TencentDB for MySQL

自社研究カーネルTXSQL

製品ドキュメント

カタログ：

自社研究カーネルTXSQL
TXSQLカーネル概要

最終更新日：：2023-04-10 16:09:48

TXSQLは、Tencent Cloud Databaseチームが保守するMySQLブランチであり、オリジナルMySQLバージョンと100%互換性があり、TXSQLはまたエンタープライズクラスの透過的データ暗号化、監査、動的スレッドプール、暗号化関数、バックアップ・リカバリ機能など、MySQL Enterprise Editionと同様な機能を提供します。

TXSQLは、InnoDBストレージエンジン、クエリの最適化、レプリケーションのパフォーマンスなどを大幅に最適化するだけではなく、TencentDB for MySQLの使いやすさと保守性を向上させ、MySQLのすべての機能をユーザーに提供するとともに、エンタープライズクラスの災害復旧、リカバリ、監視、パフォーマンスの最適化、読み書き分離、透過的データ暗号化、監査などの高度な機能を提供します。

TXSQLカーネルに関する他の情報：

- TencentDB for MySQLカーネルのバージョンに関する最新情報については、カーネルバージョンのアップグレードに関する最新情報をご参照ください。
- TencentDB for MySQLは、自動または手動によるカーネルマイナーバージョンのアップグレードに対応しています。カーネルマイナーバージョンのアップグレードをご参照ください。
- TencentDB for MySQLは、CVMを通じてMySQLインスタンスにログインしてカーネルマイナーバージョンを確認することをサポートします。カーネルマイナーバージョンの確認をご参照ください。

機能系特性
アイドル状態のトランザクションの自動kill

最終更新日：：2022-02-28 15:23:33

機能の説明

一定時間以上アイドル状態のトランザクションをKillすることで、速やかにリソースをリリースします。

サポートするバージョン

- カーネルバージョン MySQL 5.6 20180915およびそれ以降
- カーネルバージョン MySQL 5.7 20180918およびそれ以降
- カーネルバージョン MySQL 8.0 20200630およびそれ以降

ユースケース

トランザクションが開始状態の接続（begin、start transactionを使用した明示的な、または暗黙的なトランザクションの開始）について、タイムアウト時間内に次のステートメントが実行されない場合、接続をkillします。

利用説明

パラメータcdb_kill_idle_trans_timeoutによって、この機能を有効にするかどうかを制御します。0は不使用、0以外ではオンになります。sessionのwait_timeout値と比べて小さい方の値を取ります。

パラメータ名	動的	タイプ	デフォルト	パラメータ値範囲	説明
cdb_kill_idle_trans_timeout	YES	ulong	0	[0, 31536000]	0はこの機能をオフにすることを表します。そうしない場合はcdb_kill_idle_trans_time秒のアイドル状態のトランザクションはkillされます

説明：
ユーザーは現在、上記パラメータのパラメータ値を直接変更することはできません。変更する場合は、チケットを提出から変更することができます。

並列レプリケーション

最終更新日：：2023-01-04 17:09:08

機能の説明

公式のMySQL 5.6以前のバージョンでは、slaveノードで再生を行い、masterノードでbinlogを同期する際はすべてシングルスレッド方式でしたが、5.6およびそれ以降のバージョンでは並列方式に変更されています。ただし公式の並列はdatabaseとlogical clockに基づいて行われるため、並列の粒度が大きく、多くの状況下では理想的な並列効果が得られません。

Tencent Cloud TXSQLカーネルチームは並列レプリケーション機能を最適化し、tableごとの並列をサポートしました。これは粒度をテーブルに分割することに相当し、並列性を向上させてマスター/スレーブの遅延を減少させることができます。

サポートするバージョン

- カーネルバージョン MySQL 8.0 20201230以降
- カーネルバージョン MySQL 5.7 20180530以降
- カーネルバージョン MySQL 5.6 20170830以降

ユースケース

この機能は主に一部の負荷について、slaveマシンのbinlog再生速度を向上させ、マスターマシンとスレーブマシンのdelay減少を可能にします。

利用説明

MySQL 5.6および5.7バージョンでは、パラメータslave_parallel_typeを新しく追加された値TABLEに設定することでこの機能を有効にすることができます。MySQL 8.0バージョンはTABLEモードをサポートしていません。
さらに、information_schemaでステータステーブルcdb_slave_thread_statusを追加し、ステータス情報の表示に用います。

- MySQL 5.6バージョンの関連パラメータの説明
- MySQL 5.7バージョンの関連パラメータの説明
- MySQL 8.0バージョンの関連パラメータの説明

パラメータ名	動的	タイプ	デフォルト	パラメータ値範囲	説明
slave_parallel_type	YES	char*	SCHEMA	SCHEMA/TABLE	スレーブマシンの並列レプリケーションレベル： • SCHEMAはオブジェクトレベルのレプリケーションであり、異なるオブジェクトのレプリケーションイベントを並行して実行できます。 • TABLEはテーブルレベルのレプリケーションであり、異なるテーブルのレプリケーションイベントを並行して実行できます。

動的スレッドプール

最終更新日：：2022-01-04 10:59:48

機能の説明

スレッドプール（Thread_pool）は一定数のワークスレッドを用いて接続リクエストを処理するもので、通常はOLTPワークロードのシーンに比較的適しています。しかし、 リクエストにスロークエリが多い状況でスレッドプールの不足が発生すると、高遅延操作においてワークスレッドがブロックされ、新しいリクエストに迅速に応答できなくなり、システムのスループットが既存のone-thread-per-connection（Per_thread）方式より逆に低下する場合があります。

Per_thread方式とThread_pool方式にはそれぞれに長所と短所があり、システムは業務のタイプに応じて柔軟に切り替えを行う必要があります。残念ながら、現時点ではこの2種類の方式を切り替えるには、サーバーの再起動が必須です。通常、2種類の方式の切り替えニーズは業務のピーク時間帯に生じるため、その時点でのサーバー強制再起動は業務に重大な影響を与えます。

Per_thread方式とThread_pool方式の切り替えの柔軟性を高めるため、TencentDB for MySQLはスレッドプールの動的切り替えの最適化、すなわちデータベースサービスを再起動せずに、スレッドプールを動的にオンまたはオフにすることを可能にしました。

サポートするバージョン

- カーネルバージョン MySQL 8.0 20201230およびそれ以降
- カーネルバージョン MySQL 5.7 20201230およびそれ以降

ユースケース

パフォーマンスに敏感で、業務のタイプに応じて柔軟にデータベースの実行方式を調整する必要がある業務に適します。

パフォーマンスへの影響

- pool-of-threadsをone-thread-per-connectionに切り替えるプロセス自体はqueryの堆積や、パフォーマンスへの影響をもたらしません。

- one-thread-per-connectionをpool-of-threadsに切り替えるプロセスは、その前にスレッドプールがスリープ状態にあるため、QPSが極めて高くかつストレスが持続的に高い状況下では、一定のリクエストの蓄積が存在する可能性があります。対処方法は次のとおりです。
  - 方法1：thread_pool_oversubscribeを適宜増大させるとともに、thread_pool_stall_limitを適宜小さくし、スレッドプールを迅速にアクティブ化します。堆積したSQLを消化後、状況に応じて上記の変更を元に戻します。
  - 方法2：SQLの蓄積が生じた際、業務トラフィックを一時的に数秒間停止または低下させ、pool-of-threadsが完全にアクティブ化されるのを待ってから、持続的な高ストレス業務トラフィックを再開させます。

利用説明

追加されたthread_handling_switch_modeはスレッドプール動的切り替え機能の制御に用います。オプション値およびその意味は次のとおりです。

オプション値	意味
disabled	方式の動的移動禁止
stable	新しい接続のみ移動
fast	新しい接続 + 新しいリクエストをすべて移動。デフォルトモード
sharp	現在アクティブな接続をkillし、ユーザーに強制的に再接続させることで、切り替えの効果を素早く発揮

`show threadpool status` に追加されたステータスは次のとおりです。

- connections_moved_from_per_threadは、Per_threadからThread_poolに移動したconnectionsの数を表します。
- connections_moved_to_per_threadは、Thread_poolからPer_threadに移動したconnectionsの数を表します。
- events_consumedは、各スレッドプールのワークスレッドで消費されたeventsの総数を表します。Thread_poolがPer_threadに移動した後は、eventsの総数はそれ以上増加しません。
- average_wait_usecs_in_queueは、各eventのqueueにおける平均待機時間を表します。

`show full processlist` に追加されたステータスは次のとおりです。

- Moved_to_per_threadは、その接続がPer_threadに移動した回数を表します。
- Moved_to_thread_poolは、その接続がThread_poolに移動した回数を表します.

関連パラメータステータス説明

スレッドプール関連パラメータの紹介：

パラメータ名	動的	タイプ	デフォルト	パラメータ値範囲	説
thread_pool_idle_timeout	Yes	uint	60	[1, UINT_MAX]	wo ネ 場 の
thread_pool_oversubscribe	Yes	uint	3	[1,1000]	1に 大
thread_pool_size	Yes	uint	現在のマシンのCPU数	[1,1000]	ス
thread_pool_stall_limit	Yes	uint	500	[10, UINT_MAX]	こ と ス ラ ス な ー IO い が tim プ wo の ス
thread_pool_max_threads	Yes	uint	100000	[1,100000]	ス ス

パラメータ名	動的	タイプ	デフォルト	パラメータ値範囲	説
thread_pool_high_prio_mode	Yes, session	enum	transactions	transactions\statement\none	高あす。• ン1にthrが先まthrプれま• がま• すー
thread_pool_high_prio_tickets	Yes, session	uint	UINT_MAX	[0, UINT_MAX]	traてtic
threadpool_workaround_epoll_bug	Yes	bool	false	true/false	linしlin

`show threadpool status` コマンドによって表示される関連ステータスの紹介：

ステータス名	説明
groupid	スレッドグループID
connection_count	スレッドグループのユーザー接続数
thread_count	スレッドグループ内のワークスレッド数
havelistener	スレッドグループに現在listenerが存在するか
active_thread_count	スレッドグループ内のアクティブworker数

ステータス名	説明
waiting_thread_count	スレッドグループ内の待機中のworker数（wait_beginをコールしたworker）
waiting_threads_size	スレッドグループ内の、処理を必要とするネットワークイベントがなく、スリープ状態でウェイクアップ待機中のworker数（thread_pool_idle_timeout秒待機後に自動的に破棄）
queue_size	スレッドグループの一般優先度キューの長さ
high_prio_queue_size	スレッドグループの高優先度キューの長さ
get_high_prio_queue_num	スレッドグループ内でイベントを高優先度キューから取り出した総回数
get_normal_queue_num	スレッドグループ内でイベントを一般優先度キューから取り出した総回数
create_thread_num	スレッドグループ内のworkerスレッド作成総数
wake_thread_num	スレッドグループ内でwaiting_threadsキューからウェイクアップしたworkerの総数
oversubscribed_num	スレッドグループ内のworkerが、スレッドグループが現在oversubscribed状態にあることを発見し、スリープに入る準備をした回数
mysql_cond_timedwait_num	スレッドグループ内のworkerがwaiting_threadsキューに入った総回数
check_stall_nolistener	スレッドグループでtimerスレッドによるcheck_stallチェック中に発見されたlistenerなしの総回数
check_stall_stall	スレッドグループでtimerスレッドによるcheck_stallチェック中にstall状態と判定された総回数
max_req_latency_us	スレッドグループ内のユーザー接続がキュー内で待機した最長時間（単位はミリ秒）
conns_timeout_killed	スレッドグループ内のユーザー接続が、クライアントから新しいメッセージがない時間が閾値（net_wait_timeout）を超えたためにkillされた総回数
connections_moved_in	他のスレッドグループからこのスレッドグループに移動した接続の総数

ステータス名	説明
connections_moved_out	このスレッドグループから他のスレッドグループに移動した接続の総数
connections_moved_from_per_thread	one-thread-per-connection方式からこのスレッドグループに移動した接続の総数
connections_moved_to_per_thread	このスレッドグループからone-thread-per-connection方式に移動した接続の総数
events_consumed	スレッドグループが処理したeventsの総数
average_wait_usecs_in_queue	スレッドグループ内の全eventsのキュー内平均待機時間

NOWAIT構文のサポート

最終更新日：：2022-01-04 10:59:48

機能の説明

- DDLはNO_WAITおよびWAITオプションをサポートします。DDL操作に対し、WAITを設定することで、MDL LOCKの秒数待機し、設定した時間内にMDL LOCKを取得できなければ直接返すことができます。あるいは、NO_WAITオプションを指定し、MDL LOCKを取得できなければ直接返すこともできます。
- SELECT FOR UPDATEはNOWAITおよびSKIP LOCKEDオプションをサポートします。従来のSELECT FOR UPDATEロジックでは、目的の行データが別のトランザクションによってロックされた場合、そのトランザクションがロックをリリースするまで待つ必要がありました。しかし、seckillのようなケースで、ロックを待機したくない場合は、SKIP LOCKEDおよびNOWAITオプションによって、ロック待機不要の機能を利用できるようになりました。SKIP LOCKEDステートメントはロックされている行をスキップでき、これらの行は結果セットに表示されません。NOWAITステートメントではロックされた行に遭遇しても待機せず、エラーを表示します。

この2種類のNO WAITを使用する際のキーワードは異なることに注意する必要があります。

サポートするバージョン

- カーネルバージョン MySQL 5.7 20171130およびそれ以降では、DDLステートメントがNO_WAITおよびWAITオプションをサポートします。
- カーネルバージョン MySQL 5.7 20200630およびそれ以降（この機能はMySQL 8.0から移植されています。バージョン8.0ではネイティブサポートのため）では、SELECT FOR UPDATEステートメントがNOWAITおよびSKIP LOCKEDオプションをサポートします。

