УДК 004.658.2

АЛЬТЕРНАТИВНОЕ ПРИМЕНЕНИЕ ПОЛУСИНХРОННОЙ РЕПЛИКАЦИИ В MYSQL

Харин Н.И., Першенкова В.Г.

ФГБОУ ВПО «Московский государственный технический университет имени Н.Э. Баумана», Москва, e-mail: niki_kharin@mail.ru, v.pershenkova@yahoo.com

В статье рассматриваются различные варианты репликации в системе управления базами данных MySQL. Приводится сравнение алгоритмов работы двух основных подходов к полусинхронной репликации: нормальной полусинхронной репликации и полусинхронной репликации без потерь. Рассматривается вариант оптимальной настройки сервера для уменьшения времени перевода одного из подчиненных серверов в статус главного. Даны рекомендации для основных настроек сервера, необходимых для правильной и эффективной работы полусинхронной репликации в MySQL. Рассматривается альтернатива классическому подходу к полусинхронной репликации в виде использования бинарных журналов в качестве подчиненных серверов, данные в которые поступают в полусинхронном режиме. В свою очередь репликация между различными дата-центрами происходит в асинхронном режиме. Показано, что преимущества использования двоичных журналов являются решающими при работе высоконагруженных систем.

Ключевые слова: системы управления базами данных, реляционные базы данных, репликация, полусинхронная репликация, отказоустойчивая репликация

ALTERNATIVE USE OF SEMISYNCHRONOUS REPLICATION IN MYSQL

Kharin N.I., Pershenkova V.G.

Federal budget-funded institution Bauman Moscow State Technical University, Moscow, e-mail: niki_kharin@mail.ru, v.pershenkova@yahoo.com

The article discusses various replication options at the database management system MySQL. A comparison of the algorithms two main approaches to semi-synchronous replication: semi-synchronous normal replication and semi-synchronous replication without loss. We consider a variant of the optimal server configuration to reduce the time of translation of one of the slave servers in the status of the main. Recommendations are given for basic server settings, necessary for the proper and efficient operation of semi-synchronous replication in MySQL. Is considered alternative to the classical approach to semi-synchronous replication in the form of using binary logs as slave servers, which receive data in semi-synchronous mode. In turn, the replication between different data centers is happening asynchronously. It is shown that advantages of using binary logs are critical when working heavily loaded systems.

Keywords: the database management system, relational database, replication, semi-synchronous replication, failover replication

Рост сложности проблем проектирования современных баз данных и необходимость их решать в сжатые сроки определяет актуальность разработки комплексных методов и средств, позволяющих решать проектные задачи на качественно новом уровне.

Одной из наиболее широко распространенных методологий проектирования баз данных является методология проектирования реляционных баз данных (РБД).

В области разработки РБД развитие методов и средств их проектирования связано с созданием таких подходов, которые обеспечили бы, кроме реализации традиционных технологий, эффективное использование технологий человеко-машинного проектирования, использование существующей информации в рассматриваемой предметной области [1].

Многие современные системы управления базами данных (СУБД) предъявляют очень высокие требования к скорости отработки поисковых запросов при условии одновременной работы большого количества клиентов [2, 3]. Способом реализации таких требований может являться усовершенствование процесса репликации – это процесс, под которым понимается копирование данных из одного источника на другой (или на множество других) и наоборот, механизм синхронизации содержимого нескольких копий объекта (например, содержимого базы данных).

Введение в полусинхронную репликацию

В *MySQL* репликации по умолчанию являются асинхронными. При асинхронной репликации алгоритм работы прост – главный сервер записывает события в собственный журнал двоичных данных, но не знает, извлек и переработал ли их хотя бы один из подчиненных серверов [4]. Основным недостатком данного вида репликации является то, что в случае отказа главного сервера все транзакции, проведенные им, возможно, не будут переданы ни одному из его подчиненных серверов до поломки, и, как следствие

этого, переход от сломанного главного сервера к подчиненному может привести к потере данных. Для решения подобной проблемы был введен механизм полусинхронной репликации.

