Управление параллелизмом с низкими накладными расходами для разделенных баз данных в основной памяти

Распределенная среда систем и приложений баз данных стала реальностью. Этому способствуют как широкая распространенность корпоративных кластерных технологий, так и развитие подхода "облачных" вычислений (cloud computing), обеспечивающего возможность аренды "кластера" произвольного масштаба в облачной инфрастуктуре.

Практически общепринятой точкой зрения на организацию распределенных систем баз данных стала ориентация на архитектуры без совместно используемых ресурсов (sharing nothing). В распределенных аналитических системах подобные архитектуры обеспечивают линейное масштабирование, и основной текущей проблемой технологии является обеспечение способов распараллеливания по данным серверных приложений баз данных. По этому поводу в последнюю пару лет выполнено много исследовательских проектов и написано много статей

Естественный интерес вызывают и возможности применения распределенных систем баз данных в приложениях, которые традиционно назывались транзакционными (on-line analytical processing, OLTP). Использование распределенных систем баз данных в таких приложениях, вообще говоря, позволяет повысить производительность этих приложений, а также способствует увеличению уровней их надежности и доступности. Общим приемом для повышения производительности, надежности и доступности является разделение (partitioning) базы данных по нескольким узлам кластера, а также репликация (replication) отдельных частей базы данных в нескольких узлах. Однако узким местом в таких системах становится управление распределенными транзакциями, в особенности фиксация (comiting) таких транзакций на основе традиционных двух- и трехфазных протоколов, вызывающих недопустимо большое число сетевых передач сообщений и приводящих к снижению уровня доступности приложений.

Во многих приложениях OLTP имеются некоторые транзакции, которые производят доступ к данным из нескольких разделов. Это приводит к сетевым задержкам из-за потребности в координации транзакций, что ограничивает производительность системы баз данных и не допускает параллельного выполнения транзакций.

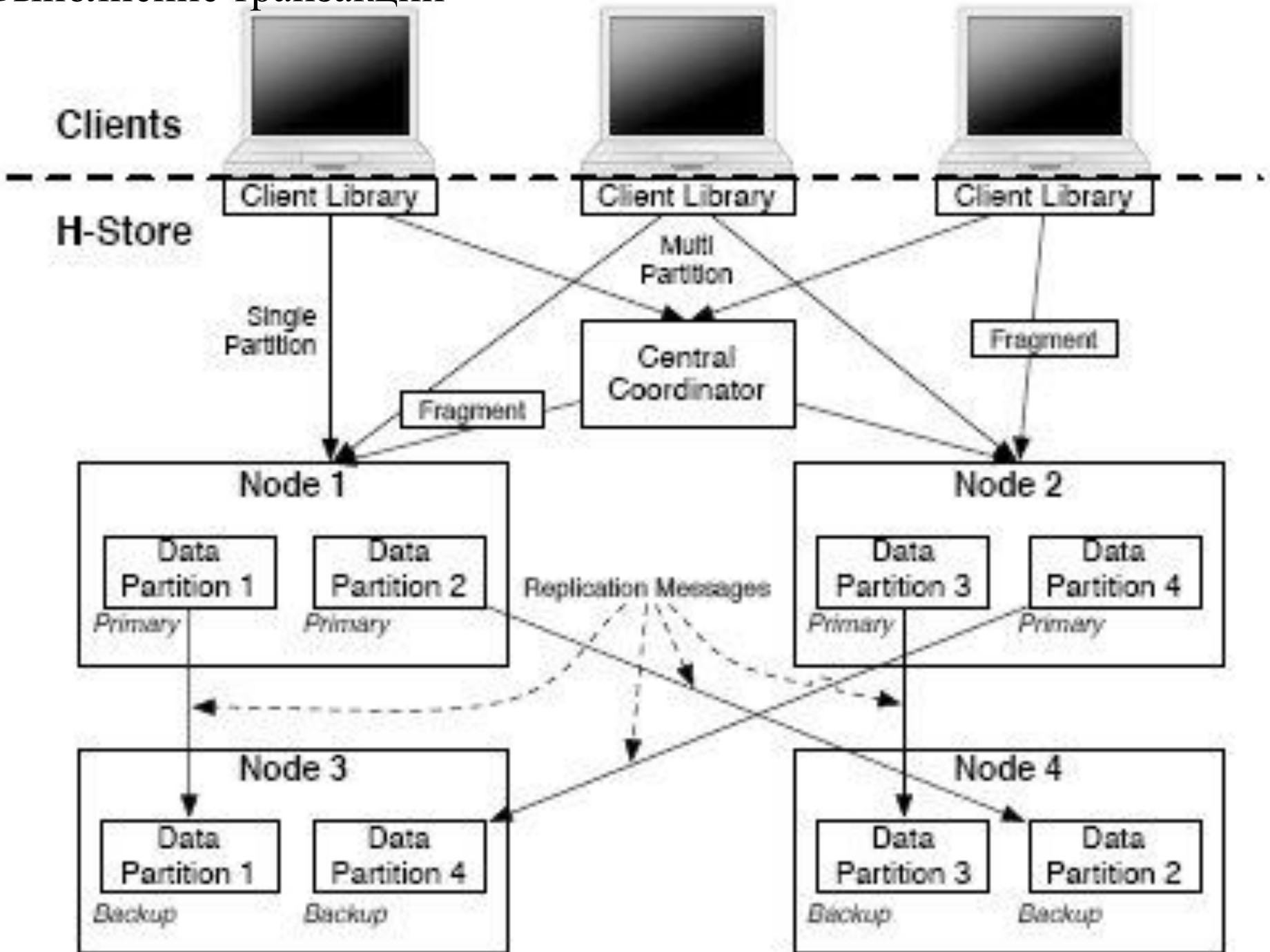
.Возможны две схемы управления параллелизмом с низкими накладными расходами. В первой схеме используются легковесные блокировки, а вторая схема обеспечивает разновидность спекулятивного управления параллелизмом, при котором избегаются накладные расходы отслеживания операций чтения и записи, но иногда выполняется работа, которую рано или поздно приходится откатить.

