -h localhost

mysql> CREATE TABLE db2.t1(a int);
Query OK, 0 rows affected (1.21 sec)
```

- `SHOW GRANTS` output after the change shows all the privileges for the `grantee` user:

```
mysql> SHOW GRANTS;
+-----------------------------------------------------------------------------------------------------------
```

```
| Grants for grantee@localhost
+-----------------------------------------------------------------------------------------------------------
| GRANT USAGE ON *.* TO 'grantee'@'localhost' IDENTIFIED BY PASSWORD '*9823FF338D44DAF02422CF24DD1F87
| GRANT ALL PRIVILEGES ON `db2`.* TO 'grantee'@'%' WITH GRANT OPTION
+-----------------------------------------------------------------------------------------------------------
2 rows in set (0.00 sec)
```

36.1.1 Version-Specific Information

- `5.7.10-1`: Feature ported from *Percona Server* 5.6

36.1.2 Other reading

- #53645 - `SHOW GRANTS` not displaying all the applicable grants

Part VIII

Diagnostics Improvements

CHAPTER

THIRTYSEVEN

USER STATISTICS

This feature adds several `INFORMATION_SCHEMA` tables, several commands, and the userstat variable. The tables and commands can be used to understand the server activity better and identify the source of the load.

The functionality is disabled by default, and must be enabled by setting `userstat` to `ON`. It works by keeping several hash tables in memory. To avoid contention over global mutexes, each connection has its own local statistics, which are occasionally merged into the global statistics, and the local statistics are then reset to 0.

37.1 Version Specific Information

- **`5.7.10-1`:** Feature ported from *Percona Server* 5.6.

37.2 Other Information

- **Author/Origin:** *Google*; *Percona* added the `INFORMATION_SCHEMA` tables and the `userstat` variable.

37.3 System Variables

variable `userstat`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type BOOLEAN

Default Value OFF

Range ON/OFF

Enables or disables collection of statistics. The default is `OFF`, meaning no statistics are gathered. This is to ensure that the statistics collection doesn't cause any extra load on the server unless desired.

variable `thread_statistics`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type BOOLEAN

Default Value OFF

Range ON/OFF

Enables or disables collection of thread statistics. The default is `OFF`, meaning no thread statistics are gathered. This is to ensure that the statistics collection doesn't cause any extra load on the server unless desired. Variable `userstat` needs to be enabled as well in order for thread statistics to be collected.

37.4 INFORMATION_SCHEMA Tables

table `INFORMATION_SCHEMA.`**`CLIENT_STATISTICS`**

Columns

- **CLIENT** – The IP address or hostname from which the connection originated.
- **TOTAL_CONNECTIONS** – The number of connections created for this client.
- **CONCURRENT_CONNECTIONS** – The number of concurrent connections for this client.
- **CONNECTED_TIME** – The cumulative number of seconds elapsed while there were connections from this client.
- **BUSY_TIME** – The cumulative number of seconds there was activity on connections from this client.
- **CPU_TIME** – The cumulative CPU time elapsed, in seconds, while servicing this client''s connections.
- **BYTES_RECEIVED** – The number of bytes received from this client's connections.
- **BYTES_SENT** – The number of bytes sent to this client's connections.
- **BINLOG_BYTES_WRITTEN** – The number of bytes written to the binary log from this client's connections.
- **ROWS_FETCHED** – The number of rows fetched by this client's connections.
- **ROWS_UPDATED** – The number of rows updated by this client's connections.
- **TABLE_ROWS_READ** – The number of rows read from tables by this client's connections. (It may be different from `ROWS_FETCHED`.)
- **SELECT_COMMANDS** – The number of `SELECT` commands executed from this client's connections.
- **UPDATE_COMMANDS** – The number of `UPDATE` commands executed from this client's connections.
- **OTHER_COMMANDS** – The number of other commands executed from this client's connections.
- **COMMIT_TRANSACTIONS** – The number of `COMMIT` commands issued by this client's connections.
- **ROLLBACK_TRANSACTIONS** – The number of `ROLLBACK` commands issued by this client's connections.

- **DENIED_CONNECTIONS** – The number of connections denied to this client.
- **LOST_CONNECTIONS** – The number of this client's connections that were terminated uncleanly.
- **ACCESS_DENIED** – The number of times this client's connections issued commands that were denied.
- **EMPTY_QUERIES** – The number of times this client's connections sent empty queries to the server.

This table holds statistics about client connections. The Percona version of the feature restricts this table's visibility to users who have the `SUPER` or `PROCESS` privilege.

Example:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.CLIENT_STATISTICS\G
*************************** 1. row ***************************
               CLIENT: 10.1.12.30
     TOTAL_CONNECTIONS: 20
CONCURRENT_CONNECTIONS: 0
        CONNECTED_TIME: 0
             BUSY_TIME: 93
              CPU_TIME: 48
        BYTES_RECEIVED: 5031
            BYTES_SENT: 276926
  BINLOG_BYTES_WRITTEN: 217
          ROWS_FETCHED: 81
          ROWS_UPDATED: 0
       TABLE_ROWS_READ: 52836023
       SELECT_COMMANDS: 26
       UPDATE_COMMANDS: 1
        OTHER_COMMANDS: 145
   COMMIT_TRANSACTIONS: 1
 ROLLBACK_TRANSACTIONS: 0
    DENIED_CONNECTIONS: 0
      LOST_CONNECTIONS: 0
         ACCESS_DENIED: 0
         EMPTY_QUERIES: 0
```

table `INFORMATION_SCHEMA.`**`INDEX_STATISTICS`**

Columns

- **TABLE_SCHEMA** – The schema (database) name.
- **TABLE_NAME** – The table name.
- **INDEX_NAME** – The index name (as visible in `SHOW CREATE TABLE`).
- **ROWS_READ** – The number of rows read from this index.

This table shows statistics on index usage. An older version of the feature contained a single column that had the `TABLE_SCHEMA`, `TABLE_NAME` and `INDEX_NAME` columns concatenated together. The *Percona* version of the feature separates these into three columns. Users can see entries only for tables to which they have `SELECT` access.

This table makes it possible to do many things that were difficult or impossible previously. For example, you can use it to find unused indexes and generate DROP commands to remove them.

Example:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INDEX_STATISTICS
   WHERE TABLE_NAME='tables_priv';
```

```
+--------------+----------------------+--------------------+-----------+
| TABLE_SCHEMA | TABLE_NAME           | INDEX_NAME         | ROWS_READ |
+--------------+----------------------+--------------------+-----------+
| mysql        | tables_priv          | PRIMARY            |         2 |
+--------------+----------------------+--------------------+-----------+
```

Note: Current implementation of index statistics doesn't support partitioned tables.

table INFORMATION_SCHEMA.**TABLE_STATISTICS**

Columns

- **TABLE_SCHEMA** – The schema (database) name.
- **TABLE_NAME** – The table name.
- **ROWS_READ** – The number of rows read from the table.
- **ROWS_CHANGED** – The number of rows changed in the table.
- **ROWS_CHANGED_X_INDEXES** – The number of rows changed in the table, multiplied by the number of indexes changed.

This table is similar in function to the INDEX_STATISTICS table.

Example:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.TABLE_STATISTICS
   WHERE TABLE_NAME=``tables_priv``;
+--------------+------------------------------+-----------+--------------+-------------------------+
| TABLE_SCHEMA | TABLE_NAME                   | ROWS_READ | ROWS_CHANGED | ROWS_CHANGED_X_INDEXES |
+--------------+------------------------------+-----------+--------------+-------------------------+
| mysql        | tables_priv                  |         2 |            0 |                       0 |
+--------------+------------------------------+-----------+--------------+-------------------------+
```

Note: Current implementation of table statistics doesn't support partitioned tables.

table INFORMATION_SCHEMA.**THREAD_STATISTICS**

Columns

- **THREAD_ID** – int(21)
- **TOTAL_CONNECTIONS** – int(21)
- **CONCURRENT_CONNECTIONS** – int(21)
- **CONNECTED_TIME** – int(21)
- **BUSY_TIME** – int(21)
- **CPU_TIME** – int(21)
- **BYTES_RECEIVED** – int(21)
- **BYTES_SENT** – int(21)
- **BINLOG_BYTES_WRITTEN** – int(21)
- **ROWS_FETCHED** – int(21)
- **ROWS_UPDATED** – int(21)
- **TABLE_ROWS_READ** – int(21)

- **SELECT_COMMANDS** – int(21)
- **UPDATE_COMMANDS** – int(21)
- **OTHER_COMMANDS** – int(21)
- **COMMIT_TRANSACTIONS** – int(21)
- **ROLLBACK_TRANSACTIONS** – int(21)
- **DENIED_CONNECTIONS** – int(21)
- **LOST_CONNECTIONS** – int(21)
- **ACCESS_DENIED** – int(21)
- **EMPTY_QUERIES** – int(21)

table `INFORMATION_SCHEMA.USER_STATISTICS`

Columns

- **USER** – The username. The value `#mysql_system_user#` appears when there is no username (such as for the slave SQL thread).
- **TOTAL_CONNECTIONS** – The number of connections created for this user.
- **CONCURRENT_CONNECTIONS** – The number of concurrent connections for this user.
- **CONNECTED_TIME** – The cumulative number of seconds elapsed while there were connections from this user.
- **BUSY_TIME** – The cumulative number of seconds there was activity on connections from this user.
- **CPU_TIME** – The cumulative CPU time elapsed, in seconds, while servicing this user's connections.
- **BYTES_RECEIVED** – The number of bytes received from this user's connections.
- **BYTES_SENT** – The number of bytes sent to this user's connections.
- **BINLOG_BYTES_WRITTEN** – The number of bytes written to the binary log from this user's connections.
- **ROWS_FETCHED** – The number of rows fetched by this user's connections.
- **ROWS_UPDATED** – The number of rows updated by this user's connections.
- **TABLE_ROWS_READ** – The number of rows read from tables by this user's connections. (It may be different from `ROWS_FETCHED`.)
- **SELECT_COMMANDS** – The number of `SELECT` commands executed from this user's connections.
- **UPDATE_COMMANDS** – The number of `UPDATE` commands executed from this user's connections.
- **OTHER_COMMANDS** – The number of other commands executed from this user's connections.
- **COMMIT_TRANSACTIONS** – The number of `COMMIT` commands issued by this user's connections.
- **ROLLBACK_TRANSACTIONS** – The number of `ROLLBACK` commands issued by this user's connections.
- **DENIED_CONNECTIONS** – The number of connections denied to this user.

- **LOST_CONNECTIONS** – The number of this user's connections that were terminated uncleanly.
- **ACCESS_DENIED** – The number of times this user's connections issued commands that were denied.
- **EMPTY_QUERIES** – The number of times this user's connections sent empty queries to the server.

This table contains information about user activity. The *Percona* version of the patch restricts this table's visibility to users who have the `SUPER` or `PROCESS` privilege.

The table gives answers to questions such as which users cause the most load, and whether any users are being abusive. It also lets you measure how close to capacity the server may be. For example, you can use it to find out whether replication is likely to start falling behind.

Example:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.USER_STATISTICS\G
*************************** 1. row ***************************
                  USER: root
     TOTAL_CONNECTIONS: 5592
CONCURRENT_CONNECTIONS: 0
        CONNECTED_TIME: 6844
             BUSY_TIME: 179
              CPU_TIME: 72
        BYTES_RECEIVED: 603344
            BYTES_SENT: 15663832
  BINLOG_BYTES_WRITTEN: 217
          ROWS_FETCHED: 9793
          ROWS_UPDATED: 0
       TABLE_ROWS_READ: 52836023
       SELECT_COMMANDS: 9701
       UPDATE_COMMANDS: 1
        OTHER_COMMANDS: 2614
   COMMIT_TRANSACTIONS: 1
 ROLLBACK_TRANSACTIONS: 0
    DENIED_CONNECTIONS: 0
      LOST_CONNECTIONS: 0
         ACCESS_DENIED: 0
         EMPTY_QUERIES: 0
```

37.5 Commands Provided

- `FLUSH CLIENT_STATISTICS`
- `FLUSH INDEX_STATISTICS`
- `FLUSH TABLE_STATISTICS`
- `FLUSH THREAD_STATISTICS`
- `FLUSH USER_STATISTICS`

These commands discard the specified type of stored statistical information.

- `SHOW CLIENT_STATISTICS`
- `SHOW INDEX_STATISTICS`
- `SHOW TABLE_STATISTICS`

- SHOW THREAD_STATISTICS
- SHOW USER_STATISTICS

These commands are another way to display the information you can get from the INFORMATION_SCHEMA tables. The commands accept WHERE clauses. They also accept but ignore LIKE clauses.

37.6 Status Variables

variable Com_show_client_statistics

Variable Type numeric

Scope Global/Session

The Com_show_client_statistics statement counter variable indicates the number of times the statement SHOW CLIENT_STATISTICS has been executed.

variable Com_show_index_statistics

Variable Type numeric

Scope Global/Session

The Com_show_index_statistics statement counter variable indicates the number of times the statement SHOW INDEX_STATISTICS has been executed.

variable Com_show_table_statistics

Variable Type numeric

Scope Global/Session

The Com_show_table_statistics statement counter variable indicates the number of times the statement SHOW TABLE_STATISTICS has been executed.

variable Com_show_thread_statistics

Variable Type numeric

Scope Global/Session

The Com_show_thread_statistics statement counter variable indicates the number of times the statement SHOW THREAD_STATISTICS has been executed.

variable Com_show_user_statistics

Variable Type numeric

Scope Global/Session

The Com_show_user_statistics statement counter variable indicates the number of times the statement SHOW USER_STATISTICS has been executed.

CHAPTER

THIRTYEIGHT

SLOW QUERY LOG

This feature adds microsecond time resolution and additional statistics to the slow query log output. It lets you enable or disable the slow query log at runtime, adds logging for the slave SQL thread, and adds fine-grained control over what and how much to log into the slow query log.

You can use *Percona-Toolkit*'s pt-query-digest tool to aggregate similar queries together and report on those that consume the most execution time.

38.1 Version Specific Information

- **`5.7.10-1`:**
  - Feature ported from *Percona Server* 5.6.

38.2 System Variables

variable `log_slow_filter`

Command Line Yes

Config File Yes

Scope Global, Session

Dynamic Yes

Filters the slow log by the query's execution plan. The value is a comma-delimited string, and can contain any combination of the following values:

- `qc_miss`: The query was not found in the query cache.
- `full_scan`: The query performed a full table scan.
- `full_join`: The query performed a full join (a join without indexes).
- `tmp_table`: The query created an implicit internal temporary table.
- `tmp_table_on_disk`: The query's temporary table was stored on disk.
- `filesort`: The query used a filesort.
- `filesort_on_disk`: The filesort was performed on disk.

Values are OR'ed together. If the string is empty, then the filter is disabled. If it is not empty, then queries will only be logged to the slow log if their execution plan matches one of the types of plans present in the filter.

For example, to log only queries that perform a full table scan, set the value to `full_scan`. To log only queries that use on-disk temporary storage for intermediate results, set the value to `tmp_table_on_disk,filesort_on_disk`.

variable `log_slow_rate_type`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Enumerated

Default Value `session`

Range `session`, `query`

Specifies semantic of `log_slow_rate_limit` - `session` or `query`.

variable `log_slow_rate_limit`

Command Line Yes

Config File Yes

Scope Global, session

Dynamic Yes

Default Value 1

Range 1-1000

Behavior of this variable depends from `log_slow_rate_type`.

Specifies that only a fraction of `session/query` should be logged. Logging is enabled for every nth `session/query`. By default, n is 1, so logging is enabled for every `session/query`. Please note: when `log_slow_rate_type` is `session` rate limiting is disabled for the replication thread.

Logging all queries might consume I/O bandwidth and cause the log file to grow large.

- When `log_slow_rate_type` is `session`, this option lets you log full sessions, so you have complete records of sessions for later analysis; but you can rate-limit the number of sessions that are logged. Note that this feature will not work well if your application uses any type of connection pooling or persistent connections. Note that you change `log_slow_rate_limit` in `session` mode, you should reconnect for get effect.
- When `log_slow_rate_type` is `query`, this option lets you log just some queries for later analysis. For example, if you set the value to 100, then one percent of queries will be logged.

Note that every query has global unique `query_id` and every connection can has it own (session) `log_slow_rate_limit`. Decision "log or no" calculated in following manner:

- if `log_slow_rate_limit` is 1 - log every query
- If `log_slow_rate_limit` > 1 - randomly log every 1/`log_slow_rate_limit` query.

This allows flexible setup logging behavior.

For example, if you set the value to 100, then one percent of `sessions/queries` will be logged. In *Percona Server* information about the `log_slow_rate_limit` has been added to the slow query log. This means that if the `log_slow_rate_limit` is effective it will be reflected in the slow query log for each written query. Example of the output looks like this:

```
Log_slow_rate_type: query  Log_slow_rate_limit: 10
```

variable `log_slow_sp_statements`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

Default Value TRUE

Range TRUE/FALSE

If `TRUE`, statements executed by stored procedures are logged to the slow if it is open.

***Percona Server* implemented improvements for logging of stored procedures to the slow query log:**

- Each query from a stored procedure is now logged to the slow query log individually
- `CALL` itself isn't logged to the slow query log anymore as this would be counting twice for the same query which would lead to incorrect results
- Queries that were called inside of stored procedures are annotated in the slow query log with the stored procedure name in which they run.

Example of the improved stored procedure slow query log entry:

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE improved_sp_log()
       BEGIN
        SELECT * FROM City;
        SELECT * FROM Country;
       END//
mysql> DELIMITER ;
mysql> CALL improved_sp_log();
```

When we check the slow query log after running the stored procedure ,with variable:*log_slow_sp_statements* set to `TRUE`, it should look like this:

```
# Time: 150109 11:38:55
# User@Host: root[root] @ localhost []
# Thread_id: 40  Schema: world  Last_errno: 0  Killed: 0
# Query_time: 0.012989  Lock_time: 0.000033  Rows_sent: 4079  Rows_examined: 4079  Rows_affected: 0
# Bytes_sent: 161085
# Stored routine: world.improved_sp_log
SET timestamp=1420803535;
SELECT * FROM City;
# User@Host: root[root] @ localhost []
# Thread_id: 40  Schema: world  Last_errno: 0  Killed: 0
# Query_time: 0.001413  Lock_time: 0.000017  Rows_sent: 4318  Rows_examined: 4318  Rows_affected: 0
# Bytes_sent: 194601
# Stored routine: world.improved_sp_log
SET timestamp=1420803535;
```

If variable `log_slow_sp_statements` is set to `FALSE`:

- Entry is added to a slow-log for a `CALL` statement only and not for any of the individual statements run in that stored procedure

- Execution time is reported for the `CALL` statement as the total execution time of the `CALL` including all its statements

If we run the same stored procedure with the variable `log_slow_sp_statements` is set to `FALSE` slow query log should look like this:

```
# Time: 150109 11:51:42
# User@Host: root[root] @ localhost []
# Thread_id: 40  Schema: world  Last_errno: 0  Killed: 0
# Query_time: 0.013947  Lock_time: 0.000000  Rows_sent: 4318  Rows_examined: 4318  Rows_affected: 0
# Bytes_sent: 194612
SET timestamp=1420804302;
CALL improved_sp_log();
```

Note: Support for logging stored procedures doesn't involve triggers, so they won't be logged even if this feature is enabled.

---

variable `log_slow_verbosity`

Command Line Yes

Config File Yes

Scope Global, session

Dynamic Yes

Specifies how much information to include in your slow log. The value is a comma-delimited string, and can contain any combination of the following values:

- `microtime`: Log queries with microsecond precision.
- `query_plan`: Log information about the query's execution plan.
- `innodb`: Log *InnoDB* statistics.
- `minimal`: Equivalent to enabling just `microtime`.
- `standard`: Equivalent to enabling `microtime,innodb`.
- `full`: Equivalent to all other values OR'ed together without the `profiling` and `profiling_use_getrusage` options.
- `profiling`: Enables profiling of all queries in all connections.
- `profiling_use_getrusage`: Enables usage of the getrusage function.

Values are OR'ed together.

For example, to enable microsecond query timing and *InnoDB* statistics, set this option to `microtime,innodb` or `standard`. To turn all options on, set the option to `full`.

variable `slow_query_log_use_global_control`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value None

Specifies which variables have global scope instead of local. Value is a "flag" variable - you can specify multiple values separated by commas

- none: All variables use local scope
- log_slow_filter: Global variable log_slow_filter has effect (instead of local)
- log_slow_rate_limit: Global variable log_slow_rate_limit has effect (instead of local)
- log_slow_verbosity: Global variable log_slow_verbosity has effect (instead of local)
- long_query_time: Global variable long_query_time has effect (instead of local)
- min_examined_row_limit: Global variable min_examined_row_limit has effect (instead of local)
- all Global variables has effect (instead of local)

variable slow_query_log_always_write_time

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value 10

This variable can be used to specify the query execution time after which the query will be written to the slow query log. It can be used to specify an additional execution time threshold for the slow query log, that, when exceeded, will cause a query to be logged unconditionally, that is, log_slow_rate_limit will not apply to it.

38.3 Other Information

38.3.1 Changes to the Log Format

The feature adds more information to the slow log output. Here is a sample log entry:

```
# Time: 130601  8:01:06.058915
# User@Host: root[root] @ localhost []  Id:    42
# Schema: imdb  Last_errno: 0  Killed: 0
# Query_time: 7.725616  Lock_time: 0.000328  Rows_sent: 4  Rows_examined: 1543720  Rows_affected: 0
# Bytes_sent: 272  Tmp_tables: 0  Tmp_disk_tables: 0  Tmp_table_sizes: 0
# QC_Hit: No  Full_scan: Yes  Full_join: No  Tmp_table: No  Tmp_table_on_disk: No
# Filesort: No  Filesort_on_disk: No  Merge_passes: 0
SET timestamp=1370073666;
SELECT id,title,production_year FROM title WHERE title = 'Bambi';
```

Another example (log_slow_verbosity =profiling):

```
# Time: 130601  8:03:20.700441
# User@Host: root[root] @ localhost []  Id:    43
# Schema: imdb  Last_errno: 0  Killed: 0
# Query_time: 7.815071  Lock_time: 0.000261  Rows_sent: 4  Rows_examined: 1543720  Rows_affected: 0
# Bytes_sent: 272
# Profile_starting: 0.000125 Profile_starting_cpu: 0.000120
Profile_checking_permissions: 0.000021 Profile_checking_permissions_cpu: 0.000021
Profile_Opening_tables: 0.000049 Profile_Opening_tables_cpu: 0.000048 Profile_init: 0.000048
Profile_init_cpu: 0.000049 Profile_System_lock: 0.000049 Profile_System_lock_cpu: 0.000048
Profile_optimizing: 0.000024 Profile_optimizing_cpu: 0.000024 Profile_statistics: 0.000036
Profile_statistics_cpu: 0.000037 Profile_preparing: 0.000029 Profile_preparing_cpu: 0.000029
Profile_executing: 0.000012 Profile_executing_cpu: 0.000012 Profile_Sending_data: 7.814583
Profile_Sending_data_cpu: 7.811634 Profile_end: 0.000013 Profile_end_cpu: 0.000012
```

```
Profile_query_end: 0.000014 Profile_query_end_cpu: 0.000014 Profile_closing_tables: 0.000023
Profile_closing_tables_cpu: 0.000023 Profile_freeing_items: 0.000051
Profile_freeing_items_cpu: 0.000050 Profile_logging_slow_query: 0.000006
Profile_logging_slow_query_cpu: 0.000006
# Profile_total: 7.815085 Profile_total_cpu: 7.812127
SET timestamp=1370073800;
SELECT id,title,production_year FROM title WHERE title = 'Bambi';
```

38.3.2 Connection and Schema Identifier

Each slow log entry now contains a connection identifier, so you can trace all the queries coming from a single connection. This is the same value that is shown in the Id column in `SHOW FULL PROCESSLIST` or returned from the `CONNECTION_ID()` function.

Each entry also contains a schema name, so you can trace all the queries whose default database was set to a particular schema.

```
# Id: 43  Schema: imdb
```

38.3.3 Microsecond Time Resolution and Extra Row Information

This is the original functionality offered by the `microslow` feature. `Query_time` and `Lock_time` are logged with microsecond resolution.

The feature also adds information about how many rows were examined for `SELECT` queries, and how many were analyzed and affected for `UPDATE`, `DELETE`, and `INSERT` queries,

```
# Query_time: 0.962742  Lock_time: 0.000202  Rows_sent: 4  Rows_examined: 1543719  Rows_affected: 0
```

Values and context:

- `Rows_examined`: Number of rows scanned - `SELECT`
- `Rows_affected`: Number of rows changed - `UPDATE`, `DELETE`, `INSERT`

38.3.4 Memory Footprint

The feature provides information about the amount of bytes sent for the result of the query and the number of temporary tables created for its execution - differentiated by whether they were created on memory or on disk - with the total number of bytes used by them.

```
# Bytes_sent: 8053  Tmp_tables: 1  Tmp_disk_tables: 0  Tmp_table_sizes: 950528
```

Values and context:

- `Bytes_sent`: The amount of bytes sent for the result of the query
- `Tmp_tables`: Number of temporary tables created on memory for the query
- `Tmp_disk_tables`: Number of temporary tables created on disk for the query
- `Tmp_table_sizes`: Total Size in bytes for all temporary tables used in the query

38.3.5 Query Plan Information

Each query can be executed in various ways. For example, it may use indexes or do a full table scan, or a temporary table may be needed. These are the things that you can usually see by running `EXPLAIN` on the query. The feature will now allow you to see the most important facts about the execution in the log file.

```
# QC_Hit: No  Full_scan: Yes  Full_join: No  Tmp_table: No  Tmp_table_on_disk: No
# Filesort: No  Filesort_on_disk: No  Merge_passes: 0
```

The values and their meanings are documented with the `log_slow_filter` option.

38.3.6 *InnoDB* Usage Information

The final part of the output is the *InnoDB* usage statistics. *MySQL* currently shows many per-session statistics for operations with `SHOW SESSION STATUS`, but that does not include those of *InnoDB*, which are always global and shared by all threads. This feature lets you see those values for a given query.

```
#   InnoDB_IO_r_ops: 6415  InnoDB_IO_r_bytes: 105103360  InnoDB_IO_r_wait: 0.001279
#   InnoDB_rec_lock_wait: 0.000000  InnoDB_queue_wait: 0.000000
#   InnoDB_pages_distinct: 6430
```

Values:

- `innodb_IO_r_ops`: Counts the number of page read operations scheduled. The actual number of read operations may be different, but since this can be done asynchronously, there is no good way to measure it.
- `innodb_IO_r_bytes`: Similar to innodb_IO_r_ops, but the unit is bytes.
- `innodb_IO_r_wait`: Shows how long (in seconds) it took *InnoDB* to actually read the data from storage.
- `innodb_rec_lock_wait`: Shows how long (in seconds) the query waited for row locks.
- `innodb_queue_wait`: Shows how long (in seconds) the query spent either waiting to enter the *InnoDB* queue or inside that queue waiting for execution.
- `innodb_pages_distinct`: Counts approximately the number of unique pages the query accessed. The approximation is based on a small hash array representing the entire buffer pool, because it could take a lot of memory to map all the pages. The inaccuracy grows with the number of pages accessed by a query, because there is a higher probability of hash collisions.

If the query did not use *InnoDB* tables, that information is written into the log instead of the above statistics.

38.4 Related Reading

- Impact of logging on MySQL's performance
- log_slow_filter Usage
- Blueprint in Launchpad

CHAPTER

THIRTYNINE

EXTENDED SHOW ENGINE *INNODB* STATUS

This feature reorganizes the output of `SHOW ENGINE INNODB STATUS` for a better readability and prints the amount of memory used by the internal hash tables. In addition, new variables are available to control the output.

This feature modified the `SHOW ENGINE INNODB STATUS` command as follows:

- Added two variables to control `SHOW ENGINE INNODB STATUS` information presented (bugfix for #29123):
  - `innodb_show_verbose_locks` - Whether to show records locked
  - `innodb_show_locks_held` - Number of locks held to print for each *InnoDB* transaction
- Added extended information about *InnoDB* internal hash table sizes (in bytes) in the `BUFFER POOL AND MEMORY` section; also added buffer pool size in bytes.
- Added additional LOG section information.

39.1 Version Specific Information

- `5.7.10-1`:

  Feature ported from *Percona Server* 5.6.

39.2 Other Information

- Author / Origin: Baron Schwartz, http://lists.mysql.com/internals/35174

39.3 System Variables

variable `innodb_show_verbose_locks`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type ULONG

Default Value 0

Range 0 - 1

Specifies to show records locked in `SHOW ENGINE INNODB STATUS`. The default is 0, which means only the higher-level information about the lock (which table and index is locked, etc.) is printed. If set to 1, then traditional *InnoDB* behavior is enabled: the records that are locked are dumped to the output.

variable `innodb_show_locks_held`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type ULONG

Default Value 10

Range 0 - 1000

Specifies the number of locks held to print for each *InnoDB* transaction in `SHOW ENGINE INNODB STATUS`.

39.4 Status Variables

The status variables here contain information available in the output of `SHOW ENGINE INNODB STATUS`, organized by the sections `SHOW ENGINE INNODB STATUS` displays. If you are familiar with the output of `SHOW ENGINE INNODB STATUS`, you will probably already recognize the information these variables contain.

39.4.1 BACKGROUND THREAD

The following variables contain information in the BACKGROUND THREAD section of the output from `SHOW ENGINE INNODB STATUS`. An example of that output is:

```
-----------------
BACKGROUND THREAD
-----------------
srv_master_thread loops: 1 srv_active, 0 srv_shutdown, 11844 srv_idle
srv_master_thread log flush and writes: 11844
```

variable `Innodb_master_thread_active_loops`

Variable Type Numeric

Scope Global

variable `Innodb_master_thread_idle_loops`

Variable Type Numeric

Scope Global

variable `Innodb_background_log_sync`

Variable Type Numeric

Scope Global

39.4.2 SEMAPHORES

The following variables contain information in the SEMAPHORES section of the output from `SHOW ENGINE INNODB STATUS`. An example of that output is:

```
----------
SEMAPHORES
----------
OS WAIT ARRAY INFO: reservation count 9664, signal count 11182
Mutex spin waits 20599, rounds 223821, OS waits 4479
RW-shared spins 5155, OS waits 1678; RW-excl spins 5632, OS waits 2592
Spin rounds per wait: 10.87 mutex, 15.01 RW-shared, 27.19 RW-excl
```

39.4.3 INSERT BUFFER AND ADAPTIVE HASH INDEX

The following variables contain information in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the output from `SHOW ENGINE INNODB STATUS`. An example of that output is:

```
-------------------------------------
INSERT BUFFER AND ADAPTIVE HASH INDEX
-------------------------------------
Ibuf: size 1, free list len 6089, seg size 6091,
44497 inserts, 44497 merged recs, 8734 merges
0.00 hash searches/s, 0.00 non-hash searches/s
```

variable `Innodb_ibuf_free_list`

Variable Type Numeric

Scope Global

variable `Innodb_ibuf_segment_size`

Variable Type Numeric

Scope Global

39.4.4 LOG

The following variables contain information in the LOG section of the output from `SHOW ENGINE INNODB STATUS`. An example of that output is:

```
LOG
---
Log sequence number 10145937666
Log flushed up to   10145937666
Pages flushed up to 10145937666
Last checkpoint at  10145937666
Max checkpoint age    80826164
Checkpoint age target 78300347
Modified age          0
Checkpoint age        0
0 pending log writes, 0 pending chkp writes
9 log i/o's done, 0.00 log i/o's/second
Log tracking enabled
Log tracked up to   10145937666
Max tracked LSN age 80826164
```

variable `Innodb_lsn_current`

Variable Type Numeric

Scope Global

variable `Innodb_lsn_flushed`

Variable Type Numeric

Scope Global

variable `Innodb_lsn_last_checkpoint`

Variable Type Numeric

Scope Global

variable `Innodb_checkpoint_age`

Variable Type Numeric

Scope Global

variable `Innodb_checkpoint_max_age`

Variable Type Numeric

Scope Global

39.4.5 BUFFER POOL AND MEMORY

The following variables contain information in the BUFFER POOL AND MEMORY section of the output from `SHOW ENGINE INNODB STATUS`. An example of that output is:

```
----------------------
BUFFER POOL AND MEMORY
----------------------
Total memory allocated 137363456; in additional pool allocated 0
Total memory allocated by read views 88
Internal hash tables (constant factor + variable factor)
    Adaptive hash index 2266736         (2213368 + 53368)
    Page hash           139112 (buffer pool 0 only)
    Dictionary cache    729463  (554768 + 174695)
    File system         824800  (812272 + 12528)
    Lock system         333248  (332872 + 376)
    Recovery system     0       (0 + 0)
Dictionary memory allocated 174695
Buffer pool size        8191
Buffer pool size, bytes 134201344
Free buffers            7481
Database pages          707
Old database pages      280
Modified db pages       0
Pending reads 0
Pending writes: LRU 0, flush list 0 single page 0
Pages made young 0, not young 0
0.00 youngs/s, 0.00 non-youngs/s
Pages read 707, created 0, written 1
0.00 reads/s, 0.00 creates/s, 0.00 writes/s
No buffer pool page gets since the last printout
Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead 0.00/s
LRU len: 707, unzip_LRU len: 0
```

variable `Innodb_mem_adaptive_hash`

Variable Type Numeric

Scope Global

variable `Innodb_mem_dictionary`

Variable Type Numeric

Scope Global

variable `Innodb_mem_total`

Variable Type Numeric

Scope Global

variable `Innodb_buffer_pool_pages_LRU_flushed`

Variable Type Numeric

Scope Global

variable `Innodb_buffer_pool_pages_made_not_young`

Variable Type Numeric

Scope Global

variable `Innodb_buffer_pool_pages_made_young`

Variable Type Numeric

Scope Global

variable `Innodb_buffer_pool_pages_old`

Variable Type Numeric

Scope Global

variable `Innodb_descriptors_memory`

Variable Type Numeric

Scope Global

This status variable shows the current size of the descriptors array (in bytes). The descriptor array is an *XtraDB* data structure that contains the information on currently running transactions.

variable `Innodb_read_views_memory`

Variable Type Numeric

Scope Global

This status variable shows the total amount of memory allocated for the *InnoDB* read view (in bytes).

39.4.6 TRANSACTIONS

The following variables contain information in the TRANSACTIONS section of the output from `SHOW INNODB STATUS`. An example of that output is:

```
------------
TRANSACTIONS
------------
Trx id counter F561FD
Purge done for trx's n:o < F561EB undo n:o < 0
```

```
History list length 19
LIST OF TRANSACTIONS FOR EACH SESSION:
---TRANSACTION 0, not started, process no 993, OS thread id 140213152634640
mysql thread id 15933, query id 32109 localhost root
show innodb status
---TRANSACTION F561FC, ACTIVE 29 sec, process no 993, OS thread id 140213152769808 updating or delet
mysql tables in use 1, locked 1
```

variable **Innodb_max_trx_id**

Variable Type Numeric

Scope Global

variable **Innodb_oldest_view_low_limit_trx_id**

Variable Type Numeric

Scope Global

variable **Innodb_purge_trx_id**

Variable Type Numeric

Scope Global

variable **Innodb_purge_undo_no**

Variable Type Numeric

Scope Global

39.5 INFORMATION_SCHEMA Tables

The following table contains information about the oldest active transaction in the system.

table INFORMATION_SCHEMA.**XTRADB_READ_VIEW**

Columns

- **READ_VIEW_LOW_LIMIT_TRX_NUMBER** – This is the highest transactions number at the time the view was created.
- **READ_VIEW_UPPER_LIMIT_TRX_ID** – This is the highest transactions ID at the time the view was created. This means that it should not see newer transactions with IDs bigger than or equal to that value.
- **READ_VIEW_LOW_LIMIT_TRX_ID** – This is the latest committed transaction ID at the time the oldest view was created. This means that it should see all transactions with IDs smaller than or equal to that value.

The following table contains information about the memory usage for InnoDB/XtraDB hash tables.

table INFORMATION_SCHEMA.**XTRADB_INTERNAL_HASH_TABLES**

Columns

- **INTERNAL_HASH_TABLE_NAME** – Hash table name
- **TOTAL_MEMORY** – Total amount of memory
- **CONSTANT_MEMORY** – Constant memory
- **VARIABLE_MEMORY** – Variable memory

39.6 Other reading

- SHOW INNODB STATUS walk through
- Table locks in SHOW INNODB STATUS

CHAPTER

FORTY

SHOW STORAGE ENGINES

This feature changes the comment field displayed when the `SHOW STORAGE ENGINES` command is executed and *XtraDB* is the storage engine.

Before the Change:

```
mysql> show storage engines;
+------------+---------+----------------------------------------------------------------+------------
| Engine     | Support | Comment                                                        | Transactio
+------------+---------+----------------------------------------------------------------+------------
| InnoDB     | YES     | Supports transactions, row-level locking, and foreign keys     | YES
...
+------------+---------+----------------------------------------------------------------+------------
```

After the Change:

```
mysql> show storage engines;
+------------+---------+-----------------------------------------------------------------------------
| Engine     | Support | Comment
+------------+---------+-----------------------------------------------------------------------------
| InnoDB     | YES     | Percona-XtraDB, Supports transactions, row-level locking, and foreign keys
...
+------------+---------+-----------------------------------------------------------------------------
```

40.1 Version-Specific Information

- `5.7.10-1`: Feature ported from *Percona Server* 5.6

CHAPTER

FORTYONE

PROCESS LIST

This page describes Percona changes to both the standard *MySQL* `SHOW PROCESSLIST` command and the standard *MySQL* `INFORMATION_SCHEMA` table `PROCESSLIST`.

41.1 Version Specific Information

- `5.7.10-1`:
  - Feature ported from *Percona Server* 5.6

41.2 INFORMATION_SCHEMA Tables

table `INFORMATION_SCHEMA.`**`PROCESSLIST`**

This table implements modifications to the standard *MySQL* `INFORMATION_SCHEMA` table `PROCESSLIST`.

Columns

- **ID** – The connection identifier.
- **USER** – The *MySQL* user who issued the statement.
- **HOST** – The host name of the client issuing the statement.
- **DB** – The default database, if one is selected, otherwise NULL.
- **COMMAND** – The type of command the thread is executing.
- **TIME** – The time in seconds that the thread has been in its current state.
- **STATE** – An action, event, or state that indicates what the thread is doing.
- **INFO** – The statement that the thread is executing, or NULL if it is not executing any statement.
- **TIME_MS** – The time in milliseconds that the thread has been in its current state.
- **ROWS_EXAMINED** – The number of rows examined by the statement being executed (*NOTE:* This column is not updated for each examined row so it does not necessarily show an up-to-date value while the statement is executing. It only shows a correct value after the statement has completed.).
- **ROWS_SENT** – The number of rows sent by the statement being executed.
- **ROWS_READ** – The number of rows read by the statement being executed.

41.3 Example Output

Table PROCESSLIST:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST;

+----+------+-----------+--------------------+---------+------+-----------+-----------------------
| ID | USER | HOST      | DB                 | COMMAND | TIME | STATE     | INFO
+----+------+-----------+--------------------+---------+------+-----------+-----------------------
| 12 | root | localhost | information_schema | Query   |    0 | executing | select * from processlist
+----+------+-----------+--------------------+---------+------+-----------+-----------------------
```

CHAPTER

FORTYTWO

MISC. INFORMATION_SCHEMA TABLES

This page lists the `INFORMATION_SCHEMA` tables added to standard *MySQL* by *Percona Server* that don't exist elsewhere in the documentation.

42.1 Temporary tables

Note: This feature implementation is considered ALPHA quality.

Only the temporary tables that were explicitly created with *CREATE TEMPORARY TABLE* or *ALTER TABLE* are shown, and not the ones created to process complex queries.

table `INFORMATION_SCHEMA.GLOBAL_TEMPORARY_TABLES`

Version Info

- **5.6.5-60.0** – Feature introduced

Columns

- **SESSION_ID** – *MySQL* connection id
- **TABLE_SCHEMA** – Schema in which the temporary table is created
- **TABLE_NAME** – Name of the temporary table
- **ENGINE** – Engine of the temporary table
- **NAME** – Internal name of the temporary table
- **TABLE_ROWS** – Number of rows of the temporary table
- **AVG_ROW_LENGTH** – Average row length of the temporary table
- **DATA_LENGTH** – Size of the data (Bytes)
- **INDEX_LENGTH** – Size of the indexes (Bytes)
- **CREATE_TIME** – Date and time of creation of the temporary table
- **UPDATE_TIME** – Date and time of the latest update of the temporary table

This table holds information on the temporary tables existing for all connections. You don't need the `SUPER` privilege to query this table.

table `INFORMATION_SCHEMA.TEMPORARY_TABLES`

Version Info

- **5.6.5-60.0** – Feature introduced

Columns

- **SESSION_ID** – *MySQL* connection id
- **TABLE_SCHEMA** – Schema in which the temporary table is created
- **TABLE_NAME** – Name of the temporary table
- **ENGINE** – Engine of the temporary table
- **NAME** – Internal name of the temporary table
- **TABLE_ROWS** – Number of rows of the temporary table
- **AVG_ROW_LENGTH** – Average row length of the temporary table
- **DATA_LENGTH** – Size of the data (Bytes)
- **INDEX_LENGTH** – Size of the indexes (Bytes)
- **CREATE_TIME** – Date and time of creation of the temporary table
- **UPDATE_TIME** – Date and time of the latest update of the temporary table

This table holds information on the temporary tables existing for the running connection.

42.2 Multiple Rollback Segments

Percona Server, in addition to the upstream multiple rollback segment implementation, provides the additional Information Schema table: `INFORMATION_SCHEMA.XTRADB_RSEG`.

42.3 `INFORMATION_SCHEMA` Tables

This feature provides the following table:

table `INFORMATION_SCHEMA.XTRADB_RSEG`

Columns

- **rseg_id** – rollback segment id
- **space_id** – space where the segment placed
- **physical_page_size** – physical page size
- **logical_page_size** – logical page size
- **is_compressed** – is the page compressed
- **page_no** – page number of the segment header
- **max_size** – max size in pages
- **curr_size** – current size in pages

This table shows information about all the rollback segments (the default segment and the extra segments).

Here is an example of output with `innodb_rollback_segments = 8`:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.XTRADB_RSEG;
+---------+----------+--------------------+-------------------+---------------+---------+------------+
| rseg_id | space_id | physical_page_size | logical_page_size | is_compressed | page_no | max_size   |
+---------+----------+--------------------+-------------------+---------------+---------+------------+
|       0 |        0 |              16384 |             16384 |             0 |       6 | 4294967294 |
|       1 |       24 |              16384 |             16384 |             0 |       3 | 4294967294 |
|       2 |       24 |              16384 |             16384 |             0 |       4 | 4294967294 |
|       3 |       24 |              16384 |             16384 |             0 |       5 | 4294967294 |
|       4 |       24 |              16384 |             16384 |             0 |       6 | 4294967294 |
|       5 |       24 |              16384 |             16384 |             0 |       7 | 4294967294 |
|       6 |       24 |              16384 |             16384 |             0 |       8 | 4294967294 |
|       7 |       24 |              16384 |             16384 |             0 |       9 | 4294967294 |
|       8 |       24 |              16384 |             16384 |             0 |      10 | 4294967294 |
+---------+----------+--------------------+-------------------+---------------+---------+------------+
9 rows in set (0.00 sec)
```

CHAPTER

FORTYTHREE

THREAD BASED PROFILING

Percona Server now uses thread based profiling by default, instead of process based profiling. This was implemented because with process based profiling, threads on the server, other than the one being profiled, can affect the profiling information.

Thread based profiling is using the information provided by the kernel getrusage function. Since the 2.6.26 kernel version, thread based resource usage is available with the **RUSAGE_THREAD**. This means that the thread based profiling will be used if you're running the 2.6.26 kernel or newer, or if the **RUSAGE_THREAD** has been ported back.

This feature is enabled by default if your system supports it, in other cases it uses process based profiling.

43.1 Version Specific Information

- `5.7.10-1`: Feature ported from *Percona Server* 5.6

CHAPTER

FORTYFOUR

METRICS FOR SCALABILITY MEASUREMENT

Percona Server has implemented extra scalability metrics. These metrics allow using Little's Law, queueing theory, and Universal Scalability Law to gain insights into server performance. This feature is implemented as a plugin.

44.1 Installation

Scalability Metrics plugin is shipped with *Percona Server*, but it is not installed by default. To enable the plugin you must run the following command:

```
INSTALL PLUGIN scalability_metrics SONAME 'scalability_metrics.so';
```

You can check if the plugin is loaded correctly by running:

```
SHOW PLUGINS;
```

The plugin should be listed in the output:

```
+---------------------------------+----------+--------------------+--------------------------+----------
| Name                            | Status   | Type               | Library                  | License
+---------------------------------+----------+--------------------+--------------------------+----------
...
| scalability_metrics             | ACTIVE   | AUDIT              | scalability_metrics.so   | GPL
+---------------------------------+----------+--------------------+--------------------------+----------
```

44.2 System Variables

variable `scalability_metrics_control`

Command Line Yes

Scope Global

Dynamic Yes

Variable Type String

Default Value `OFF`

Values `OFF`, `ON`, `RESET`

This variable can be used to enable and disable the collection of metrics for scalability measurement. By setting the value to `RESET` all counters will be reset while continuing to count metrics.

44.3 Status Variables

variable `scalability_metrics_elapsedtime`

Variable Type Numeric

This status variable shows total time elapsed, in microseconds, since metrics collection was started.

variable `scalability_metrics_queries`

Variable Type Numeric

This status variable shows number of completed queries since metrics collection was started.

variable `scalability_metrics_concurrency`

Variable Type Numeric

This status variable shows number of queries currently executed.

variable `scalability_metrics_totaltime`

Variable Type Numeric

This status variable shows total execution time of all queries, including the in-progress time of currently executing queries, in microseconds (ie. if two queries executed with 1 second of response time each, the result is 2 seconds).

variable `scalability_metrics_busytime`

Variable Type Numeric

This counter accounts the non-idle server time, that is, time when at least one query was executing.

44.4 Version Specific Information

- `5.7.10-1` Feature ported from *Percona Server* 5.6

44.5 Other Reading

- Fundamental performance and scalability instrumentation
- Forecasting MySQL Scalability with the Universal Scalability Law Whitepaper

CHAPTER

FORTYFIVE

RESPONSE TIME DISTRIBUTION

The slow query log provides exact information about queries that take a long time to execute. However, sometimes there are a large number of queries that each take a very short amount of time to execute. This feature provides a tool for analyzing that information by counting and displaying the number of queries according to the length of time they took to execute. The user can define time intervals that divide the range 0 to positive infinity into smaller intervals and then collect the number of commands whose execution times fall into each of those intervals.

Note that in a replication environment, the server will not take into account *any* queries executed by the slave SQL threads (whether they are slow or not) for the time distribution.

Each interval is described as:

```
(range_base ^ n; range_base ^ (n+1)]
```

The range_base is some positive number (see Limitations). The interval is defined as the difference between two nearby powers of the range base.

For example, if the range base=10, we have the following intervals:

```
(0; 10 ^ -6], (10 ^ -6; 10 ^ -5], (10 ^ -5; 10 ^ -4], ..., (10 ^ -1; 10 ^1], (10^1; 10^2]...(10^7; po
```

or

```
(0; 0.000001], (0.000001; 0.000010], (0.000010; 0.000100], ..., (0.100000; 1.0]; (1.0; 10.0]...(10000
```

For each interval, a count is made of the queries with execution times that fell into that interval.

You can select the range of the intervals by changing the range base. For example, for base range=2 we have the following intervals:

```
(0; 2 ^ -19], (2 ^ -19; 2 ^ -18], (2 ^ -18; 2 ^ -17], ..., (2 ^ -1; 2 ^1], (2 ^ 1; 2 ^ 2]...(2 ^ 25;
```

or

```
(0; 0.000001], (0.000001, 0.000003], ..., (0.25; 0.5], (0.5; 2], (2; 4]...(8388608; positive infinity
```

Small numbers look strange (i.e., don't look like powers of 2), because we lose precision on division when the ranges are calculated at runtime. In the resulting table, you look at the high boundary of the range.

For example, you may see:

```
+----------------+-------+------------+
|      time      | count |    total   |
+----------------+-------+------------|
|       0.000001 |     0 |   0.000000 |
|       0.000010 |    17 |   0.000094 |
|       0.000100 |  4301 |   0.236555 |
|       0.001000 |  1499 |   0.824450 |
```

```
|       0.010000 | 14851 |  81.680502 |
|       0.100000 |  8066 | 443.635693 |
|       1.000000 |     0 |   0.000000 |
|      10.000000 |     0 |   0.000000 |
|     100.000000 |     1 |  55.937094 |
|    1000.000000 |     0 |   0.000000 |
|   10000.000000 |     0 |   0.000000 |
|  100000.000000 |     0 |   0.000000 |
| 1000000.000000 |     0 |   0.000000 |
| TOO LONG QUERY |     0 |   0.000000 |
+----------------+-------+------------+
```

This means there were:

```
* 17 queries with 0.000001 < query execution time < = 0.000010 seconds; total execution time of the 1

* 4301 queries with 0.000010 < query execution time < = 0.000100 seconds; total execution time of the

* 1499 queries with 0.000100 < query execution time < = 0.001000 seconds; total execution time of the

* 14851 queries with 0.001000 < query execution time < = 0.010000 seconds; total execution time of th

* 8066 queries with 0.010000 < query execution time < = 0.100000 seconds; total execution time of the

* 1 query with 10.000000 < query execution time < = 100.0000 seconds; total execution time of the 1 q
```

45.1 Logging the queries in separate READ and WRITE tables

Note: This feature is considered **BETA** quality.

Percona Server is now able to log the queries response times into separate READ and WRITE INFORMATION_SCHEMA tables. The two new tables are named QUERY_RESPONSE_TIME_READ and QUERY_RESPONSE_TIME_WRITE respectively. The decision on whether a query is a `read` or a `write` is based on the type of the command. Thus, for example, an `UPDATE ... WHERE <condition>` is always logged as a `write` query even if `<condition>` is always false and thus no actual writes happen during its execution.

Following SQL commands will be considered as WRITE queries and will be logged into the QUERY_RESPONSE_TIME_WRITE table: CREATE_TABLE, CREATE_INDEX, ALTER_TABLE, TRUNCATE, DROP_TABLE, LOAD, CREATE_DB, DROP_DB, ALTER_DB, RENAME_TABLE, DROP_INDEX, CREATE_VIEW, DROP_VIEW, CREATE_TRIGGER, DROP_TRIGGER, CREATE_EVENT, ALTER_EVENT, DROP_EVENT, UPDATE, UPDATE_MULTI, INSERT, INSERT_SELECT, DELETE, DELETE_MULTI, REPLACE, REPLACE_SELECT, CREATE_USER, RENAME_USER, DROP_USER, ALTER_USER, GRANT, REVOKE, REVOKE_ALL, OPTIMIZE, CREATE_FUNCTION, CREATE_PROCEDURE, CREATE_SPFUNCTION, DROP_PROCEDURE, DROP_FUNCTION, ALTER_PROCEDURE, ALTER_FUNCTION, INSTALL_PLUGIN, and UNINSTALL_PLUGIN. Commands not listed here are considered as READ queries and will be logged into the QUERY_RESPONSE_TIME_READ table.