ユースケース

DevAPI/XPluginは現時点ではSELECT FOR UPDATE/SHAREステートメントのSKIP LOCKEDおよびNOWAITオプション使用をサポートしていません。過去の理由により、DDLのNO_WAITキーワードとSELECT FOR UPDATEのNOWAITキーワードは異なるものになっているため、注意して区別する必要があります。

利用説明

SELECT FOR UPDATE NOWAIT/SKIP LOCKED

```
###############session 1###############
MySQL [test]> create table t1(seat_id int, state int, primary key(seat_id)) engin
e=innodb;
Query OK, 0 rows affected (0.03 sec)

MySQL [test]> INSERT INTO t1 VALUES(1,0), (2,0), (3,0), (4,0);
Query OK, 4 rows affected (0.01 sec)
Records: 4 Duplicates: 0 Warnings: 0

MySQL [test]> begin;
Query OK, 0 rows affected (0.01 sec)

MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 FOR SHARE;
+---------+-------+
| seat_id | state |
+---------+-------+
| 1 | 0 |
| 2 | 0 |
+---------+-------+
2 rows in set (0.00 sec)

###############session 2###############
MySQL [test]> SET SESSION innodb_lock_wait_timeout=1;
Query OK, 0 rows affected (0.00 sec)
MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 FOR UPDATE;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 FOR UPDATE NOWAIT;
ERROR 5010 (HY000): Do not wait for lock.
MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 FOR UPDATE SKIP LOCKED;
+---------+-------+
| seat_id | state |
+---------+-------+
| 3 | 0 |
| 4 | 0 |
+---------+-------+
2 rows in set (0.00 sec)

MySQL [test]> SELECT * FROM t1 WHERE seat_id > 0 LIMIT 2 FOR UPDATE NOWAIT;
ERROR 5010 (HY000): Do not wait for lock.
MySQL [test]> SELECT * FROM t1 WHERE seat_id > 0 LIMIT 2 FOR UPDATE SKIP LOCKED;
+---------+-------+
| seat_id | state |
+---------+-------+
| 3 | 0 |
| 4 | 0 |
```

```
+---------+-------+
2 rows in set (0.00 sec)

MySQL [test]> commit;
Query OK, 0 rows affected (0.00 sec)
```

SELECT FOR SHARE NOWAIT/SKIP LOCKED

```
###############session 1###############
MySQL [test]> begin;
Query OK, 0 rows affected (0.01 sec)

MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 FOR UPDATE;
+---------+-------+
| seat_id | state |
+---------+-------+
| 1 | 0 |
| 2 | 0 |
+---------+-------+
2 rows in set (0.00 sec)

###############session 2###############
MySQL [test]> SET SESSION innodb_lock_wait_timeout=1;
Query OK, 0 rows affected (0.00 sec)

MySQL [test]> begin;
Query OK, 0 rows affected (0.00 sec)

MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 LOCK IN SHARE MODE;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 FOR SHARE;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 FOR SHARE NOWAIT;
ERROR 5010 (HY000): Do not wait for lock.
MySQL [test]> SELECT * FROM t1 WHERE state = 0 LIMIT 2 FOR SHARE SKIP LOCKED;
+---------+-------+
| seat_id | state |
+---------+-------+
| 3 | 0 |
| 4 | 0 |
+---------+-------+
2 rows in set (0.00 sec)

MySQL [test]> commit;
Query OK, 0 rows affected (0.00 sec)
```

DDLステートメントのNO_WAITおよびWAITオプション

```
ALTER TABLE `table` [NO_WAIT | WAIT [n]] `command`;
DROP TABLE `table` [NO_WAIT | WAIT [n]];
TRUNCATE TABLE `table` [NO_WAIT | WAIT [n]];
OPTIMIZE TABLE `table` [NO_WAIT | WAIT [n]];
RENAME TABLE `table_src` [NO_WAIT | WAIT [n]] TO `table_dst`;
CREATE INDEX `index` ON `table.columns` [NO_WAIT | WAIT [n]];
CREATE FULLTEXT INDEX `index` ON `table.columns` [NO_WAIT | WAIT [n]];
CREATE SPATIAL INDEX `index` ON `table.columns` [NO_WAIT | WAIT [n]];
DROP INDEX `index` ON `table` [NO_WAIT | WAIT [n]];
```

returningをサポート

最終更新日：：2021-12-22 16:25:12

機能の説明

ある種のユースケースでは、DML操作後に、直前に操作したデータ行を返す必要があります。このニーズを実現するには一般的に2つの方法があります。

- 1つ目は、トランザクションの開始後、DMLステートメントのすぐ後にSELECTステートメントを続けることです。
- 2つ目は、トリガー等を使用する比較的複雑な操作です。

前者では主にSELECTステートメントの分のオーバーヘッドが増え、後者ではSQLの実装がより複雑になり、なおかつ柔軟性も不十分になります（トリガーの作成が必要）。
このため、RETURNING構文の設計では主にそのケースに合わせた最適化が必要です。DMLステートメントの後にRETURNINGキーワードを追加することで、上記のニーズを柔軟かつ高効率に実現できます。

サポートするバージョン

カーネルバージョン MySQL 5.7 20210330およびそれ以降

ユースケース

現時点で、MySQL 5.7 20210330およびそれ以降のカーネルバージョンでは、それぞれ、INSERT ... RETURNING、REPLACE ... RETURNING、DELETE ... RETURNINGをサポートしています。この構文では、INSERT/REPLACE/DELETEステートメントによって操作されたすべての行（statment単位）を返すことができます。また、RETURNINGはprepared statements、ストアドプロシージャでの使用もサポートしています。

この機能を使用する場合は、次のいくつかの点に注意してください。

1. RETURNINGを使用する際、DELETE...RETURNINGステートメントは前のイメージデータを返し、INSERT/REPLACE...RETURNINGステートメントは後のイメージデータを返します。
2. 現時点ではUPDATE...RETURNINGステートメントはサポートしていません。
3. INSERT/REPLACEのケースで、外部テーブルの列はreturning中のサブクエリステートメントに対し、現時点では可視性を有しません。

4. INSERT/REPLACEのRETURNINGステートメントがlast_insert_id()を返す必要がある場合、このlast_insert_id()の値は、そのステートメントが実行に成功する前の値です。正確なlast_insert_id()値が必要な場合は、RETURNINGを使用してそのテーブルの自動インクリメント列IDを直接返すことをお勧めします。

利用説明

INSERT... RETURNING

```
MySQL [test]> CREATE TABLE `t1` (id1 INT);
Query OK, 0 rows affected (0.04 sec)
MySQL [test]> CREATE TABLE `t2` (id2 INT);
Query OK, 0 rows affected (0.03 sec)
MySQL [test]> INSERT INTO t2 (id2) values (1);
Query OK, 1 row affected (0.00 sec)
MySQL [test]> INSERT INTO t1 (id1) values (1) returning *, id1 * 2, id1 + 1, id1
* id1 as alias, (select * from t2);
+------+---------+---------+-------+--------------------+
| id1 | id1 * 2 | id1 + 1 | alias | (select * from t2) |
+------+---------+---------+-------+--------------------+
| 1 | 2 | 2 | 1 | 1 |
+------+---------+---------+-------+--------------------+
1 row in set (0.01 sec)
MySQL [test]> INSERT INTO t1 (id1) SELECT id2 from t2 returning id1;
+------+
| id1 |
+------+
| 1 |
+------+
1 row in set (0.01 sec)
```

REPLACE ... RETURNING

```
MySQL [test]> CREATE TABLE t1(id1 INT PRIMARY KEY, val1 VARCHAR(1));
Query OK, 0 rows affected (0.04 sec)
MySQL [test]> CREATE TABLE t2(id2 INT PRIMARY KEY, val2 VARCHAR(1));
Query OK, 0 rows affected (0.03 sec)
MySQL [test]> INSERT INTO t2 VALUES (1,'a'),(2,'b'),(3,'c');
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0
MySQL [test]> REPLACE INTO t1 (id1, val1) VALUES (1, 'a');
Query OK, 1 row affected (0.00 sec)
MySQL [test]> REPLACE INTO t1 (id1, val1) VALUES (1, 'b') RETURNING *;
+-----+------+
```

```
| id1 | val1 |
+-----+------+
| 1 | b |
+-----+------+
1 row in set (0.01 sec)
```

DELETE ... RETURNING

```
MySQL [test]> CREATE TABLE t1 (a int, b varchar(32));
Query OK, 0 rows affected (0.04 sec)
MySQL [test]> INSERT INTO t1 VALUES
-> (7,'ggggggg'), (1,'a'), (3,'ccc'),
-> (4,'dddd'), (1,'A'), (2,'BB'), (4,'DDDD'),
-> (5,'EEEEE'), (7,'GGGGGGG'), (2,'bb');
Query OK, 10 rows affected (0.03 sec)
Records: 10 Duplicates: 0 Warnings: 0
MySQL [test]> DELETE FROM t1 WHERE a=2 RETURNING *;
+------+------+
| a | b |
+------+------+
| 2 | BB |
| 2 | bb |
+------+------+
2 rows in set (0.01 sec)
MySQL [test]> DELETE FROM t1 RETURNING *;
+------+---------+
| a | b |
+------+---------+
| 7 | ggggggg |
| 1 | a |
| 3 | ccc |
| 4 | dddd |
| 1 | A |
| 4 | DDDD |
| 5 | EEEEE |
| 7 | GGGGGGG |
+------+---------+
8 rows in set (0.01 sec)
```

ストアドプロシージャ

```
MySQL [test]> CREATE TABLE `t` (id INT);
Query OK, 0 rows affected (0.03 sec)
MySQL [test]> delimiter $$
MySQL [test]> CREATE PROCEDURE test(in param INT)
```

```
-> BEGIN
-> INSERT INTO t (id) values (param) returning *;
-> END$$
Query OK, 0 rows affected (0.00 sec)
MySQL [test]> delimiter ;
MySQL [test]> CALL test(100);
+------+
| id |
+------+
| 100 |
+------+
1 row in set (0.01 sec)
Query OK, 0 rows affected (0.01 sec)
```

列圧縮

最終更新日：：2022-08-08 15:31:17

機能の説明

現在、行形式の圧縮とデータページの圧縮がありますが、これら2つの圧縮方法は、テーブル内のいくつかの大きなフィールドと他の多くの小さなフィールドを処理すると同時に、小さなフィールドの読み取りと書き込みを頻繁に行います。大きなフィールドへのアクセス頻度が低い場合、その読み取り/書き込みアクセスの際に、コンピューティングリソースに不必要な浪費が多く発生します。

列圧縮機能では、アクセス頻度の低い大きなフィールドを圧縮する一方、アクセス頻度の高い小さなフィールドは圧縮しないことが可能です。これによって、フィールドの行全体のストレージ容量を削減できるだけでなく、読み取り/書き込みアクセスの効率も向上します。

例えば、1枚の従業員表 `create table employee(id int, age int, gender boolean, other varchar(1000) primary key (id))` があり、小さなフィールドである `id,age,gender` へのアクセス頻度が比較的高い一方、大きなフィールドである `other` へのアクセス頻度が比較的低い場合は、 `other` 列を圧縮列として作成することができます。一般的な状況下では、 `other` の読み取り/書き込みの場合のみ、その列の圧縮と解凍がトリガーされ、その他の列へのアクセスでは列の圧縮と解凍はトリガーされません。これにより、行データストレージのサイズをさらに小さくできるため、アクセス頻繁が高い小さなフィールドはより速くなり、ストレージ容量は比較的大きいがアクセス頻度が比較的低いフィールドについては、ストレージ容量をさらに削減することができます。

サポートするバージョン

カーネルバージョン MySQL 5.7 20210330およびそれ以降

> 説明：
> カラム圧縮機能のデフォルトはクローズ状態です。使用する場合はチケット提出が開きます。

ユースケース

テーブル内にいくつかの大きなフィールドと他の多くの小さなフィールドがあり、小さなフィールドの読み取りと書き込みは頻繁に行われ、大きなフィールドへのアクセス頻度が低い場合、大きなフィールドを圧縮列に設定

することができます。

利用説明

サポートするデータタイプ

1. `BLOB`（`TINYBLOB`、`MEDIUMBLOB`、`LONGBLOB`を含む）
2. `TEXT`（`TINYTEXT`、`MEDIUMTEXT`、`LONGTEXT`を含む）
3. `VARCHAR`
4. `VARBINARY`

> 注意：
> このうち`LONGBLOB`と`LONGTEXT`のサポートする長さは最大$2^{32}-2$であり、公式String Type Storage Requirementsのサポートする$2^{32}-1$より1バイト少なくなっています。

サポートするDDL構文タイプ

公式のテーブル作成構文と比較すると、その中の`column_definition`の`COLUMN_FORMAT`の定義が一部異なります。また、列圧縮はInnodbストレージエンジンタイプのテーブルのみサポートしています。

```
column_definition:
data_type [NOT NULL | NULL] [DEFAULT default_value]
[AUTO_INCREMENT] [UNIQUE [KEY]] [[PRIMARY] KEY]
[COMMENT 'string']
[COLLATE collation_name]
[COLUMN_FORMAT {FIXED|DYNAMIC|DEFAULT}|COMPRESSED=[zlib]] # COMPRESSED圧縮列キーワード
[STORAGE {DISK|MEMORY}]
[reference_definition]
```