Интерфейс полусинхронной репликации начал поддерживаться в *MySQL,* начиная с версии 5.5, и был реализован в виде подключаемых модулей [4]. Для работы полусинхронного режима при подключении к главному серверу каждый ведомый сообщает о том, что он может работать в данном режиме. Когда настроена конфигурация полусинхронной репликации и есть хотя бы один подчиненный сервер, работающий в полусинхронном режиме, то при осуществлении транзакции основной поток главного сервера блокируется до того момента, пока хотя бы один из ведомых серверов не оповестит о получении всей информации, либо поток будет разблокирован по истечению тайм-аута.

Если Ваша система не является высоконагруженной, но потеря данных для Вас является критичным фактором, то Вам следует использовать асинхронную репликацию [3, 4]. В данном случае, при отказе системы, необходимо следовать следующим правилам:

– Установить следующие параметры в конфигурации сервера: *innodb_flush_log_at_trx_commit=1* и *sync_binlog=1*.

– После аварии на главном сервере необходимо подождать некоторое время (от 10 до 30 минут), до того, как главный сервер не восстановится полностью. Восстановление занимает так много времени, поскольку оно включает в себя перезагрузку операционной системы, восстановление памяти и данных файловой системы, а также восстановление используемой подсистемы хранения после сбоев.

– В случае, если все данные на главном сервере восстановятся, возможно продолжение работы без потери данных. Так как все данные есть на главном сервере, то все подчиненные сервера без проблем могут реплицировать все данные себе.

– Если все же Ваш сервер не смог восстановиться, и данные были потеряны, то возможна незамедлительная замена текущего главного сервера подчиненным, причем нет необходимости тратить время на полную перезагрузку отказавшего сервера. При этом Вы потеряете данные, но альтернатив данному решению нет.

Очевидно, что у данного подхода есть два очевидных недостатка – большое время простоя системы, необходимое для восстановления главного сервера, и отсутствие возможности полностью устранить риск потери данных. В современных реалиях при работе высоконагруженных систем такие факторы, как время отклика и сохранность всех поступающих в систему данных, являются решающими [7]. В подобного рода системах асинхронная репликация может привести к очень дорогостоящим последствиям, что делает полусинхронный подход к репликации настолько актуальным.

Нормальная полусинхронная репликация и полусинхронная репликация без потерь

В общем случае оба вида полусинхронной репликации работают по следующему алгоритму [5]:

1. Подготовка журнала двоичных данных.
2. Подготовка подсистемы хранения (синхронизация состояния подсистемы на диске с состоянием подсистемы в памяти).
3. Подтверждение журнала двоичных данных – кэширование файлов на жесткий диск.
4. Подтверждение журнала двоичных данных – синхронизация состояния данных на диске с состоянием данных в памяти.
5. Ожидание подтверждения от подчиненного сервера (при использовании полусинхронной репликации без потерь).
6. Подтверждение выполнения от подсистемы хранения – снятие блокировок с таблиц. Все изменения видны для пользователей системы.
7. Ожидание подтверждения от подчиненного сервера (при использовании нормальной полусинхронной репликации).

При использовании нормальной полусинхронной репликации подтверждение в подсистему хранения отсылается без ожидания подтверждений о записи данных в подчиненный сервер. Таким образом, изменения видны всем пользователям несмотря на то, что подчиненный сервер, работающий в полусинхронном режиме, мог не получить необходимые данные. В случае аварии на сервере может произойти потеря данных, на которые не было выслано подтверждение ни от одного из подчиненных серверов, это в свою очередь приводит к фантомным чтениям.

Пример подобного фантомного чтения приведен на рис. 1: пользователь 2 до отказа главного сервера мог считать данные (в данном случае строку со значением 3), а после перехода к промотированному подчиненному серверу данная строка была потеряна.

Использование полусинхронной репликации без потерь помогает избежать проблемы фантомного чтения. При данном виде репликации подтверждение в подсистему хранения высылается только после подтверждения, высланного от одного из подчиненных серверов. Пример работы полусинхронной репликации без потерь приведен на рис. 2.