45.2 Installing the plugins

In order to enable this feature you'll need to install the necessary plugins:

```
mysql> INSTALL PLUGIN QUERY_RESPONSE_TIME_AUDIT SONAME 'query_response_time.so';
```

This plugin is used for gathering statistics.

```
mysql> INSTALL PLUGIN QUERY_RESPONSE_TIME SONAME 'query_response_time.so';
```

This plugin provides the interface (QUERY_RESPONSE_TIME) to output gathered statistics.

```
mysql> INSTALL PLUGIN QUERY_RESPONSE_TIME_READ SONAME 'query_response_time.so';
```

This plugin provides the interface (QUERY_RESPONSE_TIME_READ) to output gathered statistics.

```
mysql> INSTALL PLUGIN QUERY_RESPONSE_TIME_WRITE SONAME 'query_response_time.so';
```

This plugin provides the interface (QUERY_RESPONSE_TIME_WRITE) to output gathered statistics.

You can check if plugins are installed correctly by running:

```
mysql> SHOW PLUGINS;

...
| QUERY_RESPONSE_TIME          | ACTIVE   | INFORMATION SCHEMA | query_response_time.so | GPL     |
| QUERY_RESPONSE_TIME_AUDIT    | ACTIVE   | AUDIT              | query_response_time.so | GPL     |
| QUERY_RESPONSE_TIME_READ     | ACTIVE   | INFORMATION SCHEMA | query_response_time.so | GPL     |
| QUERY_RESPONSE_TIME_WRITE    | ACTIVE   | INFORMATION SCHEMA | query_response_time.so | GPL     |
+------------------------------+----------+--------------------+------------------------+---------+
```

45.3 Usage

To start collecting query time metrics, query_response_time_stats should be enabled:

```
SET GLOBAL query_response_time_stats = on;
```

And to make it persistent, add the same to my.cnf:

```
[mysqld]
query_response_time_stats = on
```

45.3.1 SELECT

You can get the distribution using the query:

```
mysql> SELECT * from INFORMATION_SCHEMA.QUERY_RESPONSE_TIME
time                  count   total
0.000001              0       0.000000
0.000010              0       0.000000
0.000100              1       0.000072
0.001000              0       0.000000
0.010000              0       0.000000
0.100000              0       0.000000
1.000000              0       0.000000
10.000000             8       47.268416
100.000000            0       0.000000
1000.000000           0       0.000000
10000.000000          0       0.000000
100000.000000         0       0.000000
```

```
1000000.000000          0       0.000000
TOO LONG QUERY          0       0.000000
```

You can write a complex query like:

```
SELECT c.count, c.time,
(SELECT SUM(a.count) FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME as a WHERE a.count != 0) as query_co
(SELECT COUNT(*)     FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME as b WHERE b.count != 0) as not_zero
(SELECT COUNT(*)     FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME) as region_count
FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME as c WHERE c.count > 0;
```

Note: If `query_response_time_stats` is ON, the execution times for these two `SELECT` queries will also be collected.

45.3.2 FLUSH

Flushing can be done by setting the `query_response_time_flush` to `ON` (or `1`):

```
mysql> SET GLOBAL query_response_time_flush='ON';
```

`FLUSH` does two things:

- Clears the collected times from the `QUERY_RESPONSE_TIME`, `QUERY_RESPONSE_TIME_READ`, and `QUERY_RESPONSE_TIME_WRITE` tables
- Reads the value of `query_response_time_range_base` and uses it to set the range base for the table

Note: The execution time for the `FLUSH` query will also be collected.

45.3.3 Stored procedures

Stored procedure calls count as a single query.

45.3.4 Collect time point

Time is collected after query execution completes (before clearing data structures).

45.4 Version Specific Information

- `5.7.10-1`: Feature ported from *Percona Server* 5.6

45.5 System Variables

variable `query_response_time_flush`

Command Line Yes

Config File No

Scope Global

Dynamic No

Variable Type Boolean

Default Value OFF

Range OFF/ON

Setting this variable to `ON` will flush the statistics and re-read the `query_response_time_range_base`.

variable `query_response_time_range_base`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 10

Range 2-1000

Sets up the logarithm base for the scale.

NOTE: The variable takes effect only after this command has been executed:

```
mysql> SET GLOBAL query_response_time_flush=1;
```

variable `query_response_time_stats`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

Default Value OFF

Range ON/OFF

This variable enables and disables collection of query times.

45.6 INFORMATION_SCHEMA Tables

table `INFORMATION_SCHEMA.QUERY_RESPONSE_TIME`

Columns

- **TIME** (*VARCHAR*) – Interval range in which the query occurred
- **COUNT** (*INT(11)*) – Number of queries with execution times that fell into that interval
- **TOTAL** (*VARCHAR*) – Total execution time of the queries

table `INFORMATION_SCHEMA.QUERY_RESPONSE_TIME_READ`

Columns

- **TIME** (*VARCHAR*) – Interval range in which the query occurred
- **COUNT** (*INT(11)*) – Number of queries with execution times that fell into that interval

- **TOTAL** (*VARCHAR*) – Total execution time of the queries

table INFORMATION_SCHEMA.**QUERY_RESPONSE_TIME_WRITE**

Columns

- **TIME** (*VARCHAR*) – Interval range in which the query occurred
- **COUNT** (*INT(11)*) – Number of queries with execution times that fell into that interval
- **TOTAL** (*VARCHAR*) – Total execution time of the queries

Part IX

TokuDB

CHAPTER

FORTYSIX

TOKUDB INTRODUCTION

TokuDB is a highly scalable, zero-maintenance downtime MySQL storage engine that delivers indexing-based query acceleration, improved replication performance, unparalleled compression, and live schema modification. The *TokuDB* storage engine is a scalable, ACID and MVCC compliant storage engine that provides indexing-based query improvements, offers online schema modifications, and reduces slave lag for both hard disk drives and flash memory. This storage engine is specifically designed for high performance on write-intensive workloads which is achieved with Fractal Tree indexing.

Percona Server is compatible with the separately available *TokuDB* storage engine package. The *TokuDB* engine must be separately downloaded and then enabled as a plug-in component. This package can be installed alongside with standard *Percona Server* 5.7 releases and does not require any specially adapted version of *Percona Server*.

Warning: Only the Percona supplied *TokuDB* engine should be used with *Percona Server* 5.7. A *TokuDB* engine downloaded from other sources is not compatible. *TokuDB* file formats are not the same across *MySQL* variants. Migrating from one variant to any other variant requires a logical data dump and reload.

Additional features unique to *TokuDB* include:

- Up to 25x Data Compression
- Fast Inserts
- Eliminates Slave Lag with *Read Free Replication*
- Hot Schema Changes
- Hot Index Creation - *TokuDB* tables support insertions, deletions and queries with no down time while indexes are being added to that table
- Hot column addition, deletion, expansion, and rename - *TokuDB* tables support insertions, deletions and queries without down-time when an alter table adds, deletes, expands, or renames columns
- On-line Backup

For more information on installing and using *TokuDB* click on the following links:

46.1 TokuDB Installation

Percona Server is compatible with the separately available *TokuDB* storage engine package. The *TokuDB* engine must be separately downloaded and then enabled as a plug-in component. This package can be installed alongside with standard *Percona Server* 5.7 releases and does not require any specially adapted version of *Percona Server*.

The *TokuDB* storage engine is a scalable, ACID and MVCC compliant storage engine that provides indexing-based query improvements, offers online schema modifications, and reduces slave lag for both hard disk drives and flash memory. This storage engine is specifically designed for high performance on write-intensive workloads which is

achieved with Fractal Tree indexing. To learn more about Fractal Tree indexing, you can visit the following Wikipedia page.

Warning: Only the Percona supplied *TokuDB* engine should be used with *Percona Server* 5.7. A *TokuDB* engine downloaded from other sources is not compatible. *TokuDB* file formats are not the same across *MySQL* variants. Migrating from one variant to any other variant requires a logical data dump and reload.

46.1.1 Prerequisites

`libjemalloc` library

TokuDB storage engine requires `libjemalloc` library 3.3.0 or greater. If the version in the distribution repository is lower than that you can use one from *Percona Software Repositories* or download it from somewhere else.

If the `libjemalloc` wasn't installed and enabled before it will be automatically installed when installing the *TokuDB* storage engine package by using the **apt'** or **yum** package manager, but *Percona Server* instance should be restarted for `libjemalloc` to be loaded. This way `libjemalloc` will be loaded with `LD_PRELOAD`. You can also enable `libjemalloc` by specifying `malloc-lib` variable in the `[mysqld_safe]` section of the `my.cnf` file:

```
[mysqld_safe]
malloc-lib= /path/to/jemalloc
```

Transparent huge pages

TokuDB won't be able to start if the transparent huge pages are enabled. Transparent huge pages is feature available in the newer kernel versions. You can check if the Transparent huge pages are enabled with:

```
$ cat /sys/kernel/mm/transparent_hugepage/enabled

 [always] madvise never
```

If transparent huge pages are enabled and you try to start the TokuDB engine you'll get the following message in you `error.log`:

```
Transparent huge pages are enabled, according to /sys/kernel/mm/redhat_transparent_hugepage/enabled
Transparent huge pages are enabled, according to /sys/kernel/mm/transparent_hugepage/enabled
```

You can disable transparent huge pages permanently by passing `transparent_hugepage=never` to the kernel in your bootloader (**NOTE**: For this change to take an effect you'll need to reboot your server).

You can disable the transparent huge pages by running the following command as root (**NOTE**: Setting this will last only until the server is rebooted):

```
echo never > /sys/kernel/mm/transparent_hugepage/enabled
echo never > /sys/kernel/mm/transparent_hugepage/defrag
```

46.1.2 Installation

TokuDB storage engine for *Percona Server* is currently available in our *apt* and *yum* repositories.

You can install the *Percona Server* with *TokuDB* engine by using the apt/yum commands:

```
[root@centos ~]# yum install Percona-Server-tokudb-57.x86_64
```

or

```
root@wheezy:~# apt-get install percona-server-tokudb-5.7
```

46.1.3 Enabling the TokuDB Storage Engine

Once the *TokuDB* server package has been installed following output will be shown:

```
* This release of Percona Server is distributed with TokuDB storage engine.
   * Run the following script to enable the TokuDB storage engine in Percona Server:

    ps_tokudb_admin --enable -u <mysql_admin_user> -p[mysql_admin_pass] [-S <socket>] [-h <host> -P <

   * See http://www.percona.com/doc/percona-server/5.7/tokudb/tokudb_installation.html for more insta

   * See http://www.percona.com/doc/percona-server/5.7/tokudb/tokudb_intro.html for an introduction t
```

Percona Server has implemented `ps_tokudb_admin` script to make the enabling the *TokuDB* storage engine easier. This script will automatically disable Transparent huge pages, if they're enabled, and install and enable the *TokuDB* storage engine with all the required plugins. You need to run this script as root or with **sudo**. After you run the script with required parameters:

```
ps_tokudb_admin --enable -uroot -pPassw0rd
```

Following output will be displayed:

```
Checking if Percona server is running with jemalloc enabled...
>> Percona server is running with jemalloc enabled.

Checking transparent huge pages status on the system...
>> Transparent huge pages are currently disabled on the system.

Checking if thp-setting=never option is already set in config file...
>> Option thp-setting=never is not set in the config file.
>> (needed only if THP is not disabled permanently on the system)

Checking TokuDB plugin status...
>> TokuDB plugin is not installed.

Adding thp-setting=never option into /etc/mysql/my.cnf
>> Successfuly added thp-setting=never option into /etc/mysql/my.cnf

Installing TokuDB engine...
>> Successfuly installed TokuDB plugin.
```

If the script returns no errors, *TokuDB* storage engine should be successfully enabled on your server. You can check it out by running:

```
mysql> SHOW ENGINES;
...
 | TokuDB | YES | Tokutek TokuDB Storage Engine with Fractal Tree(tm) Technology | YES | YES | YES |
...
```

46.1.4 Enabling the TokuDB Storage Engine Manually

If you don't want to use `ps_tokudb_admin` script you'll need to manually install the storage engine ad required plugins.

```
INSTALL PLUGIN tokudb SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_file_map SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_fractal_tree_info SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_fractal_tree_block_map SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_trx SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_locks SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_lock_waits SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_background_job_status SONAME 'ha_tokudb.so';
```

After the engine has been installed it should be present in the engines list. To check if the engine has been correctly installed and active:

```
mysql> SHOW ENGINES;
...
| TokuDB | YES | Tokutek TokuDB Storage Engine with Fractal Tree(tm) Technology | YES | YES | YES |
...
```

To check if all the *TokuDB* plugins have been installed correctly you should run:

```
mysql> SHOW PLUGINS;
...
| TokuDB                        | ACTIVE   | STORAGE ENGINE     | ha_tokudb.so | GPL     |
| TokuDB_file_map               | ACTIVE   | INFORMATION SCHEMA | ha_tokudb.so | GPL     |
| TokuDB_fractal_tree_info      | ACTIVE   | INFORMATION SCHEMA | ha_tokudb.so | GPL     |
| TokuDB_fractal_tree_block_map | ACTIVE   | INFORMATION SCHEMA | ha_tokudb.so | GPL     |
| TokuDB_trx                    | ACTIVE   | INFORMATION SCHEMA | ha_tokudb.so | GPL     |
| TokuDB_locks                  | ACTIVE   | INFORMATION SCHEMA | ha_tokudb.so | GPL     |
| TokuDB_lock_waits             | ACTIVE   | INFORMATION SCHEMA | ha_tokudb.so | GPL     |
| TokuDB_background_job_status  | ACTIVE   | INFORMATION SCHEMA | ha_tokudb.so | GPL     |
...
```

46.1.5 TokuDB Version

TokuDB storage engine version can be checked with:

```
mysql> SELECT @@tokudb_version;
+-----------------+
| @@tokudb_version |
+-----------------+
| 5.7.10-1rc1     |
+-----------------+
1 row in set (0.00 sec)
```

46.1.6 Upgrade

Installing the *TokuDB* package is compatible with existing server setup and databases.

46.2 Using TokuDB

Warning: Do not move or modify any *TokuDB* files. You will break the database, and need to recover the database from a backup.

46.2.1 Fast Insertions and Richer Indexes

TokuDB's fast indexing enables fast queries through the use of rich indexes, such as covering and clustering indexes. It's worth investing some time to optimize index definitions to get the best performance from *MySQL* and *TokuDB*. Here are some resources to get you started:

- "Understanding Indexing" by Zardosht Kasheff (video)
- Rule of Thumb for Choosing Column Order in Indexes
- Covering Indexes: Orders-of-Magnitude Improvements
- Introducing Multiple Clustering Indexes
- Clustering Indexes vs. Covering Indexes
- How Clustering Indexes Sometimes Helps UPDATE and DELETE Performance
- *High Performance MySQL, 3rd Edition* by Baron Schwartz, Peter Zaitsev, Vadim Tkachenko, Copyright 2012, O'Reilly Media. See Chapter 5, *Indexing for High Performance*.

46.2.2 Clustering Secondary Indexes

One of the keys to exploiting TokuDB's strength in indexing is to make use of clustering secondary indexes.

TokuDB allows a secondary key to be defined as a clustering key. This means that all of the columns in the table are clustered with the secondary key. *Percona Server* parser and query optimizer support Multiple Clustering Keys when *TokuDB* engine is used. This means that the query optimizer will avoid primary clustered index reads and replace them by secondary clustered index reads in certain scenarios.

The parser has been extended to support following syntax:

```
CREATE TABLE ... ( ..., CLUSTERING KEY identifier (column list), ...
CREATE TABLE ... ( ..., UNIQUE CLUSTERING KEY identifier (column list), ...
CREATE TABLE ... ( ..., CLUSTERING UNIQUE KEY identifier (column list), ...
CREATE TABLE ... ( ..., CONSTRAINT identifier UNIQUE CLUSTERING KEY identifier (column list), ...
CREATE TABLE ... ( ..., CONSTRAINT identifier CLUSTERING UNIQUE KEY identifier (column list), ...

CREATE TABLE ... (... column type CLUSTERING [UNIQUE] [KEY], ...)
CREATE TABLE ... (... column type [UNIQUE] CLUSTERING [KEY], ...)

ALTER TABLE ..., ADD CLUSTERING INDEX identifier (column list), ...
ALTER TABLE ..., ADD UNIQUE CLUSTERING INDEX identifier (column list), ...
ALTER TABLE ..., ADD CLUSTERING UNIQUE INDEX identifier (column list), ...
ALTER TABLE ..., ADD CONSTRAINT identifier UNIQUE CLUSTERING INDEX identifier (column list), ...
ALTER TABLE ..., ADD CONSTRAINT identifier CLUSTERING UNIQUE INDEX identifier (column list), ...

CREATE CLUSTERING INDEX identifier ON ...
```

To define a secondary index as clustering, simply add the word `CLUSTERING` before the key definition. For example:

```
CREATE TABLE table (
  column_a INT,
  column_b INT,
  column_c INT,
  PRIMARY KEY index_a (column_a),
  CLUSTERING KEY index_b (column_b)) ENGINE = TokuDB;
```

In the previous example, the primary table is indexed on *column_a*. Additionally, there is a secondary clustering index (named *index_b*) sorted on *column_b*. Unlike non-clustered indexes, clustering indexes include all the columns of a

table and can be used as covering indexes. For example, the following query will run very fast using the clustering *index_b*:

```
SELECT column_c
  FROM table
  WHERE column_b BETWEEN 10 AND 100;
```

This index is sorted on *column_b*, making the `WHERE` clause fast, and includes *column_c*, which avoids lookups in the primary table to satisfy the query.

TokuDB makes clustering indexes feasible because of its excellent compression and very high indexing rates. For more information about using clustering indexes, see Introducing Multiple Clustering Indexes.

46.2.3 Hot Index Creation

TokuDB enables you to add indexes to an existing table and still perform inserts and queries on that table while the index is being created.

The `ONLINE` keyword is not used. Instead, the value of the `tokudb_create_index_online` client session variable is examined.

Hot index creation is invoked using the `CREATE INDEX` command after setting `tokudb_create_index_online` to `on` as follows:

```
mysql> SET tokudb_create_index_online=on;
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE INDEX index ON table (field_name);
```

Alternatively, using the `ALTER TABLE` command for creating an index will create the index offline (with the table unavailable for inserts or queries), regardless of the value of `tokudb_create_index_online`. The only way to hot create an index is to use the `CREATE INDEX` command.

Hot creating an index will be slower than creating the index offline, and progress depends how busy the mysqld server is with other tasks. Progress of the index creation can be seen by using the `SHOW PROCESSLIST` command (in another client). Once the index creation completes, the new index will be used in future query plans.

If more than one hot `CREATE INDEX` is issued for a particular table, the indexes will be created serially. An index creation that is waiting for another to complete will be shown as *Locked* in `SHOW PROCESSLIST`. We recommend that each `CREATE INDEX` be allowed to complete before the next one is started.

46.2.4 Hot Column Add, Delete, Expand, and Rename (HCADER)

TokuDB enables you to add or delete columns in an existing table, expand `char`, `varchar`, `varbinary`, and `integer` type columns in an existing table, or rename an existing column in a table with little blocking of other updates and queries. HCADER typically blocks other queries with a table lock for no more than a few seconds. After that initial short-term table locking, the system modifies each row (when adding, deleting, or expanding columns) later, when the row is next brought into main memory from disk. For column rename, all the work is done during the seconds of downtime. On-disk rows need not be modified.

To get good performance from HCADER, observe the following guidelines:

- The work of altering the table for column addition, deletion, or expansion is performed as subsequent operations touch parts of the Fractal Tree, both in the primary index and secondary indexes.

  You can force the column addition, deletion, or expansion work to be performed all at once using the standard syntax of `OPTIMIZE TABLE X`, when a column has been added to, deleted from, or expanded in table X. It is important to note that as of *TokuDB* version 7.1.0, `OPTIMIZE TABLE` is also hot, so that a table supports

updates and queries without blocking while an `OPTIMIZE TABLE` is being performed. Also, a hot `OPTIMIZE TABLE` does not rebuild the indexes, since *TokuDB* indexes do not age. Rather, they flush all background work, such as that induced by a hot column addition, deletion, or expansion.

- Each hot column addition, deletion, or expansion operation must be performed individually (with its own SQL statement). If you want to add, delete, or expand multiple columns use multiple statements.
- Avoid adding, deleting, or expanding a column at the same time as adding or dropping an index.
- The time that the table lock is held can vary. The table-locking time for HCADER is dominated by the time it takes to flush dirty pages, because MySQL closes the table after altering it. If a checkpoint has happened recently, this operation is fast (on the order of seconds). However, if the table has many dirty pages, then the flushing stage can take on the order of minutes.
- Avoid dropping a column that is part of an index. If a column to be dropped is part of an index, then dropping that column is slow. To drop a column that is part of an index, first drop the indexes that reference the column in one alter table statement, and then drop the column in another statement.
- Hot column expansion operations are only supported to `char`, `varchar`, `varbinary`, and `integer` data types. Hot column expansion is not supported if the given column is part of the primary key or any secondary keys.
- Rename only one column per statement. Renaming more than one column will revert to the standard MySQL blocking behavior. The proper syntax is as follows:

```
ALTER TABLE table
  CHANGE column_old column_new
  DATA_TYPE REQUIRED_NESS DEFAULT
```

Here's an example of how that might look:

```
ALTER TABLE table
  CHANGE column_old column_new
  INT(10) NOT NULL;
```

Notice that all of the column attributes must be specified. `ALTER TABLE table CHANGE column_old column_new;` induces a slow, blocking column rename.

- Hot column rename does not support the following data types: `TIME`, `ENUM`, `BLOB`, `TINYBLOB`, `MEDIUMBLOB`, `LONGBLOB`. Renaming columns of these types will revert to the standard MySQL blocking behavior.
- Temporary tables cannot take advantage of HCADER. Temporary tables are typically small anyway, so altering them using the standard method is usually fast.

46.2.5 Compression Details

TokuDB offers different levels of compression, which trade off between the amount of CPU used and the compression achieved. Standard compression uses less CPU but generally compresses at a lower level, high compression uses more CPU and generally compresses at a higher level. We have seen compression up to 25x on customer data.

Compression in *TokuDB* occurs on background threads, which means that high compression need not slow down your database. Indeed, in some settings, we've seen higher overall database performance with high compression.

Note: We recommend that users use standard compression on machines with six or fewer cores, and high compression on machines with more than six cores.

The ultimate choice depends on the particulars of how a database is used, and we recommend that users use the default settings unless they have profiled their system with high compression in place.

Compression is set on a per-table basis and is controlled by setting row format during a `CREATE TABLE` or `ALTER TABLE`. For example:

```
CREATE TABLE table (
  column_a INT NOT NULL PRIMARY KEY,
  column_b INT NOT NULL) ENGINE=TokuDB
  ROW_FORMAT=row_format;
```

If no row format is specified in a `CREATE TABLE`, the table is compressed using whichever row format is specified in the session variable `tokudb_row_format`. If no row format is set nor is `tokudb_row_format`, the zlib compressor is used.

`row_format` and `tokudb_row_format` variables accept the following values:

- `TOKUDB_DEFAULT`: This sets the compression to the default behavior. As of TokuDB 7.1.0, the default behavior is to compress using the zlib library. In the future this behavior may change.
- `TOKUDB_FAST`: This sets the compression to use the quicklz library.
- `TOKUDB_SMALL`: This sets the compression to use the lzma library.

In addition, you can choose a compression library directly, which will override previous values. The following libraries are available:

- `TOKUDB_ZLIB`: Compress using the zlib library, which provides mid-range compression and CPU utilization.
- `TOKUDB_QUICKLZ`: Compress using the quicklz library, which provides light compression and low CPU utilization.
- `TOKUDB_LZMA`: Compress using the lzma library, which provides the highest compression and high CPU utilization.
- `TOKUDB_SNAPPY` - This compression is using snappy library and aims for very high speeds and reasonable compression.
- `TOKUDB_UNCOMPRESSED`: This setting turns off compression and is useful for tables with data that cannot be compressed.

46.2.6 Changing Compression of a Table

Modify the compression used on a particular table with the following command:

```
ALTER TABLE table
  ROW_FORMAT=row_format;
```

Note: Changing the compression of a table only affects newly written data (dirtied blocks). After changing a table's compression you can run `OPTIMZE TABLE` to rewrite all blocks of the table and its indexes.

46.2.7 Read Free Replication

TokuDB slaves can be configured to perform significantly less read IO in order to apply changes from the master. By utilizing the power of Fractal Tree indexes:

- insert/update/delete operations can be configured to eliminate read-modify-write behavior and simply inject messages into the appropriate Fractal Tree indexes
- update/delete operations can be configured to eliminate the IO required for uniqueness checking

To enable Read Free Replication, the servers must be configured as follows:

- On the replication master:
  - Enable row based replication: set `BINLOG_FORMAT=ROW`
- On the replication slave(s):
  - The slave must be in read-only mode: set `read_only=1`
  - Disable unique checks: set `tokudb_rpl_unique_checks=0`
  - Disable lookups (read-modify-write): set `tokudb_rpl_lookup_rows=0`

Note: You can modify one or both behaviors on the slave(s).

Note: As long as the master is using row based replication, this optimization is available on a *TokuDB* slave. This means that it's available even if the master is using *InnoDB* or *MyISAM* tables, or running non-TokuDB binaries.

Warning: *TokuDB* Read Free Replication will not propagate `UPDATE` and `DELETE` events reliably if *TokuDB* table is missing the primary key which will eventually lead to data inconsistency on the slave.

46.2.8 Transactions and ACID-compliant Recovery

By default, *TokuDB* checkpoints all open tables regularly and logs all changes between checkpoints, so that after a power failure or system crash, *TokuDB* will restore all tables into their fully ACID-compliant state. That is, all committed transactions will be reflected in the tables, and any transaction not committed at the time of failure will be rolled back.

The default checkpoint period is every 60 seconds, and this specifies the time from the beginning of one checkpoint to the beginning of the next. If a checkpoint requires more than the defined checkpoint period to complete, the next checkpoint begins immediately. It is also related to the frequency with which log files are trimmed, as described below. The user can induce a checkpoint at any time by issuing the `FLUSH LOGS` command. When a database is shut down normally it is also checkpointed and all open transactions are aborted. The logs are trimmed at startup.

46.2.9 Managing Log Size

TokuDB keeps log files back to the most recent checkpoint. Whenever a log file reaches 100 MB, a new log file is started. Whenever there is a checkpoint, all log files older than the checkpoint are discarded. If the checkpoint period is set to be a very large number, logs will get trimmed less frequently. This value is set to 60 seconds by default.

TokuDB also keeps rollback logs for each open transaction. The size of each log is proportional to the amount of work done by its transaction and is stored compressed on disk. Rollback logs are trimmed when the associated transaction completes.

46.2.10 Recovery

Recovery is fully automatic with *TokuDB*. *TokuDB* uses both the log files and rollback logs to recover from a crash. The time to recover from a crash is proportional to the combined size of the log files and uncompressed size of rollback logs. Thus, if there were no long-standing transactions open at the time of the most recent checkpoint, recovery will take less than a minute.

46.2.11 Disabling the Write Cache

When using any transaction-safe database, it is essential that you understand the write-caching characteristics of your hardware. *TokuDB* provides transaction safe (ACID compliant) data storage for *MySQL*. However, if the underlying operating system or hardware does not actually write data to disk when it says it did, the system can corrupt your database when the machine crashes. For example, *TokuDB* can not guarantee proper recovery if it is mounted on an NFS volume. It is always safe to disable the write cache, but you may be giving up some performance.

For most configurations you must disable the write cache on your disk drives. On ATA/SATA drives, the following command should disable the write cache:

```
$ hdparm -W0 /dev/hda
```

There are some cases when you can keep the write cache, for example:

- Write caching can remain enabled when using XFS, but only if XFS reports that disk write barriers work. If you see one of the following messages in /var/log/messages, then you must disable the write cache:
  - `Disabling barriers, not supported with external log device`
  - `Disabling barriers, not supported by the underlying device`
  - `Disabling barriers, trial barrier write failed`

  XFS write barriers appear to succeed for single disks (with no LVM), or for very recent kernels (such as that provided by Fedora 12). For more information, see the XFS FAQ.

In the following cases, you must disable the write cache:

- If you use the ext3 filesystem
- If you use LVM (although recent Linux kernels, such as Fedora 12, have fixed this problem)
- If you use Linux's software RAID
- If you use a RAID controller with battery-backed-up memory. This may seem counter-intuitive. For more information, see the XFS FAQ

In summary, you should disable the write cache, unless you have a very specific reason not to do so.

46.2.12 Progress Tracking

TokuDB has a system for tracking progress of long running statements, thereby removing the need to define triggers to track statement execution, as follows:

- Bulk Load: When loading large tables using `LOAD DATA INFILE` commands, doing a `SHOW PROCESSLIST` command in a separate client session shows progress. There are two progress stages. The first will state something like `Inserted about 1000000 rows`. After all rows are processed like this, the next stage tracks progress by showing what fraction of the work is done (e.g. `Loading of data about 45% done`)
- Adding Indexes: When adding indexes via `ALTER TABLE` or `CREATE INDEX`, the command `SHOW PROCESSLIST` shows progress. When adding indexes via `ALTER TABLE` or `CREATE INDEX`, the command `SHOW PROCESSLIST` will include an estimation of the number of rows processed. Use this information to verify progress is being made. Similar to bulk loading, the first stage shows how many rows have been processed, and the second stage shows progress with a fraction.
- Commits and Aborts: When committing or aborting a transaction, the command `SHOW PROCESSLIST` will include an estimate of the transactional operations processed.

46.2.13 Migrating to TokuDB

To convert an existing table to use the *TokuDB* engine, run `ALTER TABLE... ENGINE=TokuDB`. If you wish to load from a file, use `LOAD DATA INFILE` and not `mysqldump`. Using `mysqldump` will be much slower. To create a file that can be loaded with `LOAD DATA INFILE`, refer to the `INTO OUTFILE` option of the SELECT Syntax.

Note: Creating this file does not save the schema of your table, so you may want to create a copy of that as well.

46.3 Getting Started with TokuDB

46.3.1 System and Hardware Requirements

Operating Systems: *TokuDB* is currently supported on 64-bit Linux only.

Memory: *TokuDB* Requires at least 1GB of main memory but for best results, we recommend to run with at least 2GB of main memory.

Disk space and configuration: Please make sure to allocate enough disk space for data, indexes and logs. In our users' experience, *TokuDB* achieves up to 25x space savings on data and indexes over *InnoDB* due to high compression.

46.3.2 Creating Tables and Loading Data

Creating TokuDB Tables

TokuDB tables are created the same way as other tables in *MySQL* by specifying `ENGINE=TokuDB` in the table definition. For example, the following command creates a table with a single column and uses the *TokuDB* storage engine to store its data:

```
CREATE TABLE table (
  id INT(11) NOT NULL) ENGINE=TokuDB;
```

Loading Data

Once *TokuDB* tables have been created, data can be inserted or loaded using standard *MySQL* insert or bulk load operations. For example, the following command loads data from a file into the table:

```
LOAD DATA INFILE file
  INTO TABLE table;
```

Note: For more information about loading data, see the *MySQL* 5.6 reference manual.

Migrating Data from an Existing Database

Use the following command to convert an existing table for the *TokuDB* storage engine:

```
ALTER TABLE table
  ENGINE=TokuDB;
```

46.3.3 Bulk Loading Data

The *TokuDB* bulk loader imports data much faster than regular *MySQL* with *InnoDB*. To make use of the loader you need flat files in either comma separated or tab separated format. The *MySQL* `LOAD DATA INFILE ...` statement will invoke the bulk loader if the table is empty. Keep in mind that while this is the most convenient and, in most cases, the fastest way to initialize a *TokuDB* table, it may not be replication safe if applied to the master

For more information, see the *MySQL* 5.6 Reference Manual: LOAD DATA INFILE.

To obtain the logical backup and then bulk load into *TokuDB*, follow these steps:

1. Create a logical backup of the original table. The easiest way to achieve this is using `SELECT ... INTO OUTFILE`. Keep in mind that the file will be created on the server.

   ```
   SELECT * FROM table
     INTO OUTFILE 'file.csv';
   ```

2. The output file should either be copied to the destination server or the client machine from which you plan to load it.

3. To load the data into the server use `LOAD DATA INFILE`. If loading from a machine other than the server use the keyword `LOCAL` to point to the file on local machine. Keep in mind that you will need enough disk space on the temporary directory on the server since the local file will be copied onto the server by the MySQL client utility.

   ```
   LOAD DATA [LOCAL] INFILE 'file.csv';
   ```

It is possible to create the CSV file using either **mysqldump** or the *MySQL* client utility as well, in which case the resulting file will reside on a local directory. In these 2 cases you have to make sure to use the correct command line options to create a file compatible with `LOAD DATA INFILE`.

The bulk loader will use more space than normal for logs and temporary files while running, make sure that your file system has enough disk space to process your load. As a rule of thumb, it should be approximately 1.5 times the size of the raw data.

Note: Please read the original *MySQL* documentation to understand the needed privileges and replication issues needed around `LOAD DATA INFILE`.

46.3.4 Considerations to Run TokuDB in Production

In most cases, the default options should be left in-place to run *TokuDB*, however it is a good idea to review some of the configuration parameters.

Memory allocation

TokuDB will allocate 50% of the installed RAM for its own cache (global variable `tokudb_cache_size`). While this is optimal in most situations, there are cases where it may lead to memory over allocation. If the system tries to allocate more memory than is available, the machine will begin swapping and run much slower than normal.

It is necessary to set the `tokudb_cache_size` to a value other than the default in the following cases:

- **Running other memory heavy processes on the same server as TokuDB**: In many cases, the database process needs to share the system with other server processes like additional database instances, http server, application server, e-mail server, monitoring systems and others. In order to properly configure TokuDB's memory consumption, it's important to understand how much free memory will be left and assign a sensible value for *TokuDB*. There is no fixed rule, but a conservative choice would be 50% of available RAM while all the other

processes are running. If the result is under 2 GB, you should consider moving some of the other processes to a different system or using a dedicated database server.

`tokudb_cache_size` is a static variable, so it needs to be set before starting the server and cannot be changed while the server is running. For example, to set up TokuDB's cache to 4G, add the following line to your `my.cnf` file:

```
tokudb_cache_size = 4G
```

- **System using InnoDB and TokuDB**: When using both the *TokuDB* and *InnoDB* storage engines, you need to manage the cache size for each. For example, on a server with 16 GB of RAM you could use the following values in your configuration file:

```
innodb_buffer_pool_size = 2G
tokudb_cache_size = 8G
```

- **Using TokuDB with Federated or FederatedX tables**: The Federated engine in *MySQL* and FederatedX in *MariaDB* allow you to connect to a table on a remote server and query it as if it were a local table (please see the *MySQL* documentation: 14.11. The FEDERATED Storage Engine for details). When accessing the remote table, these engines could import the complete table contents to the local server to execute a query. In this case, you will have to make sure that there is enough free memory on the server to handle these remote tables. For example, if your remote table is 8 GB in size, the server has to have more than 8 GB of free RAM to process queries against that table without going into swapping or causing a kernel panic and crash the MySQL process. There are no parameters to limit the amount of memory that the Federated or FederatedX engine will allocate while importing the remote dataset.

Specifying the Location for Files

As with *InnoDB*, it is possible to specify different locations than the default for TokuDB's data, log and temporary files. This way you may distribute the load and control the disk space. The following variables control file location:

- `tokudb_data_dir`: This variable defines the directory where the *TokuDB* tables are stored. The default location for TokuDB's data files is the *MySQL* data directory.
- `tokudb_log_dir`: This variable defines the directory where the *TokuDB* log files are stored. The default location for TokuDB's log files is the *MySQL* data directory. Configuring a separate log directory is somewhat involved and should be done only if absolutely necessary. We recommend to keep the data and log files under the same directory.
- `tokudb_tmp_dir`: This variable defines the directory where the *TokuDB* bulk loader stores temporary files. The bulk loader can create large temporary files while it is loading a table, so putting these temporary files on a disk separate from the data directory can be useful. For example, it can make sense to use a high-performance disk for the data directory and a very inexpensive disk for the temporary directory. The default location for TokuDB's temporary files is the MySQL data directory.

Table Maintenance

Overview

The fractal tree provides fast performance by inserting small messages in the buffers in the fractal trees instead of requiring a potential IO for an update on every row in the table as required by a B-tree. Additional background information on how fractal trees operate can be found here. For tables whose workload pattern is a high number of sequential deletes, it may be beneficial to flush these delete messages down to the basement nodes in order to allow for faster access. The way to perform this operation is via the `OPTIMIZE` command.

The following extensions to the `OPTIMIZE` command have been added in *TokuDB* version 7.5.5:

- **Hot Optimize Throttling**

  By default, table optimization will run with all available resources. To limit the amount of resources, it is possible to limit the speed of table optimization. The `tokudb_optimize_throttle` session variable determines an upper bound on how many fractal tree leaf nodes per second are optimized. The default is 0 (no upper bound) with a valid range of [0,1000000]. For example, to limit the table optimization to 1 leaf node per second, use the following setting:

  ```
  SET tokudb_optimize_throttle=1;
  ```

- **Optimize a Single Index of a Table**

  To optimize a single index in a table, the `tokudb_optimize_index_name` session variable can be set to select the index by name. For example, to optimize the primary key of a table:

  ```
  SET tokudb_optimize_index_name='primary';
  OPTIMIZE TABLE t;
  ```

- **Optimize a Subset of a Fractal Tree Index**

  For patterns where the left side of the tree has many deletions (a common pattern with increasing id or date values), it may be useful to delete a percentage of the tree. In this case, it is possible to optimize a subset of a fractal tree starting at the left side. The `tokudb_optimize_index_fraction` session variable controls the size of the sub tree. Valid values are in the range [0.0,1.0] with default 1.0 (optimize the whole tree). For example, to optimize the leftmost 10% of the primary key:

  ```
  SET tokudb_optimize_index_name='primary';
  SET tokudb_optimize_index_fraction=0.1;
  OPTIMIZE TABLE t;
  ```

46.4 TokuDB Variables

Like all storage engines, *TokuDB* has variables to tune performance and control behavior. Fractal Tree algorithms are designed for near optimal performance and TokuDB's default settings should work well in most situations, eliminating the need for complex and time consuming tuning in most cases.

- Client Session Variables
- MySQL Server Variables

46.4.1 Client Session Variables

variable `unique_checks`

For tables with unique keys, every insertion into the table causes a lookup by key followed by an insertion, if the key is not in the table. This greatly limits insertion performance. If one knows by design that the rows being inserted into the table have unique keys, then one can disable the key lookup prior to insertion as follows:

```
SET unique_checks=OFF;
```

If your primary key is an auto-increment key, and none of your secondary keys are declared to be unique, then setting `unique_checks=OFF` will provide limited performance gains. On the other hand, if your primary key has a lot of entropy (it looks random), or your secondary keys are declared unique and have a lot of entropy, then disabling unique checks can provide a significant performance boost.

If `unique_checks` is disabled when the primary key is not unique, secondary indexes may become corrupted. In this case, the indexes should be dropped and rebuilt. This behavior differs from that of InnoDB, in which uniqueness is always checked on the primary key, and setting `unique_checks` to off turns off uniqueness checking on secondary indexes only. Turning off uniqueness checking on the primary key can provide large performance boosts, but it should only be done when the primary key is known to be unique.

variable `tokudb_commit_sync`

Session variable `tokudb_commit_sync` controls whether or not the transaction log is flushed when a transaction commits. The default behavior is that the transaction log is flushed by the commit. Flushing the transaction log requires a disk write and may adversely affect the performance of your application.

To disable synchronous flushing of the transaction log, disable the `tokudb_commit_sync` session variable as follows:

```
SET tokudb_commit_sync=OFF;
```

Disabling this variable may make the system run faster. However, transactions committed since the last checkpoint are not guaranteed to survive a crash.

variable `tokudb_pk_insert_mode`

This session variable controls the behavior of primary key insertions with the command `REPLACE INTO` and `INSERT IGNORE` on tables with no secondary indexes and on tables whose secondary keys whose every column is also a column of the primary key.

For instance, the table `(column_a INT, column_b INT, column_c INT, PRIMARY KEY (column_a,column_b), KEY (column_b))` is affected, because the only column in the key of `column_b` is present in the primary key. *TokuDB* can make these insertions really fast on these tables. However, triggers may not work and row based replication definitely will not work in this mode. This variable takes the following values, to control this behavior. This only applies to tables described above, using the command `REPLACE INTO` or `INSERT IGNORE`. All other scenarios are unaffected.

- `0`: Insertions are fast, regardless of whether triggers are defined on the table. `REPLACE INTO` and `INSERT IGNORE` statements fail if row based replication is enabled.
- `1` (default): Insertions are fast, if there are no triggers defined on the table. Insertions may be slow if triggers are defined on the table. `REPLACE INTO` and `INSERT IGNORE` statements fail if row based replication is enabled.
- `2`: Insertions are slow, all triggers on the table work, and row based replication works on `REPLACE INTO` and `INSERT IGNORE` statements.

variable `tokudb_load_save_space`

This session variable changes the behavior of the bulk loader. When it is disabled the bulk loader stores intermediate data using uncompressed files (which consumes additional CPU), whereas on compresses the intermediate files. It is enabled by default.

Note: The location of the temporary disk space used by the bulk loader may be specified with the `tokudb_tmp_dir` server variable.

If a `LOAD DATA INFILE` statement fails with the error message `ERROR 1030 (HY000): Got error 1` from storage engine, then there may not be enough disk space for the optimized loader, so disable `tokudb_prelock_empty` and try again.

More information is available in *Known Issues*.

variable `tokudb_prelock_empty`

By default, in 7.1.0, *TokuDB* preemptively grabs an entire table lock on empty tables. If one transaction is doing the loading, such as when the user is doing a table load into an empty table, this default provides a considerable speedup.

However, if multiple transactions try to do concurrent operations on an empty table, all but one transaction will be locked out. Disabling tokudb_prelock_empty optimizes for this multi-transaction case by turning off preemptive pre-locking.

Note: If this variable is set to off, fast bulk loading is turned off as well.

variable tokudb_create_index_online

This variable controls whether indexes created with the CREATE INDEX command are hot (if enabled), or offline (if disabled). Hot index creation means that the table is available for inserts and queries while the index is being created. Offline index creation means that the table is not available for inserts and queries while the index is being created.

Note: Hot index creation is slower than offline index creation.

By default, tokudb_create_index_online is enabled.

variable tokudb_disable_slow_alter

This variable controls whether slow alter tables are allowed. For example, the following command is slow because HCADER does not allow a mixture of column additions, deletions, or expansions:

```
ALTER TABLE table
ADD COLUMN column_a INT,
DROP COLUMN column_b;
```

By default, tokudb_disable_slow_alter is disabled, and the engine reports back to mysql that this is unsupported resulting in the following output:

```
ERROR 1112 (42000): Table 'test_slow' uses an extension that doesn't exist in this MySQL version
```

variable tokudb_block_size

Fractal tree internal and leaf nodes default to 4,194,304 bytes (4 MB). The session variable tokudb_block_size controls the target uncompressed size of these nodes.

Changing the value of tokudb_block_size only affects subsequently created tables. The value of this variable cannot be changed for an existing table without a dump and reload.

variable tokudb_read_block_size

Fractal tree leaves are subdivided into read blocks, in order to speed up point queries. The session variable tokudb_read_block_size controls the target uncompressed size of the read blocks. The units are bytes and the default is 65,536 (64 KB). A smaller value favors read performance for point and small range scans over large range scans and higher compression. The minimum value of this variable is 4096.

Changing the value of tokudb_read_block_size only affects subsequently created tables. The value of this variable cannot be changed for an existing table without a dump and reload.

variable tokudb_read_buf_size

This variable controls the size of the buffer used to store values that are bulk fetched as part of a large range query. Its unit is bytes and its default value is 131,072 (128 KB).

A value of 0 turns off bulk fetching. Each client keeps a thread of this size, so it should be lowered if situations where there are a large number of clients simultaneously querying a table.

variable tokudb_disable_prefetching

TokuDB attempts to aggressively prefetch additional blocks of rows, which is helpful for most range queries but may create unnecessary IO for range queries with `LIMIT` clauses. Prefetching is on by default, with a value of `0`, and can be disabled by setting this variable to `1`.

variable `tokudb_row_format`

This session variable controls the default compression algorithm used to compress data when no row format is specified in the `CREATE TABLE` command. See *Compression Details*.

variable `tokudb_lock_timeout_debug`

The following values are available:

0 No lock timeouts or lock deadlocks are reported.

1 A JSON document that describes the lock conflict is stored in the `tokudb_last_lock_timeout` session variable

2 A JSON document that describes the lock conflict is printed to the MySQL error log.

Supported since 7.5.5: In addition to the JSON document describing the lock conflict, the following lines are printed to the MySQL error log:

- A line containing the blocked thread id and blocked sql
- A line containing the blocking thread id and the blocking sql.

3 A JSON document that describes the lock conflict is stored in the `tokudb_last_lock_timeout` session variable and is printed to the MySQL error log.

Supported since 7.5.5: In addition to the JSON document describing the lock conflict, the following lines are printed to the MySQL error log:

- A line containing the blocked thread id and blocked sql
- A line containing the blocking thread id and the blocking sql.

variable `tokudb_last_lock_timeout`

This session variable contains a JSON document that describes the last lock conflict seen by the current MySQL client. It gets set when a blocked lock request times out or a lock deadlock is detected.

The `tokudb_lock_timeout_debug` session variable must have bit 0 set for this behavior, otherwise this session variable will be empty.

variable `tokudb_bulk_fetch`

This session variable determines if our bulk fetch algorithm is used for `SELECT` and `DELETE` statements. `SELECT` statements include pure `SELECT ...` statements, as well as `INSERT INTO table-name ... SELECT ...`, `CREATE TABLE table-name ... SELECT ...`, `REPLACE INTO table-name ... SELECT ...`, `INSERT IGNORE INTO table-name ... SELECT ...`, and `INSERT INTO table-name ... SELECT ... ON DUPLICATE KEY UPDATE`.

By default, `tokudb_bulk_fetch` is enabled.

variable `tokudb_support_xa`

This session variable defines whether or not the prepare phase of an XA transaction performs an `fsync()`.

By default, `tokudb_support_xa` is enabled.

variable `tokudb_optimize_throttling`
Supported since 7.5.5

By default, table optimization will run with all available resources. To limit the amount of resources, it is possible to limit the speed of table optimization. The `tokudb_optimize_throttling` session variable determines an upper bound on how many fractal tree leaf nodes per second are optimized. The default is 0 (no upper bound) with a valid range of [0,1000000].

variable `tokudb_optimize_index_name`
Supported since 7.5.5

To optimize a single index in a table, the `tokudb_optimize_index_name` session variable can be enabled to select the index by name.

variable `tokudb_optimize_index_fraction`
Supported since 7.5.5

For patterns where the left side of the tree has many deletions (a common pattern with increasing id or date values), it may be useful to delete a percentage of the tree. In this case, it's possible to optimize a subset of a fractal tree starting at the left side. The `tokudb_optimize_index_fraction` session variable controls the size of the sub tree. Valid values are in the range [0.0,1.0] with default 1.0 (optimize the whole tree).

variable `tokudb_backup_throttle`

This session level variable throttles the write rate in bytes per second of the backup to prevent Hot Backup from crowding out other jobs in the system. The default and max values are 18446744073709551615

variable `tokudb_backup_dir`
Supported since 7.5.5

When enabled, this session level variable serves two purposes, to point to the destination directory where the backups will be dumped and to kick off the backup as soon as it is set.

variable `tokudb_backup_exclude`

Command Line Yes

Config File Yes

Scope Global/Session

Dynamic Yes

Variable Type String

Default Value `(mysqld_safe\.pid)+`

Use this variable to set a regular expression that defines source files excluded from backup. For example, to exclude all `lost+found` directories, use the following command:

```
mysql> set tokudb_backup_exclude='/lost\\+found($|/)';
```

variable `tokudb_backup_last_error`
Supported since 7.5.5

This session variable will contain the error number from the last backup. 0 indicates success.

variable `tokudb_backup_last_error_string`
Supported since 7.5.5

This session variable will contain the error string from the last backup.

46.4.2 MySQL Server Variables

variable `tokudb_loader_memory_size`

Limits the amount of memory that the *TokuDB* bulk loader will use for each loader instance, defaults to 100 MB. Increasing this value may provide a performance benefit when loading extremely large tables with several secondary indexes.

Note: Memory allocated to a loader is taken from the TokuDB cache, defined as `tokudb_cache_size`, and may impact the running workload's performance as existing cached data must be ejected for the loader to begin.

---

variable `tokudb_fsync_log_period`

Controls the frequency, in milliseconds, for `fsync()` operations. If set to 0 then the `fsync()` behavior is only controlled by the `tokudb_commit_sync`, which is on or off. The default values is `0`.

variable `tokudb_cache_size`

This variable configures the size in bytes of the *TokuDB* cache table. The default cache table size is 1/2 of physical memory. Percona highly recommends using the default setting if using buffered IO, if using direct IO then consider setting this parameter to 80% of available memory.

Consider decreasing `tokudb_cache_size` if excessive swapping is causing performance problems. Swapping may occur when running multiple mysql server instances or if other running applications use large amounts of physical memory.

variable `tokudb_directio`

When enabled, *TokuDB* employs Direct IO rather than Buffered IO for writes. When using Direct IO, consider increasing `tokudb_cache_size` from its default of 1/2 physical memory.

By default, `tokudb_directio` is disabled.

variable `tokudb_lock_timeout`

This variable controls the amount of time that a transaction will wait for a lock held by another transaction to be released. If the conflicting transaction does not release the lock within the lock timeout, the transaction that was waiting for the lock will get a lock timeout error. The units are milliseconds. A value of 0 disables lock waits. The default value is 4000 (four seconds).

If your application gets a `lock wait timeout` error (-30994), then you may find that increasing the `tokudb_lock_timeout` may help. If your application gets a `deadlock found` error (-30995), then you need to abort the current transaction and retry it.

variable `tokudb_data_dir`

This variable configures the directory name where the *TokuDB* tables are stored. The default location is the *MySQL* data directory.

variable `tokudb_log_dir`

This variable specifies the directory where the *TokuDB* log files are stored. The default location is the MySQL data directory. Configuring a separate log directory is somewhat involved. Please contact Percona support for more details.

Warning: After changing *TokuDB* log directory path, the old *TokuDB* recovery log file should be moved to new directory prior to start of *MySQL* server and log file's owner must be the `mysql` user. Otherwise server will fail to initialize the *TokuDB* store engine restart.

variable `tokudb_tmp_dir`

This variable specifies the directory where the *TokuDB* bulk loader stores temporary files. The bulk loader can create large temporary files while it is loading a table, so putting these temporary files on a disk separate from the data directory can be useful.

