Разбираемся с блокировками в MySQL Server.

Дмитрий Ленев. Июнь 2014 г.

Legal notice.

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.

О чем этот доклад?

Почему блокировки важны?

- Обеспечивают I в ACID
- Влияют на поведение и производительность приложений

О чем этот доклад:

- Обзор основных видов блокировок в MySQL Server
- Типичные проблемы для каждого вида
- Возможные пути решения этих проблем

Виды блокировок в MySQL

- Глобальная блокировка для чтения (GRL)
- Блокировки метаданных (MDL)
- Унаследованные табличные блокировки (THR_LOCK)
- Блокировки в InnoDB/других storage engine
- Ожидание FLUSH TABLE

- Пользовательские блокировки (GET_LOCK(),...)

Глобальная блокировка для чтения/ Global Read Lock

- Используется FLUSH TABLES WITH READ LOCK и SET @@read_only=1
- Блокирует изменения данных и метаданных, фиксацию транзакций
- Блокирует индивидуальные команды, а не транзакции

- Использует менеджер MDL (с 5.5)
- Два типа захватов:
  - DML и COMMIT - IX
  - FTWRL и SET @@read_only - S

	IX	S
IX	+	-
S	-	+

GRL: Как это выглядит?

id	user	command	time	state	info
2	root	Sleep	3		NULL
5	root	Query	0	executing	select id, user, command ...
3	root	Query	2	Waiting for global read lock	flush tables with read lock
4	root	Query	1	Waiting for global read lock	insert into t1 values (1)

Активный LOCK TABLE t1 WRITE в соединении 2 блокирует FLUSH TABLES WITH READ LOCK из соединения 3, тем самым блокируя все остальные изменения в сервере.

GRL: Типичные проблемы и решения

Длительный DML и LOCK TABLES WRITE блокируют FLUSH TABLES WITH READ LOCK (см. I_S.PROCESSLIST)

Возможные пути решения:

- Избегайте FLUSH TABLES WITH READ LOCK – используйте InnoDB + средства типа MySQL Enterprise Backup (c –only-innodb[-with-frm])
- Избегайте LOCK TABLES WRITE и длительного DML (используйте LIMIT)
- Убивайте соединения с активным LOCK TABLES WRITE или длительным DML (можно найти через “mysqladmin debug” и P_S)

Блокировки метаданных / MDL

- Защищают метаданные таблиц и других объектов (представлений, процедур)
- Изолируют запросы/DML от DDL
- Поддерживают разные типы объектов и иерархии блокировок
- Объекты идентифицируются ключом (тип, имя базы, имя объекта)
- Поддерживают обнаружения тупиков (поиск в графе ожиданий, эмпирики)
- Обычно блокировки удерживаются до конца транзакции
- Появились в 5.5 как решение для ошибок (#989)
- Эволюция в сторону основного менеджера блокировок для крупных объектов

MDL: как это работает для БД?

- CREATE/DROP DATABASE захватывают блокировку на БД в режиме X
- CREATE/DROP TABLE/VIEW/ FUNCTION/PROCEDURE захватывают блокировку на БД в режиме IX
- RENAME TABLE захватывает блокировки в режиме IX на исходную и конечную БД

	IX	X
IX	+	-
X	-	-

MDL: как это работает для других объектов?

- S/SH — запросы к I_S, PREPARE, для процедур
- SR — DML при чтении данных
- SW — DML изменение данных
- SU/SNW — ALTER TABLE
- SNRW — LOCK TABLES WRITE
- X — RENAME/ DROP/ ALTER TABLE, другие DDL
- Есть еще приоритеты

	S/SH	SR	SW	SU	SNW	SNRW	X
S/SH	+	+	+	+	+	+	-
SR	+	+	+	+	+	-	-
SW	+	+	+	+	-	-	-
SU	+	+	+	-	-	-	-
SNW	+	+	-	-	-	-	-
SNRW	+	-	-	-	-	-	-
X	-	-	-	-	-	-	-

MDL: Как это выглядит?

id	user	command	time	state	info
3	root	Query	2	Waiting for table metadata lock	alter table t1 add column j ...
4	root	Query	1	Waiting for table metadata lock	select * from t1
5	root	Query	0	executing	select id, user, command ...
2	root	Query	3	Sending data	Select count(*) from t1, t1 as b, ...