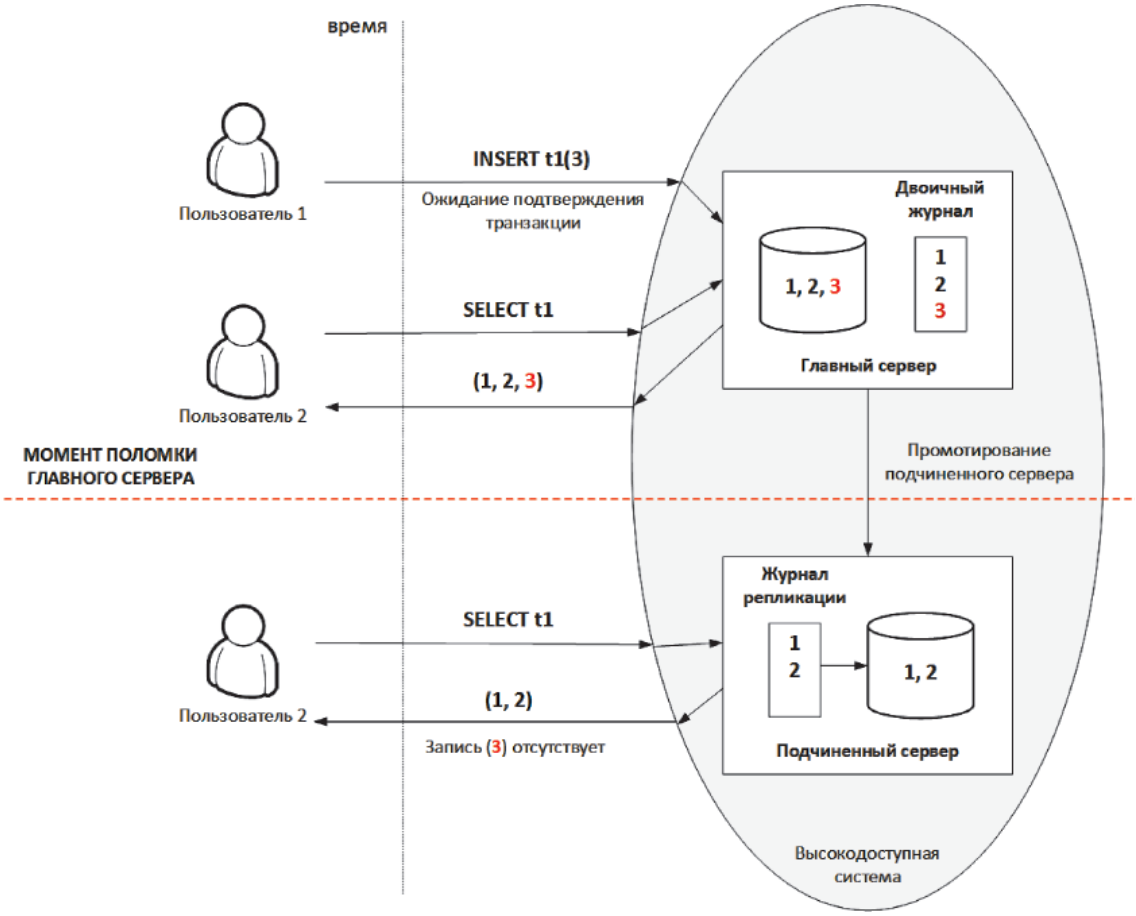

Рис. 1. Фантомные чтения в нормальной полусинхронной репликации

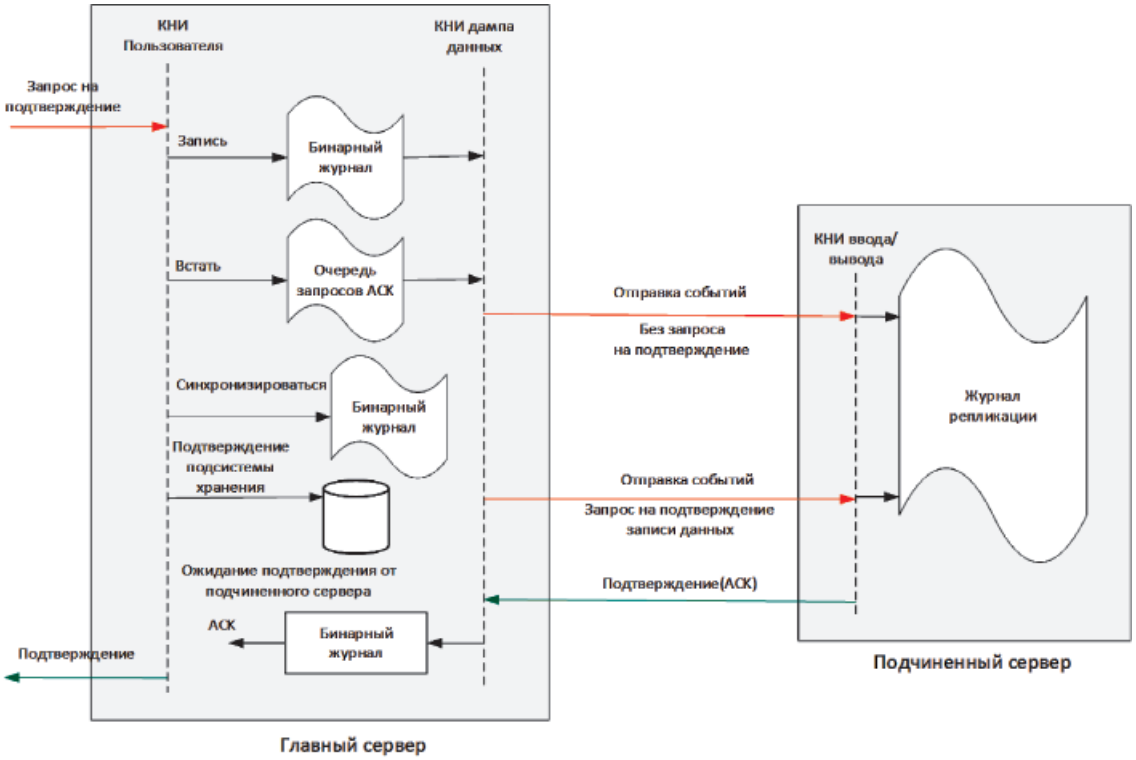

Рис. 2. Полусинхронная репликация без потерь

В данном случае риск фантомных чтений минимален. Только в том случае, если и главный сервер, и подчиненный одновременно откажут, все данные о последних транзакциях будут потеряны.

Уменьшение временных затрат на поддержание долговечности

Под долговечностью, в общем случае, понимают фиксацию всех изменений, внесенных в ходе транзакций. Причем при любом отказе системы данные «долговечные» изменения не могут быть потеряны. При отказе главного сервера перевод одного из подчиненных серверов, работающих в полусинхронном режиме, в статус главного осуществляется за счет применения к нему журналов отличий серверов. Стоит отметить, что долговечность отказавшего сервера не играет в данном процессе никакой роли, и именно поэтому при конфигурации сервера такие параметры, как *innodb_flush_log_at_trx_commit* и *sync_binlog,* должны быть установлены в любые значения кроме 1. Таким образом, можно безопасно снизить долговечность, что в свою очередь приведет к еще большему выигрышу:

– Снижение времени выполнения групповых транзакций.

– Снижение количества операций ввода/вывода за счет снижения количества вызовов функции синхронизации данных в оперативной памяти и на диске. В целом общая нагрузка на диск снижается.

– Уменьшение количества записываемых данных. В некоторых случаях подобные настройки уменьшают количество записываемых данных в два раза, что при использовании, например, флеш-памяти увеличивает ее время жизни.