For example, it can make sense to use a high-performance disk for the data directory and a very inexpensive disk for the temporary directory. The default location for temporary files is the *MySQL* data directory.

variable tokudb_checkpointing_period

This variable specifies the time in seconds between the beginning of one checkpoint and the beginning of the next. The default time between *TokuDB* checkpoints is 60 seconds. We recommend leaving this variable unchanged.

variable tokudb_write_status_frequency

TokuDB shows statement progress of queries, inserts, deletes, and updates in `SHOW PROCESSLIST`. Queries are defined as reads, and inserts, deletes, and updates are defined as writes.

Progress for updated is controlled by `tokudb_write_status_frequency`, which is set to 1000, that is, progress is measured every 1000 writes.

For slow queries, it can be helpful to set this variable and `tokudb_read_status_frequency` to 1, and then run `SHOW PROCESSLIST` several times to understand what progress is being made.

variable tokudb_read_status_frequency

TokuDB shows statement progress of queries, inserts, deletes, and updates in `SHOW PROCESSLIST`. Queries are defined as reads, and inserts, deletes, and updates are defined as writes.

Progress for reads is controlled by `tokudb_read_status_frequency` which is set to 10,000.

For slow queries, it can be helpful to set this variable and `tokudb_write_status_frequency` to 1, and then run `SHOW PROCESSLIST` several times to understand what progress is being made.

variable tokudb_fs_reserve_percent

This variable controls the percentage of the file system that must be available for inserts to be allowed. By default, this is set to 5. We recommend that this reserve be at least half the size of your physical memory. See *Full Disks* for more information.

variable tokudb_cleaner_period

This variable specifies how often in seconds the cleaner thread runs. The default value is 1. Setting this variable to 0 turns off cleaner threads.

variable tokudb_cleaner_iterations

This variable specifies how many internal nodes get processed in each `tokudb_cleaner_period` period. The default value is 5. Setting this variable to 0 turns off cleaner threads.

variable tokudb_backup_throttle

This variable specifies the maximum number of bytes per second the copier of a hot backup process will consume. Lowering its value will cause the hot backup operation to take more time but consume less IO on the server. The default value is `18446744073709551615`.

```
mysql> set tokudb_backup_throttle=1000000;
```

variable tokudb_rpl_lookup_rows

When disabled, *TokuDB* replication slaves skip row lookups for *delete row* log events and *update row* log events, which eliminates all associated read IO for these operations.

Warning: *TokuDB* Read Free Replication will not propagate `UPDATE` and `DELETE` events reliably if *TokuDB* table is missing the primary key which will eventually lead to data inconsistency on the slave.

Note: Optimization is only enabled when `read_only` is `1` and `binlog_format` is `ROW`.

By default, tokudb_rpl_lookup_rows is enabled.

variable tokudb_rpl_lookup_rows_delay

This server variable allows for simulation of long disk reads by sleeping for the given number of microseconds prior to the row lookup query, it should only be set to a non-zero value for testing.

By default, tokudb_rpl_lookup_rows_delay is disabled.

variable tokudb_rpl_unique_checks

When disabled, *TokuDB* replication slaves skip uniqueness checks on inserts and updates, which eliminates all associated read IO for these operations.

Note: Optimization is only enabled when read_only is 1 and binlog_format is ROW.

By default, tokudb_rpl_unique_checks is enabled.

variable tokudb_rpl_unique_checks_delay

This server variable allows for simulation of long disk reads by sleeping for the given number of microseconds prior to the row lookup query, it should only be set to a non-zero value for testing.

By default, tokudb_rpl_unique_checks_delay is disabled.

variable tokudb_backup_plugin_version
Supported since 7.5.5:

This server variable documents the version of the *TokuBackup* plugin

variable tokudb_backup_version``
Supported since 7.5.5:

This server variable documents the version of the hot backup library.

variable tokudb_backup_allowed_prefix
Supported since 7.5.5:

This system-level variable restricts the location of the destination directory where the backups can be located. Attempts to backup to a location outside of the directory this variable points to or its children will result in an error.

The default is null, backups have no restricted locations. This read only variable can be set in the my.cnf file and displayed with the SHOW VARIABLES command.

```
mysql> SHOW VARIABLES like 'tokudb_backup_allowed_prefix';
+------------------------------+-----------+
| Variable_name                | Value     |
+------------------------------+-----------+
| tokudb_backup_allowed_prefix | /dumpdir  |
+------------------------------+-----------+
```

variable tokudb_rpl_check_readonly
Supported since 7.5.5:

The *TokuDB* replication code will run row events from the binlog with RFR when the slave is in read only mode. The tokudb_rpl_check_readonly variable is used to disable the slave read only check in the *TokuDB* replication code.

This allows RFR to run when the slave is NOT read only. By default, tokudb_rpl_check_readonly is enabled (check slave read only). Do NOT change this value unless you completely understand the implications!

variable tokudb_fanout

Command Line Yes

Config File Yes

Scope Session/Global

Dynamic Yes

Variable Type Numeric

Range 2-16384

Default Value 16

This variable controls the Fractal Tree fanout.

variable `tokudb_client_pool_threads`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Numeric

Range 0 - 1024

Default Value 0

variable `tokudb_cachetable_pool_threads`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Numeric

Range 0 - 1024

Default Value 0

variable `tokudb_checkpoint_pool_threads`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Numeric

Range 0 - 1024

Default Value 0

variable `tokudb_enable_partial_eviction`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Boolean

Range ON/OFF

Default Value ON

This variable is used to control if partial eviction of nodes is enabled or disabled.

variable `tokudb_compress_buffers_before_eviction`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Boolean

Range ON/OFF

Default Value ON

When this variable is enabled it allows the evictor to compress unused internal node partitions in order to reduce memory requirements as a first step of partial eviction before fully evicting the partition and eventually the entire node.

variable `tokudb_strip_frm_data`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Boolean

Range ON/OFF

Default Value OFF

When this variable is set to `ON` during the startup server will check all the status files and remove the embedded `.frm` metadata. This variable can be used to assist in *TokuDB* data recovery. **WARNING:** Use this variable only if you know what you're doing otherwise it could lead to data loss.

46.5 Percona TokuBackup

Percona *TokuBackup* is an open-source hot backup utility for *MySQL* servers running the *TokuDB* storage engine (including *Percona Server* and *MariaDB*). It does not lock your database during backup. The *TokuBackup* library intercepts system calls that write files and duplicates the writes to the backup directory.

Note: This feature is currently considered *Experimental*

- Installing From Binaries
- Making a Backup
- Restoring From Backup
- Advanced Configuration
  - Monitoring Progress
  - Excluding Source Files
  - Throttling Backup Rate
  - Restricting Backup Target
  - Reporting Errors
- Limitations and known issues

46.5.1 Installing From Binaries

TokuBackup is included with *Percona Server* `5.7.10-1` and later versions. Installation can be performed with the `ps_tokudb_admin` script.

To install *Percona TokuBackup*:

1. Run `ps_tokudb_admin --enable-backup` to add the `preload-hotbackup` option into **[mysqld_safe]** section of `my.cnf`.

```
$ sudo ps_tokudb_admin --enable-backup
Checking SELinux status...
INFO: SELinux is disabled.

Checking if preload-hotbackup option is already set in config file...
INFO: Option preload-hotbackup is not set in the config file.

Checking TokuBackup plugin status...
INFO: TokuBackup plugin is not installed.

Adding preload-hotbackup option into /etc/my.cnf
INFO: Successfully added preload-hotbackup option into /etc/my.cnf
PLEASE RESTART MYSQL SERVICE AND RUN THIS SCRIPT AGAIN TO FINISH INSTALLATION!
```

2. Restart mysql service

```
$ sudo service mysql restart
```

3. Run `ps_tokudb_admin --enable-backup` again to finish installation of *TokuBackup* plugin

```
$ sudo ps_tokudb_admin --enable-backup
Checking SELinux status...
INFO: SELinux is disabled.

Checking if preload-hotbackup option is already set in config file...
INFO: Option preload-hotbackup is set in the config file.

Checking TokuBackup plugin status...
INFO: TokuBackup plugin is not installed.

Checking if Percona Server is running with libHotBackup.so preloaded...
INFO: Percona Server is running with libHotBackup.so preloaded.

Installing TokuBackup plugin...
INFO: Successfully installed TokuBackup plugin.
```

46.5.2 Making a Backup

To run *Percona TokuBackup*, the backup destination directory must exist, be writable and owned by the same user under which *MySQL* server is running (usually `mysql`) and empty. Once this directory is created, the backup can be run using the following command:

```
mysql> set tokudb_backup_dir='/path_to_empty_directory';
```

Note: Setting the `tokudb_backup_dir` variable automatically starts the backup process to the specified directory. Percona TokuBackup will take full backup each time, currently there is no incremental backup option

46.5.3 Restoring From Backup

Percona TokuBackup does not have any functionality for restoring a backup. You can use **rsync** or **cp** to restore the files. You should check that the restored files have the correct ownership and permissions.

Note: Make sure that the datadir is empty and that *MySQL* server is shut down before restoring from backup. You can't restore to a datadir of a running mysqld instance (except when importing a partial backup).

The following example shows how you might use the **rsync** command to restore the backup:

```
$ rsync -avrP /data/backup/ /var/lib/mysql/
```

Since attributes of files are preserved, in most cases you will need to change their ownership to *mysql* before starting the database server. Otherwise, the files will be owned by the user who created the backup.

```
$ chown -R mysql:mysql /var/lib/mysql
```

If you have changed default *TokuDB* data directory (`tokudb_data_dir`) or *TokuDB* log directory (`tokudb_log_dir`) or both of them, you will see separate folders for each setting in backup directory after taking backup. You'll need to restore each folder separately:

```
$ rsync -avrP /data/backup/mysql_data_dir/ /var/lib/mysql/
$ rsync -avrP /data/backup/tokudb_data_dir/ /path/to/original/tokudb_data_dir/
$ rsync -avrP /data/backup/tokudb_log_dir/ /path/to/original/tokudb_log_dir/
$ chown -R mysql:mysql /var/lib/mysql
$ chown -R mysql:mysql /path/to/original/tokudb_data_dir
$ chown -R mysql:mysql /path/to/original/tokudb_log_dir
```

46.5.4 Advanced Configuration

- Monitoring Progress
- Excluding Source Files
- Throttling Backup Rate
- Restricting Backup Target
- Reporting Errors

Monitoring Progress

TokuBackup updates the *PROCESSLIST* state while the backup is in progress. You can see the output by running `SHOW PROCESSLIST` or `SHOW FULL PROCESSLIST`.

Excluding Source Files

You can exclude certain files and directories based on a regular expression set in the tokudb_backup_exclude session variable. If the source file name matches the excluded regular expression, then the source file is excluded from backup.

For example, to exclude all lost+found directories from backup, use the following command:

```
mysql> SET tokudb_backup_exclude='/lost\\+found($|/)';
```

Note: In *Percona Server* 5.7.10-3 to address bug #125, server pid file is excluded by default. If you're providing your own additions to the exclusions and have the pid file in the default location, you will need to add the mysqld_safe.pid entry.

Throttling Backup Rate

You can throttle the backup rate using the tokudb_backup_throttle session-level variable. This variable throttles the write rate in bytes per second of the backup to prevent TokuBackup from crowding out other jobs in the system. The default and max value is 18446744073709551615.

```
mysql> SET tokudb_backup_throttle=1000000;
```

Restricting Backup Target

You can restrict the location of the destination directory where the backups can be located using the tokudb_backup_allowed_prefix system-level variable. Attempts to backup to a location outside of the specified directory or its children will result in an error.

The default is null, backups have no restricted locations. This read-only variable can be set in the my.cnf configuration file and displayed with the SHOW VARIABLES command:

```
mysql> SHOW VARIABLES LIKE 'tokudb_backup_allowed_prefix';
+------------------------------+-----------+
| Variable_name                | Value     |
+------------------------------+-----------+
| tokudb_backup_allowed_prefix | /dumpdir  |
+------------------------------+-----------+
```

Reporting Errors

Percona TokuBackup uses two variables to capture errors. They are tokudb_backup_last_error and tokudb_backup_last_error_string. When *TokuBackup* encounters an error, these will report on the error number and the error string respectively. For example, the following output shows these parameters following an attempted backup to a directory that was not empty:

```
mysql> SET tokudb_backup_dir='/tmp/backupdir';
ERROR 1231 (42000): Variable 'tokudb_backup_dir' can't be set to the value of '/tmp/backupdir'

mysql> SELECT @@tokudb_backup_last_error;
+----------------------------+
| @@tokudb_backup_last_error |
+----------------------------+
|                         17 |
+----------------------------+
```

```
mysql> SELECT @@tokudb_backup_last_error_string;
+-------------------------------------------------+
| @@tokudb_backup_last_error_string               |
+-------------------------------------------------+
| tokudb backup couldn't create needed directories. |
+-------------------------------------------------+
```

46.5.5 Limitations and known issues

- You must disable *InnoDB* asynchronous IO if backing up *InnoDB* tables with *TokuBackup*. Otherwise you will have inconsistent, unrecoverable backups. The appropriate setting is `innodb_use_native_aio=0`.
- To be able to run Point-In-Time-Recovery you'll need to manually get the binary log position.
- Transactional storage engines (*TokuDB* and *InnoDB*) will perform recovery on the backup copy of the database when it is first started.
- Tables using non-transactional storage engines (*MyISAM*) are not locked during the copy and may report issues when starting up the backup. It is best to avoid operations that modify these tables at the end of a hot backup operation (adding/changing users, stored procedures, etc.).
- The database is copied locally to the path specified in `/path/to/backup`. This folder must exist, be writable, be empty, and contain enough space for a full copy of the database.
- *TokuBackup* always makes a backup of the *MySQL* `datadir` and optionally the `tokudb_data_dir`, `tokudb_log_dir`, and the binary log folder. The latter three are only backed up separately if they are not the same as or contained in the *MySQL* `datadir`. None of these three folders can be a parent of the *MySQL* `datadir`.
- No other directory structures are supported. All *InnoDB*, *MyISAM*, and other storage engine files must be within the *MySQL* `datadir`.
- *TokuBackup* does not follow symbolic links.
- *TokuBackup* does not backup *MySQL* configuration file(s).
- *TokuBackup* does not backup tablespaces if they are out of `datadir`.
- Due to upstream bug #80183, *TokuBackup* can't recover backed-up table data if backup was taken while running `OPTIMIZE TABLE` or `ALTER TABLE ... TABLESPACE`.
- *TokuBackup* doesn't support incremental backups.

46.6 TokuDB Troubleshooting

- Known Issues
- Lock Visualization in TokuDB
- Engine Status
- Global Status

46.6.1 Known Issues

Replication and binary logging: *TokuDB* supports binary logging and replication, with one restriction. *TokuDB* does not implement a lock on the auto-increment function, so concurrent insert statements with one or more of the

statements inserting multiple rows may result in a non-deterministic interleaving of the auto-increment values. When running replication with these concurrent inserts, the auto-increment values on the slave table may not match the auto-increment values on the master table. Note that this is only an issue with Statement Based Replication (SBR), and not Row Based Replication (RBR).

For more information about auto-increment and replication, see the *MySQL* Reference Manual: AUTO_INCREMENT handling in InnoDB.

In addition, when using the `REPLACE INTO` or `INSERT IGNORE` on tables with no secondary indexes or tables where secondary indexes are subsets of the primary, the session variable `tokudb_pk_insert_mode` controls whether row based replication will work.

Uninformative error message: The `LOAD DATA INFILE` command can sometimes produce `ERROR 1030 (HY000): Got error 1 from storage engine`. The message should say that the error is caused by insufficient disk space for the temporary files created by the loader.

Transparent Huge Pages: *TokuDB* will refuse to start if transparent huge pages are enabled. Transparent huge page support can be disabled by issuing the following as root:

```
# echo never > /sys/kernel/mm/redhat_transparent_hugepage/enabled
```

Note: The previous command needs to be executed after every reboot, because it defaults to `always`.

XA behavior vs. InnoDB: *InnoDB* forces a deadlocked XA transaction to abort, *TokuDB* does not.

46.6.2 Lock Visualization in TokuDB

TokuDB uses key range locks to implement serializable transactions, which are acquired as the transaction progresses. The locks are released when the transaction commits or aborts (this implements two phase locking).

TokuDB stores these locks in a data structure called the lock tree. The lock tree stores the set of range locks granted to each transaction. In addition, the lock tree stores the set of locks that are not granted due to a conflict with locks granted to some other transaction. When these other transactions are retired, these pending lock requests are retried. If a pending lock request is not granted before the lock timer expires, then the lock request is aborted.

Lock visualization in *TokuDB* exposes the state of the lock tree with tables in the information schema. We also provide a mechanism that may be used by a database client to retrieve details about lock conflicts that it encountered while executing a transaction.

The `TOKUDB_TRX` table

The `TOKUDB_TRX` table in the `INFORMATION_SCHEMA` maps *TokuDB* transaction identifiers to *MySQL* client identifiers. This mapping allows one to associate a *TokuDB* transaction with a *MySQL* client operation.

The following query returns the *MySQL* clients that have a live *TokuDB* transaction:

```
SELECT * FROM INFORMATION_SCHEMA.TOKUDB_TRX,
  INFORMATION_SCHEMA.PROCESSLIST
  WHERE trx_mysql_thread_id = id;
```

The `TOKUDB_LOCKS` table

The `tokudb_locks` table in the information schema contains the set of locks granted to *TokuDB* transactions.

The following query returns all of the locks granted to some *TokuDB* transaction:

```
SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCKS;
```

The following query returns the locks granted to some *MySQL* client:

```
SELECT id FROM INFORMATION_SCHEMA.TOKUDB_LOCKS,
  INFORMATION_SCHEMA.PROCESSLIST
  WHERE locks_mysql_thread_id = id;
```

The `TOKUDB_LOCK_WAITS` table

The `tokudb_lock_waits` table in the information schema contains the set of lock requests that are not granted due to a lock conflict with some other transaction.

The following query returns the locks that are waiting to be granted due to a lock conflict with some other transaction:

```
SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCK_WAITS;
```

The `tokudb_lock_timeout_debug` session variable

The `tokudb_lock_timeout_debug` session variable controls how lock timeouts and lock deadlocks seen by the database client are reported.

The following values are available:

0 No lock timeouts or lock deadlocks are reported.

1 A JSON document that describes the lock conflict is stored in the `tokudb_last_lock_timeout` session variable

2 A JSON document that describes the lock conflict is printed to the *MySQL* error log.

Supported since 7.5.5: In addition to the JSON document describing the lock conflict, the following lines are printed to the MySQL error log:

- A line containing the blocked thread id and blocked SQL
- A line containing the blocking thread id and the blocking SQL.

3 A JSON document that describes the lock conflict is stored in the `tokudb_last_lock_timeout` session variable and is printed to the *MySQL* error log.

Supported since 7.5.5: In addition to the JSON document describing the lock conflict, the following lines are printed to the *MySQL* error log:

- A line containing the blocked thread id and blocked SQL
- A line containing the blocking thread id and the blocking SQL.

The `tokudb_last_lock_timeout` session variable

The `tokudb_last_lock_timeout` session variable contains a JSON document that describes the last lock conflict seen by the current *MySQL* client. It gets set when a blocked lock request times out or a lock deadlock is detected. The `tokudb_lock_timeout_debug` session variable should have bit `0` set (decimal `1`).

Example

Suppose that we create a table with a single column that is the primary key.

```
mysql> SHOW CREATE TABLE table;

Create Table: CREATE TABLE `table` (
`id` int(11) NOT NULL,
PRIMARY KEY (`id`)) ENGINE=TokuDB DEFAULT CHARSET=latin1
```

Suppose that we have 2 *MySQL* clients with ID's 1 and 2 respectively. Suppose that *MySQL* client 1 inserts some values into `table`. *TokuDB* transaction 51 is created for the insert statement. Since autocommit is disabled, transaction 51 is still live after the insert statement completes, and we can query the `tokudb_locks` table in information schema to see the locks that are held by the transaction.

```
mysql> SET AUTOCOMMIT=OFF;
mysql> INSERT INTO table VALUES (1),(10),(100);

Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCKS;

+--------------+-----------------------+---------------+----------------+-----------------+----------
| locks_trx_id | locks_mysql_thread_id | locks_dname   | locks_key_left | locks_key_right | locks_tab
+--------------+-----------------------+---------------+----------------+-----------------+----------
|           51 |                     1 | ./test/t-main | 0001000000     | 0001000000      | test
|           51 |                     1 | ./test/t-main | 000a000000     | 000a000000      | test
|           51 |                     1 | ./test/t-main | 0064000000     | 0064000000      | test
+--------------+-----------------------+---------------+----------------+-----------------+----------

mysql> SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCK_WAITS;

Empty set (0.00 sec)
```

The keys are currently hex dumped.

Now we switch to the other *MySQL* client with ID 2.

```
mysql> INSERT INTO table VALUES (2),(20),(100);
```

The insert gets blocked since there is a conflict on the primary key with value 100.

The granted *TokuDB* locks are:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCKS;

+--------------+-----------------------+---------------+----------------+-----------------+----------
| locks_trx_id | locks_mysql_thread_id | locks_dname   | locks_key_left | locks_key_right | locks_tab
+--------------+-----------------------+---------------+----------------+-----------------+----------
|           51 |                     1 | ./test/t-main | 0001000000     | 0001000000      | test
|           51 |                     1 | ./test/t-main | 000a000000     | 000a000000      | test
|           51 |                     1 | ./test/t-main | 0064000000     | 0064000000      | test
|           51 |                     1 | ./test/t-main | 0002000000     | 0002000000      | test
|           51 |                     1 | ./test/t-main | 0014000000     | 0014000000      | test
+--------------+-----------------------+---------------+----------------+-----------------+----------
```

The locks that are pending due to a conflict are:

```
SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCK_WAITS;

+------------------+-----------------+-----------------+---------------------+---------------------
| requesting_trx_id | blocking_trx_id | lock_waits_dname | lock_waits_key_left | lock_waits_key_right
+------------------+-----------------+-----------------+---------------------+---------------------
|                62 |              51 | ./test/t-main    | 0064000000          | 0064000000
+------------------+-----------------+-----------------+---------------------+---------------------
```

Eventually, the lock for client 2 times out, and we can retrieve a JSON document that describes the conflict.

```
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction

mysql> SELECT @@TOKUDB_LAST_LOCK_TIMEOUT;

+-----------------------------------------------------------------------------------------------------------
| @@tokudb_last_lock_timeout
+-----------------------------------------------------------------------------------------------------------
| "mysql_thread_id":2, "dbname":"./test/t-main", "requesting_txnid":62, "blocking_txnid":51, "key":"0
+-----------------------------------------------------------------------------------------------------------

ROLLBACK;
```

Since transaction 62 was rolled back, all of the locks taken by it are released.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCKS;

+-------------+----------------------+--------------+----------------+-----------------+----------
| locks_trx_id | locks_mysql_thread_id | locks_dname   | locks_key_left | locks_key_right | locks_tab
+-------------+----------------------+--------------+----------------+-----------------+----------
|           51 |                    1 | ./test/t-main | 0001000000     | 0001000000      | test
|           51 |                    1 | ./test/t-main | 000a000000     | 000a000000      | test
|           51 |                    1 | ./test/t-main | 0064000000     | 0064000000      | test
|           51 |                    2 | ./test/t-main | 0002000000     | 0002000000      | test
|           51 |                    2 | ./test/t-main | 0014000000     | 0014000000      | test
+-------------+----------------------+--------------+----------------+-----------------+----------
```

46.6.3 Engine Status

Engine status provides details about the inner workings of *TokuDB* and can be useful in tuning your particular environment. The engine status can be determined by running the following command:

```
SHOW ENGINE tokudb STATUS;
```

The following is a reference of table status statements:

cachetable: cleaner executions Total number of times the cleaner thread loop has executed.

cachetable: cleaner iterations This is the number of cleaner operations that are performed every cleaner period.

cachetable: cleaner period *TokuDB* includes a cleaner thread that optimizes indexes in the background. This variable is the time, in seconds, between the completion of a group of cleaner operations and the beginning of the next group of cleaner operations. The cleaner operations run on a background thread performing work that does not need to be done on the client thread.

cachetable: evictions Number of blocks evicted from cache.

cachetable: long time waiting on cache pressure Total time, in microseconds, waiting on cache pressure to subside for more than 1 second.

cachetable: miss This is a count of how many times the application was unable to access your data in the internal cache.

cachetable: miss time This is the total time, in microseconds, of how long the database has had to wait for a disk read to complete.

cachetable: number of long waits on cache pressure The number of times a thread was stalled for more than 1 second due to cache pressure.

cachetable: number of waits on cache pressure The number of times a thread was stalled due to cache pressure.

cachetable: prefetches This is the total number of times that a block of memory has been prefetched into the database's cache. Data is prefetched when the database's algorithms determine that a block of memory is likely to be accessed by the application.

cachetable: size cachepressure The number of bytes causing cache pressure (the sum of buffers and work done counters), helps to understand if cleaner threads are keeping up with workload.

cachetable: size current This is a count, in bytes, of how much of your uncompressed data is currently in the database's internal cache.

cachetable: size currently cloned data for checkpoint Amount of memory, in bytes, currently used for cloned nodes. During the checkpoint operation, dirty nodes are cloned prior to serialization/compression, then written to disk. After which, the memory for the cloned block is returned for re-use.

cachetable: size leaf The number of bytes of leaf nodes in the cache.

cachetable: size limit This is a count, in bytes, of how much of your uncompressed data will fit in the database's internal cache.

cachetable: size nonleaf The number of bytes of non-leaf nodes in the cache.

cachetable: size rollback The number of bytes of rollback nodes in the cache.

cachetable: size writing This is the number of bytes that are currently queued up to be written to disk.

cachetable: time waiting on cache pressure Total time, in microseconds, waiting on cache pressure to subside.

checkpoint: begin time Cumulative time (in microseconds) required to mark all dirty nodes as pending a checkpoint.

checkpoint: checkpoints failed This is the number of checkpoints that have failed for any reason.

checkpoint: checkpoints taken This is the number of complete checkpoints that have been taken.

checkpoint: footprint Where the database is in the checkpoint process.

checkpoint: last checkpoint began This is the time the last checkpoint began. If a checkpoint is currently in progress, then this time may be later than the time the last checkpoint completed.

Note: If no checkpoint has ever taken place, then this value will be `Dec 31, 1969` on Linux hosts.

---

checkpoint: last complete checkpoint began This is the time the last complete checkpoint started. Any data that changed after this time will not be captured in the checkpoint.

checkpoint: last complete checkpoint ended This is the time the last complete checkpoint ended.

checkpoint: last complete checkpoint LSN This is the Log Sequence Number of the last complete checkpoint.

checkpoint: long checkpoint begin count The total number of times a checkpoint begin took more than 1 second.

checkpoint: long checkpoint begin time The total time, in microseconds, of long checkpoint begins. A long checkpoint begin is one taking more than 1 second.

checkpoint: non-checkpoint client wait on cs lock The number of times a non-checkpoint client thread waited for the checkpoint-safe lock.

checkpoint: non-checkpoint client wait on mo lock The number of times a non-checkpoint client thread waited for the multi-operation lock.

checkpoint: period This is the interval in seconds between the end of an automatic checkpoint and the beginning of the next automatic checkpoint.

checkpoint: time spent during checkpoint (begin and end phases) Time (in seconds) required to complete all checkpoints.

checkpoint: time spent during last checkpoint (begin and end phases) Time (in seconds) required to complete the last checkpoint.

checkpoint: waiters max This is the maximum number of threads ever simultaneously waiting for the checkpoint-safe lock to perform a checkpoint.

checkpoint: waiters now This is the current number of threads simultaneously waiting for the checkpoint-safe lock to perform a checkpoint.

checkpoint: checkpoint end time The time spent in checkpoint end operation in seconds.

checkpoint: long checkpoint end time The time spent in checkpoint end operation in seconds.

checkpoint: long checkpoint end count This is the count of end_checkpoint operations that exceeded 1 minute.

context: promotion blocked by a flush Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a buffer flush from parent to child.

context: promotion blocked by a full eviction (should never happen) Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a full eviction.

context: promotion blocked by a full fetch (should never happen) Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a full fetch.

context: promotion blocked by a message application Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of message application (applying fresh ancestors messages to a basement node).

context: promotion blocked by a message injection Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of message injection.

context: promotion blocked by a partial eviction (should never happen) Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a partial eviction.

context: promotion blocked by a partial fetch (should never happen) Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a partial fetch.

context: promotion blocked by something uninstrumented Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of something uninstrumented.

context: promotion blocked by the cleaner thread Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a cleaner thread.

context: something uninstrumented blocked by something uninstrumented Number of times node `rwlock` contention was observed for an uninstrumented process because of something uninstrumented.

context: tree traversals blocked by a flush Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a buffer flush from parent to child.

context: tree traversals blocked by a full eviction Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a full eviction.

context: tree traversals blocked by a full fetch Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a full fetch.

context: tree traversals blocked by a message application Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of message application (applying fresh ancestors messages to a basement node).

context: tree traversals blocked by a message injection Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of message injection.

context: tree traversals blocked by a partial eviction Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a partial eviction.

context: tree traversals blocked by a partial fetch Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a partial fetch.

context: tree traversals blocked by a the cleaner thread Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a cleaner thread.

context: tree traversals blocked by something uninstrumented Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of something uninstrumented.

db closes Number of db close operations.

db opens Number of db open operations.

dictionary broadcast updates This is the number of broadcast updates that have been successfully performed. A broadcast update is an update that affects all rows in a dictionary.

dictionary broadcast updates fail This is the number of broadcast updates that have failed.

dictionary deletes This is the total number of rows that have been deleted from all primary and secondary indexes combined, if those deletes have been done with a separate recovery log entry per index.

dictionary deletes fail This is the number of single-index delete operations that failed.

dictionary inserts This is the total number of rows that have been inserted into all primary and secondary indexes combined, when those inserts have been done with a separate recovery log entry per index. For example, inserting a row into a table with one primary and two secondary indexes will increase this count by three, if the inserts were done with separate recovery log entries.

dictionary inserts fail This is the number of single-index insert operations that failed.

dictionary multi deletes This is the total number of rows that have been deleted from all primary and secondary indexes combined, when those deletes have been done with a single recovery log entry for the entire row.

dictionary multi deletes fail This is the number of multi-index delete operations that failed.

dictionary multi inserts This is the total number of rows that have been inserted into all primary and secondary indexes combined, when those inserts have been done with a single recovery log entry for the entire row. (For example, inserting a row into a table with one primary and two secondary indexes will normally increase this count by three).

dictionary multi inserts fail This is the number of multi-index insert operations that failed.

dictionary multi updates This is the total number of rows that have been updated in all primary and secondary indexes combined, if those updates have been done with a single recovery log entry for the entire row.

dictionary multi updates fail This is the number of multi-index update operations that failed.

dictionary updates This is the total number of rows that have been updated in all primary and secondary indexes combined, if those updates have been done with a separate recovery log entry per index.

dictionary updates fail This is the number of single-index update operations that failed.

disk free space This is a gross estimate of how much of your file system is available. Possible displays in this field are:

- More than twice the reserve ("more than 10 percent of total file system space")
- Less than twice the reserve
- Less than the reserve
- File system is completely full

filesystem: ENOSPC redzone state The state of how much disk space exists with respect to the red zone value. Valid values are:

0 Space is available

1 Warning, with 2x of redzone value. Operations are allowed, but engine status prints a warning.

2 In red zone, insert operations are blocked

3 All operations are blocked

filesystem: fsync count This is the total number of times the database has flushed the operating system's file buffers to disk.

filesystem: fsync time This the total time, in microseconds, used to fsync to disk.

filesystem: long fsync count This is the total number of times the database has flushed the operating system's file buffers to disk and this operation required more than 1 second.

filesystem: long fsync time This the total time, in microseconds, used to fsync to disk when the operation required more than 1 second.

filesystem: most recent disk full This is the most recent time when the disk file system was entirely full. If the disk has never been full, then this value will be "Dec 31, 1969" on Linux hosts.

filesystem: number of operations rejected by enospc prevention (red zone) This is the number of database inserts that have been rejected because the amount of disk free space was less than the reserve.

filesystem: number of write operations that returned ENOSPC This is the number of times that an attempt to write to disk failed because the disk was full. If the disk is full, this number will continue increasing until space is available.

filesystem: threads currently blocked by full disk This is the number of threads that are currently blocked because they are attempting to write to a full disk. This is normally zero. If this value is non-zero, then a warning will appear in the "disk free space" field.

ft: basements decompressed as a target of a query Number of basement nodes decompressed for queries.

ft: basements decompressed for prefetch Number of basement nodes decompressed by a prefetch thread.

ft: basements decompressed for prelocked range Number of basement nodes decompressed by queries aggressively.

ft: basements decompressed for write Number of basement nodes decompressed for writes.

ft: basement nodes deserialized with fixed-keysize The number of basement nodes deserialized where all keys had the same size, leaving the basement in a format that is optimal for in-memory workloads.

ft: basement nodes deserialized with variable-keysize The number of basement nodes deserialized where all keys did not have the same size, and thus ineligible for an in-memory optimization.

ft: basements fetched as a target of a query (bytes) Number of basement node bytes fetched from disk for queries.

ft: basements fetched as a target of a query Number of basement nodes fetched from disk for queries.

ft: basements fetched as a target of a query (seconds) Number of seconds waiting for IO when fetching basement nodes from disk for queries.

ft: basements fetched for prefetch (bytes) Number of basement node bytes fetched from disk by a prefetch thread.

ft: basements fetched for prefetch Number of basement nodes fetched from disk by a prefetch thread.

ft: basements fetched for prefetch (seconds) Number of seconds waiting for IO when fetching basement nodes from disk by a prefetch thread.

ft: basements fetched for prelocked range (bytes) Number of basement node bytes fetched from disk aggressively.

ft: basements fetched for prelocked range Number of basement nodes fetched from disk aggressively.

ft: basements fetched for prelocked range (seconds) Number of seconds waiting for IO when fetching basement nodes from disk aggressively.

ft: basements fetched for write (bytes) Number of basement node bytes fetched from disk for writes.

ft: basements fetched for write Number of basement nodes fetched from disk for writes.

ft: basements fetched for write (seconds) Number of seconds waiting for IO when fetching basement nodes from disk for writes.

ft: broadcast messages injected at root How many broadcast messages injected at root.

ft: buffers decompressed as a target of a query Number of buffers decompressed for queries.

ft: buffers decompressed for prefetch Number of buffers decompressed by a prefetch thread.

ft: buffers decompressed for prelocked range Number of buffers decompressed by queries aggressively.

ft: buffers decompressed for write Number of buffers decompressed for writes.

ft: buffers fetched as a target of a query (bytes) Number of buffer bytes fetched from disk for queries.

ft: buffers fetched as a target of a query Number of buffers fetched from disk for queries.

ft: buffers fetched as a target of a query (seconds) Number of seconds waiting for IO when fetching buffers from disk for queries.

ft: buffers fetched for prefetch (bytes) Number of buffer bytes fetched from disk by a prefetch thread.

ft: buffers fetched for prefetch Number of buffers fetched from disk by a prefetch thread.

ft: buffers fetched for prefetch (seconds) Number of seconds waiting for IO when fetching buffers from disk by a prefetch thread.

ft: buffers fetched for prelocked range (bytes) Number of buffer bytes fetched from disk aggressively.

ft: buffers fetched for prelocked range Number of buffers fetched from disk aggressively.

ft: buffers fetched for prelocked range (seconds) Number of seconds waiting for IO when fetching buffers from disk aggressively.

ft: buffers fetched for write (bytes) Number of buffer bytes fetched from disk for writes.

ft: buffers fetched for write Number of buffers fetched from disk for writes.

ft: buffers fetched for write (seconds) Number of seconds waiting for IO when fetching buffers from disk for writes.

ft: bytes of messages currently in trees (estimate) How many bytes of messages currently in trees (estimate).

ft: bytes of messages flushed from h1 nodes to leaves How many bytes of messages flushed from h1 nodes to leaves.

ft: bytes of messages injected at root (all trees) How many bytes of messages injected at root (for all trees).

ft: descriptor set This is the number of time a descriptor was updated when the entire dictionary was updated (for example, when the schema has been changed).

ft: leaf compression to memory (seconds) Total time, in seconds, spent compressing leaf nodes.

ft: leaf decompression to memory (seconds) Total time, in seconds, spent decompressing leaf nodes.

ft: leaf deserialization to memory (seconds) Total time, in seconds, spent deserializing leaf nodes.

ft: leaf node full evictions (bytes) The number of bytes freed by evicting full leaf nodes from the cache.

ft: leaf node full evictions The number of times a full leaf node was evicted from the cache.

ft: leaf node partial evictions (bytes) The number of bytes freed by evicting partitions of leaf nodes from the cache.

ft: leaf node partial evictions The number of times a partition of a leaf node was evicted from the cache.

ft: leaf nodes created Number of leaf nodes created.

ft: leaf nodes destroyed Number of leaf nodes destroyed.

ft: leaf nodes flushed to disk (for checkpoint) (bytes) Number of bytes of leaf nodes flushed to disk for checkpoint.

ft: leaf nodes flushed to disk (for checkpoint) Number of leaf nodes flushed to disk for checkpoint.

ft: leaf nodes flushed to disk (for checkpoint) (seconds) Number of seconds waiting for IO when writing leaf nodes flushed to disk for checkpoint.

ft: leaf nodes flushed to disk (for checkpoint) (uncompressed bytes) Number of uncompressed bytes of leaf nodes flushed to disk for checkpoint.

ft: leaf nodes flushed to disk (not for checkpoint) (bytes) Number of bytes of leaf nodes flushed to disk, not for checkpoint.

ft: leaf nodes flushed to disk (not for checkpoint) Number of leaf nodes flushed to disk, not for checkpoint.

ft: leaf nodes flushed to disk (not for checkpoint) (seconds) Number of seconds waiting for IO when writing leaf nodes flushed to disk, not for checkpoint.

ft: leaf nodes flushed to disk (not for checkpoint) (uncompressed bytes) Number of bytes of leaf nodes flushed to disk, not for checkpoint.

ft: leaf serialization to memory (seconds) Total time, in seconds, spent serializing leaf nodes.

ft: messages ignored by leaf due to msn The number of messages that were ignored by a leaf because it had already been applied.

ft: messages injected at root How many messages injected at root.

ft: nonleaf compression to memory (seconds) Total time, in seconds, spent compressing non leaf nodes.

ft: nonleaf decompression to memory (seconds) Total time, in seconds, spent decompressing non leaf nodes.

ft: nonleaf deserialization to memory (seconds) Total time, in seconds, spent deserializing non leaf nodes.

ft: nonleaf node full evictions (bytes) The number of bytes freed by evicting full nonleaf nodes from the cache.

ft: nonleaf node full evictions The number of times a full nonleaf node was evicted from the cache.

ft: nonleaf node partial evictions (bytes) The number of bytes freed by evicting partitions of nonleaf nodes from the cache.

ft: nonleaf node partial evictions The number of times a partition of a nonleaf node was evicted from the cache.

ft: nonleaf nodes created Number of nonleaf nodes created.

ft: nonleaf nodes destroyed Number of nonleaf nodes destroyed.

ft: nonleaf nodes flushed to disk (for checkpoint) (bytes) Number of bytes of nonleaf nodes flushed to disk for checkpoint.

ft: nonleaf nodes flushed to disk (for checkpoint) Number of nonleaf nodes flushed to disk for checkpoint.

ft: nonleaf nodes flushed to disk (for checkpoint) (seconds) Number of seconds waiting for IO when writing nonleaf nodes flushed to disk for checkpoint.

ft: nonleaf nodes flushed to disk (for checkpoint) (uncompressed bytes) Number of uncompressed bytes of nonleaf nodes flushed to disk for checkpoint.

ft: nonleaf nodes flushed to disk (not for checkpoint) (bytes) Number of bytes of nonleaf nodes flushed to disk, not for checkpoint.

ft: nonleaf nodes flushed to disk (not for checkpoint) Number of nonleaf nodes flushed to disk, not for checkpoint.

ft: nonleaf nodes flushed to disk (not for checkpoint) (seconds) Number of seconds waiting for IO when writing nonleaf nodes flushed to disk, not for check- point.

ft: nonleaf nodes flushed to disk (not for checkpoint) (uncompressed bytes) Number of uncompressed bytes of nonleaf nodes flushed to disk, not for checkpoint.

ft: nonleaf serialization to memory (seconds) Total time, in seconds, spent serializing non leaf nodes.

ft: pivots fetched for prefetch (bytes) Number of bytes of pivot nodes fetched by a prefetch thread.

ft: pivots fetched for prefetch Number of pivot nodes fetched by a prefetch thread.

ft: pivots fetched for prefetch (seconds) Number seconds waiting for IO when fetching pivot nodes by a prefetch thread.

ft: pivots fetched for query (bytes) Number of bytes of pivot nodes fetched for queries.

ft: pivots fetched for query Number of pivot nodes fetched for queries.

ft: pivots fetched for query (seconds) Number of seconds waiting for IO when fetching pivot nodes for queries.

ft: pivots fetched for write (bytes) Number of bytes of pivot nodes fetched for writes.

ft: pivots fetched for write Number of pivot nodes fetched for writes.

ft: pivots fetched for write (seconds) Number of seconds waiting for IO when fetching pivot nodes for writes.

ft: promotion: h1 roots injected into Number of times a message stopped at a root with height 1.

ft: promotion: injections at depth 0 Number of times a message stopped at depth 0.

ft: promotion: injections at depth 1 Number of times a message stopped at depth 1.

ft: promotion: injections at depth 2 Number of times a message stopped at depth 2.

ft: promotion: injections at depth 3 Number of times a message stopped at depth 3.

ft: promotion: injections lower than depth 3 Number of times a message was promoted past depth 3.

ft: promotion: leaf roots injected into Number of times a message stopped at a root with height 0.

ft: promotion: roots split Number of times the root split during promotion.

ft: promotion: stopped anyway, after locking the child Number of times a message stopped before a child which had been locked.

ft: promotion: stopped at height 1 Number of times a message stopped because it had reached height 1.

ft: promotion: stopped because of a nonempty buffer Number of times a message stopped because it reached a nonempty buffer.

ft: promotion: stopped because the child was locked or not at all in memory Number of times a message stopped because it could not cheaply get access to a child.

ft: promotion: stopped because the child was not fully in memory Number of times a message stopped because it could not cheaply get access to a child.

ft: promotion: succeeded in using the rightmost leaf shortcut Rightmost insertions used the rightmost-leaf pin path, meaning that the Fractal Tree index detected and properly optimized rightmost inserts.

ft: promotion: tried the rightmost leaf shortcut but failed (child reactive) Rightmost insertions did not use the rightmost-leaf pin path, due to the leaf being too large (needed to split).

ft: promotion: tried the rightmost leaf shortcut but failed (out-of-bounds) Rightmost insertions did not use the rightmost-leaf pin path, due to the insert not actually being into the rightmost leaf node.

ft: searches requiring more tries than the height of the tree Number of searches that required more tries than the height of the tree.

ft: searches requiring more tries than the height of the tree plus three Number of searches that required more tries than the height of the tree plus three.

ft: total search retries due to TRY AGAIN Total number of search retries due to TRY AGAIN.

ft: uncompressed / compressed bytes written (leaf) Ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for leaf nodes.

ft: uncompressed / compressed bytes written (nonleaf) Ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for nonleaf nodes.

ft: uncompressed / compressed bytes written (overall) Ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for all nodes.

ft flusher: cleaner thread leaf merges in progress The number of cleaner thread leaf merges in progress.

ft flusher: cleaner thread leaf merges successful The number of times the cleaner thread successfully merges a leaf.

ft flusher: height-greater-than-one nodes flushed by cleaner thread Number of nodes of height > 1 whose message buffers are flushed by cleaner thread.

ft flusher: height-one nodes flushed by cleaner thread Number of nodes of height one whose message buffers are flushed by cleaner thread.

ft flusher: leaf node balances Number of times a leaf node is balanced.

ft flusher: leaf node merges Number of times leaf nodes are merged.

ft flusher: leaf node splits Number of leaf nodes split.

ft flusher: max bytes in a buffer flushed by cleaner thread Max number of bytes in message buffer flushed by cleaner thread.

ft flusher: max workdone in a buffer flushed by cleaner thread Max workdone value of any message buffer flushed by cleaner thread.

ft flusher: min bytes in a buffer flushed by cleaner thread Min number of bytes in message buffer flushed by cleaner thread.

ft flusher: min workdone in a buffer flushed by cleaner thread Min workdone value of any message buffer flushed by cleaner thread.

ft flusher: nodes cleaned which had empty buffers Number of nodes that are selected by cleaner, but whose buffers are empty.

ft flusher: nodes dirtied by cleaner thread Number of nodes that are made dirty by the cleaner thread.

ft flusher: nodes dirtied by cleaner thread leaf merges The number of nodes dirtied by the "flush from root" process to merge a leaf node.

ft flusher: nonleaf node merges Number of times nonleaf nodes are merged.

ft flusher: nonleaf node splits Number of nonleaf nodes split.

ft flusher: number of flushes that read something off disk Number of flushes that had to read a child (or part) off disk.

ft flusher: number of flushes that triggered 1 cascading flush Number of flushes that triggered 1 cascading flush.

ft flusher: number of flushes that triggered 2 cascading flushes Number of flushes that triggered 2 cascading flushes.

ft flusher: number of flushes that triggered 3 cascading flushes Number of flushes that triggered 3 cascading flushes.

ft flusher: number of flushes that triggered 4 cascading flushes Number of flushes that triggered 4 cascading flushes.

ft flusher: number of flushes that triggered 5 cascading flushes Number of flushes that triggered 5 cascading flushes.

ft flusher: number of flushes that triggered another flush in child Number of flushes that triggered another flush in the child.

ft flusher: number of flushes that triggered over 5 cascading flushes Number of flushes that triggered more than 5 cascading flushes.

ft flusher: number of in memory flushes Number of in-memory flushes.

ft flusher: times cleaner thread tries to merge a leaf The number of times the cleaner thread tries to merge a leaf.

ft flusher: total bytes in buffers flushed by cleaner thread Total number of bytes in message buffers flushed by cleaner thread.

ft flusher: total nodes potentially flushed by cleaner thread Total number of nodes whose buffers are potentially flushed by cleaner thread.

ft flusher: total number of flushes done by flusher threads or cleaner threads Total number of flushes done by flusher threads or cleaner threads.

ft flusher: total workdone in buffers flushed by cleaner thread Total workdone value of message buffers flushed by cleaner thread.

handlerton: primary key bytes inserted Total number of bytes inserted into all primary key indexes.

hot: max number of flushes from root ever required to optimize a tree The maximum number of flushes from the root ever required to optimize a tree.

hot: operations aborted The number of HOT operations that have been aborted.

hot: operations ever started The number of HOT operations that have begun.

hot: operations successfully completed The number of HOT operations that have successfully completed.

indexer: max number of indexers that ever existed simultaneously This is the maximum number of indexers that ever existed simultaneously.

indexer: number of calls to indexer->abort() This is the number of indexers that were aborted.

indexer: number of calls to indexer->build() failed This is the total number of times that indexes were unable to be created using a indexer

indexer: number of calls to indexer->build() succeeded This is the total number of times that indexes were created using a indexer.

indexer: number of calls to indexer->close() that failed This is the number of indexers that were unable to create the requested index(es).

indexer: number of calls to indexer->close() that succeeded This is the number of indexers that successfully created the requested index(es).

indexer: number of calls to toku indexer create indexer() that failed This is the number of times a indexer was requested but could not be created.

indexer: number of indexers currently in existence This is the number of indexers that currently exist.

indexer: number of indexers successfully created This is the number of times one of our internal objects, a indexer, has been created.

le: expanded This is the number of times that an expanded memory mechanism was used to store a new or modified row on disk.

le: max committed xr This is the maximum number of committed transaction records that were stored on disk in a new or modified row.

le: max memsize This is the maximum number of bytes that were stored on disk as a new or modified row. This is the maximum uncompressed size of any row stored in *TokuDB* that was created or modified since the server started.

le: max provisional xr This is the maximum number of provisional transaction records that were stored on disk in a new or modified row.

le: size of leafentries after garbage collection (during message application) Total number of bytes of leaf nodes data after performing garbage collection for non-flush events.

le: size of leafentries after garbage collection (outside message application) Total number of bytes of leaf nodes data after performing garbage collection for flush events.

le: size of leafentries before garbage collection (during message application) Total number of bytes of leaf nodes data before performing garbage collection for non-flush events.

le: size of leafentries before garbage collection (outside message application) Total number of bytes of leaf nodes data before performing garbage collection for flush events.

loader: max number of loaders that ever existed simultaneously This is the maximum number of loaders that ever existed simultaneously.

loader: number of calls to loader->abort() This is the number of loaders that were aborted.

loader: number of calls to loader->close() that failed This is the number of loaders that were unable to create the requested table.

loader: number of calls to loader->close() that succeeded This is the number of loaders that successfully created the requested table.

loader: number of calls to loader->put() failed This is the total number of rows that were unable to be inserted using a loader.

loader: number of calls to loader->put() succeeded This is the total number of rows that were inserted using a loader.

loader: number of calls to toku loader create loader() that failed This is the number of times a loader was requested but could not be created.

loader: number of loaders currently in existence This is the number of loaders that currently exist.

loader: number of loaders successfully created This is the number of times one of our internal objects, a loader, has been created.

locktree: latest post-escalation memory size Size of the locktree, in bytes, after most current locktree escalation.

locktree: long time waiting for locks Total time, in microseconds, of the long waits.

locktree: long time waiting on lock escalation Total time, in microseconds, of the long waits for lock escalation to free up memory.

locktree: memory size Count, in bytes, that the locktree is currently using.

locktree: memory size limit Maximum number of bytes that the locktree is allowed to use.

locktree: number of lock timeouts Count of the number of times that a lock request timed out.

locktree: number of locktrees eligible for the STO Number of locktrees eligible for "single transaction optimizations".

locktree: number of locktrees open now Number of locktrees currently open.

locktree: number of lock waits Number of times that a lock request could not be acquired because of a conflict with some other transaction.

locktree: number of long lock waits Number of lock waits greater than 1 second in duration.

locktree: number of long waits on lock escalation Number of times that a client thread had to wait on lock escalation and the wait time was greater than 1 second.

locktree: number of pending lock requests Number of requesters waiting for a lock grant.

locktree: number of times a locktree ended the STO early Total number of times a "single transaction optimization" was ended early due to another trans- action starting.

locktree: number of times lock escalation ran Number of times the locktree needed to run lock escalation to reduce its memory footprint.

locktree: number of waits on lock escalation When the sum of the sizes of locks taken reaches the lock tree limit, we run lock escalation on a background thread. The clients threads need to wait for escalation to consolidate locks and free up memory. This counter counts the number of times a client thread has to wait on lock escalation.

locktree: time spent ending the STO early (seconds) Total number of seconds ending "single transaction optimizations".

locktree: time spent running escalation (seconds) Total number of seconds spent performing locktree escalation.

locktree: time waiting for locks Total time, in microseconds, spend by some client waiting for a lock conflict to be resolved.

locktree: time waiting on lock escalation Total time, in microseconds, that a client thread spent waiting for lock escalation to free up memory.

logger: next LSN This is the next unassigned Log Sequence Number. It will be assigned to the next entry in the recovery log.

logger: number of long logger write operations Number of times a logger write operation required 100ms or more.

logger: writes (bytes) Number of bytes the logger has written to disk.

logger: writes Number of times the logger has written to disk.

logger: writes (seconds) Number of seconds waiting for IO when writing logs to disk.

logger: writes (uncompressed bytes) Number of uncompressed the logger has written to disk.

max open dbs Max number of simultaneously open DBs.

memory: estimated maximum memory footprint Maximum memory footprint of the storage engine, the max value of (used - freed).

memory: largest attempted allocation size Largest number of bytes in a single successful malloc() operation.

memory: mallocator version Version string from in-use memory allocator.

memory: mmap threshold The threshold for malloc to use mmap.

memory: number of bytes freed Total number of mallocated bytes freed (used - freed = bytes in use).

memory: number of bytes requested Total number of bytes requested from mallocator.

memory: number of bytes used (requested + overhead) Total number of bytes allocated by mallocator.

memory: number of free operations Number of calls to free().

memory: number of malloc operations Number of calls to malloc().

memory: number of malloc operations that failed Number of failed calls to malloc().

memory: number of realloc operations Number of calls to realloc().

memory: number of realloc operations that failed Number of failed calls to realloc().

memory: size of the last failed allocation attempt Largest number of bytes in a single failed malloc() operation.

num open dbs now Number of currently open DBs.

period, in ms, that recovery log is automatically fsynced `fsync()` frequency in milliseconds.

time now Current date/time on server.

time of engine startup This is the time when the *TokuDB* storage engine started up. Normally, this is when `mysqld` started.

time of environment creation This is the time when the *TokuDB* storage engine was first started up. Normally, this is when `mysqld` was initially installed with *TokuDB* 5.x. If the environment was upgraded from *TokuDB* 4.x (4.2.0 or later), then this will be displayed as "Dec 31, 1969" on Linux hosts.

txn: aborts This is the total number of transactions that have been aborted.

txn: begin This is the number of transactions that have been started.

txn: begin read only Number of read only transactions started.

txn: successful commits This is the total number of transactions that have been committed.