簡単なサンプルを次に示します。

```
CREATE TABLE t1(
id INT PRIMARY KEY,
b BLOB COMPRESSED
);
```

ここでは圧縮アルゴリズムを省略し、デフォルトの`zlib`圧縮アルゴリズムを選択しています。圧縮アルゴリズムキーワードを表示させて指定することも可能ですが、現時点でサポートしているのは`zlib`圧縮アルゴリズムのみです。

```
CREATE TABLE t1(
id INT PRIMARY KEY,
b BLOB COMPRESSED=zlib
);
```

サポートするDDL構文を次にまとめます。

create tableに関するもの：

DDL	圧縮属性継承の有無
`CREATE TABLE t2 LIKE t1;`	Y
`CREATE TABLE t2 SELECT * FROM t1;`	Y
`CREATE TABLE t2(a BLOB) SELECT * FROM t1;`	N

alter tableに関するもの：

DDL	説明
`ALTER TABLE t1 MODIFY COLUMN a BLOB;`	圧縮列を非圧縮に変更
`ALTER TABLE t1 MODIFY COLUMN a BLOB COMPRESSED;`	非圧縮列を圧縮に変更

追加された変数の説明

パラメータ名	動的	タイプ	デフォルト	パラメータ値範囲	説明
innodb_column_compression_zlib_wrap	Yes	bool	TRUE	true/false	オンにッダーとし、adlます

パラメータ名	動的	タイプ	デフォルト	パラメータ値範囲	説明
innodb_column_compression_zlib_strategy	Yes	Integer	0	[0,4]	列圧縮 シー。最 4で、0 の圧縮 Z_DEFA Z_FILTI Z_HUFI Z_RLE、 応しま 一般的 には Z_DEFA 最適な ータに
innodb_column_compression_zlib_level	Yes	Integer	6	[0,9]	列圧縮 ル。最 で、0は します。 ど、圧 くなり 長くな
innodb_column_compression_threshold	Yes	Integer	256	[0, 0xffffffff]	列圧縮 い値。最 0xffffffff、 さがこ 場合の れ、そ ジナル 圧縮へ ます
innodb_column_compression_pct	Yes	Integer	100	[1, 100]	列圧縮 最小値 単位は **ータサ** **サイズ** 合のみ そうで ルデー ヘッダ

説明：
ユーザーは現在、上記パラメータのパラメータ値を直接変更することはできません。変更する場合は、チケットを提出から変更することができます。

追加されたステータスの説明

名称	タイプ	説明
`Innodb_column_compressed`	Integer	列圧縮の圧縮回数。非圧縮形式と圧縮形式の2種類のステータスの圧縮を含む
`Innodb_column_decompressed`	Integer	列圧縮の解凍回数。非圧縮形式と圧縮形式の2種類のステータスの解凍を含む

追加されたエラーの説明

名称	範囲	説明
`Compressed column '%-.192s' can't be used in key specification`	圧縮する列名を指定	インデックスのある列に対し圧縮属性を指定することはできません
`Unknown compression method: %s"`	DDLステートメントの中で指定する圧縮アルゴリズム名	`create table` または `alter table` の際に `zlib` 以外の不正な圧縮アルゴリズムを指定しています
`Compressed column '%-.192s' can't be used in column format specification`	圧縮する列名を指定	同一の列に、すでに `COLUMN_FORMAT` 属性が指定されている場合は圧縮属性を指定できません。この `COLUMN_FORMAT` はNDBでのみ使用できます
`Alter table ... discard/import tablespace not support column compression`	\	圧縮列のあるテーブルでは `Alter table ... discard/import tablespace` ステートメントは実行できません

パフォーマンス

全体のパフォーマンスはDDLとDMLに関するものに分けられます。
DDLに関しては、sysbenchを使用してテストしています。

- 列圧縮はCOPYアルゴリズムのDDLに対して比較的大きなパフォーマンス面の影響があり、圧縮後のパフォーマンスはそれ以前より7倍から8倍遅くなりました。
- inplaceへの影響については圧縮後のデータ量サイズに左右されます。圧縮した後、全体のデータサイズが低下すれば、DDLのパフォーマンスは上昇します。その逆であれば、パフォーマンスは一定程度低下します。
- instantについては、列圧縮がこのタイプのDDLに影響することは基本的にありません。

DMLに関しては、最も一般的な圧縮のケース（圧縮比1:1.8）を考えます。8列のテーブルがあり、テーブル内に1つの大きなvarcharタイプの列があり、その挿入データの長さは1～6000の間で均一かつランダムであり、挿入する文字は0 -9 、a - bの間でランダムです。その他のいくつかの列データのタイプはchar(60)またはintタイプです。このとき、非圧縮列の挿入、削除および照会に対しては10%以内の上昇がみられましたが、非圧縮列の更新には10%以内の低下がみられ、圧縮列の更新には15%以内のパフォーマンス低下がみられました。これは、更新のプロセスにおいて、MySQLは先にこの行の値を読み出した後にその行の更新後の値を書き込むため、更新プロセス全体が1回の解凍と圧縮をトリガーしますが、一方で挿入と照会は圧縮または解凍を1回しか行わないためです。

注意事項

1. 論理エクスポートに関しては、論理エクスポートの際にcreate tableにはCompressed関連のキーワードが付与されます。このため、インポートの際はTencentDB for MySQL内部でサポートされます。その他のMySQLブランチおよび公式バージョンは次のとおりです。
   - 公式のバージョン番号が5.7.18より小さい場合は、直接インポートできます。
   - 公式のバージョン番号が5.7.18かそれより大きい場合は、論理エクスポート後に圧縮キーワードを削除する必要があります。
2. DTSによって他のクラウドまたはユーザーにエクスポートする場合は、binlogの同期中に互換性の問題が発生する可能性があるため、圧縮キーワード付きのDDLステートメントをスキップすることができます。

フラッシュバッククエリー

最終更新日：：2023-02-22 16:13:27

機能の説明

データベースの運用保守中に誤動作が発生することがあります、このような誤動作は、業務に重大な影響を及ぼす可能性があります。誤動作によって業務に影響が及んだ場合、一般的なリカバリ手段としては、ロールバックやクローン作成などがありますが、データの変更が少ない場合や、緊急の障害修復では、エラーが発生しやすく時間がかかり、また、データ量が多い場合にはリカバリ時間を把握できません。
TXSQLチームは、Innodbエンジン上でフラッシュバッククエリー機能を設計・実装しました。簡単なSQL文だけで誤動作前の履歴データを照会することができます。特定のSQL構文で特定の時点のデータを照会することで、大量のデータ照会と回復時間を節約し、誤動作後のデータを迅速に復元し、業務の迅速な再開を保証します。

サポートするバージョン

- カーネルバージョン MySQL 5.7 20220715以降
- カーネルバージョン MySQL 8.0 20220331以降

カーネルマイナーバージョンのアップデートについて、カーネルマイナーバージョンのアップデートをご参照ください。

ユースケース

フラッシュバッククエリー機能はデータベースの運用保守プロセスの誤動作後、履歴データを迅速にクエリーするために使用されます。
この機能を使用する場合は、次のいくつかの点に注意してください：

- Innodb物理テーブルのみがサポートされます。viewやその他のエンジンはサポートされません。last_insert_id()など、実際の列が対応していない関数はサポートされません。
- 秒単位のフラッシュバッククエリーのみがサポートされます。100%の正確性は保証されていません。1秒以内に複数の変更があった場合、いずれかが問合せされる可能性があります。
- フラッシュバッククエリーでは、プライマリキーのみがサポートされます（またはGEN_CLUST_INDEX）。
- prepared statementおよびstored procedureでは使用できません。
- DDLはサポートされていません。表をDDLする（例えば、truncate table。この場合はごみ箱からリカバリすることをお勧めします）と、フラッシュバッククエリーの結果が予期しない可能性があります。

- 同じ文内の同じテーブルに複数のフラッシュバッククエリー時間が指定されている場合、現在のクエリー時間から最も遠い時間が選択されます。
- マスター/スレーブインスタンスには時間差があるため、フラッシュバッククエリーを同じ時間に指定すると、マスター/スレーブインスタンスで得られる結果が異なる場合があります。
- フラッシュバッククエリーを有効にすると、undoログのクリーンアップが遅延し、メモリー使用量が増加します。特に業務アクセスがビジーなインスタンスについて、Innodb_backquery_windowの設定が過度に大きくなることは推奨されません（900～1800の範囲をお勧めします）。
- データベースインスタンスが再起動またはcrashした場合、再起動またはcrash前の履歴情報を問い合せることはできません。指定した時間は、サポートされている範囲内でなければなりません（サポートされている範囲は、状態変数Innodb_backquery_up_timeとInnodb_backquery_low_timeを使用して確認でき、`show status like'%backquery%'` を実行します）。

利用説明

フラッシュバッククエリーでは新しいAS OF構文が提供されます。パラメータ設定でInnodb_backquery_enableパラメータをONに設定し、フラッシュバッククエリー機能をオンにして、特定の構文を使用して指定した時刻のデータを問合せします。構文は次のとおりです：

```
SELECT ... FROM <テーブル名>
AS OF TIMESTAMP <時間>;
```

指定時刻の問い合わせ例

```
MySQL [test]> create table t1(id int,c1 int) engine=innodb;
Query OK, 0 rows affected (0.06 sec)

MySQL [test]> insert into t1 values(1,1),(2,2),(3,3),(4,4);
Query OK, 4 rows affected (0.01 sec)
Records: 4 Duplicates: 0 Warnings: 0

MySQL [test]> select now();
+---------------------+
| now() |
+---------------------+
| 2022-02-17 16:01:01 |
+---------------------+
1 row in set (0.00 sec)

MySQL [test]> delete from t1 where id=4;
Query OK, 1 row affected (0.00 sec)
```

```
MySQL [test]> select * from t1;
+------+------+
| id | c1 |
+------+------+
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
+------+------+
3 rows in set (0.00 sec)

MySQL [test]> select * from t1 as of timestamp '2022-02-17 16:01:01';
+------+------+
| id | c1 |
+------+------+
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
+------+------+
4 rows in set (0.00 sec)
```

履歴データからのテーブル作成例

```
create table t3 select * from t1 as of timestamp '2022-02-17 16:01:01';
```

テーブルに履歴データを挿入する例

```
insert into t4 select * from t1 as of timestamp '2022-02-17 16:01:01';
```

パラメータの説明

フラッシュバッククエリー機能に設定可能なパラメータの説明を次の表に示します。

パラメータ名	パラメータ範囲	タイプ	デフォルト値	数値範囲
Innodb_backquery_enable	グロバールパラメータ	Boolean	OFF	ON\OFF

パラメータ名	パラメータ範囲	タイプ	デフォルト値	数値範囲
Innodb_backquery_window	グロバールパラメータ	Integer	900	1 - 86400
Innodb_backquery_history_limit	グロバールパラメータ	Integer	8000000	1 - 9223372036854476000

パフォーマンス系特性
大規模トランザクションレプリケーション

最終更新日：：2022-03-02 22:27:46

機能の説明

row方式では、1つのステートメントによって多くの行が更新される大規模トランザクションは、1行ごとに1個のeventを生成します。一方では大量のbinlogが生成され、また一方ではレプリケーション時にスレーブデータベースがapplyする場合は速度が比較的遅いため、スレーブデータベースのレプリケーション遅延が発生してしまいます。
Tencent Cloudカーネルチームは大規模トランザクションレプリケーションのケースを分析および最適化し、この機能を開発しました。大規模トランザクションレプリケーション最適化機能は、大規模トランザクションを自動的に認識し、row方式のbinlogをstatement形式のbinlogに変換することで、binlogを減少させるとともにレプリケーション効率を高めます。

サポートするバージョン

- カーネルバージョン MySQL 5.6 20210630およびそれ以降
- カーネルバージョン MySQL 5.7 20200630およびそれ以降
- カーネルバージョン MySQL 8.0 20200830およびそれ以降

ユースケース

- この機能は主にrow方式において、プライマリキーのないテーブルの大規模トランザクション再生速度を向上させるものであり、プライマリキーのないテーブルの再生が遅いことが遅延の原因であることが確実な場合にオンにできます。
- この機能は主に、row方式において大規模トランザクションが存在し、レプリケーション速度が遅いケースに対応します。

パフォーマンスデータ

レプリケーション時間はupdateシーンでは85%、insertシーンでは約30%、それぞれ減少します。

利用説明

大規模トランザクションレプリケーションの最適化機能は、SQLの過去に実行した統計状況を基に、それが大規模トランザクションである可能性の有無を判断します。大規模トランザクションと認識し、かつ最適化が可能と判断すると、分離レベルを自動的にRR（反復可能読み取り）レベルに引き上げ、binlogをStatement形式に変換することで、大規模トランザクションのスレーブデータベースでの実行時間を短縮します。このうち、

- cdb_optimize_large_trans_binlogはこの機能のスイッチです。
- cdb_sql_statisticsはSQL実行状況の統計を行うスイッチです。
- cdb_optimize_large_trans_binlog_last_affected_rows_thresholdと cdb_optimize_large_trans_binlog_aver_affected_rows_threshold は大規模トランザクションのしきい値条件を共同で構成します。
- cdb_sql_statistics_info_thresholdはメモリに保存されている過去の統計データの数です。

トランザクションの実行状況をより適切に監視するため、information_schemaデータベースのテーブル、CDB_SQL_STATISTICS を追加し、現在のトランザクションの統計情報照会に用います。