Схемы управления параллелизмом разработаны для систем разделенных данных в основной памяти типа H-Store.
В H-Store поддерживается только выполнение транзакций, которые заранее объявляются в виде хранимых процедур. Вызов каждой хранимой процедуры является одной транзакцией, которая должна быть либо аварийно завершена, либо зафиксирована до возврата результатов клиенту. Устранение непредвиденных транзакций приводит к отсутствию задержек из-за ожидания реакции пользователей, что сокращает потребность в параллелизме.
При отсутствии задержек из-за ожидания реакции пользователей или обменов с дисками в H-Store транзакции выполняются с начала до конца в одном потоке управления. Для получения преимуществ от наличия нескольких физических машин и нескольких процессоров данные должны быть разбиты на разъединенные разделы. В каждом разделе транзакции выполняются независимо. Возникает проблема нахождения способа разделения данных, при котором каждая транзакция обращается к данным только одного раздела

В последнее время в связи с этой проблемой часто упоминается теорема CAP Эрика Брювера (Eric Brewer), в которой утверждается , что в разделенной системе баз данных, в которой допускается потеря связности узлов, невозможно одновременно обеспечить доступность и согласованность данных. Однако существует альтернативная точка зрения на проблему транзакционных разделенных систем баз данных и теорему CAP- случаи потери связности узлов в распределенных разделенных системах баз данных чрезвычайно редки, и стремление к обеспечению высокого уровня доступности данных в ущерб их согласованности не является оправданным.

Было установлено, что при выполнении части тестового набора TPC-C на поддержку блокировок, защелок (latch) и буфера откатов, которая требуется при наличии многопотокового управления параллелизмом, уходит 42% команд процессора. Это говорит о том, что устранение управления параллелизмом может привести к значительному повышению производительности.

Выполнение транзакций



H-Store представляет собой инфраструктуру для системы, использующей разделение данных и однопотоковое выполнение для упрощения управления параллелизмом.

Результаты особенно важны для систем, в которых разделение данных используется для достижения параллелизма, поскольку рабочую нагрузку, не разделенную должным образом, можно выполнять без накладных расходов, свойственных управлению параллелизмом на основе синхронизационных блокировок.

Компоненты системы

Система состоит из трех типов процессов, показанных на рис. 1.

1. данные сохраняются в *разделах*, для каждого из которых один процесс отвечает за хранение данных в основной памяти и выполнение хранимых процедур с использованием одного потока выполнения. В действительности для каждого раздела имеются один основной (primary) процесс и k - 1 резервных (backup) процессов, что обеспечивает устойчивость данных к k - 1 отказу.