46.6.4 Global Status

The `INFORMATION_SCHEMA.GLOBAL_STATUS` table provides details about the inner workings of *TokuDB* and can be useful in tuning your particular environment. The statuses can be determined with the following command:

```
SELECT * FROM INFORMATION_SCHEMA.GLOBAL_STATUS;
```

The following global status parameters are available:

TOKUDB_BASEMENTS_DECOMPRESSED_FOR_WRITE Number of basement nodes decompressed for writes.

TOKUDB_BASEMENTS_DECOMPRESSED_PREFETCH Number of basement nodes decompressed by a prefetch thread.

TOKUDB_BASEMENTS_DECOMPRESSED_PRELOCKED_RANGE Number of basement nodes decompressed by queries aggressively.

TOKUDB_BASEMENTS_DECOMPRESSED_TARGET_QUERY Number of basement nodes decompressed for queries.

TOKUDB_BASEMENT_DESERIALIZATION_FIXED_KEY Number of basement nodes deserialized where all keys had the same size, leaving the basement in a format that is optimal for in-memory workloads.

TOKUDB_BASEMENT_DESERIALIZATION_VARIABLE_KEY Number of basement nodes deserialized where all keys did not have the same size, and thus ineligible for an in-memory optimization.

TOKUDB_BASEMENTS_FETCHED_FOR WRITE_BYTES Number of basement node bytes fetched from disk for writes.

TOKUDB_BASEMENTS_FETCHED_FOR WRITE Number of basement nodes fetched from disk for writes.

TOKUDB_BASEMENTS_FETCHED_FOR WRITE_SECONDS Number of seconds waiting for IO when fetching basement nodes from disk for writes.

TOKUDB_BASEMENTS_FETCHED_PREFETCH_BYTES Number of basement node bytes fetched from disk by a prefetch thread.

TOKUDB_BASEMENTS_FETCHED_PREFETCH Number of basement nodes fetched from disk by a prefetch thread.

TOKUDB_BASEMENTS_FETCHED_PREFETCH_SECONDS Number of seconds waiting for IO when fetching basement nodes from disk by a prefetch thread.

TOKUDB_BASEMENTS_FETCHED_PRELOCKED_RANGE_BYTES Number of basement node bytes fetched from disk aggressively.

TOKUDB_BASEMENTS_FETCHED_PRELOCKED_RANGE Number of basement nodes fetched from disk aggressively.

TOKUDB_BASEMENTS_FETCHED_PRELOCKED_RANGE_SECONDS Number of seconds waiting for IO when fetching basement nodes from disk aggressively.

TOKUDB_BASEMENTS_FETCHED_TARGET_QUERY_BYTES Number of basement node bytes fetched from disk for queries.

TOKUDB_BASEMENTS_FETCHED_TARGET_QUERY Number of basement nodes fetched from disk for queries.

TOKUDB_BASEMENTS_FETCHED_TARGET_QUERY_SECONDS Number of seconds waiting for IO when fetching basement nodes from disk for queries.

TOKUDB_BROADCAST_MESSAGES_INJECTED_AT_ROOT How many broadcast messages injected at root.

TOKUDB_BUFFERS_DECOMPRESSED_FOR_WRITE Number of buffers decompressed for writes.

TOKUDB_BUFFERS_DECOMPRESSED_PREFETCH Number of buffers decompressed by a prefetch thread.

TOKUDB_BUFFERS_DECOMPRESSED_PRELOCKED_RANGE Number of buffers decompressed by queries aggressively.

TOKUDB_BUFFERS_DECOMPRESSED_TARGET_QUERY Number of buffers decompressed for queries.

TOKUDB_BUFFERS_FETCHED_FOR_WRITE_BYTES Number of buffer bytes fetched from disk for writes.

TOKUDB_BUFFERS_FETCHED_FOR_WRITE Number of buffers fetched from disk for writes.

TOKUDB_BUFFERS_FETCHED_FOR_WRITE_SECONDS Number of seconds waiting for IO when fetching buffers from disk for writes.

TOKUDB_BUFFERS_FETCHED_PREFETCH_BYTES Number of buffer bytes fetched from disk by a prefetch thread.

TOKUDB_BUFFERS_FETCHED_PREFETCH Number of buffers fetched from disk by a prefetch thread.

TOKUDB_BUFFERS_FETCHED_PREFETCH_SECONDS Number of seconds waiting for IO when fetching buffers from disk by a prefetch thread.

TOKUDB_BUFFERS_FETCHED_PRELOCKED_RANGE_BYTES Number of buffer bytes fetched from disk aggressively.

TOKUDB_BUFFERS_FETCHED_PRELOCKED_RANGE Number of buffers fetched from disk aggressively.

TOKUDB_BUFFERS_FETCHED_PRELOCKED_RANGE_SECONDS Number of seconds waiting for IO when fetching buffers from disk aggressively.

TOKUDB_BUFFERS_FETCHED_TARGET_QUERY_BYTES Number of buffer bytes fetched from disk for queries.

TOKUDB_BUFFERS_FETCHED_TARGET_QUERY Number of buffers fetched from disk for queries.

TOKUDB_BUFFERS_FETCHED_TARGET_QUERY_SECONDS Number of seconds waiting for IO when fetching buffers from disk for queries.

TOKUDB_CACHETABLE_CLEANER_EXECUTIONS Total number of times the cleaner thread loop has executed.

TOKUDB_CACHETABLE_CLEANER_ITERATIONS This is the number of cleaner operations that are performed every cleaner period.

TOKUDB_CACHETABLE_CLEANER_PERIOD *TokuDB* includes a cleaner thread that optimizes indexes in the background. This variable is the time, in seconds, between the completion of a group of cleaner operations and the beginning of the next group of cleaner operations. The cleaner operations run on a background thread performing work that does not need to be done on the client thread.

TOKUDB_CACHETABLE_EVICTIONS Number of blocks evicted from cache.

TOKUDB_CACHETABLE_LONG_WAIT_PRESSURE_COUNT The number of times a thread was stalled for more than 1 second due to cache pressure.

TOKUDB_CACHETABLE_LONG_WAIT_PRESSURE_TIME Total time, in microseconds, waiting on cache pressure to subside for more than 1 second.

TOKUDB_CACHETABLE_POOL_CLIENT_NUM_THREADS

TOKUDB_CACHETABLE_POOL_CLIENT_NUM_THREADS_ACTIVE

TOKUDB_CACHETABLE_POOL_CLIENT_QUEUE_SIZE

TOKUDB_CACHETABLE_POOL_CLIENT_MAX_QUEUE_SIZE

TOKUDB_CACHETABLE_POOL_CLIENT_TOTAL_ITEMS_PROCESSED

TOKUDB_CACHETABLE_POOL_CLIENT_TOTAL_EXECUTION_TIME

TOKUDB_CACHETABLE_POOL_CACHETABLE_NUM_THREADS

TOKUDB_CACHETABLE_POOL_CACHETABLE_NUM_THREADS_ACTIVE

TOKUDB_CACHETABLE_POOL_CACHETABLE_QUEUE_SIZE

TOKUDB_CACHETABLE_POOL_CACHETABLE_MAX_QUEUE_SIZE

TOKUDB_CACHETABLE_POOL_CACHETABLE_TOTAL_ITEMS_PROCESSED

TOKUDB_CACHETABLE_POOL_CACHETABLE_TOTAL_EXECUTION_TIME

TOKUDB_CACHETABLE_POOL_CHECKPOINT_NUM_THREADS

TOKUDB_CACHETABLE_POOL_CHECKPOINT_NUM_THREADS_ACTIVE

TOKUDB_CACHETABLE_POOL_CHECKPOINT_QUEUE_SIZE

TOKUDB_CACHETABLE_POOL_CHECKPOINT_MAX_QUEUE_SIZE

TOKUDB_CACHETABLE_POOL_CHECKPOINT_TOTAL_ITEMS_PROCESSED

TOKUDB_CACHETABLE_POOL_CHECKPOINT_TOTAL_EXECUTION_TIME

TOKUDB_CACHETABLE_MISS This is a count of how many times the application was unable to access your data in the internal cache.

TOKUDB_CACHETABLE_MISS_TIME This is the total time, in microseconds, of how long the database has had to wait for a disk read to complete.

TOKUDB_CACHETABLE_PREFETCHES This is the total number of times that a block of memory has been prefetched into the database's cache. Data is prefetched when the database's algorithms determine that a block of memory is likely to be accessed by the application.

TOKUDB_CACHETABLE_SIZE_CACHEPRESSURE The number of bytes causing cache pressure (the sum of buffers and workdone counters), helps to understand if cleaner threads are keeping up with workload.

TOKUDB_CACHETABLE_SIZE_CLONED Amount of memory, in bytes, currently used for cloned nodes. During the checkpoint operation, dirty nodes are cloned prior to serialization/compression, then written to disk. After which, the memory for the cloned block is returned for re-use.

TOKUDB_CACHETABLE_SIZE_CURRENT This is a count, in bytes, of how much of your uncompressed data is currently in the database's internal cache.

TOKUDB_CACHETABLE_SIZE_LEAF The number of bytes of leaf nodes in the cache.

TOKUDB_CACHETABLE_SIZE_LIMIT This is a count, in bytes, of how much of your uncompressed data will fit in the database's internal cache.

TOKUDB_CACHETABLE_SIZE_NONLEAF The number of bytes of nonleaf nodes in the cache.

TOKUDB_CACHETABLE_SIZE_ROLLBACK The number of bytes of rollback nodes in the cache.

TOKUDB_CACHETABLE_SIZE_WRITING This is the number of bytes that are currently queued up to be written to disk.

TOKUDB_CACHETABLE_WAIT_PRESSURE_COUNT The number of times a thread was stalled due to cache pressure.

TOKUDB_CACHETABLE_WAIT_PRESSURE TIME Total time, in microseconds, waiting on cache pressure to subside.

TOKUDB_CHECKPOINT_BEGIN_TIME Cumulative time (in microseconds) required to mark all dirty nodes as pending a checkpoint.

TOKUDB_CHECKPOINT_DURATION_LAST Time (in seconds) required to complete the last checkpoint.

TOKUDB_CHECKPOINT_DURATION Time (in seconds) required to complete all checkpoints.

TOKUDB_CHECKPOINT_FAILED This is the number of checkpoints that have failed for any reason.

TOKUDB_CHECKPOINT_LAST_BEGAN This is the time the last checkpoint began. If a checkpoint is currently in progress, then this time may be later than the time the last checkpoint completed. (Note, if no checkpoint has ever taken place, then this value will be "Dec 31, 1969" on Linux hosts.)

TOKUDB_CHECKPOINT_LAST_COMPLETE_BEGAN This is the time the last complete checkpoint started. Any data that changed after this time will not be captured in the checkpoint.

TOKUDB_CHECKPOINT_LAST_COMPLETE_ENDED This is the time the last complete checkpoint ended.

TOKUDB_CHECKPOINT_LONG_CHECKPOINT_BEGIN_COUNT The total number of times a checkpoint begin took more than 1 second.

TOKUDB_CHECKPOINT_END_TIME The time spent in checkpoint end operation in seconds.

TOKUDB_CHECKPOINT_LONG_END_COUNT This is the count of end_checkpoint operations that exceeded 1 minute.

TOKUDB_CHECKPOINT_LONG_END_TIME This is the total time of long checkpoints in seconds.

TOKUDB_CHECKPOINT_LONG_CHECKPOINT_BEGIN_TIME This is the total time, in microseconds, of long checkpoint begins. A long checkpoint begin is one taking more than 1 second.

TOKUDB_CHECKPOINT_PERIOD This is the interval in seconds between the end of an automatic checkpoint and the beginning of the next automatic checkpoint.

TOKUDB_CHECKPOINT_TAKEN This is the number of complete checkpoints that have been taken.

TOKUDB_DB_CLOSES Number of db close operations.

TOKUDB_DB_OPEN_CURRENT Number of currently open DBs.

TOKUDB_DB_OPEN_MAX Max number of simultaneously open DBs.

TOKUDB_DB_OPENS Number of db open operations.

TOKUDB_DESCRIPTOR_SET This is the number of time a descriptor was updated when the entire dictionary was updated (for example, when the schema has been changed).

TOKUDB_DICTIONARY_BROADCAST_UPDATES This is the number of broadcast updates that have been successfully performed. A broadcast update is an update that affects all rows in a dictionary.

TOKUDB_DICTIONARY_UPDATES This is the total number of rows that have been updated in all primary and secondary indexes combined, if those updates have been done with a separate recovery log entry per index.

TOKUDB_FILESYSTEM_FSYNC_NUM This is the total number of times the database has flushed the operating system's file buffers to disk.

TOKUDB_FILESYSTEM_FSYNC_TIME This the total time, in microseconds, used to fsync to disk.

TOKUDB_FILESYSTEM_LONG_FSYNC_NUM This is the total number of times the database has flushed the operating system's file buffers to disk and this operation required more than 1 second.

TOKUDB_FILESYSTEM_LONG_FSYNC_TIME This the total time, in microseconds, used to fsync to disk when the operation required more than 1 second.

TOKUDB_FILESYSTEM_THREADS_BLOCKED_BY_FULL_DISK This is the number of threads that are currently blocked because they are attempting to write to a full disk. This is normally zero. If this value is non-zero, then a warning will appear in the "disk free space" field.

TOKUDB_LEAF_COMPRESSION_TO_MEMORY_SECONDS Total time, in seconds, spent compressing leaf nodes.

TOKUDB_LEAF_DECOMPRESSION_TO_MEMORY_SECONDS Total time, in seconds, spent decompressing leaf nodes.

TOKUDB_LEAF_DESERIALIZATION_TO_MEMORY_SECONDS Total time, in seconds, spent deserializing leaf nodes.

TOKUDB_LEAF_NODE_COMPRESSION_RATIO Ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for leaf nodes.

TOKUDB_LEAF_NODE_FULL_EVICTIONS_BYTES The number of bytes freed by evicting full leaf nodes from the cache.

TOKUDB_LEAF_NODE_FULL_EVICTIONS The number of times a full leaf node was evicted from the cache.

TOKUDB_LEAF_NODE_PARTIAL_EVICTIONS_BYTES The number of bytes freed by evicting partitions of leaf nodes from the cache.

TOKUDB_LEAF_NODE_PARTIAL_EVICTIONS The number of times a partition of a leaf node was evicted from the cache.

TOKUDB_LEAF_NODES_CREATED Number of leaf nodes created.

TOKUDB_LEAF_NODES_DESTROYED Number of leaf nodes destroyed.

TOKUDB_LEAF_NODES_FLUSHED_CHECKPOINT_BYTES Number of bytes of leaf nodes flushed to disk for checkpoint.

TOKUDB_LEAF_NODES_FLUSHED_CHECKPOINT Number of leaf nodes flushed to disk for checkpoint.

TOKUDB_LEAF_NODES_FLUSHED_CHECKPOINT_SECONDS Number of seconds waiting for IO when writing leaf nodes flushed to disk for checkpoint.

TOKUDB_LEAF_NODES_FLUSHED_CHECKPOINT_UNCOMPRESSED BYTES Number of uncompressed bytes of leaf nodes flushed to disk for checkpoint.

TOKUDB_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_BYTES Number of bytes of leaf nodes flushed to disk, not for checkpoint.

TOKUDB_LEAF_NODES_FLUSHED_NOT_CHECKPOINT Number of leaf nodes flushed to disk, not for checkpoint.

TOKUDB_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_SECONDS Number of seconds waiting for IO when writing leaf nodes flushed to disk, not for checkpoint.

TOKUDB_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_UNCOMPRESSED_BYTES Number of bytes of leaf nodes flushed to disk, not for checkpoint.

TOKUDB_LEAF_SERIALIZATION_TO_MEMORY_SECONDS Total time, in seconds, spent serializing leaf nodes.

TOKUDB_LOADER_NUM_CREATED This is the number of times one of our internal objects, a loader, has been created.

TOKUDB_LOADER_NUM_CURRENT This is the number of loaders that currently exist.

TOKUDB_LOADER_NUM_MAX This is the maximum number of loaders that ever existed simultaneously.

TOKUDB_LOCKTREE_ESCALATION_NUM Number of times the locktree needed to run lock escalation to reduce its memory footprint.

TOKUDB_LOCKTREE_ESCALATION_SECONDS Total number of seconds spent performing locktree escalation.

TOKUDB_LOCKTREE_LATEST_POST_ESCALATION_MEMORY_SIZE Size of the locktree, in bytes, after most current locktree escalation.

TOKUDB_LOCKTREE_LONG_WAIT_COUNT Number of lock waits greater than 1 second in duration.

TOKUDB_LOCKTREE_LONG_WAIT_ESCALATION_COUNT Number of times that a client thread had to wait on lock escalation and the wait time was greater than 1 second.

TOKUDB_LOCKTREE_LONG_WAIT_ESCALATION_TIME Total time, in microseconds, of the long waits for lock escalation to free up memory.

TOKUDB_LOCKTREE_LONG_WAIT_TIME Total time, in microseconds, of the long waits.

TOKUDB_LOCKTREE_MEMORY_SIZE Count, in bytes, that the locktree is currently using.

TOKUDB_LOCKTREE_MEMORY_SIZE_LIMIT Maximum number of bytes that the locktree is allowed to use.

TOKUDB_LOCKTREE_OPEN_CURRENT Number of locktrees currently open.

TOKUDB_LOCKTREE_PENDING_LOCK_REQUESTS Number of requesters waiting for a lock grant.

TOKUDB_LOCKTREE_STO_ELIGIBLE_NUM Number of locktrees eligible for “single transaction optimizations”.

TOKUDB_LOCKTREE_STO_ENDED_NUM Total number of times a “single transaction optimization” was ended early due to another transaction starting.

TOKUDB_LOCKTREE_STO_ENDED_SECONDS Total number of seconds ending “single transaction optimizations”.

TOKUDB_LOCKTREE_TIMEOUT_COUNT Count of the number of times that a lock request timed out.

TOKUDB_LOCKTREE_WAIT_COUNT Number of times that a lock request could not be acquired because of a conflict with some other transaction.

TOKUDB_LOCKTREE_WAIT_ESCALATION_COUNT When the sum of the sizes of locks taken reaches the lock tree limit, we run lock escalation on a background thread. The clients threads need to wait for escalation to consolidate locks and free up memory. This counter counts the number of times a client thread has to wait on lock escalation.

TOKUDB_LOCKTREE_WAIT_ESCALATION_TIME Total time, in microseconds, that a client thread spent waiting for lock escalation to free up memory.

TOKUDB_LOCKTREE_WAIT_TIME Total time, in microseconds, spend by some client waiting for a lock conflict to be resolved.

TOKUDB_LOGGER_WAIT_LONG Number of times a logger write operation required 100ms or more.

TOKUDB_LOGGER_WRITES_BYTES Number of bytes the logger has written to disk.

TOKUDB_LOGGER_WRITES Number of times the logger has written to disk.

TOKUDB_LOGGER_WRITES_SECONDS Number of seconds waiting for IO when writing logs to disk.

TOKUDB_LOGGER_WRITES_UNCOMPRESSED_BYTES Number of uncompressed the logger has written to disk.

TOKUDB_MEM_ESTIMATED_MAXIMUM_MEMORY_FOOTPRINT Maximum memory footprint of the storage engine, the max value of (used - freed).

TOKUDB_MESSAGES_FLUSHED_FROM_H1_TO_LEAVES_BYTES How many bytes of messages flushed from h1 nodes to leaves.

TOKUDB_MESSAGES_IGNORED_BY_LEAF_DUE_TO_MSN The number of messages that were ignored by a leaf because it had already been applied.

TOKUDB_MESSAGES_INJECTED_AT_ROOT_BYTES How many bytes of messages injected at root (for all trees).

TOKUDB_MESSAGES_INJECTED_AT_ROOT How many messages injected at root.

TOKUDB_MESSAGES_IN_TREES_ESTIMATE_BYTES How many bytes of messages currently in trees (estimate).

TOKUDB_NONLEAF_COMPRESSION_TO_MEMORY_SECONDS Total time, in seconds, spent compressing non leaf nodes.

TOKUDB_NONLEAF_DECOMPRESSION_TO_MEMORY_SECONDS Total time, in seconds, spent decompressing non leaf nodes.

TOKUDB_NONLEAF_DESERIALIZATION_TO_MEMORY_SECONDS Total time, in seconds, spent deserializing non leaf nodes.

TOKUDB_NONLEAF_NODE_COMPRESSION_RATIO Ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for nonleaf nodes.

TOKUDB_NONLEAF_NODE_FULL_EVICTIONS_BYTES The number of bytes freed by evicting full nonleaf nodes from the cache.

TOKUDB_NONLEAF_NODE_FULL_EVICTIONS The number of times a full nonleaf node was evicted from the cache.

TOKUDB_NONLEAF_NODE_PARTIAL_EVICTIONS_BYTES The number of bytes freed by evicting partitions of nonleaf nodes from the cache.

TOKUDB_NONLEAF_NODE_PARTIAL_EVICTIONS The number of times a partition of a nonleaf node was evicted from the cache.

TOKUDB_NONLEAF_NODES_CREATED Number of nonleaf nodes created.

TOKUDB_NONLEAF_NODES_DESTROYED Number of nonleaf nodes destroyed.

TOKUDB_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_BYTES Number of bytes of nonleaf nodes flushed to disk for checkpoint.

TOKUDB_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT Number of nonleaf nodes flushed to disk for checkpoint.

TOKUDB_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_SECONDS Number of seconds waiting for IO when writing nonleaf nodes flushed to disk for checkpoint.

TOKUDB_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_UNCOMPRESSED_BYTES Number of uncompressed bytes of nonleaf nodes flushed to disk for checkpoint.

TOKUDB_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_BYTES Number of bytes of nonleaf nodes flushed to disk, not for checkpoint.

TOKUDB_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT Number of nonleaf nodes flushed to disk, not for checkpoint.

TOKUDB_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_SECONDS Number of seconds waiting for IO when writing nonleaf nodes flushed to disk, not for check- point.

TOKUDB_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_UNCOMPRESSED_BYTES Number of uncompressed bytes of nonleaf nodes flushed to disk, not for checkpoint.

TOKUDB_NONLEAF_SERIALIZATION_TO_MEMORY_SECONDS Total time, in seconds, spent serializing non leaf nodes.

TOKUDB_OVERALL_NODE_COMPRESSION_RATIO Ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for all nodes.

TOKUDB_PIVOTS_FETCHED_FOR_PREFETCH_BYTES Number of bytes of pivot nodes fetched by a prefetch thread.

TOKUDB_PIVOTS_FETCHED_FOR_PREFETCH Number of pivot nodes fetched by a prefetch thread.

TOKUDB_PIVOTS_FETCHED_FOR_PREFETCH_SECONDS Number seconds waiting for IO when fetching pivot nodes by a prefetch thread.

TOKUDB_PIVOTS_FETCHED_FOR_QUERY_BYTES Number of bytes of pivot nodes fetched for queries.

TOKUDB_PIVOTS_FETCHED_FOR_QUERY Number of pivot nodes fetched for queries.

TOKUDB_PIVOTS_FETCHED_FOR_QUERY_SECONDS Number of seconds waiting for IO when fetching pivot nodes for queries.

TOKUDB_PIVOTS_FETCHED_FOR_WRITE_BYTES Number of bytes of pivot nodes fetched for writes.

TOKUDB_PIVOTS_FETCHED_FOR_WRITE Number of pivot nodes fetched for writes.

TOKUDB_PIVOTS_FETCHED_FOR_WRITE_SECONDS Number of seconds waiting for IO when fetching pivot nodes for writes.

TOKUDB_PROMOTION_H1_ROOTS_INJECTED_INTO Number of times a message stopped at a root with height 1.

TOKUDB_PROMOTION_INJECTIONS_AT_DEPTH_0 Number of times a message stopped at depth 0.

TOKUDB_PROMOTION_INJECTIONS_AT_DEPTH_1 Number of times a message stopped at depth 1.

TOKUDB_PROMOTION_INJECTIONS_AT_DEPTH_2 Number of times a message stopped at depth 2.

TOKUDB_PROMOTION_INJECTIONS_AT_DEPTH_3 Number of times a message stopped at depth 3.

TOKUDB_PROMOTION_INJECTIONS_LOWER_THAN_DEPTH_3 Number of times a message was promoted past depth 3.

TOKUDB_PROMOTION_LEAF_ROOTS_INJECTED_INTO Number of times a message stopped at a root with height 0.

TOKUDB_PROMOTION_ROOTS_SPLIT Number of times the root split during promotion.

TOKUDB_PROMOTION_STOPPED_AFTER_LOCKING_CHILD Number of times a message stopped before a child which had been locked.

TOKUDB_PROMOTION_STOPPED_AT_HEIGHT_1 Number of times a message stopped because it had reached height 1.

TOKUDB_PROMOTION_STOPPED_CHILD_LOCKED_OR_NOT_IN_MEMORY Number of times a message stopped because it could not cheaply get access to a child.

TOKUDB_PROMOTION_STOPPED_CHILD_NOT_FULLY_IN_MEMORY Number of times a message stopped because it could not cheaply get access to a child.

TOKUDB_PROMOTION_STOPPED_NONEMPTY_BUFFER Number of times a message stopped because it reached a nonempty buffer.

TOKUDB_TXN_ABORTS This is the total number of transactions that have been aborted.

TOKUDB_TXN_BEGIN This is the number of transactions that have been started.

TOKUDB_TXN_BEGIN_READ_ONLY Number of read only transactions started.

TOKUDB_TXN_COMMITS This is the total number of transactions that have been committed.

46.7 Frequently Asked Questions

This section contains frequently asked questions regarding *TokuDB* and related software.

- Transactional Operations
- TokuDB and the File System
- Full Disks
- Backup
- Missing Log Files
- Isolation Levels
- Lock Wait Timeout Exceeded
- Query Cache
- Row Size
- NFS & CIFS
- Using Other Storage Engines
- Using MySQL Patches with TokuDB
- Truncate Table vs Delete from Table
- Foreign Keys
- Dropping Indexes

46.7.1 Transactional Operations

What transactional operations does TokuDB support?

TokuDB supports `BEGIN TRANSACTION`, `END TRANSACTION`, `COMMIT`, `ROLLBACK`, `SAVEPOINT`, and `RELEASE SAVEPOINT`.

46.7.2 TokuDB and the File System

How can I determine which files belong to the various tables and indexes in my schemas?

The `tokudb_file_map` plugin lists all Fractal Tree Indexes and their corresponding data files. The `internal_file_name` is the actual file name (in the data folder).

```
mysql> SELECT * FROM information_schema.tokudb_file_map;
```

```
+-------------------------+---------------------------------------+---------------+-------------+---
| dictionary_name         | internal_file_name                    | table_schema  | table_name  | ta
+-------------------------+---------------------------------------+---------------+-------------+---
| ./test/tmc-key-idx_col2 | ./_test_tmc_key_idx_col2_a_14.tokudb  | test          | tmc         | ke
| ./test/tmc-main         | ./_test_tmc_main_9_14.tokudb          | test          | tmc         | ma
| ./test/tmc-status       | ./_test_tmc_status_8_14.tokudb        | test          | tmc         | st
+-------------------------+---------------------------------------+---------------+-------------+---
```

46.7.3 Full Disks

What happens when the disk system fills up?

The disk system may fill up during bulk load operations, such as `LOAD DATA IN FILE` or `CREATE INDEX`, or during incremental operations like `INSERT`.

In the bulk case, running out of disk space will cause the statement to fail with `ERROR 1030 (HY000): Got error 1 from storage engine`. The temporary space used by the bulk loader will be released. If this happens, you can use a separate physical disk for the temporary files (for more information, see `tokudb_tmp_dir`). If server runs out of free space *TokuDB* will assert the server to prevent data corruption to existing data files.

Otherwise, disk space can run low during non-bulk operations. When available space is below a user-configurable reserve (5% by default) inserts are prevented and transactions that perform inserts are aborted. If the disk becomes completely full then *TokuDB* will freeze until some disk space is made available.

Details about the disk system:

- There is a free-space reserve requirement, which is a user-configurable parameter given as a percentage of the total space in the file system. The default reserve is five percent. This value is available in the global variable `tokudb_fs_reserve_percent`. We recommend that this reserve be at least half the size of your physical memory.

  TokuDB polls the file system every five seconds to determine how much free space is available. If the free space dips below the reserve, then further table inserts are prohibited. Any transaction that attempts to insert rows will be aborted. Inserts are re-enabled when twice the reserve is available in the file system (so freeing a small amount of disk storage will not be sufficient to resume inserts). Warning messages are sent to the system error log when free space dips below twice the reserve and again when free space dips below the reserve.

  Even with inserts prohibited it is still possible for the file system to become completely full. For example this can happen because another storage engine or another application consumes disk space.

- If the file system becomes completely full, then *TokuDB* will freeze. It will not crash, but it will not respond to most SQL commands until some disk space is made available. When *TokuDB* is frozen in this state, it will still respond to the following command:

  ```
  SHOW ENGINE TokuDB STATUS;
  ```

Make disk space available will allow the storage engine to continue running, but inserts will still be prohibited until twice the reserve is free.

Note: Engine status displays a field indicating if disk free space is above twice the reserve, below twice the reserve, or below the reserve. It will also display a special warning if the disk is completely full.

- In order to make space available on this system you can:
  - Add some disk space to the filesystem.
  - Delete some non-TokuDB files manually.
  - If the disk is not completely full, you may be able to reclaim space by aborting any transactions that are very old. Old transactions can consume large volumes of disk space in the recovery log.
  - If the disk is not completely full, you can drop indexes or drop tables from your *TokuDB* databases.
  - Deleting large numbers of rows from an existing table and then closing the table may free some space, but it may not. Deleting rows may simply leave unused space (available for new inserts) inside *TokuDB* data files rather than shrink the files (internal fragmentation).

The fine print:

- The *TokuDB* storage engine can use up to three separate file systems simultaneously, one each for the data, the recovery log, and the error log. All three are monitored, and if any one of the three falls below the relevant threshold then a warning message will be issued and inserts may be prohibited.
- Warning messages to the error log are not repeated unless available disk space has been above the relevant threshold for at least one minute. This prevents excess messages in the error log if the disk free space is fluctuating around the limit.
- Even if there are no other storage engines or other applications running, it is still possible for *TokuDB* to consume more disk space when operations such as row delete and query are performed, or when checkpoints are taken. This can happen because *TokuDB* can write cached information when it is time-efficient rather than when inserts are issued by the application, because operations in addition to insert (such as delete) create log entries, and also because of internal fragmentation of *TokuDB* data files.
- The `tokudb_fs_reserve_percent` variable can not be changed once the system has started. It can only be set in `my.cnf` or on the mysqld command line.

46.7.4 Backup

How do I back up a system with TokuDB tables?

Taking backups with *Percona TokuBackup*

TokuDB is capable of performing online backups with *Percona TokuBackup*. To perform a backup, execute `backup to '/path/to/backup';`. This will create backup of the server and return when complete. The backup can be used by another server using a copy of the binaries on the source server. You can view the progress of the backup by executing `SHOW PROCESSLIST;`. *TokuBackup* produces a copy of your running *MySQL* server that is consistent at the end time of the backup process. The thread copying files from source to destination can be throttled by setting the `tokudb_backup_throttle` server variable. For more information check *Percona TokuBackup*.

The following conditions apply:

- Currently, *TokuBackup* only supports tables using the *TokuDB* storage engine and the *MyISAM* tables in the `mysql` database.

Warning: You must disable *InnoDB* asynchronous IO if backing up *InnoDB* tables via *TokuBackup* utility. Otherwise you will have inconsistent, unrecoverable backups. The appropriate setting is `innodb_use_native_aio` to `0`.

- Transactional storage engines (*TokuDB* and *InnoDB*) will perform recovery on the backup copy of the database when it is first started.
- Tables using non-transactional storage engines (*MyISAM*) are not locked during the copy and may report issues when starting up the backup. It is best to avoid operations that modify these tables at the end of a hot backup operation (adding/changing users, stored procedures, etc.).
- The database is copied locally to the path specified in `/path/to/backup`. This folder must exist, be writable, be empty, and contain enough space for a full copy of the database.
- *TokuBackup* always makes a backup of the *MySQL* `datadir` and optionally the `tokudb_data_dir`, `tokudb_log_dir`, and the binary log folder. The latter three are only backed up separately if they are not the same as or contained in the *MySQL* `datadir`. None of these three folders can be a parent of the *MySQL* `datadir`.
- A folder is created in the given backup destination for each of the source folders.
- No other directory structures are supported. All *InnoDB*, *MyISAM*, and other storage engine files must be within the *MySQL* `datadir`.
- *TokuBackup* does not follow symbolic links.

Other options for taking backups

TokuDB tables are represented in the file system with dictionary files, log files, and metadata files. A consistent copy of all of these files must be made during a backup. Copying the files while they may be modified by a running *MySQL* may result in an inconsistent copy of the database.

LVM snapshots may be used to get a consistent snapshot of all of the *TokuDB* files. The LVM snapshot may then be backed up at leisure.

The `SELECT INTO OUTFILE` statement or **mysqldump** application may also be used to get a logical backup of the database.

References

The MySQL 5.5 reference manual describes several backup methods and strategies. In addition, we recommend reading the backup and recovery chapter in the following book:

High Performance MySQL, 3rd Edition, by Baron Schwartz, Peter Zaitsev, and Vadim Tkachenko, Copyright 2012, O'Reilly Media.

Cold Backup

When *MySQL* is shut down, a copy of the *MySQL* data directory, the *TokuDB* data directory, and the *TokuDB* log directory can be made. In the simplest configuration, the *TokuDB* files are stored in the *MySQL* data directory with all of other *MySQL* files. One merely has to back up this directory.

Hot Backup using mylvmbackup

The **mylvmbackup** utility, located on Launchpad, works with *TokuDB*. It does all of the magic required to get consistent copies of all of the *MySQL* tables, including *MyISAM* tables, *InnoDB* tables, etc., creates the LVM snapshots, and backs up the snapshots.

Logical Snapshots

A logical snapshot of the databases uses a SQL statements to retrieve table rows and restore them. When used within a transaction, a consistent snapshot of the database can be taken. This method can be used to export tables from one database server and import them into another server.

The `SELECT INTO OUTFILE` statement is used to take a logical snapshot of a database. The `LOAD DATA INFILE` statement is used to load the table data. Please see the *MySQL* 5.6 reference manual for details.

Note: Please do not use the :program‘mysqlhotcopy‘ to back up *TokuDB* tables. This script is incompatible with *TokuDB*.

46.7.5 Missing Log Files

What do I do if I delete my logs files or they are otherwise missing?

You’ll need to recover from a backup. It is essential that the log files be present in order to restart the database.

46.7.6 Isolation Levels

What is the default isolation level for TokuDB?

It is repeatable-read (MVCC).

How can I change the isolation level?

TokuDB supports repeatable-read, serializable, read-uncommitted and read-committed isolation levels (other levels are not supported). *TokuDB* employs pessimistic locking, and aborts a transaction when a lock conflict is detected.

To guarantee that lock conflicts do not occur, use repeatable-read, read-uncommitted or read- committed isolation level.

46.7.7 Lock Wait Timeout Exceeded

Why do my |MySQL| clients get lock timeout errors for my update queries? And what should my application do when it gets these errors?

Updates can get lock timeouts if some other transaction is holding a lock on the rows being updated for longer than the *TokuDB* lock timeout. You may want to increase the this timeout.

If an update deadlocks, then the transaction should abort and retry.

For more information on diagnosing locking issues, see *Lock Visualization in TokuDB*.

46.7.8 Query Cache

Does TokuDB support the query cache?

Yes, you can enable the query cache in the `my.cnf` file. Please make sure that the size of the cache is set to something larger than `0`, as this, in effect, disables the cache.

46.7.9 Row Size

What is the maximum row size?

The maximum row size is 32 MiB.

46.7.10 NFS & CIFS

Can the data directories reside on a disk that is NFS or CIFS mounted?

Yes, we do have customers in production with NFS & CIFS volumes today. However, both of these disk types can pose a challenge to performance and data integrity due to their complexity. If you're seeking performance, the switching infrastructure and protocols of a traditional network were not conceptualized for low response times and can be very difficult to troubleshoot. If you're concerned with data integrity, the possible data caching at the NFS level can cause inconsistencies between the logs and data files that may never be detected in the event of a crash. If you are thinking of using a NFS or CIFS mount, we would recommend that you use synchronous mount options, which are available from the NFS mount man page, but these settings may decrease performance. For further discussion please look here.

46.7.11 Using Other Storage Engines

Can the MyISAM and InnoDB Storage Engines be used?

MyISAM and *InnoDB* can be used directly in conjunction with *TokuDB*. Please note that you should not overcommit memory between *InnoDB* and *TokuDB*. The total memory assigned to both caches must be less than physical memory.

Can the Federated Storage Engines be used?

The Federated Storage Engine can also be used, however it is disabled by default in *MySQL*. It can be enabled by either running mysqld with `--federated` as a command line parameter, or by putting `federated` in the `[mysqld]` section of the `my.cnf` file.

For more information see the *MySQL* 5.6 Reference Manual: FEDERATED Storage Engine.

46.7.12 Using MySQL Patches with TokuDB

Can I use MySQL source code patches with TokuDB?

Yes, but you need to apply Percona patches as well as your patches to *MySQL* to build a binary that works with the Percona Fractal Tree library.

46.7.13 Truncate Table vs Delete from Table

Which is faster, TRUNCATE TABLE or DELETE FROM TABLE?

Please use `TRUNCATE TABLE` whenever possible. A table truncation runs in constant time, whereas a `DELETE FROM TABLE` requires a row-by-row deletion and thus runs in time linear to the table size.

46.7.14 Foreign Keys

Does TokuDB enforce foreign key constraints?

No, *TokuDB* ignores foreign key declarations.

46.7.15 Dropping Indexes

Is dropping an index in TokuDB hot?

No, the table is locked for the amount of time it takes the file system to delete the file associated with the index.

46.8 Removing TokuDB storage engine

In case you want remove the TokuDB storage engine from *Percona Server* without causing any errors following is the recommended procedure:

46.8.1 Change the tables from TokuDB to InnoDB

If you still need the data in the TokuDB tables you'll need to alter the tables to other supported storage engine i.e., *InnoDB*:

```
mysql> ALTER TABLE City ENGINE=InnoDB;
```

Note: In case you remove the TokuDB storage engine before you've changed your tables to other supported storage engine you won't be able to access that data without re-installing the TokuDB storage engine.

46.8.2 Removing the plugins

If you're using *Percona Server* `5.6.22-72.0` or later you can use the `ps_tokudb_admin` script to remove the plugins:

```
ps_tokudb_admin --disable -uroot -pPassw0rd
```

Script output should look like this:

```
Checking if Percona server is running with jemalloc enabled...
>> Percona server is running with jemalloc enabled.

Checking transparent huge pages status on the system...
>> Transparent huge pages are currently disabled on the system.

Checking if thp-setting=never option is already set in config file...
>> Option thp-setting=never is set in the config file.

Checking TokuDB plugin status...
>> TokuDB plugin is installed.

Removing thp-setting=never option from /etc/mysql/my.cnf
>> Successfuly removed thp-setting=never option from /etc/mysql/my.cnf
```

```
Uninstalling TokuDB plugin...
>> Successfuly uninstalled TokuDB plugin.
```

Prior to *Percona Server* `5.6.22-72.0` TokuDB storage engine requires manual removal:

```
UNINSTALL PLUGIN tokudb;
UNINSTALL PLUGIN tokudb_file_map;
UNINSTALL PLUGIN tokudb_fractal_tree_info;
UNINSTALL PLUGIN tokudb_fractal_tree_block_map;
UNINSTALL PLUGIN tokudb_trx;
UNINSTALL PLUGIN tokudb_locks;
UNINSTALL PLUGIN tokudb_lock_waits;
```

After the engine and the plugins have been uninstalled you can remove the TokuDB package by using the apt/yum commands:

```
[root@centos ~]# yum remove Percona-Server-tokudb-56.x86_64
```

or

```
root@wheezy:~# apt-get remove percona-server-tokudb-5.6
```

Note: Make sure you've removed all the TokuDB specific variables from your configuration file (`my.cnf`) before you restart the server, otherwise server could show errors or warnings and won't be able to start.

46.9 Getting the Most from TokuDB

Compression: *TokuDB* compresses all data on disk, including indexes. Compression lowers cost by reducing the amount of storage required and frees up disk space for additional indexes to achieve improved query performance. Depending on the compressibility of the data, we have seen compression ratios up to 25x for high compression. Compression can also lead to improved performance since less data needs to be read from and written to disk.

Fast Insertions and Deletions: TokuDB's Fractal Tree technology enables fast indexed insertions and deletions. Fractal Trees match B-trees in their indexing sweet spot (sequential data) and are up to two orders of magnitude faster for random data with high cardinality.

Eliminates Slave Lag: *TokuDB* replication slaves can be configured to process the replication stream with virtually no read IO. Uniqueness checking is performed on the *TokuDB* master and can be skipped on all *TokuDB* slaves. Also, row based replication ensures that all before and after row images are captured in the binary logs, so the *TokuDB* slaves can harness the power of Fractal Tree indexes and bypass traditional read-modify-write behavior. This "Read Free Replication" ensures that replication slaves do not fall behind the master and can be used for read scaling, backups, and disaster recovery, without sharding, expensive hardware, or limits on what can be replicated.

Hot Index Creation: *TokuDB* allows the addition of indexes to an existing table while inserts and queries are being performed on that table. This means that *MySQL* can be run continuously with no blocking of queries or insertions while indexes are added and eliminates the down-time that index changes would otherwise require.

Hot Column Addition, Deletion, Expansion and Rename: *TokuDB* allows the addition of new columns to an existing table, the deletion of existing columns from an existing table, the expansion of `char`, `varchar`, `varbinary`, and `integer` type columns in an existing table, and the renaming of an existing column while inserts and queries are being performed on that table.

Online (Hot) Backup: The *TokuDB* can create backups of online database servers without downtime.

Fast Indexing: In practice, slow indexing often leads users to choose a smaller number of sub-optimal indexes in order to keep up with incoming data rates. These sub-optimal indexes result in disproportionately slower queries, since the

difference in speed between a query with an index and the same query when no index is available can be many orders of magnitude. Thus, fast indexing means fast queries.

Clustering Keys and Other Indexing Improvements: *TokuDB* tables are clustered on the primary key. *TokuDB* also supports clustering secondary keys, providing better performance on a broader range of queries. A clustering key includes (or clusters) all of the columns in a table along with the key. As a result, one can efficiently retrieve any column when doing a range query on a clustering key. Also, with *TokuDB*, an auto-increment column can be used in any index and in any position within an index. Lastly, *TokuDB* indexes can include up to 32 columns.

Less Aging/Fragmentation: *TokuDB* can run much longer, likely indefinitely, without the need to perform the customary practice of dump/reload or `OPTIMIZE TABLE` to restore database performance. The key is the fundamental difference with which the Fractal Tree stores data on disk. Since, by default, the Fractal Tree will store data in 4MB chunks (pre-compression), as compared to InnoDB's 16KB, *TokuDB* has the ability to avoid "database disorder" up to 250x better than InnoDB.

Bulk Loader: *TokuDB* uses a parallel loader to create tables and offline indexes. This parallel loader will use multiple cores for fast offline table and index creation.

Full-Featured Database: *TokuDB* supports fully ACID-compliant transactions, MVCC (Multi-Version Concurrency Control), serialized isolation levels, row-level locking, and XA. *TokuDB* scales with high number of client connections, even for large tables.

Lock Diagnostics: *TokuDB* provides users with the tools to diagnose locking and deadlock issues. For more information, see *Lock Visualization in TokuDB*.

Progress Tracking: Running `SHOW PROCESSLIST` when adding indexes provides status on how many rows have been processed. Running `SHOW PROCESSLIST` also shows progress on queries, as well as insertions, deletions and updates. This information is helpful for estimating how long operations will take to complete.

Fast Recovery: *TokuDB* supports very fast recovery, typically less than a minute.

CHAPTER

FORTYSEVEN

TOKUDB INSTALLATION

Percona Server is compatible with the separately available *TokuDB* storage engine package. The *TokuDB* engine must be separately downloaded and then enabled as a plug-in component. This package can be installed alongside with standard *Percona Server* 5.7 releases and does not require any specially adapted version of *Percona Server*.

The *TokuDB* storage engine is a scalable, ACID and MVCC compliant storage engine that provides indexing-based query improvements, offers online schema modifications, and reduces slave lag for both hard disk drives and flash memory. This storage engine is specifically designed for high performance on write-intensive workloads which is achieved with Fractal Tree indexing. To learn more about Fractal Tree indexing, you can visit the following Wikipedia page.

Warning: Only the Percona supplied *TokuDB* engine should be used with *Percona Server* 5.7. A *TokuDB* engine downloaded from other sources is not compatible. *TokuDB* file formats are not the same across *MySQL* variants. Migrating from one variant to any other variant requires a logical data dump and reload.

47.1 Prerequisites

47.1.1 `libjemalloc` library

TokuDB storage engine requires `libjemalloc` library 3.3.0 or greater. If the version in the distribution repository is lower than that you can use one from *Percona Software Repositories* or download it from somewhere else.

If the `libjemalloc` wasn't installed and enabled before it will be automatically installed when installing the *TokuDB* storage engine package by using the **apt'** or **yum** package manager, but *Percona Server* instance should be restarted for `libjemalloc` to be loaded. This way `libjemalloc` will be loaded with `LD_PRELOAD`. You can also enable `libjemalloc` by specifying `malloc-lib` variable in the `[mysqld_safe]` section of the `my.cnf` file:

```
[mysqld_safe]
malloc-lib= /path/to/jemalloc
```

47.1.2 Transparent huge pages

TokuDB won't be able to start if the transparent huge pages are enabled. Transparent huge pages is feature available in the newer kernel versions. You can check if the Transparent huge pages are enabled with:

```
$ cat /sys/kernel/mm/transparent_hugepage/enabled

 [always] madvise never
```

If transparent huge pages are enabled and you try to start the TokuDB engine you'll get the following message in you `error.log`:

```
Transparent huge pages are enabled, according to /sys/kernel/mm/redhat_transparent_hugepage/enabled
Transparent huge pages are enabled, according to /sys/kernel/mm/transparent_hugepage/enabled
```

You can disable transparent huge pages permanently by passing `transparent_hugepage=never` to the kernel in your bootloader (**NOTE**: For this change to take an effect you'll need to reboot your server).

You can disable the transparent huge pages by running the following command as root (**NOTE**: Setting this will last only until the server is rebooted):

```
echo never > /sys/kernel/mm/transparent_hugepage/enabled
echo never > /sys/kernel/mm/transparent_hugepage/defrag
```

47.2 Installation

TokuDB storage engine for *Percona Server* is currently available in our *apt* and *yum* repositories.