追加されたパラメータ

名称	ステータス	タイプ	デフォルト
cdb_optimize_large_trans_binlog	true	bool	false
cdb_optimize_large_trans_binlog_last_affected_rows_threshold	true	ulonglong	10000
cdb_optimize_large_trans_binlog_aver_affected_rows_threshold	true	ulonglong	10000
cdb_sql_statistics	true	bool	false
cdb_sql_statistics_info_threshold	true	ulonglong	10000

説明：

ユーザーは現在、上記パラメータのパラメータ値を直接変更することはできません。変更する場合は、チケットを提出から変更することができます。

追加されたinformation_schema.CDB_SQL_STATISTICSテーブル

名称	タイプ	説明
DIGEST_MD5	MYSQL_TYPE_STRING	このSQLのdigestから換算したMD5
DIGEST_TEXT	MYSQL_TYPE_STRING	SQLのdigestテキスト形式
SQL_COMMAND	MYSQL_TYPE_STRING	SQLコマンドのタイプ
FIRST_UPDATE_TIMESTAMP	MYSQL_TYPE_DATETIME	この統計情報の初回生成時間
LAST_UPDATE_TIMESTAMP	MYSQL_TYPE_DATETIME	この統計情報の前回更新時間
LAST_ACCESS_TIMESTAMP	MYSQL_TYPE_DATETIME	この統計情報への前回アクセス時間
EXECUTE_COUNT	MYSQL_TYPE_LONGLONG	このタイプのSQLが実行された回数
TOTAL_AFFECTED_ROWS	MYSQL_TYPE_LONGLONG	影響を受けた行の総数
AVER_AFFECTED_ROWS	MYSQL_TYPE_LONGLONG	影響を受けた行数の平均
LAST_AFFECTED_ROWS	MYSQL_TYPE_LONGLONG	前回影響を受けた行数
STMT_BINLOG_FORMAT_IF_POSSIBLE	MYSQL_TYPE_STRING	このタイプのSQLをstatement形式のbinlogに変換可能かどうか。TRUEまたはFALSE

プランキャッシュチェックの最適化

最終更新日：：2023-01-12 16:02:47

機能の説明

MySQLデータのSQL実行手順には、主に解析、準備、最適化、実行の4つの段階が含まれます。実行プランキャッシュ機能はprepare statement方式の場合にのみ作用します。prepare statement方式ではexecute時に解析と準備の2段階を省略し、実行プランによって最適化段階も省略できるため、パフォーマンスがさらに向上します。

MySQL 8.0 20210830バージョンでは(UK&PK)チェックに対してのみ作用しますが、今後のバージョンでは機能の範囲をさらに開放する予定です。

サポートするバージョン

カーネルバージョン MySQL 8.0 20210830およびそれ以降

ユースケース

クラウドオンライン上でショートクエリが比較的多く、かつprepare statement方式を使用している場合は、アプリケーションのパフォーマンスが向上します。具体的なパフォーマンスの上昇幅はオンライン業務によって異なります。

パフォーマンスへの影響

- (UK&PK)をチェックするSQLについては、遅延パフォーマンスが20%～30%、スループットパフォーマンスが20%～30%それぞれ向上しました（sysbench中のpoint_select.luaテスト）。
- メモリオーバーヘッドについては、プランキャッシュスイッチがオンの場合、オフの場合に比べてメモリ使用がやや上昇しました。

利用説明

cdb_plan_cacheスイッチを追加することでプランキャッシュをオンにするかどうかを制御し、cdb_plan_cache_statsスイッチを追加することでキャッシュのヒット状態の観察を制御します。これらのパラメータはtencentrootレベルです。

パラメータ名	ステータス	タイプ	デフォルト	パラメータ値範囲	説明
cdb_plan_cache	yes	bool	false	true/false	プランキャッシュをオン/オフにするスイッチ

> 説明：
> ユーザーは現在、上記パラメータのパラメータ値を直接変更することはできません。変更する場合は、チケットを提出から変更することができます。

`show cdb_plan_cache` コマンドを追加して、プランキャッシュのヒット状態を確認します。フィールドの意味は次のとおりです：

フィールド名	説明
sql	SQLステートメント。ここでは?の付いたSQLステートメントであり、このSQLの実行プランがすでにキャッシュされたことを表します
mode	SQLキャッシュの方式は、現時点ではprepare方式のみサポートしています
hit	本セッションのヒット回数

cdb_plan_cache_statsスイッチがオンの場合は、情報が記録されることになるため、パフォーマンスに影響する場合があります。

関連ステータス説明

show profileによってSQLの各実行段階のステータスを確認する際、SQLを実行してプランキャッシュがヒットした場合は、optimizing、statistics、preparingステータスが省略されます。

fdatasyncの使用をサポート

最終更新日：：2021-12-22 16:25:13

機能の説明

redoログファイルは現時点でfsyncシステムをコールして書き込みを行っており、これにはファイルメタデータの書き込みとファイルデータの書き込みが含まれます。ファイルメタデータの情報には、最終変更時間などのあまり重要ではない情報が含まれるため、一部のredo書き込みのケースでは、ファイルメタデータを常にストレージデバイスにフラッシュすることを避け（fdatasyncシステムのコールにより）、オーバーヘッドを減少させることが可能です。

サポートするバージョン

- カーネルバージョン MySQL 5.7 20201230およびそれ以降
- カーネルバージョン MySQL 8.0 20201230およびそれ以降

ユースケース

主に書き込みストレスが比較的大きいケースに適します。

パフォーマンスデータ

sysbench-write-onlyで多数の同時書き込みを持続的に行うケースにおいて、TPSのパフォーマンスが10%程度向上します。

利用説明

パラメータinnodb_flush_redo_using_fdatasync=true/falseを設定することで、fdatasyncを使用してredoログファイルメタデータのリアルタイム書き込みを避けるかどうかを制御します。デフォルトはfalseです。

パラメータ名	動的	タイプ	デフォルト	パラメータ値範囲	説明

パラメータ名	動的	タイプ	デフォルト	パラメータ値範囲	説明
innodb_flush_redo_using_fdatasync	yes	bool	false	true/false	fdatasync方式を使用してredoをフラッシュするか

永続的な自動インクリメント列をサポート

最終更新日：：2021-12-21 10:00:08

機能の説明

永続的な自動インクリメント列をデータページ内に実装することで、自動インクリメント値が繰り返される問題の発生を防ぎます。

サポートするバージョン

カーネルバージョン MySQL 5.7 20190830およびそれ以降

ユースケース

過去データのアーカイブなど、自動インクリメント値を繰り返させたくないケースに適します。

利用説明

カーネルはデフォルトでオンになっています。

buffer poolの初期化

最終更新日：：2021-12-22 16:25:13

機能の説明

buffer poolの初期化速度を速めて、データベースインスタンスの起動にかかる時間を短縮します。

サポートするバージョン

- カーネルバージョン MySQL 5.6 20200915およびそれ以降
- カーネルバージョン MySQL 5.7 20200630およびそれ以降

ユースケース

データベースの起動速度を向上させるために使用します。

パフォーマンステストデータ

8個のinstanceを使用した場合のパフォーマンステストデータ：

buffer_pool_size	buffer pool 初期化時間(最適化前)	buffer pool 初期化時間(最適化後)	速度上昇
50GB	2.55s	0.13s	1962%
200GB	10.28s	0.52s	1977%
500GB	25.72s	1.32s	1948%

利用説明

カーネルはデフォルトで実装されています。

FAST DDL

最終更新日：：2022-03-02 21:53:32

機能の説明

この機能はセカンダリインデックス作成プロセスの消費時間を最適化するものです。この機能をオンにすると、マルチスレッド並列処理を使用して、セカンダリインデックスデータに対し外部ソーティングを行うと同時に、flush bulk loading段階のflush listに対するロック操作を最適化し、CREATE INDEXの消費時間とDMLの並列処理への影響を効果的に低下させます。

サポートするバージョン

- カーネルバージョン MySQL 8.0 20210330およびそれ以降
- カーネルバージョン MySQL 5.7 20210331およびそれ以降

ユースケース

データベースは頻繁にDDL操作を行うため、DDLに関連するトラブルも頻繁に起こります。例えば次のようなものです。

- インデックスを追加するとインスタンスにジッターが発生し、正常な業務の読み取り/書き込みに影響が出るのはなぜですか。
- 1GBに満たないテーブルのDDL実行に10分以上かかることがあるのはなぜですか。
- 一時テーブルを使用した接続の終了時に、インスタンスにジッターが発生するのはなぜですか。

上記のようなよくあるご質問に対応するため、TXSQLカーネルチームは様々なケースを綿密に分析およびテストし、flush bulk loading段階のflush listに対するロック操作を最適化し、CREATE INDEXの消費時間とDMLの並列処理への影響を効果的に低下させることで、DDL操作による影響を低減させました。

パフォーマンスデータ

sysbenchテストでは20億行のデータをインポートしました。データ量は約453GBで、FAST DDL機能をオンにしました。

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

オンにする前の消費時間は4395秒、オンにした後の消費時間は2455秒でした。

利用説明

パラメータinnodb_fast_ddlによってこの機能をオンまたはオフにします。
パラメータinnodb_parallel_merge_threadsによって、並列外部ソーティングプロセスで使用する並列スレッド数を制御します。デフォルトは8、最大で32です。

パラメータ名	動的	タイプ	デフォルト	パラメータ値範囲	説明
innodb_fast_ddl	Yes	bool	OFF	{ON,OFF}	FAST DDLオンまたはオフ
innodb_parallel_merge_threads	Yes	Integer	8	1 - 32	merge sort時に使用する並列スレッド数

説明：
ユーザーは現在、上記のパラメータのパラメータ値を直接変更することはできません。変更が必要な場合は、ワークシートを提出して変更できます。

invisible index

最終更新日：：2021-12-22 16:25:14

機能の説明

インデックスを見えなくして、それを削除してよいかどうかを確認する機能は、多くのユーザーが必要としています。このinvisible index方式では、ユーザーはそのインデックスを削除する最終決定を行う前に、何らかのアプリケーションやデータベースユーザーが実際にそれを使用していないか（何らかのエラーが生成/報告されないか）を確認することができます。この機能はバージョン8.0から5.7へ移植されています。

サポートするバージョン

カーネルバージョン MySQL 5.7 20180918およびそれ以降

ユースケース

インデックスの削除前に、そのインデックスをinvisibleに設定することで有用性を確認でき、インデックスを安全に削除することができます。

利用説明

次のステートメントを使用すると、invisibleインデックスの作成や、あるインデックスのinvisibleインデックスへの変更ができます。

```
CREATE TABLE t1 (
i INT,
j INT,
k INT,
INDEX i_idx (i) INVISIBLE
) ENGINE = InnoDB;
CREATE INDEX j_idx ON t1 (j) INVISIBLE;
ALTER TABLE t1 ADD INDEX k_idx (k) INVISIBLE;
```

次のステートメントを使用すると、visibleインデックスに変更することができます。

```
ALTER TABLE t1 ALTER INDEX i_idx INVISIBLE;
ALTER TABLE t1 ALTER INDEX i_idx VISIBLE
```

CATSトランザクションスケジューリング

最終更新日：：2022-03-07 15:52:44

機能の説明

TXSQLに新たな並列トランザクションスケジューリングアルゴリズム（Contention-Aware Transaction Scheduling，CATS）を追加し、直接ロック競合を自動的に感知して、トランザクションの優先順位に基づいてトランザクションをスケジューリングし実行できるようになりました。
MySQLの従来の並列トランザクションはFIFO（First-In-First-Out）ルールによってトランザクションの実行順を決定します。一方CATSトランザクションスケジューリングアルゴリズムの主な原理は、トランザクションのロック保持状況に基づいて並列トランザクションの競合状況を判断するとともに、トランザクション実行の優先順位を決定し、その後優先順位に従ってトランザクションの実行順を割り振ることで、システムのトランザクション処理のスループットを向上させるものです。

サポートするバージョン

- カーネルバージョン MySQL 5.7 20190230およびそれ以降
- カーネルバージョン MySQL 8.0 20200630およびそれ以降

ユースケース

主に並列性が高く、かつロック競合が比較的著しいケースに適します。

パフォーマンスデータ

並列性が高く、ロック競合が著しいケースで、TPSパフォーマンスが50%以上向上しました。

- テスト方法：sysbench-oltp_read_writeシーン（分離レベルRR、テーブル数8、10MBデータ、paretoランダム化方式）
- テスト環境：32コア128GBオンラインインスタンス

スレッド数	FCFS(FIFO)	CATS	パフォーマンス向上
128	11999	12005	0%

スレッド数	FCFS(FIFO)	CATS	パフォーマンス向上
256	6609	10137	53%
512	3453	9365	171%
1024	2196	7015	219%

利用説明

MySQLバージョン5.7では、グローバルパラメータinnodb_trx_schedule_algorithmによってトランザクションスケジューリングのアルゴリズムを指定することができます。このパラメータのデフォルト値はautoです。
このアルゴリズムには次の3種類があります。

- auto：自動。現在のシステム状況に基づいて自動的に調整します。ロック待機スレッド数が32を超えるとCATSスケジューリングアルゴリズムを使用し、それ以外の場合はFCFSアルゴリズムを使用します。
- fcfs：先着順サービスアルゴリズム。
- cats: 競合検知スケジューリングアルゴリズム。

パラメータ名	動的	タイプ	デフォルト	パラメータ値範囲	説明
innodb_trx_schedule_algorithm	yes	string	auto	[auto,fcfs,cats]	トランザクション待機スケジューリングアルゴリズム