Требования к полусинхронной репликации

Для того чтобы снизить время на передачу данных по сети, при использовании полусинхронной репликации, необходимо, чтобы хотя бы один из подчиненных серверов, работающих в данном режиме, располагался в том же дата-центре, что и главный сервер. Иначе, помимо времени, необходимого для выполнения данной транзакции, к времени для перевода подчиненного сервера в статус главного будет прибавляться круговая задержка на передачу данных между серверами (*RTT*), что приводит к большим суммарным затратам.

Чтобы всегда поддерживать технологию полусинхронной репликации, Вам необходимо сконфигурировать ваш сервер таким образом, чтобы параметр *rpl_semi_sync_master_timeout* был очень большим, и иметь как минимум два подчиненных сервера, работающих в полусинхронном режиме.

Двоичные журналы как альтернатива подчиненному серверу

Начиная с версии *MySQL 5.6* стала доступна функция удаленной записи в двоичные журналы [5, 6]. Что наиболее интересно – удаленные резервные копии двоичных журналов могут работать так же, как подчиненный сервер, – в полусинхронном режиме. Утилита *mysqlbinlog* выполняет соединение с главным сервером, запрашивает дамп бинарных журналов и по механизму, аналогичному репликации, передает данные на необходимый удаленный сервер.

Зачастую в высоконагруженных системах главные и подчиненные серверы находятся в разных дата-центрах, что вызывает большие временные задержки при переходе от главного сервера к ведомому. Именно поэтому необходима альтернатива полусинхронной репликации, которая бы работала столь же быстро, но при этом не было необходимости иметь подчиненные серверы в одном дата-центре с главными.

Существует достаточно изящное решение данной проблемы, придуманное инженерами из компании *Facebook* совместно с *Oracle* [8].

На рис. 3 изображена модель использования бинарных журналов в роли подчиненных серверов, данные в которые поступают в полусинхронном режиме. В свою очередь репликация между различными дата-центрами происходит в асинхронном режиме.

При сравнении работы бинарных логов в качестве подчиненного сервера с классическим вариантом полусинхронной репликации первый вариант имеет большое количество преимуществ:

– Подчиненный сервер имеет большое количество накладных расходов на использование ОЗУ для синтаксического анализа запросов и составления планов работы оптимизатора.

– Подчиненный сервер при полусинхронной репликации записывает в два раза больше данных – помимо копирования двоичных журналов, так же осуществляется обновление журналов репликации.

– Алгоритм записи в журнал репликации подчиненного сервера неэффективен. Поток ввода/вывода пишет данные в буфер ядра операционной системы при каждом событии, зафиксированном в двоичном журнале. Каждая запись в буфер при выполнении транзакций в режиме *AUTOCOMMIT* состоит из трех событий: запрос на начало транзакции, тело запроса, подтверждение выполнения транзакции.