You can install the *Percona Server* with *TokuDB* engine by using the apt/yum commands:

```
[root@centos ~]# yum install Percona-Server-tokudb-57.x86_64
```

or

```
root@wheezy:~# apt-get install percona-server-tokudb-5.7
```

47.3 Enabling the TokuDB Storage Engine

Once the *TokuDB* server package has been installed following output will be shown:

```
* This release of Percona Server is distributed with TokuDB storage engine.
   * Run the following script to enable the TokuDB storage engine in Percona Server:

    ps_tokudb_admin --enable -u <mysql_admin_user> -p[mysql_admin_pass] [-S <socket>] [-h <host> -P <

   * See http://www.percona.com/doc/percona-server/5.7/tokudb/tokudb_installation.html for more insta

   * See http://www.percona.com/doc/percona-server/5.7/tokudb/tokudb_intro.html for an introduction t
```

Percona Server has implemented `ps_tokudb_admin` script to make the enabling the *TokuDB* storage engine easier. This script will automatically disable Transparent huge pages, if they're enabled, and install and enable the *TokuDB* storage engine with all the required plugins. You need to run this script as root or with **sudo**. After you run the script with required parameters:

```
ps_tokudb_admin --enable -uroot -pPassw0rd
```

Following output will be displayed:

```
Checking if Percona server is running with jemalloc enabled...
>> Percona server is running with jemalloc enabled.

Checking transparent huge pages status on the system...
>> Transparent huge pages are currently disabled on the system.

Checking if thp-setting=never option is already set in config file...
```

```
>> Option thp-setting=never is not set in the config file.
>> (needed only if THP is not disabled permanently on the system)

Checking TokuDB plugin status...
>> TokuDB plugin is not installed.

Adding thp-setting=never option into /etc/mysql/my.cnf
>> Successfuly added thp-setting=never option into /etc/mysql/my.cnf

Installing TokuDB engine...
>> Successfuly installed TokuDB plugin.
```

If the script returns no errors, *TokuDB* storage engine should be successfully enabled on your server. You can check it out by running:

```
mysql> SHOW ENGINES;
...
 | TokuDB | YES | Tokutek TokuDB Storage Engine with Fractal Tree(tm) Technology | YES | YES | YES |
...
```

47.4 Enabling the TokuDB Storage Engine Manually

If you don't want to use `ps_tokudb_admin` script you'll need to manually install the storage engine ad required plugins.

```
INSTALL PLUGIN tokudb SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_file_map SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_fractal_tree_info SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_fractal_tree_block_map SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_trx SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_locks SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_lock_waits SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_background_job_status SONAME 'ha_tokudb.so';
```

After the engine has been installed it should be present in the engines list. To check if the engine has been correctly installed and active:

```
mysql> SHOW ENGINES;
...
| TokuDB | YES | Tokutek TokuDB Storage Engine with Fractal Tree(tm) Technology | YES | YES | YES |
...
```

To check if all the *TokuDB* plugins have been installed correctly you should run:

```
mysql> SHOW PLUGINS;
...
| TokuDB                        | ACTIVE   | STORAGE ENGINE     | ha_tokudb.so | GPL     |
| TokuDB_file_map               | ACTIVE   | INFORMATION SCHEMA | ha_tokudb.so | GPL     |
| TokuDB_fractal_tree_info      | ACTIVE   | INFORMATION SCHEMA | ha_tokudb.so | GPL     |
| TokuDB_fractal_tree_block_map | ACTIVE   | INFORMATION SCHEMA | ha_tokudb.so | GPL     |
| TokuDB_trx                    | ACTIVE   | INFORMATION SCHEMA | ha_tokudb.so | GPL     |
| TokuDB_locks                  | ACTIVE   | INFORMATION SCHEMA | ha_tokudb.so | GPL     |
| TokuDB_lock_waits             | ACTIVE   | INFORMATION SCHEMA | ha_tokudb.so | GPL     |
| TokuDB_background_job_status  | ACTIVE   | INFORMATION SCHEMA | ha_tokudb.so | GPL     |
...
```

47.5 TokuDB Version

TokuDB storage engine version can be checked with:

```
mysql> SELECT @@tokudb_version;
+------------------+
| @@tokudb_version |
+------------------+
| 5.7.10-1rc1      |
+------------------+
1 row in set (0.00 sec)
```

47.6 Upgrade

Installing the *TokuDB* package is compatible with existing server setup and databases.

CHAPTER

FORTYEIGHT

USING TOKUDB

Warning: Do not move or modify any *TokuDB* files. You will break the database, and need to recover the database from a backup.

48.1 Fast Insertions and Richer Indexes

TokuDB's fast indexing enables fast queries through the use of rich indexes, such as covering and clustering indexes. It's worth investing some time to optimize index definitions to get the best performance from *MySQL* and *TokuDB*. Here are some resources to get you started:

- "Understanding Indexing" by Zardosht Kasheff (video)
- Rule of Thumb for Choosing Column Order in Indexes
- Covering Indexes: Orders-of-Magnitude Improvements
- Introducing Multiple Clustering Indexes
- Clustering Indexes vs. Covering Indexes
- How Clustering Indexes Sometimes Helps UPDATE and DELETE Performance
- *High Performance MySQL, 3rd Edition* by Baron Schwartz, Peter Zaitsev, Vadim Tkachenko, Copyright 2012, O'Reilly Media. See Chapter 5, *Indexing for High Performance*.

48.2 Clustering Secondary Indexes

One of the keys to exploiting TokuDB's strength in indexing is to make use of clustering secondary indexes.

TokuDB allows a secondary key to be defined as a clustering key. This means that all of the columns in the table are clustered with the secondary key. *Percona Server* parser and query optimizer support Multiple Clustering Keys when *TokuDB* engine is used. This means that the query optimizer will avoid primary clustered index reads and replace them by secondary clustered index reads in certain scenarios.

The parser has been extended to support following syntax:

```
CREATE TABLE ... ( ..., CLUSTERING KEY identifier (column list), ...
CREATE TABLE ... ( ..., UNIQUE CLUSTERING KEY identifier (column list), ...
CREATE TABLE ... ( ..., CLUSTERING UNIQUE KEY identifier (column list), ...
CREATE TABLE ... ( ..., CONSTRAINT identifier UNIQUE CLUSTERING KEY identifier (column list), ...
CREATE TABLE ... ( ..., CONSTRAINT identifier CLUSTERING UNIQUE KEY identifier (column list), ...

CREATE TABLE ... (... column type CLUSTERING [UNIQUE] [KEY], ...)
```

```
CREATE TABLE ... (... column type [UNIQUE] CLUSTERING [KEY], ...)

ALTER TABLE ..., ADD CLUSTERING INDEX identifier (column list), ...
ALTER TABLE ..., ADD UNIQUE CLUSTERING INDEX identifier (column list), ...
ALTER TABLE ..., ADD CLUSTERING UNIQUE INDEX identifier (column list), ...
ALTER TABLE ..., ADD CONSTRAINT identifier UNIQUE CLUSTERING INDEX identifier (column list), ...
ALTER TABLE ..., ADD CONSTRAINT identifier CLUSTERING UNIQUE INDEX identifier (column list), ...

CREATE CLUSTERING INDEX identifier ON ...
```

To define a secondary index as clustering, simply add the word `CLUSTERING` before the key definition. For example:

```
CREATE TABLE table (
  column_a INT,
  column_b INT,
  column_c INT,
  PRIMARY KEY index_a (column_a),
  CLUSTERING KEY index_b (column_b)) ENGINE = TokuDB;
```

In the previous example, the primary table is indexed on *column_a*. Additionally, there is a secondary clustering index (named *index_b*) sorted on *column_b*. Unlike non-clustered indexes, clustering indexes include all the columns of a table and can be used as covering indexes. For example, the following query will run very fast using the clustering *index_b*:

```
SELECT column_c
  FROM table
  WHERE column_b BETWEEN 10 AND 100;
```

This index is sorted on *column_b*, making the `WHERE` clause fast, and includes *column_c*, which avoids lookups in the primary table to satisfy the query.

TokuDB makes clustering indexes feasible because of its excellent compression and very high indexing rates. For more information about using clustering indexes, see Introducing Multiple Clustering Indexes.

48.3 Hot Index Creation

TokuDB enables you to add indexes to an existing table and still perform inserts and queries on that table while the index is being created.

The `ONLINE` keyword is not used. Instead, the value of the `tokudb_create_index_online` client session variable is examined.

Hot index creation is invoked using the `CREATE INDEX` command after setting `tokudb_create_index_online` to `on` as follows:

```
mysql> SET tokudb_create_index_online=on;
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE INDEX index ON table (field_name);
```

Alternatively, using the `ALTER TABLE` command for creating an index will create the index offline (with the table unavailable for inserts or queries), regardless of the value of `tokudb_create_index_online`. The only way to hot create an index is to use the `CREATE INDEX` command.

Hot creating an index will be slower than creating the index offline, and progress depends how busy the mysqld server is with other tasks. Progress of the index creation can be seen by using the `SHOW PROCESSLIST` command (in another client). Once the index creation completes, the new index will be used in future query plans.

If more than one hot `CREATE INDEX` is issued for a particular table, the indexes will be created serially. An index creation that is waiting for another to complete will be shown as *Locked* in `SHOW PROCESSLIST`. We recommend that each `CREATE INDEX` be allowed to complete before the next one is started.

48.4 Hot Column Add, Delete, Expand, and Rename (HCADER)

TokuDB enables you to add or delete columns in an existing table, expand `char`, `varchar`, `varbinary`, and `integer` type columns in an existing table, or rename an existing column in a table with little blocking of other updates and queries. HCADER typically blocks other queries with a table lock for no more than a few seconds. After that initial short-term table locking, the system modifies each row (when adding, deleting, or expanding columns) later, when the row is next brought into main memory from disk. For column rename, all the work is done during the seconds of downtime. On-disk rows need not be modified.

To get good performance from HCADER, observe the following guidelines:

- The work of altering the table for column addition, deletion, or expansion is performed as subsequent operations touch parts of the Fractal Tree, both in the primary index and secondary indexes.

  You can force the column addition, deletion, or expansion work to be performed all at once using the standard syntax of `OPTIMIZE TABLE X`, when a column has been added to, deleted from, or expanded in table X. It is important to note that as of *TokuDB* version 7.1.0, `OPTIMIZE TABLE` is also hot, so that a table supports updates and queries without blocking while an `OPTIMIZE TABLE` is being performed. Also, a hot `OPTIMIZE TABLE` does not rebuild the indexes, since *TokuDB* indexes do not age. Rather, they flush all background work, such as that induced by a hot column addition, deletion, or expansion.

- Each hot column addition, deletion, or expansion operation must be performed individually (with its own SQL statement). If you want to add, delete, or expand multiple columns use multiple statements.

- Avoid adding, deleting, or expanding a column at the same time as adding or dropping an index.

- The time that the table lock is held can vary. The table-locking time for HCADER is dominated by the time it takes to flush dirty pages, because MySQL closes the table after altering it. If a checkpoint has happened recently, this operation is fast (on the order of seconds). However, if the table has many dirty pages, then the flushing stage can take on the order of minutes.

- Avoid dropping a column that is part of an index. If a column to be dropped is part of an index, then dropping that column is slow. To drop a column that is part of an index, first drop the indexes that reference the column in one alter table statement, and then drop the column in another statement.

- Hot column expansion operations are only supported to `char`, `varchar`, `varbinary`, and `integer` data types. Hot column expansion is not supported if the given column is part of the primary key or any secondary keys.

- Rename only one column per statement. Renaming more than one column will revert to the standard MySQL blocking behavior. The proper syntax is as follows:

  ```
  ALTER TABLE table
    CHANGE column_old column_new
    DATA_TYPE REQUIRED_NESS DEFAULT
  ```

  Here's an example of how that might look:

  ```
  ALTER TABLE table
    CHANGE column_old column_new
    INT(10) NOT NULL;
  ```

Notice that all of the column attributes must be specified. `ALTER TABLE table CHANGE column_old column_new;` induces a slow, blocking column rename.

- Hot column rename does not support the following data types: `TIME`, `ENUM`, `BLOB`, `TINYBLOB`, `MEDIUMBLOB`, `LONGBLOB`. Renaming columns of these types will revert to the standard MySQL blocking behavior.
- Temporary tables cannot take advantage of HCADER. Temporary tables are typically small anyway, so altering them using the standard method is usually fast.

48.5 Compression Details

TokuDB offers different levels of compression, which trade off between the amount of CPU used and the compression achieved. Standard compression uses less CPU but generally compresses at a lower level, high compression uses more CPU and generally compresses at a higher level. We have seen compression up to 25x on customer data.

Compression in *TokuDB* occurs on background threads, which means that high compression need not slow down your database. Indeed, in some settings, we've seen higher overall database performance with high compression.

Note: We recommend that users use standard compression on machines with six or fewer cores, and high compression on machines with more than six cores.

The ultimate choice depends on the particulars of how a database is used, and we recommend that users use the default settings unless they have profiled their system with high compression in place.

Compression is set on a per-table basis and is controlled by setting row format during a `CREATE TABLE` or `ALTER TABLE`. For example:

```
CREATE TABLE table (
  column_a INT NOT NULL PRIMARY KEY,
  column_b INT NOT NULL) ENGINE=TokuDB
  ROW_FORMAT=row_format;
```

If no row format is specified in a `CREATE TABLE`, the table is compressed using whichever row format is specified in the session variable `tokudb_row_format`. If no row format is set nor is `tokudb_row_format`, the zlib compressor is used.

`row_format` and `tokudb_row_format` variables accept the following values:

- `TOKUDB_DEFAULT`: This sets the compression to the default behavior. As of TokuDB 7.1.0, the default behavior is to compress using the zlib library. In the future this behavior may change.
- `TOKUDB_FAST`: This sets the compression to use the quicklz library.
- `TOKUDB_SMALL`: This sets the compression to use the lzma library.

In addition, you can choose a compression library directly, which will override previous values. The following libraries are available:

- `TOKUDB_ZLIB`: Compress using the zlib library, which provides mid-range compression and CPU utilization.
- `TOKUDB_QUICKLZ`: Compress using the quicklz library, which provides light compression and low CPU utilization.
- `TOKUDB_LZMA`: Compress using the lzma library, which provides the highest compression and high CPU utilization.
- `TOKUDB_SNAPPY` - This compression is using snappy library and aims for very high speeds and reasonable compression.
- `TOKUDB_UNCOMPRESSED`: This setting turns off compression and is useful for tables with data that cannot be compressed.

48.6 Changing Compression of a Table

Modify the compression used on a particular table with the following command:

```
ALTER TABLE table
  ROW_FORMAT=row_format;
```

Note: Changing the compression of a table only affects newly written data (dirtied blocks). After changing a table's compression you can run `OPTIMZE TABLE` to rewrite all blocks of the table and its indexes.

48.7 Read Free Replication

TokuDB slaves can be configured to perform significantly less read IO in order to apply changes from the master. By utilizing the power of Fractal Tree indexes:

- insert/update/delete operations can be configured to eliminate read-modify-write behavior and simply inject messages into the appropriate Fractal Tree indexes
- update/delete operations can be configured to eliminate the IO required for uniqueness checking

To enable Read Free Replication, the servers must be configured as follows:

- On the replication master:
  - Enable row based replication: set `BINLOG_FORMAT=ROW`
- On the replication slave(s):
  - The slave must be in read-only mode: set `read_only=1`
  - Disable unique checks: set `tokudb_rpl_unique_checks=0`
  - Disable lookups (read-modify-write): set `tokudb_rpl_lookup_rows=0`

Note: You can modify one or both behaviors on the slave(s).

Note: As long as the master is using row based replication, this optimization is available on a *TokuDB* slave. This means that it's available even if the master is using *InnoDB* or *MyISAM* tables, or running non-TokuDB binaries.

Warning: *TokuDB* Read Free Replication will not propagate `UPDATE` and `DELETE` events reliably if *TokuDB* table is missing the primary key which will eventually lead to data inconsistency on the slave.

48.8 Transactions and ACID-compliant Recovery

By default, *TokuDB* checkpoints all open tables regularly and logs all changes between checkpoints, so that after a power failure or system crash, *TokuDB* will restore all tables into their fully ACID-compliant state. That is, all committed transactions will be reflected in the tables, and any transaction not committed at the time of failure will be rolled back.

The default checkpoint period is every 60 seconds, and this specifies the time from the beginning of one checkpoint to the beginning of the next. If a checkpoint requires more than the defined checkpoint period to complete, the next checkpoint begins immediately. It is also related to the frequency with which log files are trimmed, as described below.

The user can induce a checkpoint at any time by issuing the `FLUSH LOGS` command. When a database is shut down normally it is also checkpointed and all open transactions are aborted. The logs are trimmed at startup.

48.9 Managing Log Size

TokuDB keeps log files back to the most recent checkpoint. Whenever a log file reaches 100 MB, a new log file is started. Whenever there is a checkpoint, all log files older than the checkpoint are discarded. If the checkpoint period is set to be a very large number, logs will get trimmed less frequently. This value is set to 60 seconds by default.

TokuDB also keeps rollback logs for each open transaction. The size of each log is proportional to the amount of work done by its transaction and is stored compressed on disk. Rollback logs are trimmed when the associated transaction completes.

48.10 Recovery

Recovery is fully automatic with *TokuDB*. *TokuDB* uses both the log files and rollback logs to recover from a crash. The time to recover from a crash is proportional to the combined size of the log files and uncompressed size of rollback logs. Thus, if there were no long-standing transactions open at the time of the most recent checkpoint, recovery will take less than a minute.

48.11 Disabling the Write Cache

When using any transaction-safe database, it is essential that you understand the write-caching characteristics of your hardware. *TokuDB* provides transaction safe (ACID compliant) data storage for *MySQL*. However, if the underlying operating system or hardware does not actually write data to disk when it says it did, the system can corrupt your database when the machine crashes. For example, *TokuDB* can not guarantee proper recovery if it is mounted on an NFS volume. It is always safe to disable the write cache, but you may be giving up some performance.

For most configurations you must disable the write cache on your disk drives. On ATA/SATA drives, the following command should disable the write cache:

```
$ hdparm -W0 /dev/hda
```

There are some cases when you can keep the write cache, for example:

- Write caching can remain enabled when using XFS, but only if XFS reports that disk write barriers work. If you see one of the following messages in /var/log/messages, then you must disable the write cache:
  - `Disabling barriers, not supported with external log device`
  - `Disabling barriers, not supported by the underlying device`
  - `Disabling barriers, trial barrier write failed`

  XFS write barriers appear to succeed for single disks (with no LVM), or for very recent kernels (such as that provided by Fedora 12). For more information, see the XFS FAQ.

In the following cases, you must disable the write cache:

- If you use the ext3 filesystem
- If you use LVM (although recent Linux kernels, such as Fedora 12, have fixed this problem)
- If you use Linux's software RAID

- If you use a RAID controller with battery-backed-up memory. This may seem counter-intuitive. For more information, see the XFS FAQ

In summary, you should disable the write cache, unless you have a very specific reason not to do so.

48.12 Progress Tracking

TokuDB has a system for tracking progress of long running statements, thereby removing the need to define triggers to track statement execution, as follows:

- Bulk Load: When loading large tables using `LOAD DATA INFILE` commands, doing a `SHOW PROCESSLIST` command in a separate client session shows progress. There are two progress stages. The first will state something like `Inserted about 1000000 rows`. After all rows are processed like this, the next stage tracks progress by showing what fraction of the work is done (e.g. `Loading of data about 45% done`)
- Adding Indexes: When adding indexes via `ALTER TABLE` or `CREATE INDEX`, the command `SHOW PROCESSLIST` shows progress. When adding indexes via `ALTER TABLE` or `CREATE INDEX`, the command `SHOW PROCESSLIST` will include an estimation of the number of rows processed. Use this information to verify progress is being made. Similar to bulk loading, the first stage shows how many rows have been processed, and the second stage shows progress with a fraction.
- Commits and Aborts: When committing or aborting a transaction, the command `SHOW PROCESSLIST` will include an estimate of the transactional operations processed.

48.13 Migrating to TokuDB

To convert an existing table to use the *TokuDB* engine, run `ALTER TABLE... ENGINE=TokuDB`. If you wish to load from a file, use `LOAD DATA INFILE` and not `mysqldump`. Using `mysqldump` will be much slower. To create a file that can be loaded with `LOAD DATA INFILE`, refer to the `INTO OUTFILE` option of the SELECT Syntax.

Note: Creating this file does not save the schema of your table, so you may want to create a copy of that as well.

CHAPTER

FORTYNINE

TOKUDB BACKGROUND ANALYZE TABLE

Percona Server has an option to automatically analyze tables in the background based on a measured change in data. This has been done by implementing the background job manager that can perform operations on a background thread.

49.1 Background Jobs

Background jobs and schedule are transient in nature and are not persisted anywhere. Any currently running job will be terminated on shutdown and all scheduled jobs will be forgotten about on server restart. There can't be two jobs on the same table scheduled or running at any one point in time. If you manually invoke an `ANALYZE TABLE` that conflicts with either a pending or running job, the running job will be canceled and the users task will run immediately in the foreground. All the scheduled and running background jobs can be viewed by querying the `TOKUDB_BACKGROUND_JOB_STATUS` table.

New `tokudb_analyze_in_background` variable has been implemented in order to control if the `ANALYZE TABLE` will be dispatched to the background process or if it will be running in the foreground. To control the function of `ANALYZE TABLE` a new `tokudb_analyze_mode` variable has been implemented. This variable offers options to cancel any running or scheduled job on the specified table (`TOKUDB_ANALYZE_CANCEL`), use existing analysis algorithm (`TOKUDB_ANALYZE_STANDARD`), or to recount the logical rows in table and update persistent count (`TOKUDB_ANALYZE_RECOUNT_ROWS`).

`TOKUDB_ANALYSE_RECOUNT_ROWS` is a new mechanism that is used to perform a logical recount of all rows in a table and persist that as the basis value for the table row estimate. This mode was added for tables that have been upgraded from an older version of *TokuDB* that only reported physical row counts and never had a proper logical row count. Newly created tables/partitions will begin counting logical rows correctly from their creation and should not need to be recounted unless some odd edge condition causes the logical count to become inaccurate over time. This analysis mode has no effect on the table cardinality counts. It will take the currently set session values for `tokudb_analyze_in_background`, and `tokudb_analyze_throttle`. Changing the global or session instances of these values after scheduling will have no effect on the job.

Any background job, both pending and running, can be canceled by setting the `tokudb_analyze_mode` to `TOKUDB_ANALYZE_CANCEL` and issuing the `ANALYZE TABLE` on the table for which you want to cancel all the jobs for.

49.2 Auto analysis

To implement the background analysis and gathering of cardinality statistics on a *TokuDB* tables new `delta` value is now maintained in memory for each *TokuDB* table. This value is not persisted anywhere and it is reset to `0` on a server start. It is incremented for each `INSERT/UPDATE/DELETE` command and ignores the impact of transactions (rollback specifically). When this delta value exceeds the `tokudb_auto_analyze` percentage of rows in the table an analysis is performed according to the current session's settings. Other analysis for this table will be disabled until

this analysis completes. When this analysis completes, the delta is reset to 0 to begin recalculating table changes for the next potential analysis.

Status values are now reported to server immediately upon completion of any analysis (previously new status values were not used until the table has been closed and re-opened). Half-time direction reversal of analysis has been implemented, meaning that if a tokudb_analyze_time is in effect and the analysis has not reached the half way point of the index by the time tokudb_analyze_time/2 has been reached: it will stop the forward progress and restart the analysis from the last/rightmost row in the table, progressing leftwards and keeping/adding to the status information accumulated from the first half of the scan.

For small ratios of table_rows / tokudb_auto_analyze, auto analysis will be run for almost every change. The trigger formula is: `if (table_delta >= ((table_rows * tokudb_auto_analyze) / 100))` then run ANALYZE TABLE. If a user manually invokes an ANALYZE TABLE and tokudb_auto_analyze is enabled and there are no conflicting background jobs, the users ANALYZE TABLE will behave exactly as if the delta level has been exceeded in that the analysis is executed and delta reset to 0 upon completion.

49.3 System Variables

variable tokudb_analyze_in_background

Command Line Yes

Config File Yes

Scope Global/Session

Dynamic Yes

Variable Type Boolean

Default Value ON

When this variable is set to ON it will dispatch any ANALYZE TABLE job to a background process and return immediately, otherwise ANALYZE TABLE will run in foreground/client context.

variable tokudb_analyze_mode

Command Line Yes

Config File Yes

Scope Global/Session

Dynamic Yes

Variable Type ENUM

Default Value TOKUDB_ANALYZE_STANDARD

Range TOKUDB_ANALYZE_CANCEL, TOKUDB_ANALYZE_STANDARD, TOKUDB_ANALYZE_RECOUNT_ROWS

This variable is used to control the function of ANALYZE TABLE. Possible values are:

- TOKUDB_ANALYZE_CANCEL - Cancel any running or scheduled job on the specified table.
- TOKUDB_ANALYZE_STANDARD - Use existing analysis algorithm. This is the standard table cardinality analysis mode used to obtain cardinality statistics for a tables and its indexes. It will take the currently set session values for tokudb_analyze_time, tokudb_analyze_in_background, and tokudb_analyze_throttle at the time of its scheduling, either via a user invoked ANALYZE TABLE or an auto schedule as a result of tokudb_auto_analyze threshold being hit. Changing the global or session instances of these values after scheduling will have no effect on the scheduled job.

- TOKUDB_ANALYZE_RECOUNT_ROWS - Recount logical rows in table and update persistent count. This is a new mechanism that is used to perform a logical recount of all rows in a table and persist that as the basis value for the table row estimate. This mode was added for tables that have been upgraded from an older version of *TokuDB*/PerconaFT that only reported physical row counts and never had a proper logical row count. Newly created tables/partitions will begin counting logical rows correctly from their creation and should not need to be recounted unless some odd edge condition causes the logical count to become inaccurate over time. This analysis mode has no effect on the table cardinality counts. It will take the currently set session values for tokudb_analyze_in_background, and tokudb_analyze_throttle. Changing the global or session instances of these values after scheduling will have no effect on the job.

variable tokudb_analyze_throttle

Command Line Yes

Config File Yes

Scope Global/Session

Dynamic Yes

Variable Type Numeric

Default Value 0

This variable is used to define maximum number of keys to visit per second when performing ANALYZE TABLE with either a TOKUDB_ANALYZE_STANDARD or TOKUDB_ANALYZE_RECOUNT_ROWS.

variable tokudb_analyze_time

Command Line Yes

Config File Yes

Scope Global/Session

Dynamic Yes

Variable Type Numeric

Default Value 5

This session variable controls the number of seconds an analyze operation will spend on each index when calculating cardinality. Cardinality is shown by executing the following command:

```
SELECT INDEXES FROM table_name;
```

If an analyze is never performed on a table then the cardinality is 1 for primary key indexes and unique secondary indexes, and NULL (unknown) for all other indexes. Proper cardinality can lead to improved performance of complex SQL statements.

variable tokudb_auto_analyze

Command Line Yes

Config File Yes

Scope Global/Session

Dynamic Yes

Variable Type Numeric

Default Value 30

Percentage of table change as INSERT/UPDATE/DELETE commands to trigger an ANALYZE TABLE using the current session tokudb_analyze_in_background, tokudb_analyze_mode,

tokudb_analyze_throttle, and tokudb_analyze_time settings. If this variable is enabled and tokudb_analyze_in_background variable is set to OFF, analysis will be performed directly within the client thread context that triggered the analysis. **NOTE:** *InnoDB* enabled this functionality by default when they introduced it. Due to the potential unexpected new load it might place on a server, it is disabled by default in *TokuDB*.

variable tokudb_cardinality_scale_percent

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 100

Range 0-100

Percentage to scale table/index statistics when sending to the server to make an index appear to be either more or less unique than it actually is. *InnoDB* has a hard coded scaling factor of 50%. So if a table of 200 rows had an index with 40 unique values, InnoDB would return 200/40/2 or 2 for the index. The new TokuDB formula is the same but factored differently to use percent, for the same table.index (200/40 * tokudb_cardinality_scale) / 100, for a scale of 50% the result would also be 2 for the index.

49.4 INFORMATION_SCHEMA Tables

table INFORMATION_SCHEMA.**TOKUDB_BACKGROUND_JOB_STATUS**

Columns

- **id** – Simple monotonically incrementing job id, resets to 0 on server start.
- **database_name** – Database name
- **table_name** – Table name
- **job_type** – Type of job, either TOKUDB_ANALYZE_STANDARD or TOKUDB_ANALYZE_RECOUNT_ROWS
- **job_params** – Param values used by this job in string format. For example: TOKUDB_ANALYZE_DELETE_TIME=1.0; TOKUDB_ANALYZE_TIME=5; TOKUDB_ANALYZE_THROTTLE=2048;
- **scheduler** – Either USER or AUTO to indicate if the job was explicitly scheduled by a user or if it was scheduled as an automatic trigger
- **scheduled_time** – The time the job was scheduled
- **started_time** – The time the job was started
- **status** – Current job status if running. For example: ANALYZE TABLE standard db.tbl.idx 3 of 5 50% rows 10% time scanning forward

This table holds the information on scheduled and running background ANALYZE TABLE jobs for *TokuDB* tables.

49.5 Version Specific Information

- 5.7.10-1: Feature ported from *Percona Server* 5.6

- 5.7.11-4: tokudb_analyze_in_background is now set to ON by default and tokudb_auto_analyze is set to 30

CHAPTER

FIFTY

TOKUDB VARIABLES

Like all storage engines, *TokuDB* has variables to tune performance and control behavior. Fractal Tree algorithms are designed for near optimal performance and TokuDB's default settings should work well in most situations, eliminating the need for complex and time consuming tuning in most cases.

- Client Session Variables
- MySQL Server Variables

50.1 Client Session Variables

variable `unique_checks`

For tables with unique keys, every insertion into the table causes a lookup by key followed by an insertion, if the key is not in the table. This greatly limits insertion performance. If one knows by design that the rows being inserted into the table have unique keys, then one can disable the key lookup prior to insertion as follows:

```
SET unique_checks=OFF;
```

If your primary key is an auto-increment key, and none of your secondary keys are declared to be unique, then setting `unique_checks=OFF` will provide limited performance gains. On the other hand, if your primary key has a lot of entropy (it looks random), or your secondary keys are declared unique and have a lot of entropy, then disabling unique checks can provide a significant performance boost.

If `unique_checks` is disabled when the primary key is not unique, secondary indexes may become corrupted. In this case, the indexes should be dropped and rebuilt. This behavior differs from that of InnoDB, in which uniqueness is always checked on the primary key, and setting `unique_checks` to off turns off uniqueness checking on secondary indexes only. Turning off uniqueness checking on the primary key can provide large performance boosts, but it should only be done when the primary key is known to be unique.

variable `tokudb_commit_sync`

Session variable `tokudb_commit_sync` controls whether or not the transaction log is flushed when a transaction commits. The default behavior is that the transaction log is flushed by the commit. Flushing the transaction log requires a disk write and may adversely affect the performance of your application.

To disable synchronous flushing of the transaction log, disable the `tokudb_commit_sync` session variable as follows:

```
SET tokudb_commit_sync=OFF;
```

Disabling this variable may make the system run faster. However, transactions committed since the last checkpoint are not guaranteed to survive a crash.

variable tokudb_pk_insert_mode

This session variable controls the behavior of primary key insertions with the command `REPLACE INTO` and `INSERT IGNORE` on tables with no secondary indexes and on tables whose secondary keys whose every column is also a column of the primary key.

For instance, the table `(column_a INT, column_b INT, column_c INT, PRIMARY KEY (column_a,column_b), KEY (column_b))` is affected, because the only column in the key of `column_b` is present in the primary key. *TokuDB* can make these insertions really fast on these tables. However, triggers may not work and row based replication definitely will not work in this mode. This variable takes the following values, to control this behavior. This only applies to tables described above, using the command `REPLACE INTO` or `INSERT IGNORE`. All other scenarios are unaffected.

- `0`: Insertions are fast, regardless of whether triggers are defined on the table. `REPLACE INTO` and `INSERT IGNORE` statements fail if row based replication is enabled.
- `1` (default): Insertions are fast, if there are no triggers defined on the table. Insertions may be slow if triggers are defined on the table. `REPLACE INTO` and `INSERT IGNORE` statements fail if row based replication is enabled.
- `2`: Insertions are slow, all triggers on the table work, and row based replication works on `REPLACE INTO` and `INSERT IGNORE` statements.

variable tokudb_load_save_space

This session variable changes the behavior of the bulk loader. When it is disabled the bulk loader stores intermediate data using uncompressed files (which consumes additional CPU), whereas on compresses the intermediate files. It is enabled by default.

Note: The location of the temporary disk space used by the bulk loader may be specified with the `tokudb_tmp_dir` server variable.

If a `LOAD DATA INFILE` statement fails with the error message `ERROR 1030 (HY000): Got error 1` from storage engine, then there may not be enough disk space for the optimized loader, so disable `tokudb_prelock_empty` and try again.

More information is available in *Known Issues*.

variable tokudb_prelock_empty

By default, in `7.1.0`, *TokuDB* preemptively grabs an entire table lock on empty tables. If one transaction is doing the loading, such as when the user is doing a table load into an empty table, this default provides a considerable speedup.

However, if multiple transactions try to do concurrent operations on an empty table, all but one transaction will be locked out. Disabling `tokudb_prelock_empty` optimizes for this multi-transaction case by turning off preemptive pre-locking.

Note: If this variable is set to off, fast bulk loading is turned off as well.

---

variable tokudb_create_index_online

This variable controls whether indexes created with the `CREATE INDEX` command are hot (if enabled), or offline (if disabled). Hot index creation means that the table is available for inserts and queries while the index is being created. Offline index creation means that the table is not available for inserts and queries while the index is being created.

Note: Hot index creation is slower than offline index creation.

By default, `tokudb_create_index_online` is enabled.

variable tokudb_disable_slow_alter

This variable controls whether slow alter tables are allowed. For example, the following command is slow because HCADER does not allow a mixture of column additions, deletions, or expansions:

```
ALTER TABLE table
ADD COLUMN column_a INT,
DROP COLUMN column_b;
```

By default, `tokudb_disable_slow_alter` is disabled, and the engine reports back to mysql that this is unsupported resulting in the following output:

```
ERROR 1112 (42000): Table 'test_slow' uses an extension that doesn't exist in this MySQL version
```

variable `tokudb_block_size`

Fractal tree internal and leaf nodes default to 4,194,304 bytes (4 MB). The session variable `tokudb_block_size` controls the target uncompressed size of these nodes.

Changing the value of `tokudb_block_size` only affects subsequently created tables. The value of this variable cannot be changed for an existing table without a dump and reload.

variable `tokudb_read_block_size`

Fractal tree leaves are subdivided into read blocks, in order to speed up point queries. The session variable `tokudb_read_block_size` controls the target uncompressed size of the read blocks. The units are bytes and the default is 65,536 (64 KB). A smaller value favors read performance for point and small range scans over large range scans and higher compression. The minimum value of this variable is 4096.

Changing the value of `tokudb_read_block_size` only affects subsequently created tables. The value of this variable cannot be changed for an existing table without a dump and reload.

variable `tokudb_read_buf_size`

This variable controls the size of the buffer used to store values that are bulk fetched as part of a large range query. Its unit is bytes and its default value is 131,072 (128 KB).

A value of `0` turns off bulk fetching. Each client keeps a thread of this size, so it should be lowered if situations where there are a large number of clients simultaneously querying a table.

variable `tokudb_disable_prefetching`

TokuDB attempts to aggressively prefetch additional blocks of rows, which is helpful for most range queries but may create unnecessary IO for range queries with `LIMIT` clauses. Prefetching is on by default, with a value of `0`, and can be disabled by setting this variable to `1`.

variable `tokudb_row_format`

This session variable controls the default compression algorithm used to compress data when no row format is specified in the `CREATE TABLE` command. See *Compression Details*.

variable `tokudb_lock_timeout_debug`

The following values are available:

0 No lock timeouts or lock deadlocks are reported.

1 A JSON document that describes the lock conflict is stored in the `tokudb_last_lock_timeout` session variable

2 A JSON document that describes the lock conflict is printed to the MySQL error log.

Supported since 7.5.5: In addition to the JSON document describing the lock conflict, the following lines are printed to the MySQL error log:

- A line containing the blocked thread id and blocked sql

- A line containing the blocking thread id and the blocking sql.

3 A JSON document that describes the lock conflict is stored in the `tokudb_last_lock_timeout` session variable and is printed to the MySQL error log.

Supported since 7.5.5: In addition to the JSON document describing the lock conflict, the following lines are printed to the MySQL error log:

- A line containing the blocked thread id and blocked sql
- A line containing the blocking thread id and the blocking sql.

variable `tokudb_last_lock_timeout`

This session variable contains a JSON document that describes the last lock conflict seen by the current MySQL client. It gets set when a blocked lock request times out or a lock deadlock is detected.

The `tokudb_lock_timeout_debug` session variable must have bit 0 set for this behavior, otherwise this session variable will be empty.

variable `tokudb_bulk_fetch`

This session variable determines if our bulk fetch algorithm is used for `SELECT` and `DELETE` statements. `SELECT` statements include pure `SELECT ...` statements, as well as `INSERT INTO table-name ... SELECT ...`, `CREATE TABLE table-name ... SELECT ...`, `REPLACE INTO table-name ... SELECT ...`, `INSERT IGNORE INTO table-name ... SELECT ...`, and `INSERT INTO table-name ... SELECT ... ON DUPLICATE KEY UPDATE`.

By default, `tokudb_bulk_fetch` is enabled.

variable `tokudb_support_xa`

This session variable defines whether or not the prepare phase of an XA transaction performs an `fsync()`.

By default, `tokudb_support_xa` is enabled.

variable `tokudb_optimize_throttling`
Supported since 7.5.5

By default, table optimization will run with all available resources. To limit the amount of resources, it is possible to limit the speed of table optimization. The `tokudb_optimize_throttling` session variable determines an upper bound on how many fractal tree leaf nodes per second are optimized. The default is 0 (no upper bound) with a valid range of [0,1000000].

variable `tokudb_optimize_index_name`
Supported since 7.5.5

To optimize a single index in a table, the `tokudb_optimize_index_name` session variable can be enabled to select the index by name.

variable `tokudb_optimize_index_fraction`
Supported since 7.5.5

For patterns where the left side of the tree has many deletions (a common pattern with increasing id or date values), it may be useful to delete a percentage of the tree. In this case, it's possible to optimize a subset of a fractal tree starting at the left side. The `tokudb_optimize_index_fraction` session variable controls the size of the sub tree. Valid values are in the range [0.0,1.0] with default 1.0 (optimize the whole tree).

variable `tokudb_backup_throttle`

This session level variable throttles the write rate in bytes per second of the backup to prevent Hot Backup from crowding out other jobs in the system. The default and max values are 18446744073709551615

variable tokudb_backup_dir
Supported since 7.5.5

When enabled, this session level variable serves two purposes, to point to the destination directory where the backups will be dumped and to kick off the backup as soon as it is set.

variable tokudb_backup_exclude

Command Line Yes

Config File Yes

Scope Global/Session

Dynamic Yes

Variable Type String

Default Value `(mysqld_safe\.pid)+`

Use this variable to set a regular expression that defines source files excluded from backup. For example, to exclude all `lost+found` directories, use the following command:

```
mysql> set tokudb_backup_exclude='/lost\\+found($|/)';
```

variable tokudb_backup_last_error
Supported since 7.5.5

This session variable will contain the error number from the last backup. 0 indicates success.

variable tokudb_backup_last_error_string
Supported since 7.5.5

This session variable will contain the error string from the last backup.

50.2 MySQL Server Variables

variable tokudb_loader_memory_size

Limits the amount of memory that the *TokuDB* bulk loader will use for each loader instance, defaults to 100 MB. Increasing this value may provide a performance benefit when loading extremely large tables with several secondary indexes.

Note: Memory allocated to a loader is taken from the TokuDB cache, defined as `tokudb_cache_size`, and may impact the running workload's performance as existing cached data must be ejected for the loader to begin.

---

variable tokudb_fsync_log_period

Controls the frequency, in milliseconds, for `fsync()` operations. If set to 0 then the `fsync()` behavior is only controlled by the `tokudb_commit_sync`, which is on or off. The default values is `0`.

variable tokudb_cache_size

This variable configures the size in bytes of the *TokuDB* cache table. The default cache table size is 1/2 of physical memory. Percona highly recommends using the default setting if using buffered IO, if using direct IO then consider setting this parameter to 80% of available memory.

Consider decreasing `tokudb_cache_size` if excessive swapping is causing performance problems. Swapping may occur when running multiple mysql server instances or if other running applications use large amounts of physical memory.

variable tokudb_directio

When enabled, *TokuDB* employs Direct IO rather than Buffered IO for writes. When using Direct IO, consider increasing `tokudb_cache_size` from its default of 1/2 physical memory.

By default, `tokudb_directio` is disabled.

variable `tokudb_lock_timeout`

This variable controls the amount of time that a transaction will wait for a lock held by another transaction to be released. If the conflicting transaction does not release the lock within the lock timeout, the transaction that was waiting for the lock will get a lock timeout error. The units are milliseconds. A value of 0 disables lock waits. The default value is 4000 (four seconds).

If your application gets a `lock wait timeout` error (-30994), then you may find that increasing the `tokudb_lock_timeout` may help. If your application gets a `deadlock found` error (-30995), then you need to abort the current transaction and retry it.

variable `tokudb_data_dir`

This variable configures the directory name where the *TokuDB* tables are stored. The default location is the *MySQL* data directory.

variable `tokudb_log_dir`

This variable specifies the directory where the *TokuDB* log files are stored. The default location is the MySQL data directory. Configuring a separate log directory is somewhat involved. Please contact Percona support for more details.

Warning: After changing *TokuDB* log directory path, the old *TokuDB* recovery log file should be moved to new directory prior to start of *MySQL* server and log file's owner must be the `mysql` user. Otherwise server will fail to initialize the *TokuDB* store engine restart.

variable `tokudb_tmp_dir`

This variable specifies the directory where the *TokuDB* bulk loader stores temporary files. The bulk loader can create large temporary files while it is loading a table, so putting these temporary files on a disk separate from the data directory can be useful.

For example, it can make sense to use a high-performance disk for the data directory and a very inexpensive disk for the temporary directory. The default location for temporary files is the *MySQL* data directory.

variable `tokudb_checkpointing_period`

This variable specifies the time in seconds between the beginning of one checkpoint and the beginning of the next. The default time between *TokuDB* checkpoints is 60 seconds. We recommend leaving this variable unchanged.

variable `tokudb_write_status_frequency`

TokuDB shows statement progress of queries, inserts, deletes, and updates in `SHOW PROCESSLIST`. Queries are defined as reads, and inserts, deletes, and updates are defined as writes.

Progress for updated is controlled by `tokudb_write_status_frequency`, which is set to 1000, that is, progress is measured every 1000 writes.

For slow queries, it can be helpful to set this variable and `tokudb_read_status_frequency` to 1, and then run `SHOW PROCESSLIST` several times to understand what progress is being made.

variable `tokudb_read_status_frequency`

TokuDB shows statement progress of queries, inserts, deletes, and updates in `SHOW PROCESSLIST`. Queries are defined as reads, and inserts, deletes, and updates are defined as writes.

Progress for reads is controlled by `tokudb_read_status_frequency` which is set to 10,000.

For slow queries, it can be helpful to set this variable and `tokudb_write_status_frequency` to 1, and then run `SHOW PROCESSLIST` several times to understand what progress is being made.

variable tokudb_fs_reserve_percent

This variable controls the percentage of the file system that must be available for inserts to be allowed. By default, this is set to 5. We recommend that this reserve be at least half the size of your physical memory. See *Full Disks* for more information.

variable tokudb_cleaner_period

This variable specifies how often in seconds the cleaner thread runs. The default value is 1. Setting this variable to 0 turns off cleaner threads.

variable tokudb_cleaner_iterations

This variable specifies how many internal nodes get processed in each `tokudb_cleaner_period` period. The default value is 5. Setting this variable to 0 turns off cleaner threads.

variable tokudb_backup_throttle

This variable specifies the maximum number of bytes per second the copier of a hot backup process will consume. Lowering its value will cause the hot backup operation to take more time but consume less IO on the server. The default value is `18446744073709551615`.

```
mysql> set tokudb_backup_throttle=1000000;
```

variable tokudb_rpl_lookup_rows

When disabled, *TokuDB* replication slaves skip row lookups for *delete row* log events and *update row* log events, which eliminates all associated read IO for these operations.

Warning: *TokuDB* Read Free Replication will not propagate `UPDATE` and `DELETE` events reliably if *TokuDB* table is missing the primary key which will eventually lead to data inconsistency on the slave.

Note: Optimization is only enabled when `read_only` is `1` and `binlog_format` is `ROW`.

By default, `tokudb_rpl_lookup_rows` is enabled.

variable tokudb_rpl_lookup_rows_delay

This server variable allows for simulation of long disk reads by sleeping for the given number of microseconds prior to the row lookup query, it should only be set to a non-zero value for testing.

By default, `tokudb_rpl_lookup_rows_delay` is disabled.

variable tokudb_rpl_unique_checks

When disabled, *TokuDB* replication slaves skip uniqueness checks on inserts and updates, which eliminates all associated read IO for these operations.

Note: Optimization is only enabled when `read_only` is `1` and `binlog_format` is `ROW`.

By default, `tokudb_rpl_unique_checks` is enabled.

variable tokudb_rpl_unique_checks_delay

This server variable allows for simulation of long disk reads by sleeping for the given number of microseconds prior to the row lookup query, it should only be set to a non-zero value for testing.

By default, `tokudb_rpl_unique_checks_delay` is disabled.

variable tokudb_backup_plugin_version
Supported since 7.5.5:

This server variable documents the version of the *TokuBackup* plugin

variable `tokudb_backup_version`
Supported since 7.5.5:

This server variable documents the version of the hot backup library.

variable `tokudb_backup_allowed_prefix`
Supported since 7.5.5:

This system-level variable restricts the location of the destination directory where the backups can be located. Attempts to backup to a location outside of the directory this variable points to or its children will result in an error.

The default is null, backups have no restricted locations. This read only variable can be set in the `my.cnf` file and displayed with the `SHOW VARIABLES` command.

```
mysql> SHOW VARIABLES like 'tokudb_backup_allowed_prefix';
+-----------------------------+-----------+
| Variable_name               | Value     |
+-----------------------------+-----------+
| tokudb_backup_allowed_prefix | /dumpdir  |
+-----------------------------+-----------+
```

variable `tokudb_rpl_check_readonly`
Supported since 7.5.5:

The *TokuDB* replication code will run row events from the binlog with RFR when the slave is in read only mode. The `tokudb_rpl_check_readonly` variable is used to disable the slave read only check in the *TokuDB* replication code.

This allows RFR to run when the slave is NOT read only. By default, `tokudb_rpl_check_readonly` is enabled (check slave read only). Do NOT change this value unless you completely understand the implications!

variable `tokudb_fanout`

Command Line Yes

Config File Yes

Scope Session/Global

Dynamic Yes

Variable Type Numeric

Range 2-16384

Default Value 16

This variable controls the Fractal Tree fanout.

variable `tokudb_client_pool_threads`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Numeric

Range 0 - 1024

Default Value 0

variable `tokudb_cachetable_pool_threads`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Numeric

Range 0 - 1024

Default Value 0

variable `tokudb_checkpoint_pool_threads`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Numeric

Range 0 - 1024

Default Value 0

variable `tokudb_enable_partial_eviction`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Boolean

Range ON/OFF

Default Value ON

This variable is used to control if partial eviction of nodes is enabled or disabled.

variable `tokudb_compress_buffers_before_eviction`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Boolean

Range ON/OFF

Default Value ON

When this variable is enabled it allows the evictor to compress unused internal node partitions in order to reduce memory requirements as a first step of partial eviction before fully evicting the partition and eventually the entire node.

variable `tokudb_strip_frm_data`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Boolean

Range ON/OFF

Default Value OFF

When this variable is set to `ON` during the startup server will check all the status files and remove the embedded `.frm` metadata. This variable can be used to assist in *TokuDB* data recovery. **WARNING:** Use this variable only if you know what you're doing otherwise it could lead to data loss.

CHAPTER

FIFTYONE

TOKUDB TROUBLESHOOTING

- Known Issues
- Lock Visualization in TokuDB
- Engine Status
- Global Status

51.1 Known Issues

Replication and binary logging: *TokuDB* supports binary logging and replication, with one restriction. *TokuDB* does not implement a lock on the auto-increment function, so concurrent insert statements with one or more of the statements inserting multiple rows may result in a non-deterministic interleaving of the auto-increment values. When running replication with these concurrent inserts, the auto-increment values on the slave table may not match the auto-increment values on the master table. Note that this is only an issue with Statement Based Replication (SBR), and not Row Based Replication (RBR).

For more information about auto-increment and replication, see the *MySQL* Reference Manual: AUTO_INCREMENT handling in InnoDB.

In addition, when using the `REPLACE INTO` or `INSERT IGNORE` on tables with no secondary indexes or tables where secondary indexes are subsets of the primary, the session variable `tokudb_pk_insert_mode` controls whether row based replication will work.

Uninformative error message: The `LOAD DATA INFILE` command can sometimes produce `ERROR 1030 (HY000): Got error 1 from storage engine`. The message should say that the error is caused by insufficient disk space for the temporary files created by the loader.

Transparent Huge Pages: *TokuDB* will refuse to start if transparent huge pages are enabled. Transparent huge page support can be disabled by issuing the following as root:

```
# echo never > /sys/kernel/mm/redhat_transparent_hugepage/enabled
```

Note: The previous command needs to be executed after every reboot, because it defaults to `always`.

XA behavior vs. InnoDB: *InnoDB* forces a deadlocked XA transaction to abort, *TokuDB* does not.

51.2 Lock Visualization in TokuDB

TokuDB uses key range locks to implement serializable transactions, which are acquired as the transaction progresses. The locks are released when the transaction commits or aborts (this implements two phase locking).

TokuDB stores these locks in a data structure called the lock tree. The lock tree stores the set of range locks granted to each transaction. In addition, the lock tree stores the set of locks that are not granted due to a conflict with locks granted to some other transaction. When these other transactions are retired, these pending lock requests are retried. If a pending lock request is not granted before the lock timer expires, then the lock request is aborted.

Lock visualization in *TokuDB* exposes the state of the lock tree with tables in the information schema. We also provide a mechanism that may be used by a database client to retrieve details about lock conflicts that it encountered while executing a transaction.

51.2.1 The `TOKUDB_TRX` table

The `TOKUDB_TRX` table in the `INFORMATION_SCHEMA` maps *TokuDB* transaction identifiers to *MySQL* client identifiers. This mapping allows one to associate a *TokuDB* transaction with a *MySQL* client operation.

The following query returns the *MySQL* clients that have a live *TokuDB* transaction:

```
SELECT * FROM INFORMATION_SCHEMA.TOKUDB_TRX,
  INFORMATION_SCHEMA.PROCESSLIST
  WHERE trx_mysql_thread_id = id;
```

51.2.2 The `TOKUDB_LOCKS` table

The `tokudb_locks` table in the information schema contains the set of locks granted to *TokuDB* transactions.

The following query returns all of the locks granted to some *TokuDB* transaction:

```
SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCKS;
```

The following query returns the locks granted to some *MySQL* client:

```
SELECT id FROM INFORMATION_SCHEMA.TOKUDB_LOCKS,
  INFORMATION_SCHEMA.PROCESSLIST
  WHERE locks_mysql_thread_id = id;
```

51.2.3 The `TOKUDB_LOCK_WAITS` table

The `tokudb_lock_waits` table in the information schema contains the set of lock requests that are not granted due to a lock conflict with some other transaction.

The following query returns the locks that are waiting to be granted due to a lock conflict with some other transaction:

```
SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCK_WAITS;
```

51.2.4 The `tokudb_lock_timeout_debug` session variable

The `tokudb_lock_timeout_debug` session variable controls how lock timeouts and lock deadlocks seen by the database client are reported.

The following values are available:

0 No lock timeouts or lock deadlocks are reported.

1 A JSON document that describes the lock conflict is stored in the `tokudb_last_lock_timeout` session variable

2 A JSON document that describes the lock conflict is printed to the *MySQL* error log.

Supported since 7.5.5: In addition to the JSON document describing the lock conflict, the following lines are printed to the MySQL error log:

- A line containing the blocked thread id and blocked SQL
- A line containing the blocking thread id and the blocking SQL.

3 A JSON document that describes the lock conflict is stored in the `tokudb_last_lock_timeout` session variable and is printed to the *MySQL* error log.

Supported since 7.5.5: In addition to the JSON document describing the lock conflict, the following lines are printed to the *MySQL* error log:

- A line containing the blocked thread id and blocked SQL
- A line containing the blocking thread id and the blocking SQL.

51.2.5 The `tokudb_last_lock_timeout` session variable

The `tokudb_last_lock_timeout` session variable contains a JSON document that describes the last lock conflict seen by the current *MySQL* client. It gets set when a blocked lock request times out or a lock deadlock is detected. The `tokudb_lock_timeout_debug` session variable should have bit `0` set (decimal `1`).