説明：
ユーザーは現在、上記パラメータのパラメータ値を直接変更することはできません。変更する場合は、チケットを提出から変更することができます。

MySQL バージョン8.0ではautoアルゴリズムが常に用いられ、設定はできません。

コンピューティングプッシュダウン

最終更新日：：2022-03-02 22:30:48

機能の説明

この機能は単一テーブルクエリのLIMIT/OFFSETまたはSUM操作をInnoDBにプッシュダウンすることで、照会の遅延を効果的に低下させるものです。

- LIMIT/OFFSETをセカンダリインデックスにプッシュダウンする際、この機能によってテーブルからの「リターン」操作が回避され、スキャンコストが効果的に低下します。
- SUM操作をInnoDBにプッシュダウンする際、InnoDB層でコンピューティングを行って「最終」の結果を返すことで、Server層とInnoDBエンジン層が「1行ごとに」記録を繰り返すコストを節約します。

サポートするバージョン

- LIMIT/OFFSETの最適化に対応するカーネルバージョン MySQL 5.7 20180530
- SUM操作のプッシュダウンの最適化に対応するカーネルバージョン MySQL 5.7 20180918

ユースケース

- この機能は主に、単一テーブルクエリにLIMIT/OFFSETまたはSUMが存在するケース、例えば `Select *from tbl Limit 10”、“Select* from tbl Limit 10,2` 、 `Select sum(c1) from tbl` などのステートメントに対応します。
- 最適化できないケース：
  - 照会ステートメントにdistinct、group by、havingが存在する場合。
  - ネストの照会が存在する場合。
  - FULLTEXTインデックスを使用している場合。
  - order byが存在し、かつオプティマイザがindexを利用してorder byを実装できない場合。
  - マルチレンジリード（MRR）を使用している場合。
  - SQL_CALC_FOUND_ROWSが存在する場合。

パフォーマンスデータ

sysbenchに100万行のデータをインポート後：

- `select * from sbtest1 limit 1000000,1;` の実行時間は6.3秒から2.8秒に短縮されました。
- `select sum(k) from sbtest1;` の実行時間は5.4秒から1.5秒に短縮されました。

利用説明

SQLの実行中に、対応する機能制御パラメータのオンオフ状況に基づいて、クエリオプティマイザが自動的に照会実行計画を書き換えることで、コンピューティングプッシュダウンの最適化を行います。
パラメータは次のとおりです。

パラメータ名	動的	タイプ	デフォルト	パラメータ値範囲	説明
cdb_enable_offset_pushdown	Yes	bool	ON	{ON,OFF}	LIMIT/OFFSETプッシュダウンを制御。デフォルトではオン
cdb_enable_sumagg_pushdown	Yes	bool	OFF	{ON,OFF}	SUMプッシュダウンを制御。デフォルトではオフ

説明：
ユーザーは現在、上記パラメータのパラメータ値を直接変更することはできません。変更する場合は、チケットを提出から変更することができます。

セキュリティ系特性
透過的データ暗号化

最終更新日：：2021-12-21 10:00:09

機能の説明

TXSQLでは、MySQLの透過的暗号化システムを踏襲し、KEYRINGプラグインのもう一つの実装であるKEYRING_KMSを提供しています。これはKEYRINGにTencent Cloudのエンタープライズレベルの Key Management Service(KMS)を統合したものです。

KMSはTencent Cloudのデータおよびキーのセキュリティを保護するキーサービスであり、サービスにかかわる各フローではいずれも安全性の高いプロトコル通信を用いることで、サービスの高度なセキュリティを保証します。また分散型クラスター管理およびホットバックアップを提供し、サービスの高い信頼性と高可用性を保証します。

KMSには二重キーシステムを採用し、カスタマーマスターキー（CMK）とデータキー（Datakey）という2種類のキーがこれに関与します。カスタマーマスターキーはデータキーの暗号化またはパスワード、証明書、設定ファイル等のスモールパケットデータ（最大4KB）に用いられます。データキーは業務データの暗号化に用いられます。大量の業務データはストレージまたは通信の過程で、データキーを使用して対称暗号化方式で暗号化され、データキーはさらにカスタマーマスターキーによって非対称暗号化方式で暗号化されて保護されます。二重キーシステムにより、データをメモリとファイル内の両方で確実に暗号化することができます。

サポートするバージョン

- カーネルバージョン MySQL 5.7 20171130およびそれ以降
- カーネルバージョン MySQL 8.0 20200630およびそれ以降

ユースケース

透過的データ暗号化とは、データの暗号化/復号操作をユーザーに対して透明にすることを指し、データファイルに対するリアルタイムな I/Oの暗号化と復号をサポートしています。データをディスクに書き込む前に暗号化して、ディスクから内部記憶装置に読み込む時に復号しますので、静的データの暗号化におけるコンプライアンス要件を満たすことができます。

利用説明

透過的データ暗号化機能を有効にするには、透過的データ暗号化をご参照ください。

監査

最終更新日：：2023-05-19 11:39:15

機能の説明

Tencent CloudはTencentDB for MySQLインスタンスにデータベース監査機能を提供します。データベースのアクセス、SQLステートメントの実行状況（ステートメントの起動時間、スキャン行数、ロック待機時間、CPU使用時間、クライアントIP、ユーザー名、SQLコンテンツ等を含む）を記録し、企業のリスク制御実施を助け、データセキュリティレベルを引き上げます。

ユースケース

この機能は、データベースが遭遇するリスクを警告し、SQLインジェクションや異常な操作などのデータベースのリスクとなる行為を対象に記録して、アラートを出したい場合に適しています。

パフォーマンスへの影響

監査には同期監査と非同期監査の2種類の方式があります。同期監査ではすべての監査ログを同期的に記録し、パフォーマンスへの影響は平均6%未満です。非同期監査では、インスタンスのパフォーマンスはほとんど影響を受けないことが保証されます。パフォーマンスロスは最大3%未満にとどまり、業界トップレベルです。

利用説明

MySQLの監査機能を有効にするには、監査の操作 をご参照ください。

安定性系特性
秒レベル高速列作成

最終更新日：：2022-10-28 10:15:27

機能の説明

高速列作成機能とは、データディクショナリだけを変更する方法によって、大きなテーブルでの高速列作成を実現する機能です。これまで列作成操作で必須だったデータコピーが不要になるため、大きなテーブルの列作成にかかる時間が大幅に短縮され、システムへの影響を軽減できます。

サポートするバージョン

- カーネルバージョン MySQL 5.7 20190830およびそれ以降
- カーネルバージョン MySQL 8.0 20200630およびそれ以降

ユースケース

データ量が大きいテーブルに列追加操作を行う必要があるケースに適しています。

パフォーマンスデータ

データ量が5GBのテーブルでテストを実施し、1列追加の操作が40秒から1秒以内に短縮されました。

利用説明

- Instant Add Column構文
  Alter Tableにalgorithm=instantサブステートメントが追加され、列作成操作を次のステートメントによって行うことができます：

```
ALTER TABLE t1 ADD COLUMN c INT, ADD COLUMN d INT DEFAULT 1000, ALGORITHM=INSTA
NT;
```

- パラメータinnodb_alter_table_default_algorithmが追加され、inplace、instantに設定することが可能です。このパラメータはデフォルトではinplaceであり、このパラメータを設定することでAlter Tableのデフォルトのアルゴリズムを調整できます。例えば次のようになります：

```
SET @@global.innodb_alter_table_default_algorithm=instant;
```

このパラメータによってデフォルトのアルゴリズムを指定した後、アルゴリズムを特に指定しない場合は、デフォルトのアルゴリズムを使用してAlter Tableの操作を行います。

Instant Add Column制限

- 1つのステートメントには列の追加操作のみがあり、同じステートメント内の他の操作はサポートされていません。
- 追加された列は最後に配置され、列の順序の変更はサポートされていません。
- 行の形式がCOMPRESSEDのテーブルでは、列の高速追加はサポートされていません。
- 全文索引が既にあるテーブルへの列の高速追加はサポートされていません。
- 一時テーブルへの列の高速追加はサポートされていません。

大きなテーブルの非同期削除

最終更新日：：2023-02-22 16:19:17

機能の説明

この機能は主にデータファイルの大きなテーブルを削除し、IOのジッターを防止するために用いられます。

DROP TABLEは、元のデータベースファイル(.ibd)を新しい臨時ファイルに再命名して、成功が返されます。臨時ファイルはinnodb_async_drop_tmp_dirにより指定されるディレクトリに保存され、バックグラウンドでバッチでtruncateします。truncateされるファイルサイズはinnodb_async_truncate_sizeによって制御されます。テーブルの非同期削除機能の有効化はinnodb_async_truncate_work_enabledパラメータによって制御されます。

この機能はユーザーにより操作する必要はなく、カーネルにより自動的に実行されます。テーブル削除の際に、他のディレクトリでテーブルのデータファイルにハード接続を作成することで実現できます。drop table実行後、このファイルのハード接続のみが削除されます。その後、バックグラウンドスレッドのスキャンによってハード接続のディレクトリの中に削除すべきファイルがあると分かり次第、先ほどdropされたテーブルデータファイルをバックグラウンドで自動的にtruncateします。

サポートするバージョン

- カーネルバージョン MySQL 5.6 20220303以降
- カーネルバージョン MySQL 5.7 20190203以降
- カーネルバージョン MySQL 8.0 20200630以降

ユースケース

この機能は削除しようとするテーブルデータファイルが非常に大きいケースに適しています。

利用説明

- MySQL 5.6、5.7バージョンでは、innodb_async_truncate_work_enabledをONに設定すると、DROP TABLEが非同期方式に変更されます。デフォルトではOFFになっています。
- MySQL 8.0バージョンでは、innodb_table_drop_modeをASYNC_DROPに設定すると、DROP TABLEが非同期方式に変更されます。デフォルトではSYNC_DROPになっています。

- truncateするたびのファイルの大きさはinnodb_async_truncate_sizeによって制御されます（MySQL 5.6バージョンは現在サポートされません）。
- innodb_fast_ddlパラメータをオンにすると、大きなテーブルの非同期削除能力がより効果的になります。

- MySQL 5.6関連パラメータの説明
- MySQL 5.7関連パラメータの説明
- MySQL 8.0関連パラメータの説明

パラメータ名	動的	タイプ	デフォルト	パラメータ値範囲	説明
innodb_async_truncate_work_enabled	Yes	string	OFF	ON/OFF	大きなテーブルの非同期削除をオンにするかどうか。

説明：
ユーザーは現在、上記パラメータのパラメータ値を直接変更することはできません。変更する場合は、チケットを提出から変更することができます。

ホットスポットの更新

最終更新日：：2021-12-21 10:00:09

機能の説明

頻繁な更新やseckill系の業務ケースに対応し、ホットスポットの行データに対するupdate操作のパフォーマンスを大幅に最適化します。ホットスポット更新の自動検出をオンにすると、システムが単行のホットスポット更新の有無を自動的に検出し、もしあれば大量の並列updateキューを実行させることで、大量の行ロックによる並列パフォーマンスの低下を減少させます。

サポートするバージョン

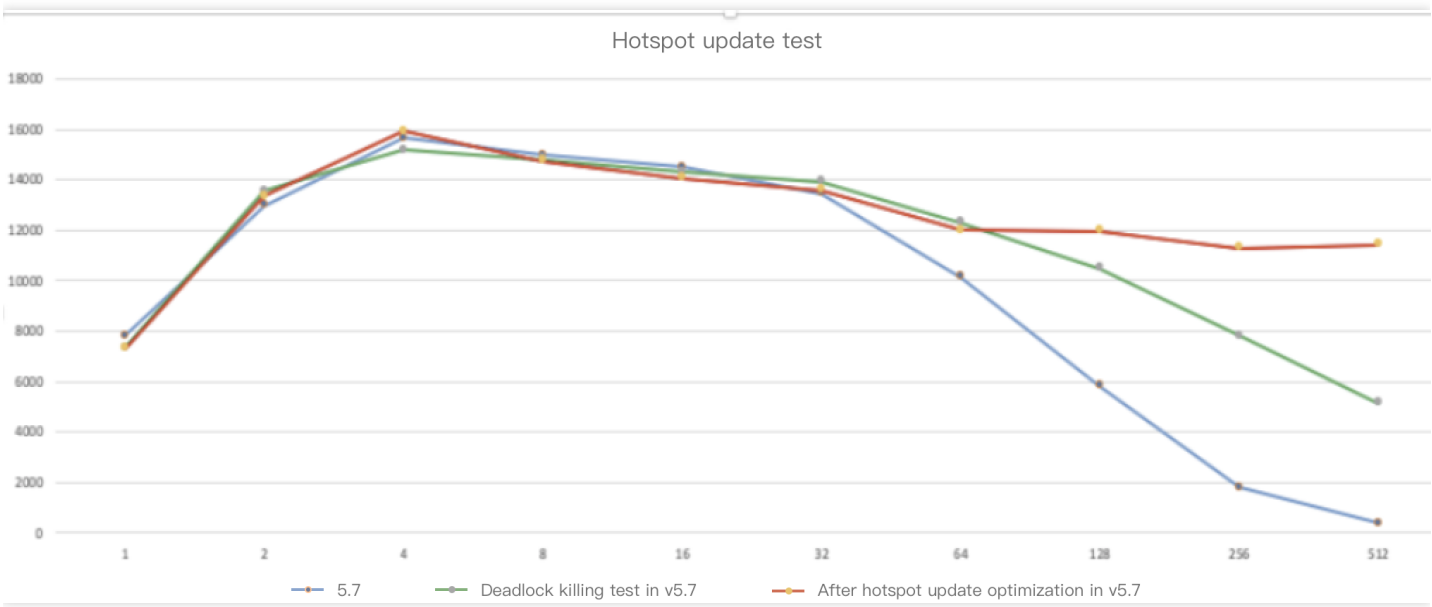
- カーネルバージョン MySQL 5.7 20200630およびそれ以降
- カーネルバージョン MySQL 8.0 20200830およびそれ以降