Смена режима блокировки с SU на X в ALTER TABLE заблокирован долгим SELECTом в соединении 2, как результат SELECT в соединении 4 также блокирован

MDL: Проблемы и решения (1)

Проблема:

- Длинные/забытые транзакции блокируют DDL и создают затор

Признаки:

- Легко увидеть в I_S.PROCESSLIST

Решения:

- Избегать длинных транзакций/LOCK TABLES
- Разделять нормальные операции и DDL по времени
- Найти и убить соединения с проблемными транзакциями:
  - mysqladmin debug – для поиска соединений с LOCK TABLES
  - I_S.INNODB_TRX/INNODB_LOCK_MONITOR – для поиска соединений с длинными InnoDB транзакциями
  - P_S.METADATA_LOCKS (5.7)

MDL: Проблемы и решения (2)

Проблема:

- Прерывание и откат транзакций из-за MDL тупиков
  - ALTER TABLE
  - RENAME/DROP TABLES

Признаки:

- Ошибка ER_LOCK_DEADLOCK на клиенте

Решения:

- Сделать клиент устойчивым к таким ошибкам
- Захватывать сначала SW блокировку, а потом SR
- Предварительный захват блокировок

Унаследованные табличные блокировки (THR_LOCK)

- Защищает данные в таблицах для простых storage engine (MyISAM, Heap, Archive, CSV,...)
- Изолируют запросы и операторы (и DML, и DDL) друг от друга
- Применяются только для обычных таблиц
- Держатся в течение одного оператора
- Используют технику избегания тупиков, основанную на захвател блокировок в фиксированном порядке, поэтому должны захватываться одномоментно

THR_LOCK: Какие и когда?

- Чтение + нет binary log = READ
- Чтение + binary log = READ_NO_INSERT
- INSERT + @concurrent_insert=1 = WRITE_CONCUR_INSERT
- Изменение таблички + простой SE = WRITE
- InnoDB = READ или WRITE_ALLOW_WRITE
- Приоритеты!

	R/ RHP	RNI	WAW	WCI	W/ WLP
READ/READ_HIGH_PRIO	+	+	+	+	-
READ_NO_INSERT	+	+	+	-	-
WRITE_ALLOW_WRITE	+	-	-	-	-
WRITE_CONCUR_INSERT	+	-	-	-	-
WRITE/WRITE_LOW_PRIO	-	-	-	-	-

THR_LOCK: Как это выглядит?

id	user	command	time	state	info
2	root	Sleep	3		NULL
3	root	Query	2	Waiting for table level lock	update t1 set i = 1
5	root	Query	0	executing	select id, user, command ...
4	root	Query	1	Waiting for table level lock	select * from t1

LOCK TABLES READ в соединении 2 блокирует UPDATE из соединения 3 и создает пробку (см. соединение 4).

THR_LOCK: Проблемы и решения (1)

Проблема:

- Одна из табличных блокировок становится узким местом

Признаки:

- I_S.PROCESSLIST
- Переменные Table_locks_immediate/waited
- P_S.table_locks_waits_summary_by_table

Решения:

- Использовать concurrent insert
- Пересмотреть код приложения на предмет требований к блокировкам (RBR, SELECT потом UPDATE).
- InnoDB

THR_LOCK: Проблемы и решения (2)

Проблема:

- Запросы блокировки на READ “голодают”

Признаки:

- I_S.PROCESSLIST
- Переменные Table_locks_immediate/waited
- P_S.table_locks_waits_summary_by_table

Решения:

- Использовать concurrent insert
- Использовать модификаторы приоритетов (LOW_PRIORITY, HIGH_PRIORITY, @@low_priority_updates)
- --max-write-lock-count
- InnoDB

Блокировки InnoDB

- Защищают данные в таблицах
- Изолируют DML операторы друг от друга
- Бывают нескольких видов:
  - Табличные
  - На индивидуальные записи в индексах
  - На пропуски в индексах
  - Next-key lock = на запись + на пропуск перед ней
- Поддерживают обнаружение тупиков на основе графа ожиданий
- Обычно держатся в течение транзакции
- Не используются когда возможно использовать MVCC

Блокировки InnoDB: режимы и совместимость

- IS, IX, AI — только на таблицы
- S и X — на таблицы и на индивидуальные записи

	IS	IX	S	X	AI
LOCK_IS	+	+	+	-	+
LOCK_IX	+	+	-	-	+
LOCK_S	+	-	+	-	-
LOCK_X	-	-	-	-	-
LOCK_AUTO_INC	+	+	-	-	-

Блокировки InnoDB: преимущества версионности

SELECT к InnoDB табличке может быть:

Блокирующим:

- Если мы в SERIALIZABLE режиме
- Если использовался IN SHARE MODE/FOR UPDATE

Неблокирующим, использующим версионность:

- По умолчанию/в остальных случаях
- Не захватывает блокировок
- Использует consistent read view на начало транзакции (REPEATABLE READ) или оператора (READ COMMITTED)
- Больше параллелизма!

Блокировки InnoDB: табличные блокировки.