51.2.6 Example

Suppose that we create a table with a single column that is the primary key.

```
mysql> SHOW CREATE TABLE table;

Create Table: CREATE TABLE `table` (
`id` int(11) NOT NULL,
PRIMARY KEY (`id`)) ENGINE=TokuDB DEFAULT CHARSET=latin1
```

Suppose that we have 2 *MySQL* clients with ID's 1 and 2 respectively. Suppose that *MySQL* client 1 inserts some values into `table`. *TokuDB* transaction 51 is created for the insert statement. Since autocommit is disabled, transaction 51 is still live after the insert statement completes, and we can query the `tokudb_locks` table in information schema to see the locks that are held by the transaction.

```
mysql> SET AUTOCOMMIT=OFF;
mysql> INSERT INTO table VALUES (1),(10),(100);

Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCKS;

+--------------+-----------------------+---------------+----------------+-----------------+----------
| locks_trx_id | locks_mysql_thread_id | locks_dname   | locks_key_left | locks_key_right | locks_tab
+--------------+-----------------------+---------------+----------------+-----------------+----------
|           51 |                     1 | ./test/t-main | 0001000000     | 0001000000      | test
|           51 |                     1 | ./test/t-main | 000a000000     | 000a000000      | test
|           51 |                     1 | ./test/t-main | 0064000000     | 0064000000      | test
+--------------+-----------------------+---------------+----------------+-----------------+----------

mysql> SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCK_WAITS;

Empty set (0.00 sec)
```

The keys are currently hex dumped.

Now we switch to the other *MySQL* client with ID 2.

```
mysql> INSERT INTO table VALUES (2),(20),(100);
```

The insert gets blocked since there is a conflict on the primary key with value 100.

The granted *TokuDB* locks are:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCKS;
```

```
+--------------+-----------------------+---------------+----------------+-----------------+---------
| locks_trx_id | locks_mysql_thread_id | locks_dname   | locks_key_left | locks_key_right | locks_tab
+--------------+-----------------------+---------------+----------------+-----------------+---------
|           51 |                     1 | ./test/t-main | 0001000000     | 0001000000      | test
|           51 |                     1 | ./test/t-main | 000a000000     | 000a000000      | test
|           51 |                     1 | ./test/t-main | 0064000000     | 0064000000      | test
|           51 |                     1 | ./test/t-main | 0002000000     | 0002000000      | test
|           51 |                     1 | ./test/t-main | 0014000000     | 0014000000      | test
+--------------+-----------------------+---------------+----------------+-----------------+---------
```

The locks that are pending due to a conflict are:

```
SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCK_WAITS;
```

```
+-------------------+------------------+------------------+---------------------+---------------------
| requesting_trx_id | blocking_trx_id | lock_waits_dname | lock_waits_key_left | lock_waits_key_right
+-------------------+------------------+------------------+---------------------+---------------------
|                62 |               51 | ./test/t-main    | 0064000000          | 0064000000
+-------------------+------------------+------------------+---------------------+---------------------
```

Eventually, the lock for client 2 times out, and we can retrieve a JSON document that describes the conflict.

```
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
```

```
mysql> SELECT @@TOKUDB_LAST_LOCK_TIMEOUT;
```

```
+-----------------------------------------------------------------------------------------------------
| @@tokudb_last_lock_timeout
+-----------------------------------------------------------------------------------------------------
| "mysql_thread_id":2, "dbname":"./test/t-main", "requesting_txnid":62, "blocking_txnid":51, "key":"0
+-----------------------------------------------------------------------------------------------------
```

```
ROLLBACK;
```

Since transaction 62 was rolled back, all of the locks taken by it are released.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCKS;
```

```
+--------------+-----------------------+---------------+----------------+-----------------+---------
| locks_trx_id | locks_mysql_thread_id | locks_dname   | locks_key_left | locks_key_right | locks_tab
+--------------+-----------------------+---------------+----------------+-----------------+---------
|           51 |                     1 | ./test/t-main | 0001000000     | 0001000000      | test
|           51 |                     1 | ./test/t-main | 000a000000     | 000a000000      | test
|           51 |                     1 | ./test/t-main | 0064000000     | 0064000000      | test
|           51 |                     2 | ./test/t-main | 0002000000     | 0002000000      | test
|           51 |                     2 | ./test/t-main | 0014000000     | 0014000000      | test
+--------------+-----------------------+---------------+----------------+-----------------+---------
```

51.3 Engine Status

Engine status provides details about the inner workings of *TokuDB* and can be useful in tuning your particular environment. The engine status can be determined by running the following command:

```
SHOW ENGINE tokudb STATUS;
```

The following is a reference of table status statements:

cachetable: cleaner executions Total number of times the cleaner thread loop has executed.

cachetable: cleaner iterations This is the number of cleaner operations that are performed every cleaner period.

cachetable: cleaner period *TokuDB* includes a cleaner thread that optimizes indexes in the background. This variable is the time, in seconds, between the completion of a group of cleaner operations and the beginning of the next group of cleaner operations. The cleaner operations run on a background thread performing work that does not need to be done on the client thread.

cachetable: evictions Number of blocks evicted from cache.

cachetable: long time waiting on cache pressure Total time, in microseconds, waiting on cache pressure to subside for more than 1 second.

cachetable: miss This is a count of how many times the application was unable to access your data in the internal cache.

cachetable: miss time This is the total time, in microseconds, of how long the database has had to wait for a disk read to complete.

cachetable: number of long waits on cache pressure The number of times a thread was stalled for more than 1 second due to cache pressure.

cachetable: number of waits on cache pressure The number of times a thread was stalled due to cache pressure.

cachetable: prefetches This is the total number of times that a block of memory has been prefetched into the database's cache. Data is prefetched when the database's algorithms determine that a block of memory is likely to be accessed by the application.

cachetable: size cachepressure The number of bytes causing cache pressure (the sum of buffers and work done counters), helps to understand if cleaner threads are keeping up with workload.

cachetable: size current This is a count, in bytes, of how much of your uncompressed data is currently in the database's internal cache.

cachetable: size currently cloned data for checkpoint Amount of memory, in bytes, currently used for cloned nodes. During the checkpoint operation, dirty nodes are cloned prior to serialization/compression, then written to disk. After which, the memory for the cloned block is returned for re-use.

cachetable: size leaf The number of bytes of leaf nodes in the cache.

cachetable: size limit This is a count, in bytes, of how much of your uncompressed data will fit in the database's internal cache.

cachetable: size nonleaf The number of bytes of non-leaf nodes in the cache.

cachetable: size rollback The number of bytes of rollback nodes in the cache.

cachetable: size writing This is the number of bytes that are currently queued up to be written to disk.

cachetable: time waiting on cache pressure Total time, in microseconds, waiting on cache pressure to subside.

checkpoint: begin time Cumulative time (in microseconds) required to mark all dirty nodes as pending a checkpoint.

checkpoint: checkpoints failed This is the number of checkpoints that have failed for any reason.

checkpoint: checkpoints taken This is the number of complete checkpoints that have been taken.

checkpoint: footprint Where the database is in the checkpoint process.

checkpoint: last checkpoint began This is the time the last checkpoint began. If a checkpoint is currently in progress, then this time may be later than the time the last checkpoint completed.

Note: If no checkpoint has ever taken place, then this value will be `Dec 31, 1969` on Linux hosts.

---

checkpoint: last complete checkpoint began This is the time the last complete checkpoint started. Any data that changed after this time will not be captured in the checkpoint.

checkpoint: last complete checkpoint ended This is the time the last complete checkpoint ended.

checkpoint: last complete checkpoint LSN This is the Log Sequence Number of the last complete checkpoint.

checkpoint: long checkpoint begin count The total number of times a checkpoint begin took more than 1 second.

checkpoint: long checkpoint begin time The total time, in microseconds, of long checkpoint begins. A long checkpoint begin is one taking more than 1 second.

checkpoint: non-checkpoint client wait on cs lock The number of times a non-checkpoint client thread waited for the checkpoint-safe lock.

checkpoint: non-checkpoint client wait on mo lock The number of times a non-checkpoint client thread waited for the multi-operation lock.

checkpoint: period This is the interval in seconds between the end of an automatic checkpoint and the beginning of the next automatic checkpoint.

checkpoint: time spent during checkpoint (begin and end phases) Time (in seconds) required to complete all checkpoints.

checkpoint: time spent during last checkpoint (begin and end phases) Time (in seconds) required to complete the last checkpoint.

checkpoint: waiters max This is the maximum number of threads ever simultaneously waiting for the checkpoint-safe lock to perform a checkpoint.

checkpoint: waiters now This is the current number of threads simultaneously waiting for the checkpoint-safe lock to perform a checkpoint.

checkpoint: checkpoint end time The time spent in checkpoint end operation in seconds.

checkpoint: long checkpoint end time The time spent in checkpoint end operation in seconds.

checkpoint: long checkpoint end count This is the count of end_checkpoint operations that exceeded 1 minute.

context: promotion blocked by a flush Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a buffer flush from parent to child.

context: promotion blocked by a full eviction (should never happen) Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a full eviction.

context: promotion blocked by a full fetch (should never happen) Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a full fetch.

context: promotion blocked by a message application Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of message application (applying fresh ancestors messages to a basement node).

context: promotion blocked by a message injection Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of message injection.

context: promotion blocked by a partial eviction (should never happen) Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a partial eviction.

context: promotion blocked by a partial fetch (should never happen) Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a partial fetch.

context: promotion blocked by something uninstrumented Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of something uninstrumented.

context: promotion blocked by the cleaner thread Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a cleaner thread.

context: something uninstrumented blocked by something uninstrumented Number of times node `rwlock` contention was observed for an uninstrumented process because of something uninstrumented.

context: tree traversals blocked by a flush Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a buffer flush from parent to child.

context: tree traversals blocked by a full eviction Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a full eviction.

context: tree traversals blocked by a full fetch Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a full fetch.

context: tree traversals blocked by a message application Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of message application (applying fresh ancestors messages to a basement node).

context: tree traversals blocked by a message injection Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of message injection.

context: tree traversals blocked by a partial eviction Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a partial eviction.

context: tree traversals blocked by a partial fetch Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a partial fetch.

context: tree traversals blocked by a the cleaner thread Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a cleaner thread.

context: tree traversals blocked by something uninstrumented Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of something uninstrumented.

db closes Number of db close operations.

db opens Number of db open operations.

dictionary broadcast updates This is the number of broadcast updates that have been successfully performed. A broadcast update is an update that affects all rows in a dictionary.

dictionary broadcast updates fail This is the number of broadcast updates that have failed.

dictionary deletes This is the total number of rows that have been deleted from all primary and secondary indexes combined, if those deletes have been done with a separate recovery log entry per index.

dictionary deletes fail This is the number of single-index delete operations that failed.

dictionary inserts This is the total number of rows that have been inserted into all primary and secondary indexes combined, when those inserts have been done with a separate recovery log entry per index. For example, inserting a row into a table with one primary and two secondary indexes will increase this count by three, if the inserts were done with separate recovery log entries.

dictionary inserts fail This is the number of single-index insert operations that failed.

dictionary multi deletes This is the total number of rows that have been deleted from all primary and secondary indexes combined, when those deletes have been done with a single recovery log entry for the entire row.

dictionary multi deletes fail This is the number of multi-index delete operations that failed.

dictionary multi inserts This is the total number of rows that have been inserted into all primary and secondary indexes combined, when those inserts have been done with a single recovery log entry for the entire row. (For example, inserting a row into a table with one primary and two secondary indexes will normally increase this count by three).

dictionary multi inserts fail This is the number of multi-index insert operations that failed.

dictionary multi updates This is the total number of rows that have been updated in all primary and secondary indexes combined, if those updates have been done with a single recovery log entry for the entire row.

dictionary multi updates fail This is the number of multi-index update operations that failed.

dictionary updates This is the total number of rows that have been updated in all primary and secondary indexes combined, if those updates have been done with a separate recovery log entry per index.

dictionary updates fail This is the number of single-index update operations that failed.

disk free space This is a gross estimate of how much of your file system is available. Possible displays in this field are:

- More than twice the reserve ("more than 10 percent of total file system space")
- Less than twice the reserve
- Less than the reserve
- File system is completely full

filesystem: ENOSPC redzone state The state of how much disk space exists with respect to the red zone value. Valid values are:

0 Space is available

1 Warning, with 2x of redzone value. Operations are allowed, but engine status prints a warning.

2 In red zone, insert operations are blocked

3 All operations are blocked

filesystem: fsync count This is the total number of times the database has flushed the operating system's file buffers to disk.

filesystem: fsync time This the total time, in microseconds, used to fsync to disk.

filesystem: long fsync count This is the total number of times the database has flushed the operating system's file buffers to disk and this operation required more than 1 second.

filesystem: long fsync time This the total time, in microseconds, used to fsync to disk when the operation required more than 1 second.

filesystem: most recent disk full This is the most recent time when the disk file system was entirely full. If the disk has never been full, then this value will be "Dec 31, 1969" on Linux hosts.

filesystem: number of operations rejected by enospc prevention (red zone) This is the number of database inserts that have been rejected because the amount of disk free space was less than the reserve.

filesystem: number of write operations that returned ENOSPC This is the number of times that an attempt to write to disk failed because the disk was full. If the disk is full, this number will continue increasing until space is available.

filesystem: threads currently blocked by full disk This is the number of threads that are currently blocked because they are attempting to write to a full disk. This is normally zero. If this value is non-zero, then a warning will appear in the "disk free space" field.

ft: basements decompressed as a target of a query Number of basement nodes decompressed for queries.

ft: basements decompressed for prefetch Number of basement nodes decompressed by a prefetch thread.

ft: basements decompressed for prelocked range Number of basement nodes decompressed by queries aggressively.

ft: basements decompressed for write Number of basement nodes decompressed for writes.

ft: basement nodes deserialized with fixed-keysize The number of basement nodes deserialized where all keys had the same size, leaving the basement in a format that is optimal for in-memory workloads.

ft: basement nodes deserialized with variable-keysize The number of basement nodes deserialized where all keys did not have the same size, and thus ineligible for an in-memory optimization.

ft: basements fetched as a target of a query (bytes) Number of basement node bytes fetched from disk for queries.

ft: basements fetched as a target of a query Number of basement nodes fetched from disk for queries.

ft: basements fetched as a target of a query (seconds) Number of seconds waiting for IO when fetching basement nodes from disk for queries.

ft: basements fetched for prefetch (bytes) Number of basement node bytes fetched from disk by a prefetch thread.

ft: basements fetched for prefetch Number of basement nodes fetched from disk by a prefetch thread.

ft: basements fetched for prefetch (seconds) Number of seconds waiting for IO when fetching basement nodes from disk by a prefetch thread.

ft: basements fetched for prelocked range (bytes) Number of basement node bytes fetched from disk aggressively.

ft: basements fetched for prelocked range Number of basement nodes fetched from disk aggressively.

ft: basements fetched for prelocked range (seconds) Number of seconds waiting for IO when fetching basement nodes from disk aggressively.

ft: basements fetched for write (bytes) Number of basement node bytes fetched from disk for writes.

ft: basements fetched for write Number of basement nodes fetched from disk for writes.

ft: basements fetched for write (seconds) Number of seconds waiting for IO when fetching basement nodes from disk for writes.

ft: broadcast messages injected at root How many broadcast messages injected at root.

ft: buffers decompressed as a target of a query Number of buffers decompressed for queries.

ft: buffers decompressed for prefetch Number of buffers decompressed by a prefetch thread.

ft: buffers decompressed for prelocked range Number of buffers decompressed by queries aggressively.

ft: buffers decompressed for write Number of buffers decompressed for writes.

ft: buffers fetched as a target of a query (bytes) Number of buffer bytes fetched from disk for queries.

ft: buffers fetched as a target of a query Number of buffers fetched from disk for queries.

ft: buffers fetched as a target of a query (seconds) Number of seconds waiting for IO when fetching buffers from disk for queries.

ft: buffers fetched for prefetch (bytes) Number of buffer bytes fetched from disk by a prefetch thread.

ft: buffers fetched for prefetch Number of buffers fetched from disk by a prefetch thread.

ft: buffers fetched for prefetch (seconds) Number of seconds waiting for IO when fetching buffers from disk by a prefetch thread.

ft: buffers fetched for prelocked range (bytes) Number of buffer bytes fetched from disk aggressively.

ft: buffers fetched for prelocked range Number of buffers fetched from disk aggressively.

ft: buffers fetched for prelocked range (seconds) Number of seconds waiting for IO when fetching buffers from disk aggressively.

ft: buffers fetched for write (bytes) Number of buffer bytes fetched from disk for writes.

ft: buffers fetched for write Number of buffers fetched from disk for writes.

ft: buffers fetched for write (seconds) Number of seconds waiting for IO when fetching buffers from disk for writes.

ft: bytes of messages currently in trees (estimate) How many bytes of messages currently in trees (estimate).

ft: bytes of messages flushed from h1 nodes to leaves How many bytes of messages flushed from h1 nodes to leaves.

ft: bytes of messages injected at root (all trees) How many bytes of messages injected at root (for all trees).

ft: descriptor set This is the number of time a descriptor was updated when the entire dictionary was updated (for example, when the schema has been changed).

ft: leaf compression to memory (seconds) Total time, in seconds, spent compressing leaf nodes.

ft: leaf decompression to memory (seconds) Total time, in seconds, spent decompressing leaf nodes.

ft: leaf deserialization to memory (seconds) Total time, in seconds, spent deserializing leaf nodes.

ft: leaf node full evictions (bytes) The number of bytes freed by evicting full leaf nodes from the cache.

ft: leaf node full evictions The number of times a full leaf node was evicted from the cache.

ft: leaf node partial evictions (bytes) The number of bytes freed by evicting partitions of leaf nodes from the cache.

ft: leaf node partial evictions The number of times a partition of a leaf node was evicted from the cache.

ft: leaf nodes created Number of leaf nodes created.

ft: leaf nodes destroyed Number of leaf nodes destroyed.

ft: leaf nodes flushed to disk (for checkpoint) (bytes) Number of bytes of leaf nodes flushed to disk for checkpoint.

ft: leaf nodes flushed to disk (for checkpoint) Number of leaf nodes flushed to disk for checkpoint.

ft: leaf nodes flushed to disk (for checkpoint) (seconds) Number of seconds waiting for IO when writing leaf nodes flushed to disk for checkpoint.

ft: leaf nodes flushed to disk (for checkpoint) (uncompressed bytes) Number of uncompressed bytes of leaf nodes flushed to disk for checkpoint.

ft: leaf nodes flushed to disk (not for checkpoint) (bytes) Number of bytes of leaf nodes flushed to disk, not for checkpoint.

ft: leaf nodes flushed to disk (not for checkpoint) Number of leaf nodes flushed to disk, not for checkpoint.

ft: leaf nodes flushed to disk (not for checkpoint) (seconds) Number of seconds waiting for IO when writing leaf nodes flushed to disk, not for checkpoint.

ft: leaf nodes flushed to disk (not for checkpoint) (uncompressed bytes) Number of bytes of leaf nodes flushed to disk, not for checkpoint.

ft: leaf serialization to memory (seconds) Total time, in seconds, spent serializing leaf nodes.

ft: messages ignored by leaf due to msn The number of messages that were ignored by a leaf because it had already been applied.

ft: messages injected at root How many messages injected at root.

ft: nonleaf compression to memory (seconds) Total time, in seconds, spent compressing non leaf nodes.

ft: nonleaf decompression to memory (seconds) Total time, in seconds, spent decompressing non leaf nodes.

ft: nonleaf deserialization to memory (seconds) Total time, in seconds, spent deserializing non leaf nodes.

ft: nonleaf node full evictions (bytes) The number of bytes freed by evicting full nonleaf nodes from the cache.

ft: nonleaf node full evictions The number of times a full nonleaf node was evicted from the cache.

ft: nonleaf node partial evictions (bytes) The number of bytes freed by evicting partitions of nonleaf nodes from the cache.

ft: nonleaf node partial evictions The number of times a partition of a nonleaf node was evicted from the cache.

ft: nonleaf nodes created Number of nonleaf nodes created.

ft: nonleaf nodes destroyed Number of nonleaf nodes destroyed.

ft: nonleaf nodes flushed to disk (for checkpoint) (bytes) Number of bytes of nonleaf nodes flushed to disk for checkpoint.

ft: nonleaf nodes flushed to disk (for checkpoint) Number of nonleaf nodes flushed to disk for checkpoint.

ft: nonleaf nodes flushed to disk (for checkpoint) (seconds) Number of seconds waiting for IO when writing nonleaf nodes flushed to disk for checkpoint.

ft: nonleaf nodes flushed to disk (for checkpoint) (uncompressed bytes) Number of uncompressed bytes of nonleaf nodes flushed to disk for checkpoint.

ft: nonleaf nodes flushed to disk (not for checkpoint) (bytes) Number of bytes of nonleaf nodes flushed to disk, not for checkpoint.

ft: nonleaf nodes flushed to disk (not for checkpoint) Number of nonleaf nodes flushed to disk, not for checkpoint.

ft: nonleaf nodes flushed to disk (not for checkpoint) (seconds) Number of seconds waiting for IO when writing nonleaf nodes flushed to disk, not for check- point.

ft: nonleaf nodes flushed to disk (not for checkpoint) (uncompressed bytes) Number of uncompressed bytes of nonleaf nodes flushed to disk, not for checkpoint.

ft: nonleaf serialization to memory (seconds) Total time, in seconds, spent serializing non leaf nodes.

ft: pivots fetched for prefetch (bytes) Number of bytes of pivot nodes fetched by a prefetch thread.

ft: pivots fetched for prefetch Number of pivot nodes fetched by a prefetch thread.

ft: pivots fetched for prefetch (seconds) Number seconds waiting for IO when fetching pivot nodes by a prefetch thread.

ft: pivots fetched for query (bytes) Number of bytes of pivot nodes fetched for queries.

ft: pivots fetched for query Number of pivot nodes fetched for queries.

ft: pivots fetched for query (seconds) Number of seconds waiting for IO when fetching pivot nodes for queries.

ft: pivots fetched for write (bytes) Number of bytes of pivot nodes fetched for writes.

ft: pivots fetched for write Number of pivot nodes fetched for writes.

ft: pivots fetched for write (seconds) Number of seconds waiting for IO when fetching pivot nodes for writes.

ft: promotion: h1 roots injected into Number of times a message stopped at a root with height 1.

ft: promotion: injections at depth 0 Number of times a message stopped at depth 0.

ft: promotion: injections at depth 1 Number of times a message stopped at depth 1.

ft: promotion: injections at depth 2 Number of times a message stopped at depth 2.

ft: promotion: injections at depth 3 Number of times a message stopped at depth 3.

ft: promotion: injections lower than depth 3 Number of times a message was promoted past depth 3.

ft: promotion: leaf roots injected into Number of times a message stopped at a root with height 0.

ft: promotion: roots split Number of times the root split during promotion.

ft: promotion: stopped anyway, after locking the child Number of times a message stopped before a child which had been locked.

ft: promotion: stopped at height 1 Number of times a message stopped because it had reached height 1.

ft: promotion: stopped because of a nonempty buffer Number of times a message stopped because it reached a nonempty buffer.

ft: promotion: stopped because the child was locked or not at all in memory Number of times a message stopped because it could not cheaply get access to a child.

ft: promotion: stopped because the child was not fully in memory Number of times a message stopped because it could not cheaply get access to a child.

ft: promotion: succeeded in using the rightmost leaf shortcut Rightmost insertions used the rightmost-leaf pin path, meaning that the Fractal Tree index detected and properly optimized rightmost inserts.

ft: promotion: tried the rightmost leaf shortcut but failed (child reactive) Rightmost insertions did not use the rightmost-leaf pin path, due to the leaf being too large (needed to split).

ft: promotion: tried the rightmost leaf shortcut but failed (out-of-bounds) Rightmost insertions did not use the rightmost-leaf pin path, due to the insert not actually being into the rightmost leaf node.

ft: searches requiring more tries than the height of the tree Number of searches that required more tries than the height of the tree.

ft: searches requiring more tries than the height of the tree plus three Number of searches that required more tries than the height of the tree plus three.

ft: total search retries due to TRY AGAIN Total number of search retries due to TRY AGAIN.

ft: uncompressed / compressed bytes written (leaf) Ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for leaf nodes.

ft: uncompressed / compressed bytes written (nonleaf) Ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for nonleaf nodes.

ft: uncompressed / compressed bytes written (overall) Ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for all nodes.

ft flusher: cleaner thread leaf merges in progress The number of cleaner thread leaf merges in progress.

ft flusher: cleaner thread leaf merges successful The number of times the cleaner thread successfully merges a leaf.

ft flusher: height-greater-than-one nodes flushed by cleaner thread Number of nodes of height > 1 whose message buffers are flushed by cleaner thread.

ft flusher: height-one nodes flushed by cleaner thread Number of nodes of height one whose message buffers are flushed by cleaner thread.

ft flusher: leaf node balances Number of times a leaf node is balanced.

ft flusher: leaf node merges Number of times leaf nodes are merged.

ft flusher: leaf node splits Number of leaf nodes split.

ft flusher: max bytes in a buffer flushed by cleaner thread Max number of bytes in message buffer flushed by cleaner thread.

ft flusher: max workdone in a buffer flushed by cleaner thread Max workdone value of any message buffer flushed by cleaner thread.

ft flusher: min bytes in a buffer flushed by cleaner thread Min number of bytes in message buffer flushed by cleaner thread.

ft flusher: min workdone in a buffer flushed by cleaner thread Min workdone value of any message buffer flushed by cleaner thread.

ft flusher: nodes cleaned which had empty buffers Number of nodes that are selected by cleaner, but whose buffers are empty.

ft flusher: nodes dirtied by cleaner thread Number of nodes that are made dirty by the cleaner thread.

ft flusher: nodes dirtied by cleaner thread leaf merges The number of nodes dirtied by the "flush from root" process to merge a leaf node.

ft flusher: nonleaf node merges Number of times nonleaf nodes are merged.

ft flusher: nonleaf node splits Number of nonleaf nodes split.

ft flusher: number of flushes that read something off disk Number of flushes that had to read a child (or part) off disk.

ft flusher: number of flushes that triggered 1 cascading flush Number of flushes that triggered 1 cascading flush.

ft flusher: number of flushes that triggered 2 cascading flushes Number of flushes that triggered 2 cascading flushes.

ft flusher: number of flushes that triggered 3 cascading flushes Number of flushes that triggered 3 cascading flushes.

ft flusher: number of flushes that triggered 4 cascading flushes Number of flushes that triggered 4 cascading flushes.

ft flusher: number of flushes that triggered 5 cascading flushes Number of flushes that triggered 5 cascading flushes.

ft flusher: number of flushes that triggered another flush in child Number of flushes that triggered another flush in the child.

ft flusher: number of flushes that triggered over 5 cascading flushes Number of flushes that triggered more than 5 cascading flushes.

ft flusher: number of in memory flushes Number of in-memory flushes.

ft flusher: times cleaner thread tries to merge a leaf The number of times the cleaner thread tries to merge a leaf.

ft flusher: total bytes in buffers flushed by cleaner thread Total number of bytes in message buffers flushed by cleaner thread.

ft flusher: total nodes potentially flushed by cleaner thread Total number of nodes whose buffers are potentially flushed by cleaner thread.

ft flusher: total number of flushes done by flusher threads or cleaner threads Total number of flushes done by flusher threads or cleaner threads.

ft flusher: total workdone in buffers flushed by cleaner thread Total workdone value of message buffers flushed by cleaner thread.

handlerton: primary key bytes inserted Total number of bytes inserted into all primary key indexes.

hot: max number of flushes from root ever required to optimize a tree The maximum number of flushes from the root ever required to optimize a tree.

hot: operations aborted The number of HOT operations that have been aborted.

hot: operations ever started The number of HOT operations that have begun.

hot: operations successfully completed The number of HOT operations that have successfully completed.

indexer: max number of indexers that ever existed simultaneously This is the maximum number of indexers that ever existed simultaneously.

indexer: number of calls to indexer->abort() This is the number of indexers that were aborted.

indexer: number of calls to indexer->build() failed This is the total number of times that indexes were unable to be created using a indexer

indexer: number of calls to indexer->build() succeeded This is the total number of times that indexes were created using a indexer.

indexer: number of calls to indexer->close() that failed This is the number of indexers that were unable to create the requested index(es).

indexer: number of calls to indexer->close() that succeeded This is the number of indexers that successfully created the requested index(es).

indexer: number of calls to toku indexer create indexer() that failed This is the number of times a indexer was requested but could not be created.

indexer: number of indexers currently in existence This is the number of indexers that currently exist.

indexer: number of indexers successfully created This is the number of times one of our internal objects, a indexer, has been created.

le: expanded This is the number of times that an expanded memory mechanism was used to store a new or modified row on disk.

le: max committed xr This is the maximum number of committed transaction records that were stored on disk in a new or modified row.

le: max memsize This is the maximum number of bytes that were stored on disk as a new or modified row. This is the maximum uncompressed size of any row stored in *TokuDB* that was created or modified since the server started.

le: max provisional xr This is the maximum number of provisional transaction records that were stored on disk in a new or modified row.

le: size of leafentries after garbage collection (during message application) Total number of bytes of leaf nodes data after performing garbage collection for non-flush events.

le: size of leafentries after garbage collection (outside message application) Total number of bytes of leaf nodes data after performing garbage collection for flush events.

le: size of leafentries before garbage collection (during message application) Total number of bytes of leaf nodes data before performing garbage collection for non-flush events.

le: size of leafentries before garbage collection (outside message application) Total number of bytes of leaf nodes data before performing garbage collection for flush events.

loader: max number of loaders that ever existed simultaneously This is the maximum number of loaders that ever existed simultaneously.

loader: number of calls to loader->abort() This is the number of loaders that were aborted.

loader: number of calls to loader->close() that failed This is the number of loaders that were unable to create the requested table.

loader: number of calls to loader->close() that succeeded This is the number of loaders that successfully created the requested table.

loader: number of calls to loader->put() failed This is the total number of rows that were unable to be inserted using a loader.

loader: number of calls to loader->put() succeeded This is the total number of rows that were inserted using a loader.

loader: number of calls to toku loader create loader() that failed This is the number of times a loader was requested but could not be created.

loader: number of loaders currently in existence This is the number of loaders that currently exist.

loader: number of loaders successfully created This is the number of times one of our internal objects, a loader, has been created.

locktree: latest post-escalation memory size Size of the locktree, in bytes, after most current locktree escalation.

locktree: long time waiting for locks Total time, in microseconds, of the long waits.

locktree: long time waiting on lock escalation Total time, in microseconds, of the long waits for lock escalation to free up memory.

locktree: memory size Count, in bytes, that the locktree is currently using.

locktree: memory size limit Maximum number of bytes that the locktree is allowed to use.

locktree: number of lock timeouts Count of the number of times that a lock request timed out.

locktree: number of locktrees eligible for the STO Number of locktrees eligible for "single transaction optimizations".

locktree: number of locktrees open now Number of locktrees currently open.

locktree: number of lock waits Number of times that a lock request could not be acquired because of a conflict with some other transaction.

locktree: number of long lock waits Number of lock waits greater than 1 second in duration.

locktree: number of long waits on lock escalation Number of times that a client thread had to wait on lock escalation and the wait time was greater than 1 second.

locktree: number of pending lock requests Number of requesters waiting for a lock grant.

locktree: number of times a locktree ended the STO early Total number of times a "single transaction optimization" was ended early due to another trans- action starting.

locktree: number of times lock escalation ran Number of times the locktree needed to run lock escalation to reduce its memory footprint.

locktree: number of waits on lock escalation When the sum of the sizes of locks taken reaches the lock tree limit, we run lock escalation on a background thread. The clients threads need to wait for escalation to consolidate locks and free up memory. This counter counts the number of times a client thread has to wait on lock escalation.

locktree: time spent ending the STO early (seconds) Total number of seconds ending "single transaction optimizations".

locktree: time spent running escalation (seconds) Total number of seconds spent performing locktree escalation.

locktree: time waiting for locks Total time, in microseconds, spend by some client waiting for a lock conflict to be resolved.

locktree: time waiting on lock escalation Total time, in microseconds, that a client thread spent waiting for lock escalation to free up memory.

logger: next LSN This is the next unassigned Log Sequence Number. It will be assigned to the next entry in the recovery log.

logger: number of long logger write operations Number of times a logger write operation required 100ms or more.

logger: writes (bytes) Number of bytes the logger has written to disk.

logger: writes Number of times the logger has written to disk.

logger: writes (seconds) Number of seconds waiting for IO when writing logs to disk.

logger: writes (uncompressed bytes) Number of uncompressed the logger has written to disk.

max open dbs Max number of simultaneously open DBs.

memory: estimated maximum memory footprint Maximum memory footprint of the storage engine, the max value of (used - freed).

memory: largest attempted allocation size Largest number of bytes in a single successful malloc() operation.

memory: mallocator version Version string from in-use memory allocator.

memory: mmap threshold The threshold for malloc to use mmap.

memory: number of bytes freed Total number of mallocated bytes freed (used - freed = bytes in use).

memory: number of bytes requested Total number of bytes requested from mallocator.

memory: number of bytes used (requested + overhead) Total number of bytes allocated by mallocator.

memory: number of free operations Number of calls to free().

memory: number of malloc operations Number of calls to malloc().

memory: number of malloc operations that failed Number of failed calls to malloc().

memory: number of realloc operations Number of calls to realloc().

memory: number of realloc operations that failed Number of failed calls to realloc().

memory: size of the last failed allocation attempt Largest number of bytes in a single failed malloc() operation.

num open dbs now Number of currently open DBs.

period, in ms, that recovery log is automatically fsynced `fsync()` frequency in milliseconds.

time now Current date/time on server.

time of engine startup This is the time when the *TokuDB* storage engine started up. Normally, this is when `mysqld` started.

time of environment creation This is the time when the *TokuDB* storage engine was first started up. Normally, this is when `mysqld` was initially installed with *TokuDB* 5.x. If the environment was upgraded from *TokuDB* 4.x (4.2.0 or later), then this will be displayed as "Dec 31, 1969" on Linux hosts.

txn: aborts This is the total number of transactions that have been aborted.

txn: begin This is the number of transactions that have been started.

txn: begin read only Number of read only transactions started.

txn: successful commits This is the total number of transactions that have been committed.

51.4 Global Status

The INFORMATION_SCHEMA.GLOBAL_STATUS table provides details about the inner workings of *TokuDB* and can be useful in tuning your particular environment. The statuses can be determined with the following command:

```
SELECT * FROM INFORMATION_SCHEMA.GLOBAL_STATUS;
```

The following global status parameters are available:

TOKUDB_BASEMENTS_DECOMPRESSED_FOR_WRITE Number of basement nodes decompressed for writes.

TOKUDB_BASEMENTS_DECOMPRESSED_PREFETCH Number of basement nodes decompressed by a prefetch thread.

TOKUDB_BASEMENTS_DECOMPRESSED_PRELOCKED_RANGE Number of basement nodes decompressed by queries aggressively.

TOKUDB_BASEMENTS_DECOMPRESSED_TARGET_QUERY Number of basement nodes decompressed for queries.

TOKUDB_BASEMENT_DESERIALIZATION_FIXED_KEY Number of basement nodes deserialized where all keys had the same size, leaving the basement in a format that is optimal for in-memory workloads.

TOKUDB_BASEMENT_DESERIALIZATION_VARIABLE_KEY Number of basement nodes deserialized where all keys did not have the same size, and thus ineligible for an in-memory optimization.

TOKUDB_BASEMENTS_FETCHED_FOR WRITE_BYTES Number of basement node bytes fetched from disk for writes.

TOKUDB_BASEMENTS_FETCHED_FOR WRITE Number of basement nodes fetched from disk for writes.

TOKUDB_BASEMENTS_FETCHED_FOR WRITE_SECONDS Number of seconds waiting for IO when fetching basement nodes from disk for writes.

TOKUDB_BASEMENTS_FETCHED_PREFETCH_BYTES Number of basement node bytes fetched from disk by a prefetch thread.

TOKUDB_BASEMENTS_FETCHED_PREFETCH Number of basement nodes fetched from disk by a prefetch thread.

TOKUDB_BASEMENTS_FETCHED_PREFETCH_SECONDS Number of seconds waiting for IO when fetching basement nodes from disk by a prefetch thread.

TOKUDB_BASEMENTS_FETCHED_PRELOCKED_RANGE_BYTES Number of basement node bytes fetched from disk aggressively.

TOKUDB_BASEMENTS_FETCHED_PRELOCKED_RANGE Number of basement nodes fetched from disk aggressively.

TOKUDB_BASEMENTS_FETCHED_PRELOCKED_RANGE_SECONDS Number of seconds waiting for IO when fetching basement nodes from disk aggressively.

TOKUDB_BASEMENTS_FETCHED_TARGET_QUERY_BYTES Number of basement node bytes fetched from disk for queries.

TOKUDB_BASEMENTS_FETCHED_TARGET_QUERY Number of basement nodes fetched from disk for queries.

TOKUDB_BASEMENTS_FETCHED_TARGET_QUERY_SECONDS Number of seconds waiting for IO when fetching basement nodes from disk for queries.

TOKUDB_BROADCAST_MESSAGES_INJECTED_AT_ROOT How many broadcast messages injected at root.

TOKUDB_BUFFERS_DECOMPRESSED_FOR_WRITE Number of buffers decompressed for writes.

TOKUDB_BUFFERS_DECOMPRESSED_PREFETCH Number of buffers decompressed by a prefetch thread.

TOKUDB_BUFFERS_DECOMPRESSED_PRELOCKED_RANGE Number of buffers decompressed by queries aggressively.

TOKUDB_BUFFERS_DECOMPRESSED_TARGET_QUERY Number of buffers decompressed for queries.

TOKUDB_BUFFERS_FETCHED_FOR_WRITE_BYTES Number of buffer bytes fetched from disk for writes.

TOKUDB_BUFFERS_FETCHED_FOR_WRITE Number of buffers fetched from disk for writes.

TOKUDB_BUFFERS_FETCHED_FOR_WRITE_SECONDS Number of seconds waiting for IO when fetching buffers from disk for writes.

TOKUDB_BUFFERS_FETCHED_PREFETCH_BYTES Number of buffer bytes fetched from disk by a prefetch thread.

TOKUDB_BUFFERS_FETCHED_PREFETCH Number of buffers fetched from disk by a prefetch thread.

TOKUDB_BUFFERS_FETCHED_PREFETCH_SECONDS Number of seconds waiting for IO when fetching buffers from disk by a prefetch thread.

TOKUDB_BUFFERS_FETCHED_PRELOCKED_RANGE_BYTES Number of buffer bytes fetched from disk aggressively.

TOKUDB_BUFFERS_FETCHED_PRELOCKED_RANGE Number of buffers fetched from disk aggressively.

TOKUDB_BUFFERS_FETCHED_PRELOCKED_RANGE_SECONDS Number of seconds waiting for IO when fetching buffers from disk aggressively.

TOKUDB_BUFFERS_FETCHED_TARGET_QUERY_BYTES Number of buffer bytes fetched from disk for queries.

TOKUDB_BUFFERS_FETCHED_TARGET_QUERY Number of buffers fetched from disk for queries.

TOKUDB_BUFFERS_FETCHED_TARGET_QUERY_SECONDS Number of seconds waiting for IO when fetching buffers from disk for queries.

TOKUDB_CACHETABLE_CLEANER_EXECUTIONS Total number of times the cleaner thread loop has executed.

TOKUDB_CACHETABLE_CLEANER_ITERATIONS This is the number of cleaner operations that are performed every cleaner period.

TOKUDB_CACHETABLE_CLEANER_PERIOD *TokuDB* includes a cleaner thread that optimizes indexes in the background. This variable is the time, in seconds, between the completion of a group of cleaner operations and the beginning of the next group of cleaner operations. The cleaner operations run on a background thread performing work that does not need to be done on the client thread.

TOKUDB_CACHETABLE_EVICTIONS Number of blocks evicted from cache.

TOKUDB_CACHETABLE_LONG_WAIT_PRESSURE_COUNT The number of times a thread was stalled for more than 1 second due to cache pressure.

TOKUDB_CACHETABLE_LONG_WAIT_PRESSURE_TIME Total time, in microseconds, waiting on cache pressure to subside for more than 1 second.

TOKUDB_CACHETABLE_POOL_CLIENT_NUM_THREADS

TOKUDB_CACHETABLE_POOL_CLIENT_NUM_THREADS_ACTIVE

TOKUDB_CACHETABLE_POOL_CLIENT_QUEUE_SIZE

TOKUDB_CACHETABLE_POOL_CLIENT_MAX_QUEUE_SIZE

TOKUDB_CACHETABLE_POOL_CLIENT_TOTAL_ITEMS_PROCESSED

TOKUDB_CACHETABLE_POOL_CLIENT_TOTAL_EXECUTION_TIME

TOKUDB_CACHETABLE_POOL_CACHETABLE_NUM_THREADS

TOKUDB_CACHETABLE_POOL_CACHETABLE_NUM_THREADS_ACTIVE

TOKUDB_CACHETABLE_POOL_CACHETABLE_QUEUE_SIZE

TOKUDB_CACHETABLE_POOL_CACHETABLE_MAX_QUEUE_SIZE

TOKUDB_CACHETABLE_POOL_CACHETABLE_TOTAL_ITEMS_PROCESSED

TOKUDB_CACHETABLE_POOL_CACHETABLE_TOTAL_EXECUTION_TIME

TOKUDB_CACHETABLE_POOL_CHECKPOINT_NUM_THREADS

TOKUDB_CACHETABLE_POOL_CHECKPOINT_NUM_THREADS_ACTIVE

TOKUDB_CACHETABLE_POOL_CHECKPOINT_QUEUE_SIZE

TOKUDB_CACHETABLE_POOL_CHECKPOINT_MAX_QUEUE_SIZE

TOKUDB_CACHETABLE_POOL_CHECKPOINT_TOTAL_ITEMS_PROCESSED

TOKUDB_CACHETABLE_POOL_CHECKPOINT_TOTAL_EXECUTION_TIME

TOKUDB_CACHETABLE_MISS This is a count of how many times the application was unable to access your data in the internal cache.

TOKUDB_CACHETABLE_MISS_TIME This is the total time, in microseconds, of how long the database has had to wait for a disk read to complete.

TOKUDB_CACHETABLE_PREFETCHES This is the total number of times that a block of memory has been prefetched into the database's cache. Data is prefetched when the database's algorithms determine that a block of memory is likely to be accessed by the application.

TOKUDB_CACHETABLE_SIZE_CACHEPRESSURE The number of bytes causing cache pressure (the sum of buffers and workdone counters), helps to understand if cleaner threads are keeping up with workload.

TOKUDB_CACHETABLE_SIZE_CLONED Amount of memory, in bytes, currently used for cloned nodes. During the checkpoint operation, dirty nodes are cloned prior to serialization/compression, then written to disk. After which, the memory for the cloned block is returned for re-use.

TOKUDB_CACHETABLE_SIZE_CURRENT This is a count, in bytes, of how much of your uncompressed data is currently in the database's internal cache.

TOKUDB_CACHETABLE_SIZE_LEAF The number of bytes of leaf nodes in the cache.

TOKUDB_CACHETABLE_SIZE_LIMIT This is a count, in bytes, of how much of your uncompressed data will fit in the database's internal cache.

TOKUDB_CACHETABLE_SIZE_NONLEAF The number of bytes of nonleaf nodes in the cache.

TOKUDB_CACHETABLE_SIZE_ROLLBACK The number of bytes of rollback nodes in the cache.

TOKUDB_CACHETABLE_SIZE_WRITING This is the number of bytes that are currently queued up to be written to disk.

TOKUDB_CACHETABLE_WAIT_PRESSURE_COUNT The number of times a thread was stalled due to cache pressure.

TOKUDB_CACHETABLE_WAIT_PRESSURE TIME Total time, in microseconds, waiting on cache pressure to subside.

TOKUDB_CHECKPOINT_BEGIN_TIME Cumulative time (in microseconds) required to mark all dirty nodes as pending a checkpoint.

TOKUDB_CHECKPOINT_DURATION_LAST Time (in seconds) required to complete the last checkpoint.

TOKUDB_CHECKPOINT_DURATION Time (in seconds) required to complete all checkpoints.

TOKUDB_CHECKPOINT_FAILED This is the number of checkpoints that have failed for any reason.

TOKUDB_CHECKPOINT_LAST_BEGAN This is the time the last checkpoint began. If a checkpoint is currently in progress, then this time may be later than the time the last checkpoint completed. (Note, if no checkpoint has ever taken place, then this value will be "Dec 31, 1969" on Linux hosts.)

TOKUDB_CHECKPOINT_LAST_COMPLETE_BEGAN This is the time the last complete checkpoint started. Any data that changed after this time will not be captured in the checkpoint.

TOKUDB_CHECKPOINT_LAST_COMPLETE_ENDED This is the time the last complete checkpoint ended.

TOKUDB_CHECKPOINT_LONG_CHECKPOINT_BEGIN_COUNT The total number of times a checkpoint begin took more than 1 second.

TOKUDB_CHECKPOINT_END_TIME The time spent in checkpoint end operation in seconds.

TOKUDB_CHECKPOINT_LONG_END_COUNT This is the count of end_checkpoint operations that exceeded 1 minute.

TOKUDB_CHECKPOINT_LONG_END_TIME This is the total time of long checkpoints in seconds.

TOKUDB_CHECKPOINT_LONG_CHECKPOINT_BEGIN_TIME This is the total time, in microseconds, of long checkpoint begins. A long checkpoint begin is one taking more than 1 second.

TOKUDB_CHECKPOINT_PERIOD This is the interval in seconds between the end of an automatic checkpoint and the beginning of the next automatic checkpoint.

TOKUDB_CHECKPOINT_TAKEN This is the number of complete checkpoints that have been taken.

TOKUDB_DB_CLOSES Number of db close operations.

TOKUDB_DB_OPEN_CURRENT Number of currently open DBs.

TOKUDB_DB_OPEN_MAX Max number of simultaneously open DBs.

TOKUDB_DB_OPENS Number of db open operations.

TOKUDB_DESCRIPTOR_SET This is the number of time a descriptor was updated when the entire dictionary was updated (for example, when the schema has been changed).

TOKUDB_DICTIONARY_BROADCAST_UPDATES This is the number of broadcast updates that have been successfully performed. A broadcast update is an update that affects all rows in a dictionary.

TOKUDB_DICTIONARY_UPDATES This is the total number of rows that have been updated in all primary and secondary indexes combined, if those updates have been done with a separate recovery log entry per index.

TOKUDB_FILESYSTEM_FSYNC_NUM This is the total number of times the database has flushed the operating system's file buffers to disk.

TOKUDB_FILESYSTEM_FSYNC_TIME This the total time, in microseconds, used to fsync to disk.

TOKUDB_FILESYSTEM_LONG_FSYNC_NUM This is the total number of times the database has flushed the operating system's file buffers to disk and this operation required more than 1 second.

TOKUDB_FILESYSTEM_LONG_FSYNC_TIME This the total time, in microseconds, used to fsync to disk when the operation required more than 1 second.

TOKUDB_FILESYSTEM_THREADS_BLOCKED_BY_FULL_DISK This is the number of threads that are currently blocked because they are attempting to write to a full disk. This is normally zero. If this value is non-zero, then a warning will appear in the "disk free space" field.

TOKUDB_LEAF_COMPRESSION_TO_MEMORY_SECONDS Total time, in seconds, spent compressing leaf nodes.

TOKUDB_LEAF_DECOMPRESSION_TO_MEMORY_SECONDS Total time, in seconds, spent decompressing leaf nodes.

TOKUDB_LEAF_DESERIALIZATION_TO_MEMORY_SECONDS Total time, in seconds, spent deserializing leaf nodes.

TOKUDB_LEAF_NODE_COMPRESSION_RATIO Ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for leaf nodes.

TOKUDB_LEAF_NODE_FULL_EVICTIONS_BYTES The number of bytes freed by evicting full leaf nodes from the cache.

TOKUDB_LEAF_NODE_FULL_EVICTIONS The number of times a full leaf node was evicted from the cache.

TOKUDB_LEAF_NODE_PARTIAL_EVICTIONS_BYTES The number of bytes freed by evicting partitions of leaf nodes from the cache.

TOKUDB_LEAF_NODE_PARTIAL_EVICTIONS The number of times a partition of a leaf node was evicted from the cache.

TOKUDB_LEAF_NODES_CREATED Number of leaf nodes created.

TOKUDB_LEAF_NODES_DESTROYED Number of leaf nodes destroyed.

TOKUDB_LEAF_NODES_FLUSHED_CHECKPOINT_BYTES Number of bytes of leaf nodes flushed to disk for checkpoint.

TOKUDB_LEAF_NODES_FLUSHED_CHECKPOINT Number of leaf nodes flushed to disk for checkpoint.

TOKUDB_LEAF_NODES_FLUSHED_CHECKPOINT_SECONDS Number of seconds waiting for IO when writing leaf nodes flushed to disk for checkpoint.

TOKUDB_LEAF_NODES_FLUSHED_CHECKPOINT_UNCOMPRESSED BYTES Number of uncompressed bytes of leaf nodes flushed to disk for checkpoint.

TOKUDB_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_BYTES Number of bytes of leaf nodes flushed to disk, not for checkpoint.

TOKUDB_LEAF_NODES_FLUSHED_NOT_CHECKPOINT Number of leaf nodes flushed to disk, not for checkpoint.

TOKUDB_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_SECONDS Number of seconds waiting for IO when writing leaf nodes flushed to disk, not for checkpoint.

TOKUDB_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_UNCOMPRESSED_BYTES Number of bytes of leaf nodes flushed to disk, not for checkpoint.

TOKUDB_LEAF_SERIALIZATION_TO_MEMORY_SECONDS Total time, in seconds, spent serializing leaf nodes.

TOKUDB_LOADER_NUM_CREATED This is the number of times one of our internal objects, a loader, has been created.

TOKUDB_LOADER_NUM_CURRENT This is the number of loaders that currently exist.

TOKUDB_LOADER_NUM_MAX This is the maximum number of loaders that ever existed simultaneously.

TOKUDB_LOCKTREE_ESCALATION_NUM Number of times the locktree needed to run lock escalation to reduce its memory footprint.

TOKUDB_LOCKTREE_ESCALATION_SECONDS Total number of seconds spent performing locktree escalation.

TOKUDB_LOCKTREE_LATEST_POST_ESCALATION_MEMORY_SIZE Size of the locktree, in bytes, after most current locktree escalation.

TOKUDB_LOCKTREE_LONG_WAIT_COUNT Number of lock waits greater than 1 second in duration.

TOKUDB_LOCKTREE_LONG_WAIT_ESCALATION_COUNT Number of times that a client thread had to wait on lock escalation and the wait time was greater than 1 second.

TOKUDB_LOCKTREE_LONG_WAIT_ESCALATION_TIME Total time, in microseconds, of the long waits for lock escalation to free up memory.

TOKUDB_LOCKTREE_LONG_WAIT_TIME Total time, in microseconds, of the long waits.

TOKUDB_LOCKTREE_MEMORY_SIZE Count, in bytes, that the locktree is currently using.