ユースケース

シングルポイントまたはマルチポイントのプライマリキーがあり、更新ストレスが非常に大きいケース（seckillのケース）に適しています。

パフォーマンスデータ

同時実行数が多い場合、プライマリキーがある条件下でのシングルポイント並列updateでは10倍以上のパフォーマンス向上を実現します。



利用説明

ホットスポット更新の保護

SQLトラフィック制限

最終更新日：：2021-12-21 10:00:09

機能の説明

キーワードを設定することで、特定のSQLの同一時間内に同時実行可能な並列度を制限します。

サポートするバージョン

- カーネルバージョン MySQL 5.7 20200330およびそれ以降
- カーネルバージョン MySQL 5.6 20200915およびそれ以降

ユースケース

同時実行数が多く、大量のリソースを占有し、システムのパフォーマンスを低下させるSQLに対応します。

ユースケース

SQLトラフィック制限

statement outline

最終更新日：：2022-03-07 10:39:41

機能の説明

SQLチューニングはデータベースのパフォーマンス最適化において非常に重要なプロセスの一つです。オプティマイザが適切な実行計画を選択できないことによる影響を防ぐため、TXSQLはOUTLINE機能を提供し、ユーザーが実行計画をバインドできるようにしました。MySQLデータベースにはHINTによって実行計画を人為的にバインドできる機能があります。HINT情報には、SQLがどの最適化ルールを採用しているか、どのアルゴリズムを実行しているか、データスキャンにどのインデックスを採用しているかなどが含まれます。OUTLINEは主にHINTによって照会計画を指定するもので、弊社はシステムテーブルmysql.outlineを提供し、ユーザーが計画バインドルールを追加できるようにしています。この機能をオンにするかどうかはスイッチ（cdb_opt_outline_enabled）で制御します。

サポートするバージョン

カーネルバージョン MySQL 8.0 20201230およびそれ以降

ユースケース

オンライン実行計画のインデックス選択ミスなど、オンラインでの実行計画に誤りがあったが、業務上、SQLを変更して新バージョンをリリースする解決方法を取りたくないケース。

パフォーマンスへの影響

- cdb_opt_outline_enabledスイッチがオンになっている状態で、outlineにヒットしないSQLの実行効率は影響を受けません。
- outlineにヒットしたSQLの実行効率は正常な実行よりは低下しますが、一般的にoutlineバインドによる上昇は、それまでの計画パフォーマンスに比べて数倍の上昇となります。
- このスイッチを使用する場合は、問い合わせと運用保守またはカーネルの担当者を置き、発生する可能性のあるバインドエラーによるパフォーマンスの後退を防止する必要があります。

利用説明

OUTLINE構文の設定に用いる新しい構文形式：

- OUTLINE情報を設定する：`outline "sql" set outline_info "outline";`
- OUTLINE情報を空にする：`outline reset ""; outline reset all;`
- OUTLINE情報をフラッシュする：`outline flush;`

次にOUTLINEの主な使用方法を紹介します。以下のschemaを例に説明します。

```
create table t1(a int, b int, c int, primary key(a));
create table t2(a int, b int, c int, unique key idx2(a));
create table t3(a int, b int, c int, unique key idx3(a));
```

パラメータ名	動的	タイプ	デフォルト	パラメータ値範囲	説明
cdb_opt_outline_enabled	yes	bool	fasle	true/false	outline機能をオンにするかどうか

説明：
ユーザーは現在、上記パラメータのパラメータ値を直接変更することはできません。変更する場合は、チケットを提出から変更することができます。

OUTLINEのバインド

OUTLINEの直接バインド方式とは、1文のSQLを、SQLの意味は変えずに別の1文に置き換えるもので、いくつかのHINT情報を追加して、どう実行するかをオプティマイザに通知するだけのものです。
構文形式は `outline "sql" set outline_info "outline";` です。outline_infoの後の文字列は必ず"OUTLINE:"で開始し、"OUTLINE:"の後にHINTの後のSQLを追加することに注意してください。例えば、 `select *from t1, t2 where t1.a = t2.a` というSQLのt2テーブルにa列のインデックスを加えます。

```
outline "select* from t1, t2 where t1.a = t2.a" set outline_info "OUTLINE:select
* from t1, t2 use index(idx2) where t1.a = t2.a";
```

optimizer hintのバインド

機能をさらに柔軟にするため、TXSQLでは、SQL中にoptimizer hintを増分追加することが許容されています。同様の機能はoutlineの直接バインドによっても実現できます。
構文形式は `outline "sql" set outline_info "outline";` です。outline_infoの後の文字列は必

ず"OPT:"で開始し、"OPT:"の後を追加したいoptimizer hint情報とすることに注意してください。例えば、s `elect *from t1 where t1.a in (select b from t2)` というSQLにMATERIALIZATION/DUPSWEEDOUTのSEMIJOINを指定します。

```
outline "select* from t1 where t1.a in (select b from t2)" set outline_info "OPT:
2#qb_name(qb2)";
outline "select * from t1 where t1.a in (select b from t2)" set outline_info "OP
T:1#SEMIJOIN(@qb2 MATERIALIZATION, DUPSWEEDOUT)";
```

オリジナルのSQLステートメントにOPTIMIZERのHINTを追加する場合は、1回につき1個のHINTの追加のみサポートしています。構文については3つの点に注意する必要があります。

- OPTキーワードは必ず"のすぐ後に入力します。
- バインドしたい新しいステートメントの前には必ず':'を入力します。
- 2つのフィールドを追加したい場合は（query blockの番号#optimizer hintの文字列）、間を必ず#で区切ります（ `ie. "OPT:1#max_execution_time(1000)"` ）。

index hintのバインド

機能をさらに柔軟にするため、TXSQLでは、SQL中にindex hintを増分追加することが許容されています。同様の機能はoutlineの直接バインドによっても実現できます。
構文形式は `outline "sql" set outline_info "outline";` です。outline_infoの後の文字列は必ず"INDEX:"で開始し、"INDEX:"の後を追加したいindex hint情報とすることに注意してください。
次に例を挙げます。 `select *from t1 where t1.a in (select t1.a from t1 where t1.b in (select t1.a from t1 left join t2 on t1.a = t2.a))` というSQLのquery block 3上のデータベースtest下のt1テーブルにUSE INDEXのインデックスidx1を追加します。タイプはFOR JOINです。

```
outline "select* from t1 where t1.a in (select t1.a from t1 where t1.b in (select
t1.a from t1 left join t2 on t1.a = t2.a))" set outline_info "INDEX:3#test#t1#idx
1#1#0";
```

オリジナルのSQLステートメントにINDEXのHINTを追加する場合は、1回につき1個のHINTの追加のみサポートしています。構文については4つの点に注意してください。

- INDEXキーワードは必ず"のすぐ後に入力します。
- バインドしたい新しいステートメントの前には必ず':'を入力します。
- 5つのフィールド（query blockの番号 #db_name#table_name#index_name#index_type#clause）を追加する必要があります。
- その中のindex_typeにはさらに3つの値があり（0はINDEX_HINT_IGNORE、1はINDEX_HINT_USE、2はINDEX_HINT_FORCE）、clauseには3つの値があり（1はFOR JOIN、2はFOR ORDER BY、3はFOR GROUP

BY）、間は必ず#で区切ります（ `ie. "INDEX:2#test#t2#idx2#1#0"` は2番目のquery block中のtest.t2テーブル中に、タイプがUSE INDEX FOR JOINであるidx1インデックスをバインドすることを表す）。

あるSQLに対応するOUTLINE情報を削除

TXSQLでは、ユーザーがあるSQLステートメントのOUTLINEバインド情報を削除することが許容されています。構文は `outline reset "sql";` であり、 `select *from t1, t2 where t1.a = t2.a` のoutline情報を削除する場合は、 `outline reset "select* from t1, t2 where t1.a = t2.a";` となります。

すべてのOUTLINE情報を空にする

TXSQLでは、ユーザーがカーネル内のすべてのOUTLINEバインド情報を削除することが許容されています。構文は `outline reset all` 、実行ステートメントは `outline reset all;` です。

オンライン業務中には、時にいくつかの非常に特殊な問題が発生し、強制的にインデックスをバインドする必要が生じることがあります。その場合は直接OUTLINEを設定してバインドすることができます。
OUTLINEの設定後に起こる可能性のあるパフォーマンスの後退を分析し、許容可能なパフォーマンス後退の範囲でバインドを行う必要があります。 必要に応じ、カーネル担当者と相談してください。

関連パラメータステータス説明

TXSQLはユーザーのSQLのOUTLINEバインドを確認するための様々な方法を提供しています。まず、mysql.outlineテーブルを通じてユーザーのOUTLINE設定状況を確認できます。次に、show cdb_outline_info、select * from information_schema.cdb_outline_infoという2つのインターフェースによって、メモリ内のOUTLINE情報を確認できます。入力したSQLが変更可能かどうかは、メモリ内にOUTLINE情報があるかどうかによって決まるため、ユーザーはこの2つのインターフェースを使用してデバッグを行うことができます。

mysql.outlineシステムテーブルが追加され、ユーザーが設定したOUTLINE情報のレコードはこのテーブルに保存されます。このテーブルのフィールドは次のとおりです。

フィールド名	説明
Id	OUTLINE設定情報番号
Digest	オリジナルSQLステートメントのハッシュ値
Digest_text	オリジナルSQLステートメントの指紋情報テキスト
Outline_text	OUTLINEバインド後のSQLステートメントの指紋情報テキスト

show cdb_outline_infoまたはselect * from information_schema.cdb_outline_infoでもメモリ内のレコードを確認でき、SQLを実行するとその中のOUTLINEレコードバインド計画にヒットする場合があります。パラメータは次の

とおりです。

フィールド名	説明
origin	オリジナルSQLステートメントの指紋
outline	OUTLINEバインド後のSQLステートメントの指紋

TXRocksエンジン
TXRocks概要

最終更新日：：2022-09-16 14:18:18

TXRocks概要

RocksDBは非常に人気の高い高性能の永続的なKV (key-value)ストレージであり、TXRocksはTencent TXSQLチームがこれに基づいて開発したトランザクション型ストレージエンジンです。

TXRocksストレージエンジンを使用する理由

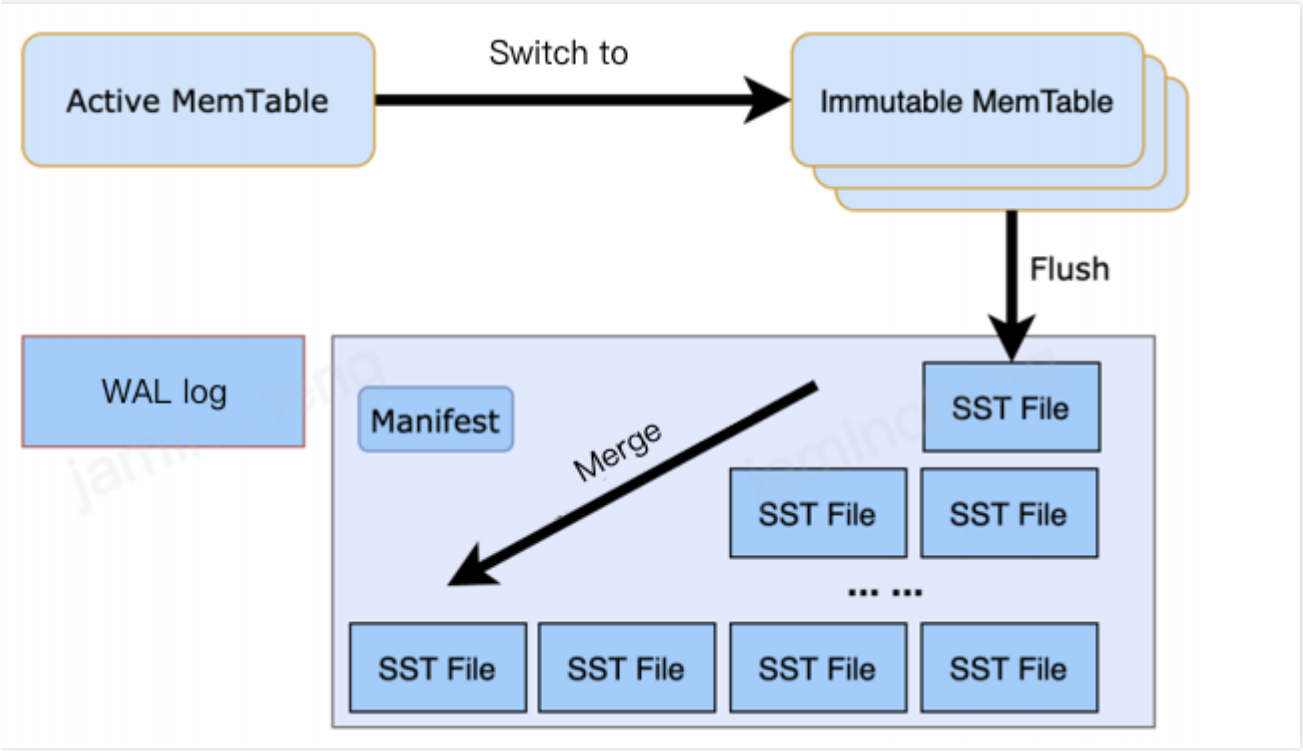
TXRocksトランザクション型ストレージエンジンはRocksDB LSM Treeストレージ構造のおかげで、InnoDBページの半分格納と断片化の無駄を減らすと同時に、コンパクトな形式のストレージを使うことができます。そのためTXRocksはInnoDBと近い性能を維持する前提の下で、ストレージ空間はInnoDBによりも半分以上節約することができ、さらにトランザクションの読み書きパフォーマンスが要求され、大量のデータを保存するビジネスに最適です。