- IS — захватываются блокирующими чтениями и DML операторами которые читают данные
- IX — захватываются DML которые меняет данные
- Табличные S и X блокировки захватываются только LOCK TABLES и некотороми фазами ALTER TABLE

- AUTO_INC блокировки захватываются INSERT (детали зависят от —innodb_auto_inc_mode) и удерживаются в течение выполнения оператора а не транзакции

Блокировки InnoDB: строчные блокировки.

- SELECT в SERIALIZABLE режиме захватывает S next-key блокировки на кажду запись в индексе что он проходит.
- SELECT IN SHARE MODE/FOR UPDATE всегда захватывают S или X next-key блокировки соответственно.
- UPDATE и DELETE захватывают X next-key блокировки на каждую просмотренную запись.
- INSERT захватывает IX блокировку на пропуск куда намечена вставка, X блокировку на запись которая вставляется и S блокировки на встреченные дубликаты

- Если захватывается X блокировка на вторичный индекс, то захватывается блокировка и на первичный ключ
- Это упрощенная картина (innodb-locks-unsafe-for-binlog)

Блокировки InnoDB: Пример

Для таблицы:

CREATE TABLE t1 (pk INT PRIMARY KEY, sec_key INT, t INT, KEY(sec_key))

И оператора UPDATE t1 SET t=10 WHERE sec_key=22

Будут захвачены (при некоторых предположениях):

- IX блокировка на таблицу t1
- X блокировка на строчку 22 в ключе sec_key
- X блокировка на пропуск (22, 23) в ключе sec_key
- X блокировка на строчку в PRIMARY KEY

Блокировки InnoDB: Как это выглядит?

Используем I_S.INNODB_TRX так как I_S.PROCESSLIST не слишком помогает

trx_id	trx_state	trx_wait_started	trx_mysql_thread_id	trx_query
140461753338968	LOCK WAIT	2013-09-11 10:49:26	3	select * from t1 for update
140461753337968	RUNNING	NULL	2	NULL

Транзакция в соединении 2 захватила X блокировку на какие-то строчки из 't1' и блокирует SELECT FOR UPDATE из соединения 3.

Больше информации доступно в innodb_lock_monitor и I_S.INNODB_LOCKS

Блокировки InnoDB: Проблемы и решения (1)

Проблема: Тупики

Признаки:

- ER_LOCK_DEADLOCK на клиенте
- Сообщения в SHOW ENGINE INNODB STATUS
- Сообщения в логе ошибок при --innodb_print_all_deadlocks=1

Решения:

- Сделайте клиентов устойчивыми, перезапускайте транзакции
- Уменьшите размер транзакций
- Используйте одинаковый порядок для изменений и блокирующих чтений
- Задумайтесь о более слабом уровне изоляции (RBR!)

Блокировки InnoDB: Проблемы и решения (2)

Проблема:

- Блокировки InnoDB становятся узким местом.

Признаки:

- Значения Innodb_row_lock_* переменных
- SHOW ENGINE INNODB STATUS + INNODB_LOCK_MONITOR
- I_S.INNODB_TRX/INNODB_LOCKS/INNODB_LOCK_WAITS

Решения:

- Проверьте планы запросов, используйте индексы
- Попытайтесь уменьшить размер транзакций
- Подумайте об использовании более слабого уровня изоляции (RBR!)

Блокировки InnoDB: Проблемы и решения (3)

Проблема: Блокировки на конкретные «горячие» строчки в InnoDB таблице становятся узким местом.

Признаки:

- Значения Innodb_row_lock_* переменных
- SHOW ENGINE INNODB STATUS + INNODB_LOCK_MONITOR
- I_S.INNODB_TRX/INNODB_LOCKS/INNODB_LOCK_WAITS

Решения:

- Проверьте действительно ли «горячие» строчки должны быть таковыми по логике приложения
- Постарайтесь уменьшить время в течение которого удерживается блокировка
- Подумайте о разбиении «горячих» строк
- Подумайте об использовании более слабого уровня изоляции (RBR!)

Блокировки InnoDB: Проблемы и решения (4)

Проблема: AUTO_INC блокировка становится узким местом

Признаки:

- Значения Innodb_row_lock_* переменных
- SHOW ENGINE INNODB STATUS + INNODB_LOCK_MONITOR
- I_S.INNODB_TRX/INNODB_LOCKS/INNODB_LOCK_WAITS

Решения:

- Обновитесь до 5.5
- Используйте по крайней мере innodb_autoinc_lock_mode=1
- Избегайте INSERT с неизвестным заранее числом строк
- Используйте RBR

Ожидание FLUSH TABLE

Что такое FLUSH TABLE?