TOKUDB_LOCKTREE_MEMORY_SIZE_LIMIT Maximum number of bytes that the locktree is allowed to use.

TOKUDB_LOCKTREE_OPEN_CURRENT Number of locktrees currently open.

TOKUDB_LOCKTREE_PENDING_LOCK_REQUESTS Number of requesters waiting for a lock grant.

TOKUDB_LOCKTREE_STO_ELIGIBLE_NUM Number of locktrees eligible for "single transaction optimizations".

TOKUDB_LOCKTREE_STO_ENDED_NUM Total number of times a "single transaction optimization" was ended early due to another transaction starting.

TOKUDB_LOCKTREE_STO_ENDED_SECONDS Total number of seconds ending "single transaction optimizations".

TOKUDB_LOCKTREE_TIMEOUT_COUNT Count of the number of times that a lock request timed out.

TOKUDB_LOCKTREE_WAIT_COUNT Number of times that a lock request could not be acquired because of a conflict with some other transaction.

TOKUDB_LOCKTREE_WAIT_ESCALATION_COUNT When the sum of the sizes of locks taken reaches the lock tree limit, we run lock escalation on a background thread. The clients threads need to wait for escalation to consolidate locks and free up memory. This counter counts the number of times a client thread has to wait on lock escalation.

TOKUDB_LOCKTREE_WAIT_ESCALATION_TIME Total time, in microseconds, that a client thread spent waiting for lock escalation to free up memory.

TOKUDB_LOCKTREE_WAIT_TIME Total time, in microseconds, spend by some client waiting for a lock conflict to be resolved.

TOKUDB_LOGGER_WAIT_LONG Number of times a logger write operation required 100ms or more.

TOKUDB_LOGGER_WRITES_BYTES Number of bytes the logger has written to disk.

TOKUDB_LOGGER_WRITES Number of times the logger has written to disk.

TOKUDB_LOGGER_WRITES_SECONDS Number of seconds waiting for IO when writing logs to disk.

TOKUDB_LOGGER_WRITES_UNCOMPRESSED_BYTES Number of uncompressed the logger has written to disk.

TOKUDB_MEM_ESTIMATED_MAXIMUM_MEMORY_FOOTPRINT Maximum memory footprint of the storage engine, the max value of (used - freed).

TOKUDB_MESSAGES_FLUSHED_FROM_H1_TO_LEAVES_BYTES How many bytes of messages flushed from h1 nodes to leaves.

TOKUDB_MESSAGES_IGNORED_BY_LEAF_DUE_TO_MSN The number of messages that were ignored by a leaf because it had already been applied.

TOKUDB_MESSAGES_INJECTED_AT_ROOT_BYTES How many bytes of messages injected at root (for all trees).

TOKUDB_MESSAGES_INJECTED_AT_ROOT How many messages injected at root.

TOKUDB_MESSAGES_IN_TREES_ESTIMATE_BYTES How many bytes of messages currently in trees (estimate).

TOKUDB_NONLEAF_COMPRESSION_TO_MEMORY_SECONDS Total time, in seconds, spent compressing non leaf nodes.

TOKUDB_NONLEAF_DECOMPRESSION_TO_MEMORY_SECONDS Total time, in seconds, spent decompressing non leaf nodes.

TOKUDB_NONLEAF_DESERIALIZATION_TO_MEMORY_SECONDS Total time, in seconds, spent deserializing non leaf nodes.

TOKUDB_NONLEAF_NODE_COMPRESSION_RATIO Ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for nonleaf nodes.

TOKUDB_NONLEAF_NODE_FULL_EVICTIONS_BYTES The number of bytes freed by evicting full nonleaf nodes from the cache.

TOKUDB_NONLEAF_NODE_FULL_EVICTIONS The number of times a full nonleaf node was evicted from the cache.

TOKUDB_NONLEAF_NODE_PARTIAL_EVICTIONS_BYTES The number of bytes freed by evicting partitions of nonleaf nodes from the cache.

TOKUDB_NONLEAF_NODE_PARTIAL_EVICTIONS The number of times a partition of a nonleaf node was evicted from the cache.

TOKUDB_NONLEAF_NODES_CREATED Number of nonleaf nodes created.

TOKUDB_NONLEAF_NODES_DESTROYED Number of nonleaf nodes destroyed.

TOKUDB_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_BYTES Number of bytes of nonleaf nodes flushed to disk for checkpoint.

TOKUDB_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT Number of nonleaf nodes flushed to disk for checkpoint.

TOKUDB_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_SECONDS Number of seconds waiting for IO when writing nonleaf nodes flushed to disk for checkpoint.

TOKUDB_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_UNCOMPRESSED_BYTES Number of uncompressed bytes of nonleaf nodes flushed to disk for checkpoint.

TOKUDB_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_BYTES Number of bytes of nonleaf nodes flushed to disk, not for checkpoint.

TOKUDB_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT Number of nonleaf nodes flushed to disk, not for checkpoint.

TOKUDB_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_SECONDS Number of seconds waiting for IO when writing nonleaf nodes flushed to disk, not for check- point.

TOKUDB_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_UNCOMPRESSED_BYTES Number of uncompressed bytes of nonleaf nodes flushed to disk, not for checkpoint.

TOKUDB_NONLEAF_SERIALIZATION_TO_MEMORY_SECONDS Total time, in seconds, spent serializing non leaf nodes.

TOKUDB_OVERALL_NODE_COMPRESSION_RATIO Ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for all nodes.

TOKUDB_PIVOTS_FETCHED_FOR_PREFETCH_BYTES Number of bytes of pivot nodes fetched by a prefetch thread.

TOKUDB_PIVOTS_FETCHED_FOR_PREFETCH Number of pivot nodes fetched by a prefetch thread.

TOKUDB_PIVOTS_FETCHED_FOR_PREFETCH_SECONDS Number seconds waiting for IO when fetching pivot nodes by a prefetch thread.

TOKUDB_PIVOTS_FETCHED_FOR_QUERY_BYTES Number of bytes of pivot nodes fetched for queries.

TOKUDB_PIVOTS_FETCHED_FOR_QUERY Number of pivot nodes fetched for queries.

TOKUDB_PIVOTS_FETCHED_FOR_QUERY_SECONDS Number of seconds waiting for IO when fetching pivot nodes for queries.

TOKUDB_PIVOTS_FETCHED_FOR_WRITE_BYTES Number of bytes of pivot nodes fetched for writes.

TOKUDB_PIVOTS_FETCHED_FOR_WRITE Number of pivot nodes fetched for writes.

TOKUDB_PIVOTS_FETCHED_FOR_WRITE_SECONDS Number of seconds waiting for IO when fetching pivot nodes for writes.

TOKUDB_PROMOTION_H1_ROOTS_INJECTED_INTO Number of times a message stopped at a root with height 1.

TOKUDB_PROMOTION_INJECTIONS_AT_DEPTH_0 Number of times a message stopped at depth 0.

TOKUDB_PROMOTION_INJECTIONS_AT_DEPTH_1 Number of times a message stopped at depth 1.

TOKUDB_PROMOTION_INJECTIONS_AT_DEPTH_2 Number of times a message stopped at depth 2.

TOKUDB_PROMOTION_INJECTIONS_AT_DEPTH_3 Number of times a message stopped at depth 3.

TOKUDB_PROMOTION_INJECTIONS_LOWER_THAN_DEPTH_3 Number of times a message was promoted past depth 3.

TOKUDB_PROMOTION_LEAF_ROOTS_INJECTED_INTO Number of times a message stopped at a root with height 0.

TOKUDB_PROMOTION_ROOTS_SPLIT Number of times the root split during promotion.

TOKUDB_PROMOTION_STOPPED_AFTER_LOCKING_CHILD Number of times a message stopped before a child which had been locked.

TOKUDB_PROMOTION_STOPPED_AT_HEIGHT_1 Number of times a message stopped because it had reached height 1.

TOKUDB_PROMOTION_STOPPED_CHILD_LOCKED_OR_NOT_IN_MEMORY Number of times a message stopped because it could not cheaply get access to a child.

TOKUDB_PROMOTION_STOPPED_CHILD_NOT_FULLY_IN_MEMORY Number of times a message stopped because it could not cheaply get access to a child.

TOKUDB_PROMOTION_STOPPED_NONEMPTY_BUFFER Number of times a message stopped because it reached a nonempty buffer.

TOKUDB_TXN_ABORTS This is the total number of transactions that have been aborted.

TOKUDB_TXN_BEGIN This is the number of transactions that have been started.

TOKUDB_TXN_BEGIN_READ_ONLY Number of read only transactions started.

TOKUDB_TXN_COMMITS This is the total number of transactions that have been committed.

CHAPTER

FIFTYTWO

PERCONA TOKUBACKUP

Percona *TokuBackup* is an open-source hot backup utility for *MySQL* servers running the *TokuDB* storage engine (including *Percona Server* and *MariaDB*). It does not lock your database during backup. The *TokuBackup* library intercepts system calls that write files and duplicates the writes to the backup directory.

Note: This feature is currently considered *Experimental*

- Installing From Binaries
- Making a Backup
- Restoring From Backup
- Advanced Configuration
  - Monitoring Progress
  - Excluding Source Files
  - Throttling Backup Rate
  - Restricting Backup Target
  - Reporting Errors
- Limitations and known issues

52.1 Installing From Binaries

TokuBackup is included with *Percona Server* `5.7.10-1` and later versions. Installation can be performed with the `ps_tokudb_admin` script.

To install *Percona TokuBackup*:

1. Run `ps_tokudb_admin --enable-backup` to add the `preload-hotbackup` option into **[mysqld_safe]** section of `my.cnf`.

```
$ sudo ps_tokudb_admin --enable-backup
Checking SELinux status...
INFO: SELinux is disabled.

Checking if preload-hotbackup option is already set in config file...
INFO: Option preload-hotbackup is not set in the config file.

Checking TokuBackup plugin status...
INFO: TokuBackup plugin is not installed.

Adding preload-hotbackup option into /etc/my.cnf
INFO: Successfully added preload-hotbackup option into /etc/my.cnf
PLEASE RESTART MYSQL SERVICE AND RUN THIS SCRIPT AGAIN TO FINISH INSTALLATION!
```

2. Restart mysql service

```
$ sudo service mysql restart
```

3. Run `ps_tokudb_admin --enable-backup` again to finish installation of *TokuBackup* plugin

```
$ sudo ps_tokudb_admin --enable-backup
Checking SELinux status...
INFO: SELinux is disabled.

Checking if preload-hotbackup option is already set in config file...
INFO: Option preload-hotbackup is set in the config file.

Checking TokuBackup plugin status...
INFO: TokuBackup plugin is not installed.

Checking if Percona Server is running with libHotBackup.so preloaded...
INFO: Percona Server is running with libHotBackup.so preloaded.

Installing TokuBackup plugin...
INFO: Successfully installed TokuBackup plugin.
```

52.2 Making a Backup

To run *Percona TokuBackup*, the backup destination directory must exist, be writable and owned by the same user under which *MySQL* server is running (usually `mysql`) and empty. Once this directory is created, the backup can be run using the following command:

```
mysql> set tokudb_backup_dir='/path_to_empty_directory';
```

Note: Setting the `tokudb_backup_dir` variable automatically starts the backup process to the specified directory. Percona TokuBackup will take full backup each time, currently there is no incremental backup option

---

52.3 Restoring From Backup

Percona TokuBackup does not have any functionality for restoring a backup. You can use **rsync** or **cp** to restore the files. You should check that the restored files have the correct ownership and permissions.

Note: Make sure that the datadir is empty and that *MySQL* server is shut down before restoring from backup. You can't restore to a datadir of a running mysqld instance (except when importing a partial backup).

The following example shows how you might use the **rsync** command to restore the backup:

```
$ rsync -avrP /data/backup/ /var/lib/mysql/
```

Since attributes of files are preserved, in most cases you will need to change their ownership to *mysql* before starting the database server. Otherwise, the files will be owned by the user who created the backup.

```
$ chown -R mysql:mysql /var/lib/mysql
```

If you have changed default *TokuDB* data directory (`tokudb_data_dir`) or *TokuDB* log directory (`tokudb_log_dir`) or both of them, you will see separate folders for each setting in backup directory after taking backup. You'll need to restore each folder separately:

```
$ rsync -avrP /data/backup/mysql_data_dir/ /var/lib/mysql/
$ rsync -avrP /data/backup/tokudb_data_dir/ /path/to/original/tokudb_data_dir/
$ rsync -avrP /data/backup/tokudb_log_dir/ /path/to/original/tokudb_log_dir/
$ chown -R mysql:mysql /var/lib/mysql
$ chown -R mysql:mysql /path/to/original/tokudb_data_dir
$ chown -R mysql:mysql /path/to/original/tokudb_log_dir
```

52.4 Advanced Configuration

- Monitoring Progress
- Excluding Source Files
- Throttling Backup Rate
- Restricting Backup Target
- Reporting Errors

52.4.1 Monitoring Progress

TokuBackup updates the *PROCESSLIST* state while the backup is in progress. You can see the output by running `SHOW PROCESSLIST` or `SHOW FULL PROCESSLIST`.

52.4.2 Excluding Source Files

You can exclude certain files and directories based on a regular expression set in the `tokudb_backup_exclude` session variable. If the source file name matches the excluded regular expression, then the source file is excluded from backup.

For example, to exclude all `lost+found` directories from backup, use the following command:

```
mysql> SET tokudb_backup_exclude='/lost\\+found($|/)';
```

Note: In *Percona Server* `5.7.10-3` to address bug #125, server `pid` file is excluded by default. If you're providing your own additions to the exclusions and have the `pid` file in the default location, you will need to add the mysqld_safe.pid entry.

52.4.3 Throttling Backup Rate

You can throttle the backup rate using the `tokudb_backup_throttle` session-level variable. This variable throttles the write rate in bytes per second of the backup to prevent TokuBackup from crowding out other jobs in the system. The default and max value is 18446744073709551615.

```
mysql> SET tokudb_backup_throttle=1000000;
```

52.4.4 Restricting Backup Target

You can restrict the location of the destination directory where the backups can be located using the `tokudb_backup_allowed_prefix` system-level variable. Attempts to backup to a location outside of the specified directory or its children will result in an error.

The default is `null`, backups have no restricted locations. This read-only variable can be set in the `my.cnf` configuration file and displayed with the `SHOW VARIABLES` command:

```
mysql> SHOW VARIABLES LIKE 'tokudb_backup_allowed_prefix';
+-------------------------------+-----------+
| Variable_name                 | Value     |
+-------------------------------+-----------+
| tokudb_backup_allowed_prefix | /dumpdir  |
+-------------------------------+-----------+
```

52.4.5 Reporting Errors

Percona TokuBackup uses two variables to capture errors. They are `tokudb_backup_last_error` and `tokudb_backup_last_error_string`. When *TokuBackup* encounters an error, these will report on the error number and the error string respectively. For example, the following output shows these parameters following an attempted backup to a directory that was not empty:

```
mysql> SET tokudb_backup_dir='/tmp/backupdir';
ERROR 1231 (42000): Variable 'tokudb_backup_dir' can't be set to the value of '/tmp/backupdir'

mysql> SELECT @@tokudb_backup_last_error;
+-----------------------------+
| @@tokudb_backup_last_error |
+-----------------------------+
|                          17 |
+-----------------------------+

mysql> SELECT @@tokudb_backup_last_error_string;
+-------------------------------------------------+
| @@tokudb_backup_last_error_string               |
+-------------------------------------------------+
| tokudb backup couldn't create needed directories. |
+-------------------------------------------------+
```

52.5 Limitations and known issues

- You must disable *InnoDB* asynchronous IO if backing up *InnoDB* tables with *TokuBackup*. Otherwise you will have inconsistent, unrecoverable backups. The appropriate setting is `innodb_use_native_aio=0`.
- To be able to run Point-In-Time-Recovery you'll need to manually get the binary log position.
- Transactional storage engines (*TokuDB* and *InnoDB*) will perform recovery on the backup copy of the database when it is first started.
- Tables using non-transactional storage engines (*MyISAM*) are not locked during the copy and may report issues when starting up the backup. It is best to avoid operations that modify these tables at the end of a hot backup operation (adding/changing users, stored procedures, etc.).
- The database is copied locally to the path specified in `/path/to/backup`. This folder must exist, be writable, be empty, and contain enough space for a full copy of the database.

- *TokuBackup* always makes a backup of the *MySQL* `datadir` and optionally the `tokudb_data_dir`, `tokudb_log_dir`, and the binary log folder. The latter three are only backed up separately if they are not the same as or contained in the *MySQL* `datadir`. None of these three folders can be a parent of the *MySQL* `datadir`.
- No other directory structures are supported. All *InnoDB*, *MyISAM*, and other storage engine files must be within the *MySQL* `datadir`.
- *TokuBackup* does not follow symbolic links.
- *TokuBackup* does not backup *MySQL* configuration file(s).
- *TokuBackup* does not backup tablespaces if they are out of `datadir`.
- Due to upstream bug #80183, *TokuBackup* can't recover backed-up table data if backup was taken while running `OPTIMIZE TABLE` or `ALTER TABLE ... TABLESPACE`.
- *TokuBackup* doesn't support incremental backups.

CHAPTER

FIFTYTHREE

FREQUENTLY ASKED QUESTIONS

This section contains frequently asked questions regarding *TokuDB* and related software.

- Transactional Operations
- TokuDB and the File System
- Full Disks
- Backup
- Missing Log Files
- Isolation Levels
- Lock Wait Timeout Exceeded
- Query Cache
- Row Size
- NFS & CIFS
- Using Other Storage Engines
- Using MySQL Patches with TokuDB
- Truncate Table vs Delete from Table
- Foreign Keys
- Dropping Indexes

53.1 Transactional Operations

What transactional operations does TokuDB support?

TokuDB supports `BEGIN TRANSACTION`, `END TRANSACTION`, `COMMIT`, `ROLLBACK`, `SAVEPOINT`, and `RELEASE SAVEPOINT`.

53.2 TokuDB and the File System

How can I determine which files belong to the various tables and indexes in my schemas?

The `tokudb_file_map` plugin lists all Fractal Tree Indexes and their corresponding data files. The `internal_file_name` is the actual file name (in the data folder).

```
mysql> SELECT * FROM information_schema.tokudb_file_map;

+-------------------------+-----------------------------------------+---------------+-------------+---
| dictionary_name         | internal_file_name                      | table_schema  | table_name  | ta
+-------------------------+-----------------------------------------+---------------+-------------+---
| ./test/tmc-key-idx_col2 | ./_test_tmc_key_idx_col2_a_14.tokudb    | test          | tmc         | ke
```

```
| ./test/tmc-main          | ./_test_tmc_main_9_14.tokudb          | test          | tmc          | ma
| ./test/tmc-status        | ./_test_tmc_status_8_14.tokudb        | test          | tmc          | st
+--------------------------+---------------------------------------+---------------+--------------+--
```

53.3 Full Disks

What happens when the disk system fills up?

The disk system may fill up during bulk load operations, such as `LOAD DATA IN FILE` or `CREATE INDEX`, or during incremental operations like `INSERT`.

In the bulk case, running out of disk space will cause the statement to fail with `ERROR 1030 (HY000): Got error 1 from storage engine`. The temporary space used by the bulk loader will be released. If this happens, you can use a separate physical disk for the temporary files (for more information, see `tokudb_tmp_dir`). If server runs out of free space *TokuDB* will assert the server to prevent data corruption to existing data files.

Otherwise, disk space can run low during non-bulk operations. When available space is below a user-configurable reserve (5% by default) inserts are prevented and transactions that perform inserts are aborted. If the disk becomes completely full then *TokuDB* will freeze until some disk space is made available.

Details about the disk system:

- There is a free-space reserve requirement, which is a user-configurable parameter given as a percentage of the total space in the file system. The default reserve is five percent. This value is available in the global variable `tokudb_fs_reserve_percent`. We recommend that this reserve be at least half the size of your physical memory.

  TokuDB polls the file system every five seconds to determine how much free space is available. If the free space dips below the reserve, then further table inserts are prohibited. Any transaction that attempts to insert rows will be aborted. Inserts are re-enabled when twice the reserve is available in the file system (so freeing a small amount of disk storage will not be sufficient to resume inserts). Warning messages are sent to the system error log when free space dips below twice the reserve and again when free space dips below the reserve.

  Even with inserts prohibited it is still possible for the file system to become completely full. For example this can happen because another storage engine or another application consumes disk space.

- If the file system becomes completely full, then *TokuDB* will freeze. It will not crash, but it will not respond to most SQL commands until some disk space is made available. When *TokuDB* is frozen in this state, it will still respond to the following command:

  ```
  SHOW ENGINE TokuDB STATUS;
  ```

  Make disk space available will allow the storage engine to continue running, but inserts will still be prohibited until twice the reserve is free.

  Note: Engine status displays a field indicating if disk free space is above twice the reserve, below twice the reserve, or below the reserve. It will also display a special warning if the disk is completely full.

- In order to make space available on this system you can:
  - Add some disk space to the filesystem.
  - Delete some non-TokuDB files manually.
  - If the disk is not completely full, you may be able to reclaim space by aborting any transactions that are very old. Old transactions can consume large volumes of disk space in the recovery log.
  - If the disk is not completely full, you can drop indexes or drop tables from your *TokuDB* databases.

  - Deleting large numbers of rows from an existing table and then closing the table may free some space, but it may not. Deleting rows may simply leave unused space (available for new inserts) inside *TokuDB* data files rather than shrink the files (internal fragmentation).

The fine print:

- The *TokuDB* storage engine can use up to three separate file systems simultaneously, one each for the data, the recovery log, and the error log. All three are monitored, and if any one of the three falls below the relevant threshold then a warning message will be issued and inserts may be prohibited.
- Warning messages to the error log are not repeated unless available disk space has been above the relevant threshold for at least one minute. This prevents excess messages in the error log if the disk free space is fluctuating around the limit.
- Even if there are no other storage engines or other applications running, it is still possible for *TokuDB* to consume more disk space when operations such as row delete and query are performed, or when checkpoints are taken. This can happen because *TokuDB* can write cached information when it is time-efficient rather than when inserts are issued by the application, because operations in addition to insert (such as delete) create log entries, and also because of internal fragmentation of *TokuDB* data files.
- The `tokudb_fs_reserve_percent` variable can not be changed once the system has started. It can only be set in `my.cnf` or on the mysqld command line.

53.4 Backup

How do I back up a system with TokuDB tables?

53.4.1 Taking backups with *Percona TokuBackup*

TokuDB is capable of performing online backups with *Percona TokuBackup*. To perform a backup, execute `backup to '/path/to/backup';`. This will create backup of the server and return when complete. The backup can be used by another server using a copy of the binaries on the source server. You can view the progress of the backup by executing `SHOW PROCESSLIST;`. *TokuBackup* produces a copy of your running *MySQL* server that is consistent at the end time of the backup process. The thread copying files from source to destination can be throttled by setting the `tokudb_backup_throttle` server variable. For more information check *Percona TokuBackup*.

The following conditions apply:

- Currently, *TokuBackup* only supports tables using the *TokuDB* storage engine and the *MyISAM* tables in the `mysql` database.

  Warning: You must disable *InnoDB* asynchronous IO if backing up *InnoDB* tables via *TokuBackup* utility. Otherwise you will have inconsistent, unrecoverable backups. The appropriate setting is `innodb_use_native_aio` to `0`.

- Transactional storage engines (*TokuDB* and *InnoDB*) will perform recovery on the backup copy of the database when it is first started.
- Tables using non-transactional storage engines (*MyISAM*) are not locked during the copy and may report issues when starting up the backup. It is best to avoid operations that modify these tables at the end of a hot backup operation (adding/changing users, stored procedures, etc.).
- The database is copied locally to the path specified in `/path/to/backup`. This folder must exist, be writable, be empty, and contain enough space for a full copy of the database.

- *TokuBackup* always makes a backup of the *MySQL* `datadir` and optionally the `tokudb_data_dir`, `tokudb_log_dir`, and the binary log folder. The latter three are only backed up separately if they are not the same as or contained in the *MySQL* `datadir`. None of these three folders can be a parent of the *MySQL* `datadir`.
- A folder is created in the given backup destination for each of the source folders.
- No other directory structures are supported. All *InnoDB*, *MyISAM*, and other storage engine files must be within the *MySQL* `datadir`.
- *TokuBackup* does not follow symbolic links.

53.4.2 Other options for taking backups

TokuDB tables are represented in the file system with dictionary files, log files, and metadata files. A consistent copy of all of these files must be made during a backup. Copying the files while they may be modified by a running *MySQL* may result in an inconsistent copy of the database.

LVM snapshots may be used to get a consistent snapshot of all of the *TokuDB* files. The LVM snapshot may then be backed up at leisure.

The `SELECT INTO OUTFILE` statement or **mysqldump** application may also be used to get a logical backup of the database.

References

The MySQL 5.5 reference manual describes several backup methods and strategies. In addition, we recommend reading the backup and recovery chapter in the following book:

High Performance MySQL, 3rd Edition, by Baron Schwartz, Peter Zaitsev, and Vadim Tkachenko, Copyright 2012, O'Reilly Media.

Cold Backup

When *MySQL* is shut down, a copy of the *MySQL* data directory, the *TokuDB* data directory, and the *TokuDB* log directory can be made. In the simplest configuration, the *TokuDB* files are stored in the *MySQL* data directory with all of other *MySQL* files. One merely has to back up this directory.

Hot Backup using mylvmbackup

The **mylvmbackup** utility, located on Launchpad, works with *TokuDB*. It does all of the magic required to get consistent copies of all of the *MySQL* tables, including *MyISAM* tables, *InnoDB* tables, etc., creates the LVM snapshots, and backs up the snapshots.

Logical Snapshots

A logical snapshot of the databases uses a SQL statements to retrieve table rows and restore them. When used within a transaction, a consistent snapshot of the database can be taken. This method can be used to export tables from one database server and import them into another server.

The `SELECT INTO OUTFILE` statement is used to take a logical snapshot of a database. The `LOAD DATA INFILE` statement is used to load the table data. Please see the *MySQL* 5.6 reference manual for details.

Note: Please do not use the :program‘mysqlhotcopy‘ to back up *TokuDB* tables. This script is incompatible with *TokuDB*.

53.5 Missing Log Files

What do I do if I delete my logs files or they are otherwise missing?

You’ll need to recover from a backup. It is essential that the log files be present in order to restart the database.

53.6 Isolation Levels

What is the default isolation level for TokuDB?

It is repeatable-read (MVCC).

How can I change the isolation level?

TokuDB supports repeatable-read, serializable, read-uncommitted and read-committed isolation levels (other levels are not supported). *TokuDB* employs pessimistic locking, and aborts a transaction when a lock conflict is detected.

To guarantee that lock conflicts do not occur, use repeatable-read, read-uncommitted or read- committed isolation level.

53.7 Lock Wait Timeout Exceeded

Why do my |MySQL| clients get lock timeout errors for my update queries? And what should my application do when it gets these errors?

Updates can get lock timeouts if some other transaction is holding a lock on the rows being updated for longer than the *TokuDB* lock timeout. You may want to increase the this timeout.

If an update deadlocks, then the transaction should abort and retry.

For more information on diagnosing locking issues, see *Lock Visualization in TokuDB*.

53.8 Query Cache

Does TokuDB support the query cache?

Yes, you can enable the query cache in the `my.cnf` file. Please make sure that the size of the cache is set to something larger than `0`, as this, in effect, disables the cache.

53.9 Row Size

What is the maximum row size?

The maximum row size is 32 MiB.

53.10 NFS & CIFS

Can the data directories reside on a disk that is NFS or CIFS mounted?

Yes, we do have customers in production with NFS & CIFS volumes today. However, both of these disk types can pose a challenge to performance and data integrity due to their complexity. If you're seeking performance, the switching infrastructure and protocols of a traditional network were not conceptualized for low response times and can be very difficult to troubleshoot. If you're concerned with data integrity, the possible data caching at the NFS level can cause inconsistencies between the logs and data files that may never be detected in the event of a crash. If you are thinking of using a NFS or CIFS mount, we would recommend that you use synchronous mount options, which are available from the NFS mount man page, but these settings may decrease performance. For further discussion please look here.

53.11 Using Other Storage Engines

Can the MyISAM and InnoDB Storage Engines be used?

MyISAM and *InnoDB* can be used directly in conjunction with *TokuDB*. Please note that you should not overcommit memory between *InnoDB* and *TokuDB*. The total memory assigned to both caches must be less than physical memory.

Can the Federated Storage Engines be used?

The Federated Storage Engine can also be used, however it is disabled by default in *MySQL*. It can be enabled by either running mysqld with `--federated` as a command line parameter, or by putting `federated` in the `[mysqld]` section of the `my.cnf` file.

For more information see the *MySQL* 5.6 Reference Manual: FEDERATED Storage Engine.

53.12 Using MySQL Patches with TokuDB

Can I use MySQL source code patches with TokuDB?

Yes, but you need to apply Percona patches as well as your patches to *MySQL* to build a binary that works with the Percona Fractal Tree library.

53.13 Truncate Table vs Delete from Table

Which is faster, TRUNCATE TABLE or DELETE FROM TABLE?

Please use `TRUNCATE TABLE` whenever possible. A table truncation runs in constant time, whereas a `DELETE FROM TABLE` requires a row-by-row deletion and thus runs in time linear to the table size.

53.14 Foreign Keys

Does TokuDB enforce foreign key constraints?

No, *TokuDB* ignores foreign key declarations.

53.15 Dropping Indexes

Is dropping an index in TokuDB hot?

No, the table is locked for the amount of time it takes the file system to delete the file associated with the index.

CHAPTER

FIFTYFOUR

REMOVING TOKUDB STORAGE ENGINE

In case you want remove the TokuDB storage engine from *Percona Server* without causing any errors following is the recommended procedure:

54.1 Change the tables from TokuDB to InnoDB

If you still need the data in the TokuDB tables you'll need to alter the tables to other supported storage engine i.e., *InnoDB*:

```
mysql> ALTER TABLE City ENGINE=InnoDB;
```

Note: In case you remove the TokuDB storage engine before you've changed your tables to other supported storage engine you won't be able to access that data without re-installing the TokuDB storage engine.

54.2 Removing the plugins

If you're using *Percona Server* `5.6.22-72.0` or later you can use the `ps_tokudb_admin` script to remove the plugins:

```
ps_tokudb_admin --disable -uroot -pPassw0rd
```

Script output should look like this:

```
Checking if Percona server is running with jemalloc enabled...
>> Percona server is running with jemalloc enabled.

Checking transparent huge pages status on the system...
>> Transparent huge pages are currently disabled on the system.

Checking if thp-setting=never option is already set in config file...
>> Option thp-setting=never is set in the config file.

Checking TokuDB plugin status...
>> TokuDB plugin is installed.

Removing thp-setting=never option from /etc/mysql/my.cnf
>> Successfuly removed thp-setting=never option from /etc/mysql/my.cnf

Uninstalling TokuDB plugin...
>> Successfuly uninstalled TokuDB plugin.
```

Prior to *Percona Server* 5.6.22-72.0 TokuDB storage engine requires manual removal:

```
UNINSTALL PLUGIN tokudb;
UNINSTALL PLUGIN tokudb_file_map;
UNINSTALL PLUGIN tokudb_fractal_tree_info;
UNINSTALL PLUGIN tokudb_fractal_tree_block_map;
UNINSTALL PLUGIN tokudb_trx;
UNINSTALL PLUGIN tokudb_locks;
UNINSTALL PLUGIN tokudb_lock_waits;
```

After the engine and the plugins have been uninstalled you can remove the TokuDB package by using the apt/yum commands:

```
[root@centos ~]# yum remove Percona-Server-tokudb-56.x86_64
```

or

```
root@wheezy:~# apt-get remove percona-server-tokudb-5.6
```

Note: Make sure you've removed all the TokuDB specific variables from your configuration file (my.cnf) before you restart the server, otherwise server could show errors or warnings and won't be able to start.

Part X

Reference

CHAPTER

FIFTYFIVE

LIST OF UPSTREAM *MYSQL* BUGS FIXED IN *PERCONA SERVER* 5.7

Upstream bug #80607 - main.log_tables-big unstable on loaded hosts
Launchpad bug #1554043
Upstream state Verified (checked on 2016-03-11)
Fix Released `5.7.11-4`
Upstream fix N/A

Upstream bug #80606 - my_write, my_pwrite no longer safe to call from THD-less server utility...
Launchpad bug #1552682
Upstream state N/A
Fix Released `5.7.11-4`
Upstream fix N/A

Upstream bug #80496 - buf_dblwr_init_or_load_pages now returns an error code, but caller not...
Launchpad bug #1549301
Upstream state Verified (checked on 2016-03-11)
Fix Released `5.7.11-4`
Upstream fix N/A

Upstream bug #80053 - Assertion in binlog coordinator on slave with 2 2pc handler log_slave ...
Launchpad bug #1534249
Upstream state Open (checked on 2016-03-11)
Fix Released `5.7.10-2`
Upstream fix N/A

Upstream bug #79894 - Page cleaner worker threads are not instrumented for performance schema
Launchpad bug #1532747
Upstream state Open (checked on 2016-03-11)
Fix Released `5.7.10-2`
Upstream fix N/A

Continued on next page

Table 55.1 – continued from previous page

Upstream bug #79703 - Spin rounds per wait will be negative in InnoDB status if spin waits >... **Launchpad bug** #1527160 **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-2` **Upstream fix** N/A
Upstream bug #79569 - Some –big-test tests were forgotten to update in 5.7.10 **Launchpad bug** #1525109 **Upstream state** Closed **Fix Released** `5.7.10-2` **Upstream fix** 5.7.11
Upstream bug #79117 - “change_user” command should be aware of preceding “error” command **Launchpad bug** #1172090 **Upstream state** Closed **Fix Released** `5.7.10-1` **Upstream fix** 5.7.12
Upstream bug #78894 - buf_pool_resize can lock less in checking whether AHI is on or off **Launchpad bug** #1525215 **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A
Upstream bug #77684 - DROP TABLE IF EXISTS may brake replication if slave has replication ... **Launchpad bug** #1475107 **Upstream state** Closed **Fix Released** `5.7.10-1` **Upstream fix** 5.7.12
Upstream bug #77591 - ALTER TABLE does not allow to change NULL/NOT NULL if foreign key exists **Launchpad bug** #1466414 **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A

Continued on next page

Table 55.1 – continued from previous page

Upstream bug #77399 - Deadlocks missed by INFORMATION_SCHEMA.INNODB_METRICS lock_deadlocks ... **Launchpad bug** #1466414 **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A
Upstream bug #75534 - Solve buffer pool mutex contention by splitting it **Launchpad bug** *Improved Buffer Pool Scalability* **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A
Upstream bug #75504 - btr_search_guess_on_hash makes found block young twice? **Launchpad bug** #1411694 **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A
Upstream bug #75480 - Selecting wrong pos with SHOW BINLOG EVENTS causes a potentially ... **Launchpad bug** #1409652 **Upstream state** N/A **Fix Released** `5.7.10-1` **Upstream fix** N/A
Upstream bug #74637 - make dirty page flushing more adaptive **Launchpad bug** *Multi-threaded asynchronous LRU flusher* **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-3` **Upstream fix** N/A
Upstream bug #73418 - Add –manual-lldb option to mysql-test-run.pl **Launchpad bug** #1328482 **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A
Upstream bug #72108 - Hard to read history file **Launchpad bug** #1296192 **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A

Continued on next page

Table 55.1 – continued from previous page

Upstream bug #71411 - buf_flush_LRU() does not return correct number in case of compressed ... **Launchpad bug** #1262651 **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A
Upstream bug #71183 - os_file_fsync() should handle fsync() returning EINTR **Launchpad bug** #1262651 **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A
Upstream bug #70500 - Page cleaner should perform LRU flushing regardless of server activity **Launchpad bug** #1234562 **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A
Upstream bug #70490 - Suppression is too strict on some systems **Launchpad bug** #1205196 **Upstream state** Open (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A
Upstream bug #69991 - MySQL client is broken without readline **Launchpad bug** #1266386 **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A
Upstream bug #69232 - buf_dblwr->mutex can be splited into two **Launchpad bug** *Parallel doublewrite buffer* **Upstream state** Open (checked on 2016-03-11) **Fix Released** `5.7.11-4` **Upstream fix** N/A
Upstream bug #69146 - Needless log flush order mutex acquisition in buf_pool_get_oldest_mod... **Launchpad bug** #1176496 **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A

Continued on next page

Table 55.1 – continued from previous page

Upstream bug #68714 - Remove literal statement digest values from perfschema tests **Launchpad bug** #1157078 **Upstream state** Not a Bug **Fix Released** `5.7.10-1` **Upstream fix** N/A
Upstream bug #67808 - in innodb engine, double write and multi-buffer pool instance reduce ... **Launchpad bug** *Parallel doublewrite buffer* **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.11-4` **Upstream fix** N/A
Upstream bug #63130 - CMake-based check for the presence of a system readline library is not... **Launchpad bug** #1266386 **Upstream state** Can't Repeat (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A
Upstream bug #57583 - fast index create not used during "alter table foo engine=innodb" **Launchpad bug** #1451351 **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A
Upstream bug #53645 - SHOW GRANTS not displaying all the applicable grants **Launchpad bug** #1354988 **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A
Upstream bug #49120 - mysqldump should have flag to delay creating indexes for innodb plugin... **Launchpad bug** #744103 **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A
Upstream bug #35125 - Allow the ability to set the server_id for a connection for logging to... **Launchpad BP** Blueprint **Upstream state** Verified (checked on 2016-03-11) **Fix Released** `5.7.10-1` **Upstream fix** N/A

CHAPTER

FIFTYSIX

LIST OF VARIABLES INTRODUCED IN PERCONA SERVER 5.7

56.1 System Variables

Name	Cmd-Line	Option File	Var Scope	Dynamic
audit_log_buffer_size	Yes	Yes	Global	No
audit_log_file	Yes	Yes	Global	No
audit_log_flush	Yes	Yes	Global	Yes
audit_log_format	Yes	Yes	Global	No
audit_log_handler	Yes	Yes	Global	No
audit_log_policy	Yes	Yes	Global	Yes
audit_log_rotate_on_size	Yes	Yes	Global	No
audit_log_rotations	Yes	Yes	Global	No
audit_log_strategy	Yes	Yes	Global	No
audit_log_syslog_facility	Yes	Yes	Global	No
audit_log_syslog_ident	Yes	Yes	Global	No
audit_log_syslog_priority	Yes	Yes	Global	No
csv_mode	Yes	Yes	Both	Yes
enforce_storage_engine	Yes	Yes	Global	No
expand_fast_index_creation	Yes	No	Both	Yes
extra_max_connections	Yes	Yes	Global	Yes
extra_port	Yes	Yes	Global	No
have_backup_locks	Yes	No	Global	No
have_backup_safe_binlog_info	Yes	No	Global	No
have_snapshot_cloning	Yes	No	Global	No
innodb_cleaner_lsn_age_factor	Yes	Yes	Global	Yes
innodb_corrupt_table_action	Yes	Yes	Global	Yes
innodb_empty_free_list_algorithm	Yes	Yes	Global	Yes
innodb_kill_idle_transaction	Yes	Yes	Global	Yes
innodb_max_bitmap_file_size	Yes	Yes	Global	Yes
innodb_max_changed_pages	Yes	Yes	Global	Yes
innodb_show_locks_held	Yes	Yes	Global	Yes
innodb_show_verbose_locks	Yes	Yes	Global	Yes
innodb_track_changed_pages	Yes	Yes	Global	No
innodb_use_global_flush_log_at_trx_commit	Yes	Yes	Global	Yes
log_slow_filter	Yes	Yes	Both	Yes
log_slow_rate_limit	Yes	Yes	Both	Yes
log_slow_rate_type	Yes	Yes	Global	Yes

Continued on next page

Table 56.1 – continued from previous page

Name	Cmd-Line	Option File	Var Scope	Dynamic
log_slow_sp_statements	Yes	Yes	Global	Yes
log_slow_verbosity	Yes	Yes	Both	Yes
log_warnings_suppress	Yes	Yes	Global	Yes
max_binlog_files	Yes	Yes	Global	Yes
max_slowlog_files	Yes	Yes	Global	Yes
max_slowlog_size	Yes	Yes	Global	Yes
proxy_protocol_networks	Yes	Yes	Global	No
pseudo_server_id	Yes	No	Session	Yes
query_cache_strip_comments	Yes	Yes	Global	Yes
query_response_time_flush	Yes	No	Global	No
query_response_time_range_base	Yes	Yes	Global	Yes
query_response_time_stats	Yes	Yes	Global	Yes
slow_query_log_always_write_time	Yes	Yes	Global	Yes
slow_query_log_use_global_control	Yes	Yes	Global	Yes
thread_pool_high_prio_mode	Yes	Yes	Both	Yes
thread_pool_high_prio_tickets	Yes	Yes	Both	Yes
thread_pool_idle_timeout	Yes	Yes	Global	Yes
thread_pool_max_threads	Yes	Yes	Global	Yes
thread_pool_oversubscribe	Yes	Yes	Global	Yes
thread_pool_size	Yes	Yes	Global	Yes
thread_pool_stall_limit	Yes	Yes	Global	No
thread_statistics	Yes	Yes	Global	Yes
tokudb_alter_print_error				
tokudb_analyze_delete_fraction				
tokudb_analyze_in_background	Yes	Yes	Both	Yes
tokudb_analyze_mode	Yes	Yes	Both	Yes
tokudb_analyze_throttle	Yes	Yes	Both	Yes
tokudb_analyze_time	Yes	Yes	Both	Yes
tokudb_auto_analyze	Yes	Yes	Both	Yes
tokudb_block_size				
tokudb_bulk_fetch				
tokudb_cache_size				
tokudb_cachetable_pool_threads	Yes	Yes	Global	No
tokudb_cardinality_scale_percent				
tokudb_check_jemalloc				
tokudb_checkpoint_lock				
tokudb_checkpoint_on_flush_logs				
tokudb_checkpoint_pool_threads	Yes	Yes	Global	No
tokudb_checkpointing_period				
tokudb_cleaner_iterations				
tokudb_cleaner_period				
tokudb_client_pool_threads	Yes	Yes	Global	No
tokudb_commit_sync				
tokudb_compress_buffers_before_eviction	Yes	Yes	Global	No
tokudb_create_index_online				
tokudb_data_dir				
tokudb_debug				
tokudb_directio				

Continued on next page

Table 56.1 – continued from previous page

Name	Cmd-Line	Option File	Var Scope	Dynamic
tokudb_disable_hot_alter				
tokudb_disable_prefetching				
tokudb_disable_slow_alter				
tokudb_empty_scan				
tokudb_enable_partial_eviction	Yes	Yes	Global	No
tokudb_fanout	Yes	Yes	Both	Yes
tokudb_fs_reserve_percent				
tokudb_fsync_log_period				
tokudb_hide_default_row_format				
tokudb_killed_time				
tokudb_last_lock_timeout				
tokudb_load_save_space				
tokudb_loader_memory_size				
tokudb_lock_timeout				
tokudb_lock_timeout_debug				
tokudb_log_dir				
tokudb_max_lock_memory				
tokudb_optimize_index_fraction				
tokudb_optimize_index_name				
tokudb_optimize_throttle				
tokudb_pk_insert_mode				
tokudb_prelock_empty				
tokudb_read_block_size				
tokudb_read_buf_size				
tokudb_read_status_frequency				
tokudb_row_format				
tokudb_rpl_check_readonly				
tokudb_rpl_lookup_rows				
tokudb_rpl_lookup_rows_delay				
tokudb_rpl_unique_checks				
tokudb_rpl_unique_checks_delay				
tokudb_strip_frm_data	Yes	Yes	Global	No
tokudb_support_xa				
tokudb_tmp_dir				
tokudb_version				
tokudb_write_status_frequency				
userstat	Yes	Yes	Global	Yes

56.2 Status Variables

Name	Var Type	Var Scope
Binlog_snapshot_file	String	Global
Binlog_snapshot_position	Numeric	Global
Com_lock_binlog_for_backup	Numeric	Both
Com_lock_tables_for_backup	Numeric	Both

Continued on next page

Table 56.2 – continued from previous page

Name	Var Type	Var Scope
Com_show_client_statistics	Numeric	Both
Com_show_index_statistics	Numeric	Both
Com_show_table_statistics	Numeric	Both
Com_show_thread_statistics	Numeric	Both
Com_show_user_statistics	Numeric	Both
Com_unlock_binlog	Numeric	Both
Innodb_background_log_sync	Numeric	Global
Innodb_buffer_pool_pages_LRU_flushed	Numeric	Global
Innodb_buffer_pool_pages_made_not_young	Numeric	Global
Innodb_buffer_pool_pages_made_young	Numeric	Global
Innodb_buffer_pool_pages_old	Numeric	Global
Innodb_checkpoint_age	Numeric	Global
Innodb_checkpoint_max_age	Numeric	Global
Innodb_ibuf_free_list	Numeric	Global
Innodb_ibuf_segment_size	Numeric	Global
Innodb_lsn_current	Numeric	Global
Innodb_lsn_flushed	Numeric	Global
Innodb_lsn_last_checkpoint	Numeric	Global
Innodb_master_thread_active_loops	Numeric	Global
Innodb_master_thread_idle_loops	Numeric	Global
Innodb_max_trx_id	Numeric	Global
Innodb_mem_adaptive_hash	Numeric	Global
Innodb_mem_dictionary	Numeric	Global
Innodb_oldest_view_low_limit_trx_id	Numeric	Global
Innodb_purge_trx_id	Numeric	Global
Innodb_purge_undo_no	Numeric	Global
Threadpool_idle_threads	Numeric	Global
Threadpool_threads	Numeric	Global
Tokudb_DB_OPENS		
Tokudb_DB_CLOSES		
Tokudb_DB_OPEN_CURRENT		
Tokudb_DB_OPEN_MAX		
Tokudb_LEAF_ENTRY_MAX_COMMITTED_XR		
Tokudb_LEAF_ENTRY_MAX_PROVISIONAL_XR		
Tokudb_LEAF_ENTRY_EXPANDED		
Tokudb_LEAF_ENTRY_MAX_MEMSIZE		
Tokudb_LEAF_ENTRY_APPLY_GC_BYTES_IN		
Tokudb_LEAF_ENTRY_APPLY_GC_BYTES_OUT		
Tokudb_LEAF_ENTRY_NORMAL_GC_BYTES_IN		
Tokudb_LEAF_ENTRY_NORMAL_GC_BYTES_OUT		
Tokudb_CHECKPOINT_PERIOD		
Tokudb_CHECKPOINT_FOOTPRINT		
Tokudb_CHECKPOINT_LAST_BEGAN		
Tokudb_CHECKPOINT_LAST_COMPLETE_BEGAN		
Tokudb_CHECKPOINT_LAST_COMPLETE_ENDED		
Tokudb_CHECKPOINT_DURATION		
Tokudb_CHECKPOINT_DURATION_LAST		
Tokudb_CHECKPOINT_LAST_LSN		

Continued on next page

Table 56.2 – continued from previous page

Name	Var Type	Var Scope
Tokudb_CHECKPOINT_TAKEN		
Tokudb_CHECKPOINT_FAILED		
Tokudb_CHECKPOINT_WAITERS_NOW		
Tokudb_CHECKPOINT_WAITERS_MAX		
Tokudb_CHECKPOINT_CLIENT_WAIT_ON_MO		
Tokudb_CHECKPOINT_CLIENT_WAIT_ON_CS		
Tokudb_CHECKPOINT_BEGIN_TIME		
Tokudb_CHECKPOINT_LONG_BEGIN_TIME		
Tokudb_CHECKPOINT_LONG_BEGIN_COUNT		
Tokudb_CHECKPOINT_END_TIME		
Tokudb_CHECKPOINT_LONG_END_TIME		
Tokudb_CHECKPOINT_LONG_END_COUNT		
Tokudb_CACHETABLE_MISS		
Tokudb_CACHETABLE_MISS_TIME		
Tokudb_CACHETABLE_PREFETCHES		
Tokudb_CACHETABLE_SIZE_CURRENT		
Tokudb_CACHETABLE_SIZE_LIMIT		
Tokudb_CACHETABLE_SIZE_WRITING		
Tokudb_CACHETABLE_SIZE_NONLEAF		
Tokudb_CACHETABLE_SIZE_LEAF		
Tokudb_CACHETABLE_SIZE_ROLLBACK		
Tokudb_CACHETABLE_SIZE_CACHEPRESSURE		
Tokudb_CACHETABLE_SIZE_CLONED		
Tokudb_CACHETABLE_EVICTIONS		
Tokudb_CACHETABLE_CLEANER_EXECUTIONS		
Tokudb_CACHETABLE_CLEANER_PERIOD		
Tokudb_CACHETABLE_CLEANER_ITERATIONS		
Tokudb_CACHETABLE_WAIT_PRESSURE_COUNT		
Tokudb_CACHETABLE_WAIT_PRESSURE_TIME		
Tokudb_CACHETABLE_LONG_WAIT_PRESSURE_COUNT		
Tokudb_CACHETABLE_LONG_WAIT_PRESSURE_TIME		
Tokudb_CACHETABLE_POOL_CLIENT_NUM_THREADS		
Tokudb_CACHETABLE_POOL_CLIENT_NUM_THREADS_ACTIVE		
Tokudb_CACHETABLE_POOL_CLIENT_QUEUE_SIZE		
Tokudb_CACHETABLE_POOL_CLIENT_MAX_QUEUE_SIZE		
Tokudb_CACHETABLE_POOL_CLIENT_TOTAL_ITEMS_PROCESSED		
Tokudb_CACHETABLE_POOL_CLIENT_TOTAL_EXECUTION_TIME		
Tokudb_CACHETABLE_POOL_CACHETABLE_NUM_THREADS		
Tokudb_CACHETABLE_POOL_CACHETABLE_NUM_THREADS_ACTIVE		
Tokudb_CACHETABLE_POOL_CACHETABLE_QUEUE_SIZE		
Tokudb_CACHETABLE_POOL_CACHETABLE_MAX_QUEUE_SIZE		
Tokudb_CACHETABLE_POOL_CACHETABLE_TOTAL_ITEMS_PROCESSED		
Tokudb_CACHETABLE_POOL_CACHETABLE_TOTAL_EXECUTION_TIME		
Tokudb_CACHETABLE_POOL_CHECKPOINT_NUM_THREADS		
Tokudb_CACHETABLE_POOL_CHECKPOINT_NUM_THREADS_ACTIVE		
Tokudb_CACHETABLE_POOL_CHECKPOINT_QUEUE_SIZE		
Tokudb_CACHETABLE_POOL_CHECKPOINT_MAX_QUEUE_SIZE		
Tokudb_CACHETABLE_POOL_CHECKPOINT_TOTAL_ITEMS_PROCESSED		