2. Имеется один процесс, называемый *центральным координатором* и используемый для координации всех распределенных транзакций. Это обеспечивает глобальную упорядоченность распределенных транзакций. **3.** *Клиентские* процессы являются приложениями конечных пользователей, запускающими в системе транзакции в форме вызовов хранимых процедур. Когда клиентская библиотека подключается к базе данных, она загружает часть системного каталога, в которой описываются доступные хранимые процедуры, сетевые адреса разделов и способ распределения данных. Это позволяет клиентской библиотеке направлять транзакции в соответствующие процессы.

Транзакции оформляются в виде хранимых процедур, состоящих из детерминированного кода, который перемежается операциями над базой данных. Клиент инициирует транзакцию путем посылки в систему сообщения с требованием вызова некоторой хранимой процедуры. Система распределяет работу между разделами. Каждая транзакция разбивается на *фрагменты*. Фрагмент – это часть работы, которую выполнить в точно **одном разделе**. В нем может выполняться некоторая смесь пользовательского кода и операций над базой данных. Если клиент определяет, что запрос является одноузловой транзакцией, он направляет его в основной раздел, отвечающий за требуемые данные. Для обеспечения долговечности хранения в соответствующем процессе основного раздела используется протокол репликации между основным и резервными разделами. При отсутствии сбоев процесс основного раздела получает запрос из сети и посылает его копию процессам резервных разделов. Во время ожидания их подтверждения в процессе основного раздела выполняется транзакция. Поскольку эта транзакция является одноразделной, этот процесс не блокируется.

После получения подтверждения от всех процессов резервных разделов результат транзакции посылается клиенту. Этот протокол гарантирует долговечность транзакции, если хотя бы одна сохраняется работоспособность хотя бы одной реплики.

Для выполнения одноузловых транзакций не требуется никакого управления параллелизмом. В большинстве случаев система выполняет такие транзакции без сохранения информации, требуемой для откатов, что приводит к очень низким накладным расходам. Это возможно благодаря тому, что транзакции сопровождаются аннотациями, в которых указывается, может ли произойти аварийное завершение транзакции по инициативе пользователя. Для транзакций, для которых отсутствует возможность аварийного завершения по инициативе пользователя, при использовании схем управления параллелизмом, гарантирующих отсутствие синхронизационных тупиков, журнал отката не поддерживается. В других случаях система поддерживает в основной памяти буфер отката, который освобождается при фиксации транзакции.

Чтобы обеспечить сериализуемый порядок выполнения **многораздельных транзакций** без возможности возникновения синхронизационных тупиков, они направляются в систему через центральный координатор, который определяет им глобальный порядок. Достоинством этого подхода является его простота, но понятно, что наличие центрального координатора ограничивает число одновременно выполняемых многораздельных транзакций. Чтобы обеспечить возможность одновременного выполнения большего числа транзакций, необходимо использовать несколько координаторов. Центральный координатор разбивает транзакцию на фрагменты и посылает их в разделы. После получения ответов координатор выполняет код приложения, чтобы определить, как следует продолжать выполнение транзакции, для чего может потребоваться посылка дополнительных фрагментов. В каждом разделе фрагменты данной транзакции выполняются последовательно.

Многораздельные транзакции выполняются с использованием буфера отката, а для принятия решения об успешности завершения транзакций применяется двухфазный протокол фиксации (two-phase commit, 2PC). Координатор присоединяет сообщение "подготовиться" ("prepare") протокола 2PC к последнему фрагменту транзакции.

Когда процесс основного раздела получает заключительный фрагмент, он отсылает все фрагменты транзакции процессам резервных разделов и ожидает их подтверждения до отправки окончательных результатов координатору.

Это эквивалентно принуждению участника 2PC к выталкиванию на диск своего решения о фиксации транзакции.

Когда у координатора имеются все решения участников, он завершает транзакцию, посылая сообщение "фиксация" ("commit") процессам разделов и возвращая окончательный результат приложению.

При выполнении многораздельных транзакций при ожидании данных от процессов других разделов в процессе основного раздела могут возникнуть сетевые задержки. Этот простой может стать фактором, ограничивающим производительность, даже если многораздельные транзакции составляют лишь малую долю рабочей нагрузки.