RocksDBのLSM Treeアーキテクチャ

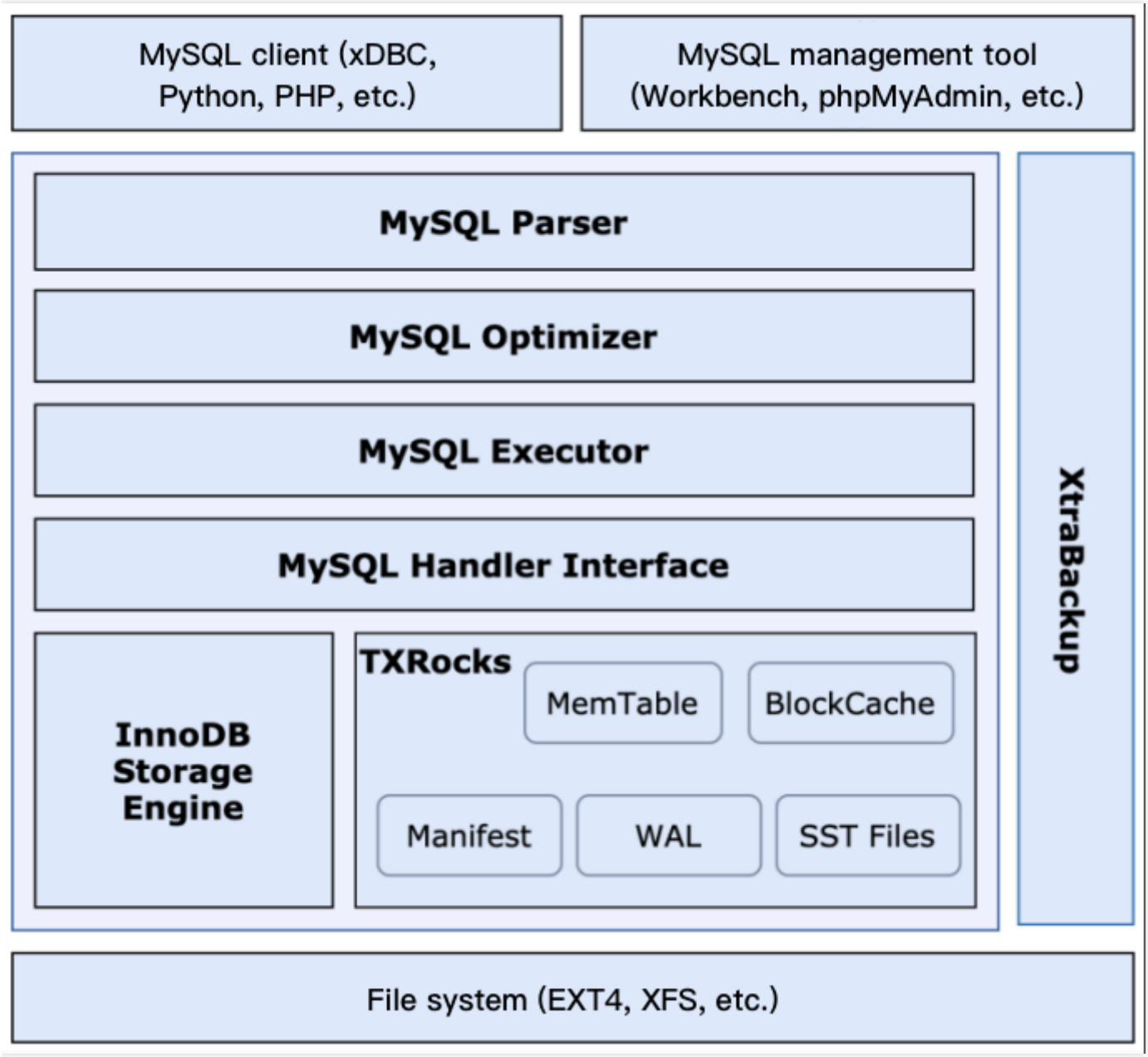
RocksDBはLSM Treeストレージ構造を使用し、データはメモリ内のMemTableとディスク上のSSTファイルの層のセットに編成されます。
書き込みリクエストは、新しいバージョンのレコードをActive MemTableに書き込むと同時に、WALログを永続的に書き込みます。書き込みリクエストにより、MemTableとWALを書き込めば返します。
Active MemTableがある程度いっぱいになったら、Active MemTableを凍結したImmutable MemTableに切り替えます。バックグラウンドスレッドはImmutable MemTableをハードディスクにブラッシュアップし、対応するSSTファイルを生成します。SSTはリフレッシュされた順序で階層化され、通常はL0～L6に分割されます。L1-L6は、各レイヤー内のSSTのレコードが順序付けられ、SSTファイル間でレコード範囲が重複することはありません。L0は、Immutable MemTableが占有するメモリ空間をできるだけ早く解放することをサポートするために、Flushが生成するL0のSSTにレコード範囲のオーバラップを許容します。

1行のレコードを読み込んだ場合、新旧順にActive MemTable、Immutable MemTable、L0、L1～L6の各コンポーネントからこの行を探し、いずれかのコンポーネントから見つかれば、最新のバージョンが見つかったことを意味し、すぐに返します。

範囲スキャンが実行されると、各レイヤーのMemTableを含む各レイヤーのデータに対して、それぞれ1つのイテレータが生成され、これらのイテレータは、マージして次のレコードを検索します。読み取りのプロセスから、LSM Treeのレイヤー数が多すぎると、読み取りのパフォーマンス、特に範囲スキャンのパフォーマンスが著しく低下することがわかります。したがって、LSM Treeの形状をより良いものに維持するために、バックグラウンドではcompaction操作を実行して、下位レイヤーのデータを上位レイヤーのデータに統合し、レイヤー数を減らしていきます。



TXRocksアーキテクチャ



TXRocksストレージエンジンのメリット

ストレージスペースをさらに節約できる

InnoDBが用いているB+Treeインデックス構造に比べて、LSM Treeはかなりの割合のメモリスペースを節約できます。

InnoDBのB+Treeのスプリットは通常、ページが半分格納となり、ページ内の空き領域が無駄になり、ページ有効利用率が低くなります。

RocksDBのSSTファイルは一般的にMB単位以上の大きさに設定されており、ファイルの4Kアライメントの場合、発生する無駄の割合が低く、SST内部もBlockに分割されていますが、Blockはアラインする必要はありません。

また、RocksDBのSSTファイルはプレフィックス圧縮を採用しており、同じプレフィックスは1部しか記録されません。同時にRocksDBの異なるレイヤーのSSTは異なる圧縮アルゴリズムを採用することができ、さらにストレージスペースのオーバーヘッドを下げることができます。トランザクションのoverheadについては、InnoDBのレコードにはtrx id、roll_ptrなどのフィールド情報が含まれているが、TXRocksの最下層のSSTファイル（ほとんどの割合のデータが含まれている）には、データは他のトランザクションオーバヘッドを保持する必要はなく、例えばレコード上のバージョン番号は十分な時間が経過すると消去できます。

書き込みの拡大率がより低い

InnoDBはIn-Placeの修正方式を採用し、1行のレコードを修正するだけでもディスク全体をブラッシュアップすることがあり、高い書き込みの拡大率とランダムな書き込みが発生します。

TXRocksはAppend-Onlyを採用しており、それよりも書き込みの拡張率が低いです。そのためTXRocksは消去回数が限られているSSDなどの製品には優しいです。

ユースケース

TXRocksは、ストレージコストが重視され、読み取り量が少ないが、書き込み量が多く、トランザクションの読み取りと書き込みのパフォーマンスが要求される、大量のデータストレージを必要とするビジネスシナリオに最適です。

TXRocksストレージエンジンの使用方法

TXRocksエンジンのご使用にあたっての注意事項をご参照ください。

最適化と今後の展開

TXRocksは業務のニーズに応じて部分的な最適化を行い、例えば最適化sum演算子プッシュダウンの最適化により、sumクエリの性能を30倍以上に最適化しました。同時にTXRocksも新しいハードウェアとの結合を積極的に模索しており、AEPを二次キャッシュとして利用し、性能を大幅に向上させ、コストパフォーマンスを高めます。

MySQLのストレージエンジンであるTXRocksは、今後、ビジネスでの使用で発生した問題に対応するために最適化され、改善されていきます。また、InnoDBの重要な補完として、TXRocksストレージエンジンは、より多くの重要なビジネスで導入され、スムーズに稼働します。

TXRocksエンジンのご使用にあたっての注意事項

最終更新日：：2022-06-13 15:37:15

TXRocksは、Tencent TXSQLチームがRocksDBに基づいて開発したトランザクション型ストレージエンジンであり、さらにストレージスペースを節約し、書き込みの拡大率が低いという利点を併せ持っています。

製品紹介

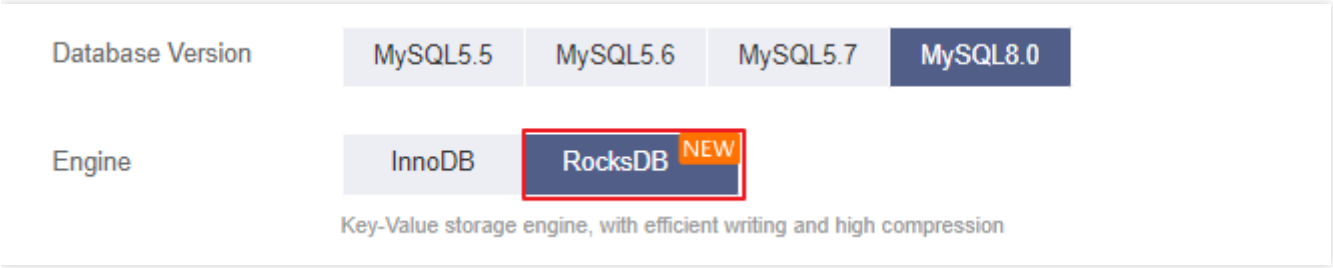
TXRocksは、Tencent TXSQLチームがRocksDBに基づいて開発したトランザクション型ストレージエンジンであり、RocksDB LSM Treeストレージ構造により、InnoDBページの半分格納と断片化の無駄を減らすと同時に、コンパクトな形式のストレージを使うことができます。そのため、TXRocksはInnoDBと近い性能を維持するの前提の下で、ストレージ空間はInnoDBによりも半分以上節約することができ、さらにトランザクションの読み書きパフォーマンスが要求され、大量のデータを保存するビジネスに最適です。

前提条件

データベースのバージョンはMySQL 5.7、8.0で、アーキテクチャは2ノードでなければなりません。

TencentDB for MySQLインスタンスの購入（RocksDBエンジン）

TencentDB for MySQL購入ページでインスタンスを購入するときに、エンジンとしてRocksDBを選択することができます。他のパラメータ項目について、MySQLインスタンスを作成をご参照ください。



説明：

RocksDBはkey-valueストレージエンジンであり、効率的な書き込み能力と高圧縮ストレージとして知られています。現時点で、エンジンとしてRocksDBを選択することをサポートしているのはMySQL 5.7、8.0だけです。

RocksDBテーブルの作成

インスタンスを作成する際に、デフォルトのエンジンをRocksDBと設定する場合、テーブル作成時のデフォルトエンジンはRocksDBです。デフォルトエンジンは、以下のコマンドで確認できます：

```
show variables like '%default_storage_engine%';
```

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

デフォルトエンジンはRocksDBである場合は、テーブル作成ステートメントではストレージエンジンの指定が許可されません。

```
mysql> create table tencent_db (id int primary key,c1 varchar(30),c2 varchar(50));
Query OK, 0 rows affected (0.00 sec)

mysql> show create table tencent_db;
+------------+-----------------------------------------------------------------------------------------------------------------
-----------------+
| Table      | Create Table
                |
+------------+-----------------------------------------------------------------------------------------------------------------
-----------------+
| tencent_db | CREATE TABLE `tencent_db` (
  `id` int(11) NOT NULL,
  `c1` varchar(30) DEFAULT NULL,
  `c2` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=RocksDB DEFAULT CHARSET=utf8 |
+------------+-----------------------------------------------------------------------------------------------------------------
-----------------+
1 row in set (0.00 sec)
```

テーブルが正常に作成された後、その後の使用方法はInnoDBと同様に、データはRocksDBエンジンに格納されます。

エンジン機能の制限

TXRocksは、次の表に示すように、エンジン機能にいくつかの制限があります：

機能分類	機能項目	TXRocks制限
DDL	Online DDL	サポートされません。例えば、ALTER TABLE ... ALOGRITHM=INSTANT機能をサポートしません。Partition 管理操作ではCOPYアルゴリズムだけをサポートします。
SQL機能	外部キー	外部キー（Foreign Key）がサポートされません
	パーティションテーブル	パーティションテーブル（Partition）がサポートされません
	生成列	生成列（Generated Columns）がサポートされません
	明示的なDefault表現式	サポートされません。例えば、CREATE TABLE t1（c1 FLOAT DEFAULT(RAND())）ENGINE=ROCKSDB; が失敗して、'Specited storage engine' is not supported for default value expressions.が表示されます
	暗号化テーブル	暗号化テーブルをサポートしません
インデックス	空間インデックス	空間インデックス（Spatial Index）、空間データ型（GEOMETRY、POINT等）をサポートしません
	フルテキストインデックス	フルテキストインデックス(Fulltext Index)をサポートしません
	多値インデックス	多値インデックス(multi-valued index)をサポートしません
レプリケーション	グループレプリケーション	グループレプリケーション(Group Replication)をサポートしません
	binlog形式	ROW形式をサポートしますが、stmtまたはmixed形式をサポートしません
	クローンプラグイン	クローンプラグイン(Clone Plugin)をサポートしません