- Выталкивает таблицу из кэша открытых таблиц
- Для простых SE (MyISAM, Archive, ...) это приводит к выталкиванию незаписанных изменений на диск
- Инициируется FLUSH TABLE, FLUSH TABLES WITH READ LOCK и некоторым DDL (ANALYZE TABLE).

Почему его можно рассматривать как блокировку?

- Помечает таблицу для выталкивания
- Это блокирует все новые операторы для этой таблицы
- Опционально ожидает окончания процесса

- FLUSH TABLE “блокировка” удерживается в течение выполнения оператора
- Поддерживается обнаружение тупиков (через MDL)

Ожидание FLUSH TABLE: Как это выглядит?

id	user	command	time	state	info
2	root	Query	2	Writing to net	select * from t1, t1 as a, ...
3	root	Sleep	1		NULL
5	root	Query	0	executing	select id, user, command ...
4	root	Query	1	Waiting for table flush	select * from t1 limit 1

ANALYZE TABLE в соединении 3 пометил таблицу 't1' для выталкивания. Процесс выталкивания ждет долгий SELECT в соединении 2, блокируя другие SELECTы к таблице 't1' (например, см. соединение 4).

Ожидание FLUSH TABLE: Проблема и решения.

Проблема: Долгие запросы и LOCK TABLES не дают FLUSH TABLE завершиться и создают затор.

Признаки:

- Легко видеть в I_S.PROCESSLIST

Решения:

- Избегайте FLUSH TABLE – используйте InnoDB + средства типа MySQL Enterprise Backup
- Избегайте LOCK TABLES и долгих запросов во время выполнения FLUSH TABLE
- Убить соединения с активными LOCK TABLES и долгими запросами (активные LOCK TABLES можно найти через mysqladmin debug и P_S)

Пользовательские блокировки

- Именованные блокировки которые явно захватываются и освобождаются клиентом
- Только X режим
- Одно соединение — одна блокировка в момент времени
- Неявно освобождаются при закрытии соединения, а не в конце транзакции или оператора

- Используются приложениями для реализации логики
- Доступ через функции:
  - GET_LOCK(name, timeout) — освободить текущую блокировку и захватить новую
  - RELEASE_LOCK(name) — освободить блокировку
  - IS_FREE/USED_LOCK(name) — проверить свободна/занята ли блокировка

Пользовательские блокировки: Как это выглядит?

id	user	command	time	state	info
2	root	Sleep	1		NULL
3	root	Query	1	User lock	select GET_LOCK(“a”, 10)
4	root	Query	0	executing	select id, user, command, ...

Соединение 3 ждет блокировку “a” захваченную соединением 2.

Пользовательские блокировки: Проблема и решение.

Проблема:

- Забытые/заснувшие блокировки создают затор

Признаки:

- Хорошо видно в I_S.PROCESSLIST

Решение:

- Найти владельцев используя функцию IS_USED_LOCK()
- Убить их используя KILL

Бонус: межсистемные тупики.

- Тупики включающие блокировки управляемые разными менеджерами блокировок
- Видимы в I_S.PROCESSLIST, I_S.INNODB_* и mysqladmin debug и P_S как несколько групп соединений ожидающих ресурсов захваченных друг-другом.

- Не могут быть обнаружены нормальными средствами в сервере
- Разрешаются за счет таймаутов:
  - @@lock_wait_timeout (1 год по умолчанию)
  - @@innodb_lock_wait_timeout (50 секунд по умолчанию)

Межсистемные тупики: Пример.

Предположим что у нас есть три соединения которые параллельно выполняют следующие транзакции:

Connection 2:
```
BEGIN;
SELECT * FROM t1;
INSERT INTO t2 VALUES (1);
```

Connection 3:
```
BEGIN;
SELECT * FROM t2 FOR UPDATE;
SELECT * FROM t1;
```

Connection 4:
```
ALTER TABLE t1 ADD COLUMN j INT;
```

Межсистемные тупики: Пример.

id	user	command	time	state	info
2	root	Query	3	update	insert into t2 values (1)
3	root	Query	1	Waiting for table metadata lock	select * from t1
4	root	Query	2	Waiting for table metadata lock	alter table t1 add column ...

trx_id	trx_state	trx_wait_started	trx_mysql_thread_id	trx_query
140082901338224	LOCK WAIT	2013-09-11 19:00:13	2	insert into t2 values (1)
140082901339224	RUNNING	NULL	3	select * from t1

Соединение 2 ожидает InnoDB блокировку на строку в 't2', которая захвачена соединением 3, которое ожидает MDL на 't1'. MDL на 't1' не может быть удовлетворен поскольку ALTER TABLE запрашивает на него X блокировку, которую не может получить из-за активной SR блокировки в соединении 2.

Спасибо!