Простейшая схема управления многораздельными транзакциями состоит в том, что до завершения активных транзакций прием на обработку других транзакций блокируется. Когда процесс некоторого раздела получает первый фрагмент некоторой многораздельной транзакции, он выполняет его и возвращает результаты. Все другие транзакции ставятся в очередь. При получении последующих фрагментов активной транзакции процесс обрабатывает их по порядку. После того как данная транзакция фиксируется или откатывается, обрабатываются транзакции, ранее поставленные в очередь.

Спекулятивная обработка одноразделных транзакций. Для каждого раздела поддерживается очередь невыполненных транзакций и очередь незафиксированных транзакций. В начале очереди незафиксированных транзакций всегда находится некоторая не спекулятивная транзакция. После того, как в разделе выполняется последний фрагмент некоторой многораздельной транзакции, в нем спекулятивным образом выполняются дополнительные транзакции, взятые из очереди невыполненных транзакций. Для каждой такой транзакции поддерживается буфер отката. Если не спекулятивная транзакция откатывается, выбирается каждая транзакция из оставшейся части очереди незафиксированных транзакций, откатывается, а затем добавляется в начало очереди невыполненных транзакций для повторного исполнения. Если не спекулятивная транзакция фиксируется, то выбираются транзакции из очереди незафиксированных транзакций, и результаты их выполнения посылаются в приложение. Когда очередь незафиксированных транзакций опустошается, система возобновляет выполнение транзакций в не спекулятивном режиме

Это схема чисто **локального спекулятивного выполнения**, когда спекулятивные результаты буферизуются внутри раздела и не демонстрируются за его пределами до тех пор, пока не станет известно, что они корректны. При такой схеме можно спекулятивно выполнить только первый фрагмент многораздельной транзакции, поскольку результаты, к которым м.б. применен откат, нельзя делать доступными вне локального раздела. Однако можно допустить спекулятивное выполнение многих многораздельных транзакций, если о спекулятивном выполнении знает координатор.

Спекулятивная обработка многораздельных транзакций. Когда многораздельные транзакции зафиксируются, и очередь незафиксированных транзакций станет пустой, процессы разделов смогут возобновить не спекулятивное выполнение транзакций. Эта схема позволяет без блокирования выполнять последовательность многораздельных транзакций, в каждой из которых имеется по одному фрагменту для каждого раздела, если все эти транзакции фиксируются. Мы называем такие транзакции **простыми многораздельными транзакциями.** Транзакции этого вида довольно распространены.

Например, если имеется некоторая таблицы, над которой в основном выполняются операции чтения, то может оказаться полезно реплицировать ее по всем разделам. Тогда операции чтения могут выполняться локально, в составе какой-либо одноразделной тразакции. Случайные операции модификации этой таблицы выполняются в виде простой многораздельной транзакции над всеми разделами. Другой пример представляет таблица, разделенная по столбцу x, доступ к записям которой основывается на значении столбца y. Такой доступ может быть обеспечен за счет обращения ко всем разделам этой таблицы, что также является простой многораздельной транзакцией.

У спекулятивной схемы выполнения транзакций имеется несколько ограничений. Во-первых, поскольку спекулятивное выполнение может применяться только после выполнения последнего фрагмента транзакции, этот подход менее эффективен при наличии транзакций с несколькими фрагментами над одним разделом.
Во-вторых, многораздельное спекулятивное выполнение транзакций можно применять только в тех случаях, когда многораздельные транзакции поступают от одного и того же координатора. Это требуется для того, чтобы координатор знал о виде завершения более ранних транзакций и мог при необходи-мости каскадно откатить несколько транзакций. Однако единственный координатор может стать узким местом системы. Чтобы система могла получить пользу от этой оптимизации при применении нескольких координаторов, каждый координатор должен распределять транзакции по пакетам. Это может приводить к задержке выполнения транзакций и требует настройки числа координаторов в соответствии с особенностями рабочей нагрузки.

Преимуществом спекулятивной схемы является то, что в этом случае не требуются синхронизационные блокировки и отслеживание наборов чтения/записи. Кроме того, возникающие накладные расходы ниже, чем у традиционных схем управления параллелизмом. Недостатком является предположение, что все транзакции конфликтуют, из-за чего временами происходят ненужные откаты.