機能分類	機能項目	TXRocks制限
	トランスポータブル表領域	トランスポータブル表領域(Transportable Tablespace)をサポートしません
トランザクションとロック	LOCK NOWAITとSKIP LOCKED	LOCK NOWAITとSKIP LOCKEDをサポートしません
	ギャップロック	ギャップロック(Gap Lock)をサポートしません
	Savepoint	Savepointをサポートしません
	LOBフィールドの一部更新	LOBフィールドの一部更新をサポートしません
	XAトランザクション	使用をお勧めしません

パラメータの説明

説明：
TencentDB for MySQLインスタンスを作成する際には、RocksDB をデフォルトストレージエンジンとして選択でき、下の表のパラメータの説明に基づいて自社の業務に合わせたパラメータテンプレートを調整することができます。

MySQL 5.7に関するパラメータリスト

パラメータ名称	再起動が必要かどうか	デフォルト値	許容値
rocksdb_use_direct_io_for_flush_and_compaction	はい	ON	ON/OFF

パラメータ名称	再起動が必要かどうか	デフォルト値	許容値
rocksdb_flush_log_at_trx_commit	いいえ	1	0/1/2
rocksdb_lock_wait_timeout	いいえ	1	1-1073741824
rocksdb_deadlock_detect	いいえ	ON	ON/OFF
rocksdb_manual_wal_flush	はい	ON	ON/OFF

MySQL 8.0関連パラメータリスト

パラメータ名称	再起動が必要かどうか	デフォルト値	許容値

パラメータ名称	再起動が必要かどうか	デフォルト値	許容値
rocksdb_flush_log_at_trx_commit	いいえ	1	0/1/2
rocksdb_lock_wait_timeout	いいえ	1	1-1073741824
rocksdb_merge_buf_size	いいえ	524288(=512K)	100-18446744073709551
rocksdb_merge_combine_read_size	いいえ	8388608 (=8M)	524288(=512K)-1844674
rocksdb_deadlock_detect	いいえ	ON	ON/OFF
rocksdb_manual_wal_flush	はい	ON	ON/OFF

RocksDB エンジンモニタリング項目

下の表はRocksDBのエンジンモニタリングメトリックです。

メトリック	説明
rocksdb_bytes_read	ディスク読み取り数

メトリック	説明
rocksdb_bytes_written	ディスク書き込み数
rocksdb_block_cache_bytes_read	データブロック読み取り数
rocksdb_block_cache_bytes_write	データブロック書き込み数
rocksdb_wal_log_capacity	WAL書き込みログのサイズ

TXRocksのコストパフォーマンス

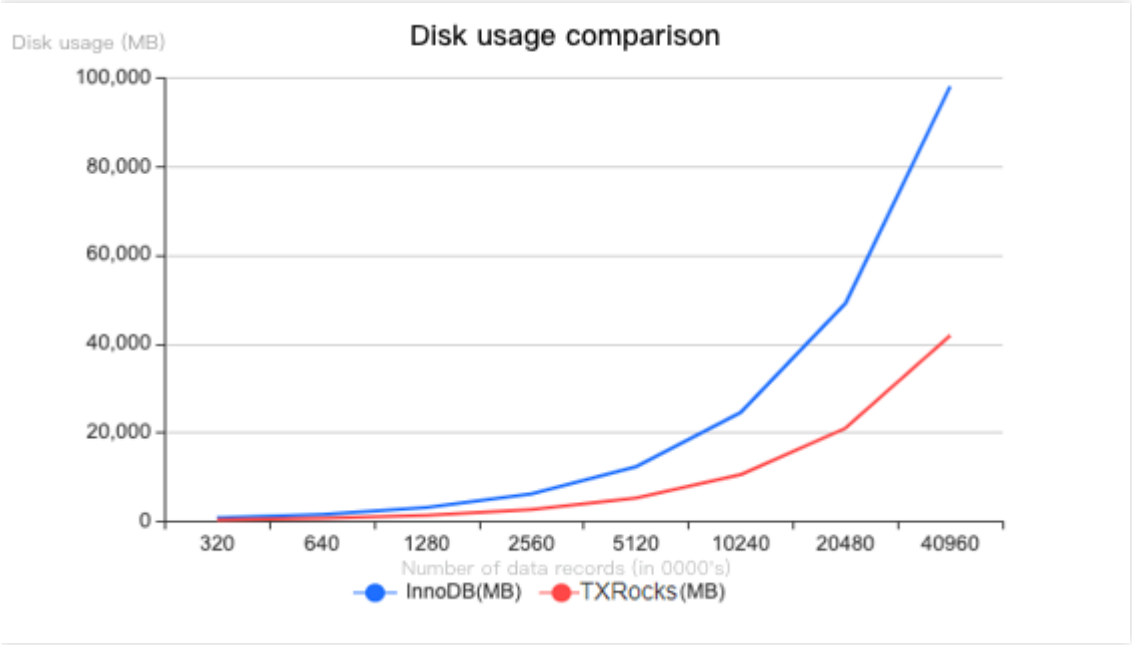
最終更新日：：2022-06-13 15:37:15

TXRocksの性能はInnoDBに近いですが、LSM Treeストレージ構造により、InnoDBページの半分格納と断片化の無駄が削減され、InnoDBによりもTXRocksのほうがストレージ容量を節約できるため、優れたコストパフォーマンスを実現できます。

背景情報

TencentDB製品の中で、TXRocksはInnoDBの重要な補足で、性能が近いということを基礎として、TXRocksは部分的な最適化と改善が行われ、ストレージ空間について、InnoDBよりもさらに節約します。以下はスペース占有と性能面から2つのエンジンを比較します。

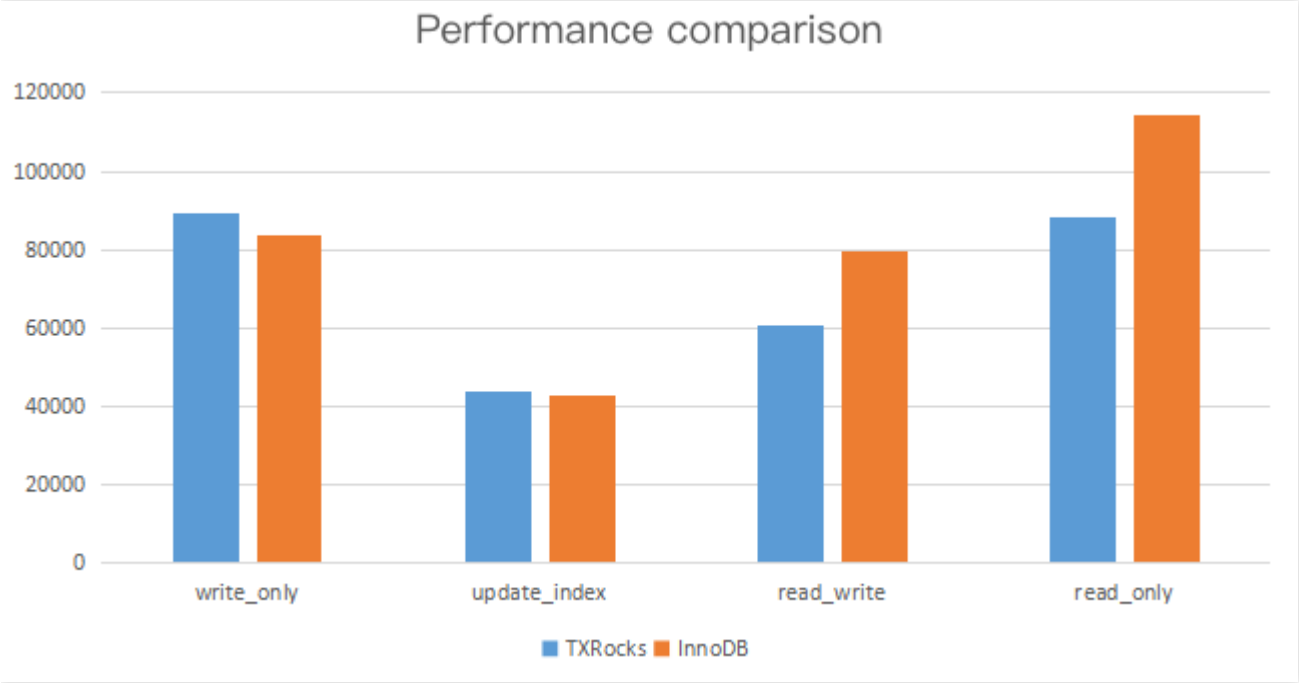
TXRocksのスペース占有はInnoDBよりも少ない



テストシナリオ：どちらのストレージエンジンもデフォルトの設定を使用し、SysBenchのデフォルトのテーブル構造を使用します。各テーブルには80万レコードが含まれ、合計テーブル数は4から512に増加しています。
上の図はテスト条件でTXRocksとInnoDBストレージエンジンをそれぞれ使用した場合のスペース占有状況を示し、左側はInnoDBとTXRocksストレージエンジンを使用した場合のハードディスクの使用状況を示しています。実測データによると、データ量が徐々に増加するにつれて、TXRocksエンジンのハードドライブの消費量はさらに緩やかになり、より多くのスペースを節約し、最大でInnoDBの42.71%しかありませんでした。プレフィックス

の重複率が高いデータを記録する場合、TXRocksのほうは圧縮率が高く、ストレージ・コスト・パフォーマンスに優れています。

TXRocksの性能がInnoDBとほぼ同じ



テストシナリオ：インスタンス8コア32GBシナリオ、6つのテーブル500万のデータ、各テストはリスタート後のコールドスタートテストであり、1caseあたり1200秒実行します。
上の図はテストシナリオ条件でTXRocksとInnoDBのストレージエンジンをそれぞれ使用した場合の性能比較であるが、比較するとTXRocksとInnoDBの性能が近いことが分かります。
sysbenchコマンドのキーパラメータ：

```
sysbench --table-size=5000000 --tables=6 --threads=32 --time=1200
```

まとめ

TXRocksは、InnoDBと同様のパフォーマンスを発揮しますが、省スペースのTencentDB for MySQLストレージエンジン製品です。ビジネスパフォーマンスのニーズを保証するとともにストレージ・コストを削減することができます。TXRocksの詳細については、TXRocksの概要をご参照ください。

TXRocksベストプラクティス

最終更新日：：2022-07-12 14:30:07

このドキュメントは、TXRocksを使用するベストプライクティスとして、大量のデータをインポートするときのインポート速度が向上することについてご説明します。

背景

- **シナリオ**：大量のデータをTXRocksエンジンのデータベースにインポートする場合、インポートを高速化する必要があります。
- **影響**：大量のデータをインポートすると、エラー `Rows inserted during bulk load must not overlap existing rows` が発生する可能性があります。

処理方法1

1. 最初にセカンダリインデックスを削除します（プライマリキーインデックスのみを保持します）。
2. 仕様とユーザーデータ量に基づいてメモリ関連パラメータを調整します。

> 説明：
> 仕様とデータ量に応じて、パラメータrocksdb_merge_buf_sizeとrocksdb_merge_combine_read_sizeを適切に大きくする必要があります。

- rocksdb_merge_buf_sizeは、インデックス作成中にマルチパスマージのときの1パスあたりのデータ量を表し、rocksdb_merge_combine_read_sizeは、マルチパスマージのときの各ウェイで消費される合計メモリを表します。
- rocksdb_block_cache_sizeは、rocksdb_block_cacheのサイズを表します。マルチパスマージのときには、一時的に小さくすることをお勧めします。

3. bulk load方式によるデータのインポート。

```
SET session rocksdb_bulk_load_allow_unsorted=1;
SET session rocksdb_bulk_load=1;
...
データのインポート
...
```

```
SET session rocksdb_bulk_load=0;
SET session rocksdb_bulk_load_allow_unsorted=0;
```

説明：
インポートしたデータが順序付けられている場合は、rocksdb_bulk_load_allow_unsortedを設定する必要はありません。
4. セカンダリインデックスの再構築には、すべてのデータのインポートが完了した後で、セカンダリインデックスを1つずつ再構築することができます。
!

- セカンダリインデックスの作成にはマルチパスマージが含まれます。rocksdb_merge_buf_sizeは1パスあたりのデータサイズ、rocksdb_merge_combine_read_sizeはマージ中にマルチパスマージに使用された合計メモリーサイズです。
- 例えば、rocksdb_merge_buf_sizeを64MB以上、rocksdb_merge_combine_read_sizeを1GB以上に設定することをお勧めします。OOMを回避するには、データのインポートが完了したら必ず元のパラメータ値に戻します。
- また、各セカンダリインデックス作成プロセスでは多くのメモリが消費されるため、同時に複数のセカンダリインデックスを作成することは推奨されません。

処理方法2

データのインポート中にunique_checkをオフにすると、インポートにおけるパフォーマンスが向上します。

```
SET unique_checks=OFF;
...
データのインポート
...
SET unique_checks=ON;
```

注意：
処理が完了したら、unique_checksを必ずONに戻してください。そうしないと、その後の通常のトランザクションによって書き込まれるinsert操作は一意性をチェックしません。