Continued on next page

Table 56.2 – continued from previous page

Name	Var Type	Var Scope
Tokudb_CACHETABLE_POOL_CHECKPOINT_TOTAL_EXECUTION_TIME		
Tokudb_LOCKTREE_MEMORY_SIZE		
Tokudb_LOCKTREE_MEMORY_SIZE_LIMIT		
Tokudb_LOCKTREE_ESCALATION_NUM		
Tokudb_LOCKTREE_ESCALATION_SECONDS		
Tokudb_LOCKTREE_LATEST_POST_ESCALATION_MEMORY_SIZE		
Tokudb_LOCKTREE_OPEN_CURRENT		
Tokudb_LOCKTREE_PENDING_LOCK_REQUESTS		
Tokudb_LOCKTREE_STO_ELIGIBLE_NUM		
Tokudb_LOCKTREE_STO_ENDED_NUM		
Tokudb_LOCKTREE_STO_ENDED_SECONDS		
Tokudb_LOCKTREE_WAIT_COUNT		
Tokudb_LOCKTREE_WAIT_TIME		
Tokudb_LOCKTREE_LONG_WAIT_COUNT		
Tokudb_LOCKTREE_LONG_WAIT_TIME		
Tokudb_LOCKTREE_TIMEOUT_COUNT		
Tokudb_LOCKTREE_WAIT_ESCALATION_COUNT		
Tokudb_LOCKTREE_WAIT_ESCALATION_TIME		
Tokudb_LOCKTREE_LONG_WAIT_ESCALATION_COUNT		
Tokudb_LOCKTREE_LONG_WAIT_ESCALATION_TIME		
Tokudb_DICTIONARY_UPDATES		
Tokudb_DICTIONARY_BROADCAST_UPDATES		
Tokudb_DESCRIPTOR_SET		
Tokudb_MESSAGES_IGNORED_BY_LEAF_DUE_TO_MSN		
Tokudb_TOTAL_SEARCH_RETRIES		
Tokudb_SEARCH_TRIES_GT_HEIGHT		
Tokudb_SEARCH_TRIES_GT_HEIGHTPLUS3		
Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT		
Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_BYTES		
Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_UNCOMPRESSED_BYTES		
Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_SECONDS		
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT		
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_BYTES		
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_UNCOMPRESSE		
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_SECONDS		
Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT		
Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_BYTES		
Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_UNCOMPRESSED_BYTES		
Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_SECONDS		
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT		
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_BYTES		
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_UNCOMPRESSED_BY		
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_SECONDS		
Tokudb_LEAF_NODE_COMPRESSION_RATIO		
Tokudb_NONLEAF_NODE_COMPRESSION_RATIO		
Tokudb_OVERALL_NODE_COMPRESSION_RATIO		
Tokudb_NONLEAF_NODE_PARTIAL_EVICTIONS		
Tokudb_NONLEAF_NODE_PARTIAL_EVICTIONS_BYTES		

Continued on next page

Table 56.2 – continued from previous page

Name	Var Type	Var Scope
Tokudb_LEAF_NODE_PARTIAL_EVICTIONS		
Tokudb_LEAF_NODE_PARTIAL_EVICTIONS_BYTES		
Tokudb_LEAF_NODE_FULL_EVICTIONS		
Tokudb_LEAF_NODE_FULL_EVICTIONS_BYTES		
Tokudb_NONLEAF_NODE_FULL_EVICTIONS		
Tokudb_NONLEAF_NODE_FULL_EVICTIONS_BYTES		
Tokudb_LEAF_NODES_CREATED		
Tokudb_NONLEAF_NODES_CREATED		
Tokudb_LEAF_NODES_DESTROYED		
Tokudb_NONLEAF_NODES_DESTROYED		
Tokudb_MESSAGES_INJECTED_AT_ROOT_BYTES		
Tokudb_MESSAGES_FLUSHED_FROM_H1_TO_LEAVES_BYTES		
Tokudb_MESSAGES_IN_TREES_ESTIMATE_BYTES		
Tokudb_MESSAGES_INJECTED_AT_ROOT		
Tokudb_BROADCASE_MESSAGES_INJECTED_AT_ROOT		
Tokudb_BASEMENTS_DECOMPRESSED_TARGET_QUERY		
Tokudb_BASEMENTS_DECOMPRESSED_PRELOCKED_RANGE		
Tokudb_BASEMENTS_DECOMPRESSED_PREFETCH		
Tokudb_BASEMENTS_DECOMPRESSED_FOR_WRITE		
Tokudb_BUFFERS_DECOMPRESSED_TARGET_QUERY		
Tokudb_BUFFERS_DECOMPRESSED_PRELOCKED_RANGE		
Tokudb_BUFFERS_DECOMPRESSED_PREFETCH		
Tokudb_BUFFERS_DECOMPRESSED_FOR_WRITE		
Tokudb_PIVOTS_FETCHED_FOR_QUERY		
Tokudb_PIVOTS_FETCHED_FOR_QUERY_BYTES		
Tokudb_PIVOTS_FETCHED_FOR_QUERY_SECONDS		
Tokudb_PIVOTS_FETCHED_FOR_PREFETCH		
Tokudb_PIVOTS_FETCHED_FOR_PREFETCH_BYTES		
Tokudb_PIVOTS_FETCHED_FOR_PREFETCH_SECONDS		
Tokudb_PIVOTS_FETCHED_FOR_WRITE		
Tokudb_PIVOTS_FETCHED_FOR_WRITE_BYTES		
Tokudb_PIVOTS_FETCHED_FOR_WRITE_SECONDS		
Tokudb_BASEMENTS_FETCHED_TARGET_QUERY		
Tokudb_BASEMENTS_FETCHED_TARGET_QUERY_BYTES		
Tokudb_BASEMENTS_FETCHED_TARGET_QUERY_SECONDS		
Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE		
Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE_BYTES		
Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE_SECONDS		
Tokudb_BASEMENTS_FETCHED_PREFETCH		
Tokudb_BASEMENTS_FETCHED_PREFETCH_BYTES		
Tokudb_BASEMENTS_FETCHED_PREFETCH_SECONDS		
Tokudb_BASEMENTS_FETCHED_FOR_WRITE		
Tokudb_BASEMENTS_FETCHED_FOR_WRITE_BYTES		
Tokudb_BASEMENTS_FETCHED_FOR_WRITE_SECONDS		
Tokudb_BUFFERS_FETCHED_TARGET_QUERY		
Tokudb_BUFFERS_FETCHED_TARGET_QUERY_BYTES		
Tokudb_BUFFERS_FETCHED_TARGET_QUERY_SECONDS		
Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE		

Continued on next page

Table 56.2 – continued from previous page

Name	Var Type	Var Scope
Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE_BYTES		
Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE_SECONDS		
Tokudb_BUFFERS_FETCHED_PREFETCH		
Tokudb_BUFFERS_FETCHED_PREFETCH_BYTES		
Tokudb_BUFFERS_FETCHED_PREFETCH_SECONDS		
Tokudb_BUFFERS_FETCHED_FOR_WRITE		
Tokudb_BUFFERS_FETCHED_FOR_WRITE_BYTES		
Tokudb_BUFFERS_FETCHED_FOR_WRITE_SECONDS		
Tokudb_LEAF_COMPRESSION_TO_MEMORY_SECONDS		
Tokudb_LEAF_SERIALIZATION_TO_MEMORY_SECONDS		
Tokudb_LEAF_DECOMPRESSION_TO_MEMORY_SECONDS		
Tokudb_LEAF_DESERIALIZATION_TO_MEMORY_SECONDS		
Tokudb_NONLEAF_COMPRESSION_TO_MEMORY_SECONDS		
Tokudb_NONLEAF_SERIALIZATION_TO_MEMORY_SECONDS		
Tokudb_NONLEAF_DECOMPRESSION_TO_MEMORY_SECONDS		
Tokudb_NONLEAF_DESERIALIZATION_TO_MEMORY_SECONDS		
Tokudb_PROMOTION_ROOTS_SPLIT		
Tokudb_PROMOTION_LEAF_ROOTS_INJECTED_INTO		
Tokudb_PROMOTION_H1_ROOTS_INJECTED_INTO		
Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_0		
Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_1		
Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_2		
Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_3		
Tokudb_PROMOTION_INJECTIONS_LOWER_THAN_DEPTH_3		
Tokudb_PROMOTION_STOPPED_NONEMPTY_BUFFER		
Tokudb_PROMOTION_STOPPED_AT_HEIGHT_1		
Tokudb_PROMOTION_STOPPED_CHILD_LOCKED_OR_NOT_IN_MEMORY		
Tokudb_PROMOTION_STOPPED_CHILD_NOT_FULLY_IN_MEMORY		
Tokudb_PROMOTION_STOPPED_AFTER_LOCKING_CHILD		
Tokudb_BASEMENT_DESERIALIZATION_FIXED_KEY		
Tokudb_BASEMENT_DESERIALIZATION_VARIABLE_KEY		
Tokudb_PRO_RIGHTMOST_LEAF_SHORTCUT_SUCCESS		
Tokudb_PRO_RIGHTMOST_LEAF_SHORTCUT_FAIL_POS		
Tokudb_RIGHTMOST_LEAF_SHORTCUT_FAIL_REACTIVE		
Tokudb_CURSOR_SKIP_DELETED_LEAF_ENTRY		
Tokudb_FLUSHER_CLEANER_TOTAL_NODES		
Tokudb_FLUSHER_CLEANER_H1_NODES		
Tokudb_FLUSHER_CLEANER_HGT1_NODES		
Tokudb_FLUSHER_CLEANER_EMPTY_NODES		
Tokudb_FLUSHER_CLEANER_NODES_DIRTIED		
Tokudb_FLUSHER_CLEANER_MAX_BUFFER_SIZE		
Tokudb_FLUSHER_CLEANER_MIN_BUFFER_SIZE		
Tokudb_FLUSHER_CLEANER_TOTAL_BUFFER_SIZE		
Tokudb_FLUSHER_CLEANER_MAX_BUFFER_WORKDONE		
Tokudb_FLUSHER_CLEANER_MIN_BUFFER_WORKDONE		
Tokudb_FLUSHER_CLEANER_TOTAL_BUFFER_WORKDONE		
Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_STARTED		
Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_RUNNING		

Continued on next page

Table 56.2 – continued from previous page

Name	Var Type	Var Scope
Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_COMPLETED		
Tokudb_FLUSHER_CLEANER_NUM_DIRTIED_FOR_LEAF_MERGE		
Tokudb_FLUSHER_FLUSH_TOTAL		
Tokudb_FLUSHER_FLUSH_IN_MEMORY		
Tokudb_FLUSHER_FLUSH_NEEDED_IO		
Tokudb_FLUSHER_FLUSH_CASCADES		
Tokudb_FLUSHER_FLUSH_CASCADES_1		
Tokudb_FLUSHER_FLUSH_CASCADES_2		
Tokudb_FLUSHER_FLUSH_CASCADES_3		
Tokudb_FLUSHER_FLUSH_CASCADES_4		
Tokudb_FLUSHER_FLUSH_CASCADES_5		
Tokudb_FLUSHER_FLUSH_CASCADES_GT_5		
Tokudb_FLUSHER_SPLIT_LEAF		
Tokudb_FLUSHER_SPLIT_NONLEAF		
Tokudb_FLUSHER_MERGE_LEAF		
Tokudb_FLUSHER_MERGE_NONLEAF		
Tokudb_FLUSHER_BALANCE_LEAF		
Tokudb_HOT_NUM_STARTED		
Tokudb_HOT_NUM_COMPLETED		
Tokudb_HOT_NUM_ABORTED		
Tokudb_HOT_MAX_ROOT_FLUSH_COUNT		
Tokudb_TXN_BEGIN		
Tokudb_TXN_BEGIN_READ_ONLY		
Tokudb_TXN_COMMITS		
Tokudb_TXN_ABORTS		
Tokudb_LOGGER_NEXT_LSN		
Tokudb_LOGGER_WRITES		
Tokudb_LOGGER_WRITES_BYTES		
Tokudb_LOGGER_WRITES_UNCOMPRESSED_BYTES		
Tokudb_LOGGER_WRITES_SECONDS		
Tokudb_LOGGER_WAIT_LONG		
Tokudb_LOADER_NUM_CREATED		
Tokudb_LOADER_NUM_CURRENT		
Tokudb_LOADER_NUM_MAX		
Tokudb_MEMORY_MALLOC_COUNT		
Tokudb_MEMORY_FREE_COUNT		
Tokudb_MEMORY_REALLOC_COUNT		
Tokudb_MEMORY_MALLOC_FAIL		
Tokudb_MEMORY_REALLOC_FAIL		
Tokudb_MEMORY_REQUESTED		
Tokudb_MEMORY_USED		
Tokudb_MEMORY_FREED		
Tokudb_MEMORY_MAX_REQUESTED_SIZE		
Tokudb_MEMORY_LAST_FAILED_SIZE		
Tokudb_MEM_ESTIMATED_MAXIMUM_MEMORY_FOOTPRINT		
Tokudb_MEMORY_MALLOCATOR_VERSION		
Tokudb_MEMORY_MMAP_THRESHOLD		
Tokudb_FILESYSTEM_THREADS_BLOCKED_BY_FULL_DISK		

Continued on next page

Table 56.2 – continued from previous page

Name	Var Type	Var Scope
Tokudb_FILESYSTEM_FSYNC_TIME		
Tokudb_FILESYSTEM_FSYNC_NUM		
Tokudb_FILESYSTEM_LONG_FSYNC_TIME		
Tokudb_FILESYSTEM_LONG_FSYNC_NUM		

CHAPTER

FIFTYSEVEN

KNOWN ISSUES AND LIMITATIONS

In *Percona Server* 5.7 super_read_only feature has been replaced with upstream implementation. There are currently two known issues compared to *Percona Server* 5.6 implementation:

- Bug #78963, `super_read_only` aborts `STOP SLAVE` if variable `relay_log_info_repository` is set to `TABLE` which could lead to a server crash in Debug builds.
- Bug #79328, `super_read_only` set as a server option has no effect.

InnoDB crash recovery might fail if `innodb_flush_method` is set to `ALL_O_DIRECT`. The workaround is to set this variable to a different value before starting up the crashed instance (bug #1529885).

CHAPTER

FIFTYEIGHT

DEVELOPMENT OF PERCONA SERVER

Percona Server is an open source project to produce a distribution of the *MySQL* server with improved performance, scalability and diagnostics.

58.1 Submitting Changes

We keep trunk in a constant state of stability to allow for a release at any time and to minimize wasted time by developers due to broken code from somebody else interfering with their day.

You should also be familiar with our *Jenkins* setup.

58.1.1 Overview

At Percona we use Bazaar for source control and launchpad for both code hosting and release management.

Changes to our software projects could be because of a new feature (blueprint) or fixing a bug (bug). Projects such as refactoring could be classed as a blueprint or a bug depending on the scope of the work.

Blueprints and bugs are targeted to specific milestones (releases). A milestone is part of a series - e.g. 1.6 is a series in Percona XtraBackup and 1.6.1, 1.6.2 and 1.6.3 are milestones in the 1.6 series.

Code is proposed for merging in the form of merge requests on launchpad.

Some software (such as Percona Xtrabackup) we maintain both a development branch and a stable branch. For example: Xtrabackup 1.6 is the current stable series, and changes that should make it into bugfix releases of 1.6 should be proposed for the 1.6 tree. However, most new features or more invasive (or smaller) bug fixes should be targeted to the next release, currently 1.7. If submitting something to 1.6, you should also propose a branch that has these changes merged to the development release (1.7). This way somebody else doesn't have to attempt to merge your code and we get to run any extra tests that may be in the tree (and check compatibility with all platforms).

For Percona Server, we have two current bzr branches on which development occurs: 5.1 and 5.5. As Percona Server is not a traditional project, instead being a set of patches against an existing product, these two branches are not related. That is, we do not merge from one to the other. To have your changes in both, you must propose two branches: one for 5.1 version of patch and one for 5.5.

58.1.2 Making a change to a project

In this case we're going to use percona-xtrabackup as an example. workflow is similar for Percona Server, but patch will need to be modified both in 5.1 and 5.5 branches.

- `bzr branch lp:percona-xtrabackup featureX` (where 'featureX' is a sensible name for the task at hand)

- (developer makes changes in featureX, testing locally)
- Developer pushes to `lp:~username/percona-xtrabackup/featureX`
- When the developer thinks the branch may be ready to be merged, they will run the branch through param build.
- If there are any build or test failures, developer fixes them (in the case of failing tests in trunk... no more tests should fail. Eventually all tests will pass in trunk)
- Developer can then submit a merge proposal to lp:percona-xtrabackup, referencing URL for the param build showing that build and test passes
- Code undergoes review
- Once code is accepted, it can be merged (see other section)

If the change also applies to a stable release (e.g. 1.6) then changes should be made on a branch of 1.6 and merged to a branch of trunk. In this case there should be two branches run through param build and two merge proposals (one for 1.6 and one with the changes merged to trunk). This prevents somebody else having to guess how to merge your changes.

58.1.3 Merging approved branches

Before code hits trunk, it goes through a "staging" branch, where some extra tests may be run (e.g. valgrind) along with testing that all branches behave well together (build and test) before pushing to trunk.

To ensure quality, **DO NOT push directly to trunk!** everything must go through adequate testing first. This ensures that at any point trunk is in a releasable state.

Please note that **ALL changes must go through staging first** This is to ensure that several approved merge requests do not interact badly with each other.

- Merge captain (for lack of a better term for the person merging approved code into trunk) may collate several approved branches that have individually passed param-build as run by the original developers.
  - Workflow would look something like this:
    * `bzr branch lp:percona-xtrabackup staging`
    * `bzr merge lp:~user/percona-xtrabackup/featureX`
    * `bzr commit -m "merge feature X"`
    * `bzr merge lp:~user/percona-xtrabackup/featureY`
    * `bzr commit -m "merge feature Y"`
    * `bzr push --overwrite lp:percona-xtrabackup/staging'`
    * Run `lp:percona-xtrabackup/staging` through param build (in future, we'll likely have a Jenkins job specifically for this)
    * If build succeeds, `bzr push lp:percona-server` (and branches will be automatically marked as 'merged'.. although bug reports will need to be manually changed to 'Fix Released')
    * If build or test fails, attempt to find which branch may be the cause, and repeat process but without that branch.
- Any failing branch will be set to 'Work in Progress' with a 'Needs fixing' review with the URL of the build in jenkins where the failure occured. This will allow developers to fix their code.

58.1.4 Resubmitting a merge request

In the event of a merge request being marked as 'Work In Progress' due to build/test failures when merging, the developer should fix up the branch, run through param build and then 'Resubmit' the merge proposal.

There is a link on launchpad to resubmit the merge proposal, this means it appears in the list of merge requests to review again rather than off in the "work in progress" section.

58.1.5 Percona Server

The same process for Percona Server, but we have different branches (and merge requests) for 5.1 and 5.5 series.

58.1.6 Upgrading MySQL base version

- Same process as other modifications.
- create local branch
- make changes
- param build
- merge request

We will need some human processes to ensure that we do not merge extra things during the time when base MySQL version is being updated to avoid making life harder for the person doing the update.

58.2 Making a release

- `bzr branch lp:project release-project-VERSION`
- build packages
- perform any final tests (as we transition, this will already have been done by jenkins)
- `bzr tag project-version`
- merge request back to lp:project including the tag (TODO: write exact bzr commands for this)

This way anybody can easily check out an old release by just using bzr to branch the specific tag.

58.3 Jenkins

Our Jenkins instance uses a mixture of VMs on physical hosts that Percona runs and Virtual Machines in Amazon EC2 that are launched on demand.

58.3.1 Basic Concepts

We have some jobs that are activated based on source control changes (new commits in a bzr repository). We have some that are "param build" - that is, a user specifies parameters for the build (e.g. the bzr tree). A param-build allows developers to ensure their branch compiles and passes tests on all supported platforms *before* submitting a merge request. This helps us maintain the quality of the main bzr branches and not block other developers work.

Jenkins is a Master/Slave system and the jenkins master schedules the builds across available machines (and may launch new VMs in EC2 to meet demand).

Most of our jobs are what's known as "matrix builds". That is, a job that will be run with several different configurations of the project (e.g. release, debug) across several platforms (e.g. on a host matching the label of "centos5-32" and a host matching label of "ubuntu-natty-32bit"). Matrix builds show a table of lights to indicate their status. Clicking "build now" on one of these queues up builds for all of the combinations.

We have some integration of our regression test suites (currently xtrabackup) with Jenkins ability to parse JUnitXML, presenting a nice user interface to any test failures.

Because building some projects is non-trivial, in order to not duplicate the list of compile instructions for each job, we use template builds. You'll see builds such as percona-xtrabackup-template which is a disabled job, but all current xtrabackup jobs point to it for the commands to build and run the test suite.

58.3.2 Percona Xtrabackup

http://jenkins.percona.com/view/Percona%20Xtrabackup/

We currently build both xtrabackup 1.6 and xtrabackup trunk (will become 1.7).

There are param-builds for 1.6 and trunk too. These should be run for each merge request (and before any collection of merged branches is pushed to trunk)

58.3.3 Percona Server

We have separate jobs for Percona Server 5.1 and Percona Server 5.5 due to the different build systems that MySQL 5.1 and 5.5 use.

The `mysql-test-run.pl` test suite is integrated with Jenkins through subunit and `subunit2junitxml` allowing us to easily see which tests passed/failed on any particular test run.

Percona Server 5.1

http://jenkins.percona.com/view/Percona%20Server%205.1/

We have trunk and param jobs. We also have a valgrind job that will run after a successful trunk build.

Percona Server 5.5

http://jenkins.percona.com/view/Percona%20Server%205.5/

Similar to 5.1, but for PS5.5 instead.

58.3.4 MySQL Builds

http://jenkins.percona.com/view/MySQL/

I've set up a few jobs in Jenkins that should help us predict the future for Percona Server. Namely, if upstream MySQL may cause us any problems.

I wanted to see if some test failures were possibly upstream, so I set up two jobs:

http://jenkins.percona.com/view/MySQL/job/mysql-5.1-url-param/ http://jenkins.percona.com/view/MySQL/job/mysql-5.5-url-param/

both of which ask for a URL to a MySQL source tarball and then do a full build and test across the platforms we have in jenkins.

But my next thought was that we could try and do this *before* the source tarballs come out - hopefully then being able to have MySQL release source tarballs that do in fact pass build and test everywhere where we're wanting to support Percona Server.

http://jenkins.percona.com/view/MySQL/job/mysql-5.1-trunk/ http://jenkins.percona.com/view/MySQL/job/mysql-5.5-trunk/

are scheduled to just try once per week (we can change the frequency if we want to) to build and test from the MySQL bzr trees.

I also have a valgrind build (same configuration as for Percona Server) to help us see if there's any new valgrind warnings (or missed suppressions).

I'm hoping that these jobs will help us catch any future problems before they become our problem. (e.g. we can easily see that the sporadic test failures we see in Percona Server are actually in upstream MySQL).

CHAPTER

FIFTYNINE

TRADEMARK POLICY

This Trademark Policy is to ensure that users of Percona-branded products or services know that what they receive has really been developed, approved, tested and maintained by Percona. Trademarks help to prevent confusion in the marketplace, by distinguishing one company's or person's products and services from another's.

Percona owns a number of marks, including but not limited to Percona, XtraDB, Percona XtraDB, XtraBackup, Percona XtraBackup, Percona Server, and Percona Live, plus the distinctive visual icons and logos associated with these marks. Both the unregistered and registered marks of Percona are protected.

Use of any Percona trademark in the name, URL, or other identifying characteristic of any product, service, website, or other use is not permitted without Percona's written permission with the following three limited exceptions.

First, you may use the appropriate Percona mark when making a nominative fair use reference to a bona fide Percona product.

Second, when Percona has released a product under a version of the GNU General Public License ("GPL"), you may use the appropriate Percona mark when distributing a verbatim copy of that product in accordance with the terms and conditions of the GPL.

Third, you may use the appropriate Percona mark to refer to a distribution of GPL-released Percona software that has been modified with minor changes for the sole purpose of allowing the software to operate on an operating system or hardware platform for which Percona has not yet released the software, provided that those third party changes do not affect the behavior, functionality, features, design or performance of the software. Users who acquire this Percona-branded software receive substantially exact implementations of the Percona software.

Percona reserves the right to revoke this authorization at any time in its sole discretion. For example, if Percona believes that your modification is beyond the scope of the limited license granted in this Policy or that your use of the Percona mark is detrimental to Percona, Percona will revoke this authorization. Upon revocation, you must immediately cease using the applicable Percona mark. If you do not immediately cease using the Percona mark upon revocation, Percona may take action to protect its rights and interests in the Percona mark. Percona does not grant any license to use any Percona mark for any other modified versions of Percona software; such use will require our prior written permission.

Neither trademark law nor any of the exceptions set forth in this Trademark Policy permit you to truncate, modify or otherwise use any Percona mark as part of your own brand. For example, if XYZ creates a modified version of the Percona Server, XYZ may not brand that modification as "XYZ Percona Server" or "Percona XYZ Server", even if that modification otherwise complies with the third exception noted above.

In all cases, you must comply with applicable law, the underlying license, and this Trademark Policy, as amended from time to time. For instance, any mention of Percona trademarks should include the full trademarked name, with proper spelling and capitalization, along with attribution of ownership to Percona Inc. For example, the full proper name for XtraBackup is Percona XtraBackup. However, it is acceptable to omit the word "Percona" for brevity on the second and subsequent uses, where such omission does not cause confusion.

In the event of doubt as to any of the conditions or exceptions outlined in this Trademark Policy, please contact trademarks@percona.com for assistance and we will do our very best to be helpful.

CHAPTER

SIXTY

INDEX OF INFORMATION_SCHEMA TABLES

This is a list of the INFORMATION_SCHEMA TABLES that exist in *Percona Server* with *XtraDB*. The entry for each table points to the page in the documentation where it's described.

- CLIENT_STATISTICS
- GLOBAL_TEMPORARY_TABLES
- INDEX_STATISTICS
- INNODB_CHANGED_PAGES
- QUERY_RESPONSE_TIME
- TABLE_STATISTICS
- TEMPORARY_TABLES
- THREAD_STATISTICS
- USER_STATISTICS
- XTRADB_INTERNAL_HASH_TABLES
- XTRADB_READ_VIEW
- XTRADB_RSEG

CHAPTER

SIXTYONE

FREQUENTLY ASKED QUESTIONS

61.1 Q: Will *Percona Server* with *XtraDB* invalidate our *MySQL* support?

A: We don't know the details of your support contract. You should check with your *Oracle* representative. We have heard anecdotal stories from *MySQL* Support team members that they have customers who use *Percona Server* with *XtraDB*, but you should not base your decision on that.

61.2 Q: Will we have to *GPL* our whole application if we use *Percona Server* with *XtraDB*?

A: This is a common misconception about the *GPL*. We suggest reading the *Free Software Foundation* 's excellent reference material on the GPL Version 2, which is the license that applies to *MySQL* and therefore to *Percona Server* with *XtraDB*. That document contains links to many other documents which should answer your questions. *Percona* is unable to give legal advice about the *GPL*.

61.3 Q: Do I need to install *Percona* client libraries?

A: No, you don't need to change anything on the clients. *Percona Server* is 100% compatible with all existing client libraries and connectors.

61.4 Q: When using the *Percona XtraBackup* to setup a replication slave on Debian based systems I'm getting: "ERROR 1045 (28000): Access denied for user 'debian-sys-maint'@'localhost' (using password: YES)"

A: In case you're using init script on Debian based system to start `mysqld`, be sure that the password for `debian-sys-maint` user has been updated and it's the same as that user's password from the server that the backup has been taken from. Password can be seen and updated in `/etc/mysql/debian.cnf`. For more information on how to set up a replication slave using *Percona XtraBackup* see this how-to.

CHAPTER

SIXTYTWO

COPYRIGHT AND LICENSING INFORMATION

62.1 Documentation Licensing

62.2 Software License

Percona Server is built upon MySQL from Oracle. Along with making our own modifications, we merge in changes from other sources such as community contributions and changes from MariaDB.

The original SHOW USER/TABLE/INDEX statistics code came from Google.

Percona does not require copyright assignment.

See the COPYING files accompanying the software distribution.

CHAPTER

SIXTYTHREE

PERCONA SERVER 5.7 RELEASE NOTES

63.1 *Percona Server* 5.7.11-4

Percona is glad to announce the GA (Generally Available) release of *Percona Server* 5.7.11-4 on March 15th, 2016 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.7.11, including all the bug fixes in it, *Percona Server* 5.7.11-4 is the current GA release in the *Percona Server* 5.7 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.7.11-4 milestone at Launchpad

63.1.1 New Features

Percona Server has implemented *Parallel doublewrite buffer*.

TokuDB Background ANALYZE TABLE feature is now enabled by default (`tokudb_analyze_in_background` is set to `ON` by default). Variable `tokudb_auto_analyze` default value has been changed from `0` to `30`. (#935)

Suppress Warning Messages feature has been removed from *Percona Server* 5.7 because *MySQL* 5.7.11 has implemented a new system variable, log_statements_unsafe_for_binlog, which implements the same effect.

63.1.2 Bugs Fixed

If `pid-file` option wasn't specified with the full path, *Ubuntu*/*Debian* `sysvinit` script wouldn't notice the server is actually running and it will timeout or in some cases even hang. Bug fixed #1549333.

Buffer pool may fail to remove dirty pages for a particular tablesspace from the flush list, as requested by, for example, `DROP TABLE` or `TRUNCATE TABLE` commands. This could lead to a crash. Bug fixed #1552673.

Audit Log Plugin worker thread may crash on write call writing fewer bytes than requested. Bug fixed #1552682 (upstream #80606).

Percona Server 5.7 `systemd` script now takes the last option specified in `my.cnf` if the same option is specified multiple times. Previously it would try to take all values which would break the script and server would fail to start. Bug fixed #1554976.

`mysqldumpslow` script has been removed because it was not compatible with *Percona Server* extended slow query log format. Please use pt-query-digest from *Percona Toolkit* instead. Bug fixed #856910.

Other bugs fixed: #1521120, #1549301 (upstream #80496), and #1554043 (upstream #80607).

63.2 *Percona Server* 5.7.10-3

Percona is glad to announce the first GA (Generally Available) release of *Percona Server* 5.7.10-3 on February 23rd, 2016 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.7.10, including all the bug fixes in it, *Percona Server* 5.7.10-3 is the current Generally Available release in the *Percona Server* 5.7 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.7.10-3 milestone at Launchpad

63.2.1 New Features

Complete list of changes between *Percona Server* 5.6 and 5.7 can be seen in *Changed in Percona Server 5.7*.

Percona Server has implemented *Multi-threaded asynchronous LRU flusher*. This work also allows to safely use `backoff` value for the `innodb_empty_free_list_algorithm` server system variable, and its default has been changed accordingly.

63.2.2 Known Issues

In *Percona Server* 5.7 super_read_only feature has been replaced with upstream implementation. There are currently two known issues compared to *Percona Server* 5.6 implementation:

- Bug #78963, `super_read_only` aborts `STOP SLAVE` if variable `relay_log_info_repository` is set to `TABLE` which could lead to a server crash in Debug builds.
- Bug #79328, `super_read_only` set as a server option has no effect.

InnoDB crash recovery might fail if `innodb_flush_method` is set to `ALL_O_DIRECT`. The workaround is to set this variable to a different value before starting up the crashed instance (bug #1529885).

63.2.3 Bugs Fixed

Percona Server `5.7.10-1` didn't write the initial root password into the log file `/var/log/mysqld.log` during the installation on *CentOS 6*. Bug fixed #1541769.

Cardinality of partitioned *TokuDB* tables became inaccurate after the changes introduced by *TokuDB Background ANALYZE TABLE* feature in *Percona Server* `5.7.10-1`. Bug fixed #925.

Running the `TRUNCATE TABLE` while *TokuDB Background ANALYZE TABLE* is enabled could lead to a server crash once analyze job tries to access the truncated table. Bug fixed #938.

Percona TokuBackup would fail with an unclear error if backup process found `mysqld_safe.pid` file (owned by root) inside the `datadir`. Fixed by excluding the `pid` file by default. Bug fixed #125.

PAM Authentication Plugin build warning has been fixed. Bug fixed #1541601.

63.3 *Percona Server* 5.7.10-2

Percona is glad to announce the second Release Candidate release of *Percona Server* 5.7.10-2 on February 5th, 2016 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.7.10, including all the bug fixes in it, *Percona Server* 5.7.10-2 is the current Release Candidate release in the *Percona Server* 5.7 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.7.10-2 milestone at Launchpad

63.3.1 New Features

Complete list of changes between *Percona Server* 5.6 and 5.7 can be seen in *Changed in Percona Server 5.7*.

5.7 binlog group commit algorithm is now supported in *TokuDB* as well.

New *TokuDB* index statistics reporting has been implemented to be compatible with the changes implemented in upstream 5.7. Following the *InnoDB* example, the default value for `tokudb_cardinality_scale_percent` has been changed from `50%` to `100%`. Implementing this also addresses a server crash deep in the optimizer code.

63.3.2 Known Issues

In *Percona Server* 5.7 super_read_only feature has been replaced with upstream implementation. There are currently two known issues compared to *Percona Server* 5.6 implementation:

- Bug #78963, `super_read_only` aborts `STOP SLAVE` if variable `relay_log_info_repository` is set to `TABLE` which could lead to a server crash in Debug builds.
- Bug #79328, `super_read_only` set as a server option has no effect.

InnoDB crash recovery might fail if `innodb_flush_method` is set to `ALL_O_DIRECT`. The workaround is to set this variable to a different value before starting up the crashed instance (bug #1529885).

63.3.3 Bugs Fixed

Clustering secondary index could not be created on a partitioned *TokuDB* table. Bug fixed #1527730 (#720).

Percona TokuBackup was failing to compile with *Percona Server* 5.7. Bug fixed #123.

Granting privileges to a user authenticating with *PAM Authentication Plugin* could lead to a server crash. Bug fixed #1521474.

TokuDB status variables were missing from *Percona Server* `5.7.10-1`. Bug fixed #1527364 (#923).

Attempting to rotate the audit log file would result in audit log file name `foo.log.%u` (literally) instead of a numeric suffix. Bug fixed #1528603.

Adding an index to an *InnoDB* temporary table while `expand_fast_index_creation` was enabled could lead to server assertion. Bug fixed #1529555.

TokuDB would not be upgraded on *Debian*/*Ubuntu* distributions while performing an upgrade from *Percona Server* 5.6 to *Percona Server* 5.7 even if explicitly requested. Bug fixed #1533580.

Server would assert when both *TokuDB* and *InnoDB* tables were used within one transaction on a replication slave which has binary log enabled and slave updates logging disabled. Bug fixed #1534249 (upstream bug #80053).

MeCab Full-Text Parser Plugin has not been included in the previous release. Bug fixed #1534617.

Fixed server assertion caused by `Performance Schema` memory key mix-up in `SET STATEMENT ... FOR ...` statements. Bug fixed #1534874.

Service name on *CentOS* 6 has been renamed from `mysqld` back to `mysql`. This change requires manual service restart after being upgraded from *Percona Server* `5.7.10-1`. Bug fixed #1542332.

Setting the `innodb_sched_priority_purge` (available only in debug builds) while purge threads were stopped would cause a server crash. Bug fixed #1368552.

Enabling *TokuDB* with `ps_tokudb_admin` script inside the Docker container would cause an error due to insufficient privileges even when running as root. In order for this script to be used inside docker containers this error has been changed to a warning that a check is impossible. Bug fixed #1520890.

Write-heavy workload with a small buffer pool could lead to a deadlock when free buffers are exhausted. Bug fixed #1521905.

InnoDB status will start printing negative values for spin rounds per wait, if the wait number, even though being accounted as a signed 64-bit integer, will not fit into a signed 32-bit integer. Bug fixed #1527160 (upstream #79703).

Percona Server 5.7 couldn't be restarted after *TokuDB* has been installed with `ps_tokudb_admin` script. Bug fixed #1527535.

Fixed memory leak when `utility_user` is enabled. Bug fixed #1530918.

Page cleaner worker threads were not instrumented for `Performance Schema`. Bug fixed #1532747 (upstream bug #79894).

Busy server was preferring LRU flushing over flush list flushing too strongly which could lead to performance degradation. Bug fixed #1534114.

`libjemalloc.so.1` was missing from binary tarball. Bug fixed #1537129.

When `cmake/make/make_binary_distribution` workflow was used to produce binary tarballs it would produce tarballs with `mysql-...` naming instead of `percona-server-...`. Bug fixed #1540385.

Added proper memory cleanup if for some reason a table is unable to be opened from a dead closed state. This prevents an assertion from happening the next time the table is attempted to be opened. Bug fixed #917.

Variable `tokudb_support_xa` has been modified to prevent setting it to anything but `ON`/`ENABLED` and to print a SQL warning anytime an attempt is made to change it, just like `innodb_support_xa`. Bug fixed #928.

Other bugs fixed: #1179451, #1534246, #1524763, #1525109 (upstream #79569), #1530102, #897, #898, #899, #900, #901, #902, #903, #905, #906, #907, #908, #909, #910, #911, #912, #913, #915, #919, and #904.

63.4 *Percona Server* 5.7.10-1

Percona is glad to announce the first Release Candidate release of *Percona Server* 5.7.10-1 on December 14th, 2015 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.7.10, including all the bug fixes in it, *Percona Server* 5.7.10-1 is the current Release Candidate release in the *Percona Server* 5.7 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.7.10-1 milestone at Launchpad

This release contains all the bug fixes from latest *Percona Server* 5.6 release (currently *Percona Server* 5.6.27-76.0).

63.4.1 New Features

Percona Server 5.7.10-1 is not available on *RHEL* 5 family of Linux distributions and *Debian* 6 (squeeze).

Complete list of changes between *Percona Server* 5.6 and 5.7 can be seen in *Changed in Percona Server 5.7*.

63.4.2 Known issues

MeCab Full-Text Parser Plugin has not been included in this release.

PAM Authentication Plugin currently isn't working correctly.

Variables `innodb_show_verbose_locks` and `innodb_show_locks_held` are not working correctly.

In *Percona Server* 5.7 super_read_only feature has been replaced with upstream implementation. There are currently two known issues compared to *Percona Server* 5.6 implementation:

- Bug #78963, `super_read_only` aborts `STOP SLAVE` if variable `relay_log_info_repository` is set to `TABLE` which could lead to a server crash in Debug builds.
- Bug #79328, `super_read_only` set as a server option has no effect.

Using primary key with a `BLOB` in *TokuDB* table could lead to a server crash (#916).

Using XA transactions with *TokuDB* could lead to a server crash (#900).

Percona TokuBackup has not been included in this release.

63.4.3 Bugs Fixed

Running `ALTER TABLE` without specifying the storage engine (without `ENGINE=` clause) or `OPTIMIZE TABLE` when `enforce_storage_engine` was enabled could lead to unrequested and unexpected storage engine changes. If done for a system table, it would circumvent regular system table storage engine compatibility checks, resulting in crashes or otherwise broken server operation. Bug fixed #1488055.

Some transaction deadlocks did not increase the `INFORMATION_SCHEMA.INNODB_METRICS lock_deadlocks` counter. Bug fixed #1466414 (upstream #77399).

Removed excessive locking during the buffer pool resize when checking whether AHI is enabled. Bug fixed #1525215 (upstream #78894).

Removed unnecessary code in InnoDB error monitor thread. Bug fixed #1521564 (upstream #79477).

With *Expanded Fast Index Creation* enabled, DDL queries involving *InnoDB* temporary tables would cause later queries on the same tables to produce warnings that their indexes were not found in the index translation table. Bug fixed #1233431.

Other bugs fixed: #371752 (upstream #45379), #1441362 (upstream #56155), #1385062 (upstream #74810), #1519201 (upstream #79391), #1515602, #1506697 (upstream #57552), #1501089 (upstream #75239), #1447527 (upstream #75368), #1384658 (upstream #74619), #1384656 (upstream #74584), and #1192052.

CHAPTER

SIXTYFOUR

GLOSSARY

ACID Set of properties that guarantee database transactions are processed reliably. Stands for *Atomicity*, *Consistency*, *Isolation*, *Durability*.

Atomicity Atomicity means that database operations are applied following a "all or nothing" rule. A transaction is either fully applied or not at all.

Consistency Consistency means that each transaction that modifies the database takes it from one consistent state to another.

Durability Once a transaction is committed, it will remain so.

Foreign Key A referential constraint between two tables. Example: A purchase order in the purchase_orders table must have been made by a customer that exists in the customers table.

Isolation The Isolation requirement means that no transaction can interfere with another.

InnoDB A *Storage Engine* for MySQL and derivatives (*Percona Server*, *MariaDB*) originally written by Innobase Oy, since acquired by Oracle. It provides *ACID* compliant storage engine with *foreign key* support. As of *MySQL* version 5.5, InnoDB became the default storage engine on all platforms.

Jenkins Jenkins is a continuous integration system that we use to help ensure the continued quality of the software we produce. It helps us achieve the aims of:

- no failed tests in trunk on any platform,
- aid developers in ensuring merge requests build and test on all platforms,
- no known performance regressions (without a damn good explanation).

LSN Log Serial Number. A term used in relation to the *InnoDB* or *XtraDB* storage engines.

MariaDB A fork of *MySQL* that is maintained primarily by Monty Program AB. It aims to add features, fix bugs while maintaining 100% backwards compatibility with MySQL.

my.cnf The file name of the default MySQL configuration file.

MyISAM A *MySQL Storage Engine* that was the default until MySQL 5.5.

MySQL An open source database that has spawned several distributions and forks. MySQL AB was the primary maintainer and distributor until bought by Sun Microsystems, which was then acquired by Oracle. As Oracle owns the MySQL trademark, the term MySQL is often used for the Oracle distribution of MySQL as distinct from the drop-in replacements such as *MariaDB* and *Percona Server*.

Percona Server Percona's branch of *MySQL* with performance and management improvements.

Storage Engine A *Storage Engine* is a piece of software that implements the details of data storage and retrieval for a database system. This term is primarily used within the *MySQL* ecosystem due to it being the first widely used relational database to have an abstraction layer around storage. It is analogous to a Virtual File System layer in an Operating System. A VFS layer allows an operating system to read and write multiple file systems (e.g.

FAT, NTFS, XFS, ext3) and a Storage Engine layer allows a database server to access tables stored in different engines (e.g. *MyISAM*, InnoDB).

XtraDB Percona's improved version of *InnoDB* providing performance, features and reliability above what is shipped by Oracle in InnoDB.

- *genindex*
- *modindex*

Symbols

5.7.10-1 (release notes), 298
5.7.10-2 (release notes), 296
5.7.10-3 (release notes), 295
5.7.11-4 (release notes), 295

A

ACID, **300**
Atomicity, **300**
audit_log_buffer_size (variable), 101
audit_log_file (variable), 101
audit_log_flush (variable), 101
audit_log_format (variable), 101
audit_log_handler (variable), 102
audit_log_policy (variable), 102
audit_log_rotate_on_size (variable), 102
audit_log_rotations (variable), 102
audit_log_strategy (variable), 100
audit_log_syslog_facility (variable), 103
audit_log_syslog_ident (variable), 103
audit_log_syslog_priority (variable), 103

B

Binlog_snapshot_file (variable), 105
Binlog_snapshot_position (variable), 105

C

CLIENT_STATISTICS (table), 110
Com_lock_binlog_for_backup (variable), 97
Com_lock_tables_for_backup (variable), 97
Com_show_client_statistics (variable), 115
Com_show_index_statistics (variable), 115
Com_show_table_statistics (variable), 115
Com_show_thread_statistics (variable), 115
Com_show_user_statistics (variable), 115
Com_unlock_binlog (variable), 97
Consistency, **300**
csv_mode (variable), 66

D

Durability, **300**

E

enforce_storage_engine (variable), 78
expand_fast_index_creation (variable), 92
extra_max_connections (variable), 45
extra_port (variable), 45

F

flush_caches (variable), 40
Foreign Key, **300**

G

GLOBAL_TEMPORARY_TABLES (table), 133

H

have_backup_locks (variable), 96
have_backup_safe_binlog_info (variable), 96
have_snapshot_cloning (variable), 105

I

INDEX_STATISTICS (table), 111
InnoDB, **300**
Innodb_background_log_sync (variable), 124
Innodb_buffer_pool_pages_LRU_flushed (variable), 127
Innodb_buffer_pool_pages_made_not_young (variable), 127
Innodb_buffer_pool_pages_made_young (variable), 127
Innodb_buffer_pool_pages_old (variable), 127
INNODB_CHANGED_PAGES (table), 87
Innodb_checkpoint_age (variable), 126
Innodb_checkpoint_max_age (variable), 126
innodb_corrupt_table_action (variable), 73
Innodb_descriptors_memory (variable), 127
innodb_empty_free_list_algorithm (variable), 47
innodb_flush_method (variable), 35
Innodb_ibuf_free_list (variable), 125
Innodb_ibuf_segment_size (variable), 125
innodb_kill_idle_transaction (variable), 77
Innodb_lsn_current (variable), 125
Innodb_lsn_flushed (variable), 126
Innodb_lsn_last_checkpoint (variable), 126
Innodb_master_thread_active_loops (variable), 124
Innodb_master_thread_idle_loops (variable), 124

innodb_max_bitmap_file_size (variable), 88
innodb_max_changed_pages (variable), 87
Innodb_max_trx_id (variable), 128
Innodb_mem_adaptive_hash (variable), 126
Innodb_mem_dictionary (variable), 127
Innodb_mem_total (variable), 127
Innodb_oldest_view_low_limit_trx_id (variable), 128
innodb_parallel_doublewrite_path (variable), 49
Innodb_purge_trx_id (variable), 128
Innodb_purge_undo_no (variable), 128
Innodb_read_views_memory (variable), 127
innodb_sched_priority_master (variable), 49
innodb_show_locks_held (variable), 124
innodb_show_verbose_locks (variable), 123
innodb_track_changed_pages (variable), 87
innodb_use_global_flush_log_at_trx_commit (variable), 35
Isolation, **300**

J

Jenkins, **300**

L

lock-for-backup (option), 97
log_slow_filter (variable), 116
log_slow_rate_limit (variable), 117
log_slow_rate_type (variable), 117
log_slow_sp_statements (variable), 118
log_slow_verbosity (variable), 119
log_warnings_suppress (variable), 51
LSN, **300**

M

MariaDB, **300**
max_binlog_files (variable), 58
max_slowlog_files (variable), 63
max_slowlog_size (variable), 63
my.cnf, **300**
MyISAM, **300**
MySQL, **300**

P

Percona Server, **300**
PROCESSLIST (table), 131
proxy_protocol_networks (variable), 67
pseudo_server_id (variable), 69

Q

query_cache_strip_comments (variable), 39
QUERY_RESPONSE_TIME (table), 143
query_response_time_flush (variable), 142
query_response_time_range_base (variable), 143
QUERY_RESPONSE_TIME_READ (table), 143
query_response_time_stats (variable), 143
QUERY_RESPONSE_TIME_WRITE (table), 144

S

scalability_metrics_busytime (variable), 138
scalability_metrics_concurrency (variable), 138
scalability_metrics_control (variable), 137
scalability_metrics_elapsedtime (variable), 138
scalability_metrics_queries (variable), 138
scalability_metrics_totaltime (variable), 138
slow_query_log_always_write_time (variable), 120
slow_query_log_use_global_control (variable), 119
Storage Engine, **300**

T

TABLE_STATISTICS (table), 112
TEMPORARY_TABLES (table), 133
thread_pool_high_prio_mode (variable), 43
thread_pool_high_prio_tickets (variable), 43
thread_pool_idle_timeout (variable), 43
thread_pool_max_threads (variable), 44
thread_pool_oversubscribe (variable), 44
thread_pool_size (variable), 44
thread_pool_stall_limit (variable), 44
THREAD_STATISTICS (table), 112
thread_statistics (variable), 109
Threadpool_idle_threads (variable), 45
Threadpool_threads (variable), 45
tokudb_analyze_in_background (variable), 217
tokudb_analyze_mode (variable), 217
tokudb_analyze_throttle (variable), 218
tokudb_analyze_time (variable), 218
tokudb_auto_analyze (variable), 218
TOKUDB_BACKGROUND_JOB_STATUS (table), 219
tokudb_backup_allowed_prefix (variable), 166, 228
tokudb_backup_dir (variable), 163, 224
tokudb_backup_exclude (variable), 163, 225
tokudb_backup_last_error (variable), 163, 225
tokudb_backup_last_error_string (variable), 163, 225
tokudb_backup_plugin_version (variable), 166, 227
tokudb_backup_throttle (variable), 163, 165, 224, 227
tokudb_backup_version‘‘ (variable), 166, 228
tokudb_block_size (variable), 161, 223
tokudb_bulk_fetch (variable), 162, 224
tokudb_cache_size (variable), 164, 225
tokudb_cachetable_pool_threads (variable), 167, 228
tokudb_cardinality_scale_percent (variable), 219
tokudb_checkpoint_pool_threads (variable), 167, 229
tokudb_checkpointing_period (variable), 165, 226
tokudb_cleaner_iterations (variable), 165, 227
tokudb_cleaner_period (variable), 165, 227
tokudb_client_pool_threads (variable), 167, 228
tokudb_commit_sync (variable), 160, 221
tokudb_compress_buffers_before_eviction (variable), 168, 229

tokudb_create_index_online (variable), 161, 222
tokudb_data_dir (variable), 164, 226
tokudb_directio (variable), 164, 225
tokudb_disable_prefetching (variable), 161, 223
tokudb_disable_slow_alter (variable), 161, 222
tokudb_enable_partial_eviction (variable), 167, 229
tokudb_fanout (variable), 166, 228
tokudb_fs_reserve_percent (variable), 165, 226
tokudb_fsync_log_period (variable), 164, 225
tokudb_last_lock_timeout (variable), 162, 224
tokudb_load_save_space (variable), 160, 222
tokudb_loader_memory_size (variable), 163, 225
tokudb_lock_timeout (variable), 164, 226
tokudb_lock_timeout_debug (variable), 162, 223
tokudb_log_dir (variable), 164, 226
tokudb_optimize_index_fraction (variable), 163, 224
tokudb_optimize_index_name (variable), 163, 224
tokudb_optimize_throttling (variable), 162, 224
tokudb_pk_insert_mode (variable), 160, 221
tokudb_prelock_empty (variable), 160, 222
tokudb_read_block_size (variable), 161, 223
tokudb_read_buf_size (variable), 161, 223
tokudb_read_status_frequency (variable), 165, 226
tokudb_row_format (variable), 162, 223
tokudb_rpl_check_readonly (variable), 166, 228
tokudb_rpl_lookup_rows (variable), 165, 227
tokudb_rpl_lookup_rows_delay (variable), 166, 227
tokudb_rpl_unique_checks (variable), 166, 227
tokudb_rpl_unique_checks_delay (variable), 166, 227
tokudb_strip_frm_data (variable), 168, 229
tokudb_support_xa (variable), 162, 224
tokudb_tmp_dir (variable), 164, 226
tokudb_write_status_frequency (variable), 165, 226

U

unique_checks (variable), 159, 221
USER_STATISTICS (table), 113
userstat (variable), 109
utility_user (variable), 81
utility_user_password (variable), 81
utility_user_privileges (variable), 82
utility_user_schema_access (variable), 81

X

XtraDB, **301**
XTRADB_INTERNAL_HASH_TABLES (table), 128
XTRADB_READ_VIEW (table), 128
XTRADB_RSEG (table), 134