Синхронизационные блокировки.

В схеме с синхронизационными блокировками транзакции при своем выполнении запрашивают синхронизационные блокировки элементов данных по чтению и записи, и выполнение транзакции, запросившей конфликтующую синхронизационную блокировку, приостанавливается. Транзакции должны сохранять информацию, требуемую для отката, чтобы иметь возможность откататься при возникновении синхронизационного тупика. Применение синхронизационных блокировок позволяет в одном разделе выполнять и фиксировать неконфликтующие транзакции во время сетевых задержек для многораздельных транзакций.

Механизм синхронизационных блокировок гарантирует, что результаты будут эквивалентны результатам выполнения транзакций в некотором последовательном порядке. Недостатком является то, что транзакции выполняются с дополнительными накладными расходами, связанными с запросами блокировок и обнаружением тупиковых ситуаций. Блокировки запрашиваются только тогда, когда имеются активные многораздельные транзакции.

В схеме синхронизационных блокировок следуют строгому двухфазному протоколу. Поскольку это гарантирует получение сериализуемого плана выполнения транзакций, клиенты посылают многораздельные транзакции прямо процессам разделов, не используя центральный координатор. Этот подход более эффективен при отсутствии конфликтующих синхронизационных блокировок, но при этом появляется возможность распределенного синхронизационного тупика.

Для распознавания локальных тупиков используется выявление наличия циклов, а наличие распределенных тупиков устанавливается с использованием механизма тайм-аутов.

При обнаружении цикла для его разрушения в жертву приносятся одноразделные транзакции, потому что их повторное выполнение обходится более дешево.

Эффективность схемы с синхронизационными блокировками зависит от наличия или отсутствия конфликтов между транзакциями. При отсутствии конфликтов транзакции выполняются параллельно. Однако при их наличии возникают дополнительные накладные расходы на приостановку и возобновление выполнения. Синхронизационные тупики в этой рабочей нагрузке невозможны, что позволило избежать влияния на производительность зависящих от реализации политик разрешения такие ситуаций. Производительность системы с использованием схемы синхронизационных блокировок при повышении частоты конфликтов падает. Система с синхронизационными блокировками превосходит по производительности систему с блокировочной схемой при наличии большого числа многораздельных транзакций, потому что в данной рабочей нагрузке каждая транзакция конфликтует только в одном разделе, так что некоторую часть работы удается выполнять параллельно.

Аварийное завершение транзакций

Спекулятивная схема основана не предположении, что транзакции будут фиксироваться. Если транзакция завершается аварийно, то спекулятивно выполненные транзакции должны откатываться и выполняться заново, что приводит к лишним тратам времени процессора. Несмотря на наличие ограничений центрального координатора, производительность системы со спекулятивной схемой все еще превосходит производительность системы с синхронизационными блокировками, пока доля аварийно завершаемых транзакций не достигает 5%. Когда доля аварийно завершающихся транзакций достигает 10%, система со спекулятивной схемой становится близка по производительности к системе с блокировочной схемой, поскольку некоторые транзакции приходится повторно выполнять много раз.

Результаты анализа показывают, что выбор наилучшего механизма управления параллелизмом зависит от свойств рабочей нагрузки. Спекулятивная схема работает значительно лучше, чем блокировочная схема и схема с синхронизационными блокировками, при наличии многораздельных транзакций с одним циклом коммуникаций и небольшой доли аварийно завершающихся транзакций. Метод синхронизационных блокировок с низкими накладными расходами оказывается предпочтительным, когда имеется много транзакций с несколькими циклами коммуникаций.

Спекулятивная схема является предпочтительной, когда имеется немного транзакций с несколькими циклами коммуникаций (транзакций общего вида) и аварийных завершений транзакций.

Родственные работы. В большинстве распределенных систем баз данных для обработки параллельных запросов используется некоторая разновидность двухфазных синхронизационных блокировок, которые лучше всего подходят при наличии рабочих загрузок с малым числом конфликтов. Другие схемы, такие как упорядочение по временны*ы*м меткам, позволяют избежать синхронизационных тупиков, допуская при этом параллельное выполнение транзакций. Для поддержки таких схем требуется поддержка наборов чтения/записи, защелок и откатов.