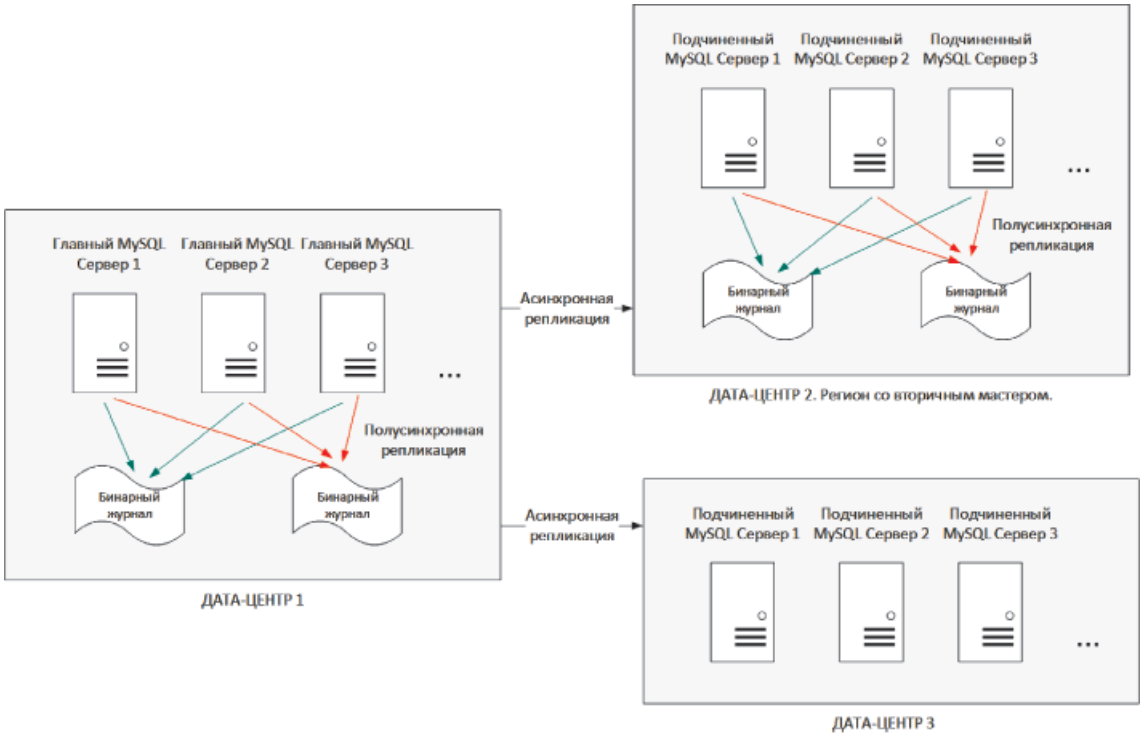

Рис. 3. Использование двоичных журналов в роли подчиненных серверов

– В подчиненном сервере возникает состояние гонки между потоками *SQL* и ввода/вывода, тогда как при записи в двоичные журналы отсутствует *SQL* поток.

Выводы

Данные преимущества использования двоичных журналов являются решающими при работе высоконагруженных систем, однако процедура перехода от отказавшего главного сервера к подчиненному является наиболее сложной, так как встает задача переноса всех двоичных журналов на новый сервер. Подобного рода процедура может быть осуществлена при помощи библиотеки *MHA for MySQL*, написанной на языке *Perl*, либо при помощи написания собственных скриптов для осуществления подобного промотирования подчиненного сервера.

Список литературы

1. Брешенков А.В., Гудзенко Д.Ю., Терехова Н.Ю. Методы проектирования реляционных баз данных на основе информации табличного вида: учеб. пос. для вузов. – М.: Изд-во МГОУ, 2012. – 184 с.

2. Виноградова М.В., Игушев Э.Г. Конструктор баз данных на основе сущностей и их реквизитов с возможностью нормализации // Наука и образование: научное издание МГТУ им. Н.Э. Баумана. – 2012. – № 1. – С. 1. – URL: http://technomag.bmstu.ru/doc/242645.html.

3. Григорьев Ю.А. Оценка времени выполнения SQL-запросов к базам данных // Наука и образование: научное издание МГТУ им. Н.Э. Баумана. – 2012. – № 1. – С. 30. – URL: http://technomag.bmstu.ru/doc/296143.html.

4. DuBois P. MySQL Cookbook: Solutions for Database Developers and Administrators. – 3rd Edition. – O'Reilly Media Inc., 2014. – 866 p.

5. Loss-less Semi-Synchronous Replication on MySQL 5.7.2. [Электронный ресурс]. – Режим доступа: http://my-replication-life.blogspot.ru/2013/09/loss-less-semi-synchronous-replication.html (дата обращения: 21.03.2016).

6. MySQL 5.7 Reference Manual. [Электронный ресурс]. – Режим доступа: http://dev.mysql.com/doc/refman/5.7/en/ (дата обращения: 21.02.2016).

7. Schwartz B., Zaitsev P., Trachenko V., Zawodny J., Lentz A., Balling D. High performance MySQL. – 3rd edition. – O'Reilly Media Inc., 2012. – 826 p.

8. Semi-Synchronous MySQL Replication at Facebook. [Электронный ресурс]. – Режим доступа: http://yoshinorimatsunobu.blogspot.ru/2014/04/semi-synchronous-replication-at-facebook.html (дата обращения: 12.03.2016).