УДК 004.724.4

ГУСЕВ Е.И.,
КУЛАКОВ А.Ю.

ИСКЛЮЧЕНИЕ БЛОКИРОВАНИЯ ОБЩЕГО РЕСУРСА В РАСПРЕДЕЛЁННЫХ СИСТЕМАХ

В статье рассматривается алгоритм реализации распределенных транзакций, исключающий блокирование общего ресурса при выполнении и фиксации транзакции. Такой подход существенно улучшает пропускную способность распределенных систем.

The article provides asynchronous algorithm for distributed transactions, which excludes share blocking during transaction execution and commit. This approach significantly increases the throughput of distributed systems.

1. Введение

В настоящее время в связи с бурным ростом Grid технологий особо остро стала проблема использования общего ресурса в распределённых системах. Имеющиеся в арсенале технологии обладают низкими показателями масштабирования, как при увеличении рабочей нагрузки, так и при увеличении количества узлов распределённой системы. Повсеместной практикой сегодня является выделение процессов, затрагивающих общие ресурсы, из общего потока распределяемых операций. На практике это выглядит как единая база данных обслуживающая многих узлов масштабированной Grid системы. Эта единая база данных (БД) становится узким местом всей системы, препятствуя дальнейшему наращиванию производительности. Следует отметить, что разработчики аппаратного обеспечения давно уловили тенденцию развития серверов БД, и предлагают довольно мощные аппаратные конфигурации для сервера центральной БД. Также следует отметить то, что разработчики системы управления базой данных (СУБД) предлагают новые, так называемые, кластерные решения, целью которых так же является увеличение производительности центральной БД. Однако оба подхода имеют существенные недостатки.

Увеличение аппаратных средств не обладает гибкостью распределения ресурсов, свойственного Grid системам. Обеспечение надёжности таких систем осуществляется также традиционными методами, что менее эффективно, чем обеспечивать надежность средствами Grid.

Как правило, в современных распределенных системах сервера СУБД проектируются с многократной избыточностью аппаратных ресурсов, чтобы исключить нехватку ресурса в случае увеличения нагрузки с течением времени и застраховаться от пиковых всплесков активности.

Кластерные системы также как и Grid системы позволяют динамически наращивать мощность. Следует отметить высокие показатели надежности таких систем. К основным недостаткам кластерных систем следует отнести:

- нелинейный рост служебного трафика между узлами кластера при увеличении узлов системы (плохая масштабируемость);
- специальные требования к оборудованию и системам хранения;
- сложность развертывания и сопровождения системы и, как следствие, более высокую ее стоимость.

Учитывая все вышеперечисленное, актуальной является задача масштабируемости Grid систем, использующих общий ресурс. Целью данной работы является предложение способа, альтернативного традиционным, позволяющего эффективнее решать проблему общего ресурса. Основной идеей является оптимизация существующих алгоритмов реализации распределённых транзакций. Предлагаемый подход – использовать асинхронную фиксацию распределённых транзакций без блокирования общего ресурса, который содержит в себе потенциал ликвидации недостатков современных систем.

2. Существующие алгоритмы распределённых транзакций

На сегодняшний день самым распространенным алгоритмом распределённых транзакций является протокол двухфазной фиксации (Two-phase commit protocol) [1,2]. Этот протокол ис-

пользуется в самых распространённых на данный момент системах [3].

Алгоритм двухфазной фиксации[4] функционирует следующим образом: одна нода объявляется координатором, а остальные ноды в сети объявляются когортами. Протокол инициируется координатором после завершения последнего шага транзакции. Когорты затем отвечают сообщением согласия или прерывания в зависимости от того обработалась транзакция когортой или нет.

Фаза запроса или голосования заключается в следующем:

1. Координатор шлет запрос фиксации (query to commit message) всем когортам и ждёт ответа от всех когорт.
2. Когорты выполняют транзакцию до точки запроса на фиксацию.
3. Каждая когорта отвечает согласием или прерыванием, если когорта обнаруживает сбой, из-за которого фиксация невозможна.

Фаза завершения транзакции зависит от результата ее выполнения. В случае успеха, то есть координатор получил согласие от всех когорт во время фазы голосования:

1. Координатор шлёт сообщение фиксация всем когортам.
2. Каждая когорта завершает операцию и освобождает ресурсы, удерживаемые во время транзакции.
3. Каждая когорта шлёт подтверждение координатору.
4. Координатор завершает транзакцию когда все подтверждения получены.

В случае неуспеха, когда хотя бы одна когорта проголосовала отрицательно во время фазы голосования (или координатор дождался критического времени) осуществляются следующие операции:

1. Координатор шлёт сообщение откат всем когортам.
2. Каждая когорта отменяет операцию используя информацию UNDO и освобождает ресурсы, удерживаемые во время транзакции.
3. Каждая когорта шлёт подтверждение координатору.
4. Координатор отменяет транзакцию когда все подтверждения получены.

В настоящее время также существует алгоритм трехфазной фиксации и алгоритм Кейдара-Долева и их модификации для различных топологий распределенной системы, когда в ней существуют цепочки связей между узлами, принимающими участие в транзакции. Алгоритм трёхфазной фиксации [5]. устраняет один из ключевых недостатков предыдущего алгоритма – невозможность восстановления сбоя координатора и члена когорты во время фазы фиксации. Если сбой произошел только у координатора и не было членов когорты, получивших сообщение фиксации – такая ситуация может быть интерпретирована как отсутствие фиксации. Иными словами алгоритм повышает устойчивость распределённой системы по сравнению с базовым алгоритмом, а не повышает её производительность.

Главным недостатком алгоритма трёхфазной фиксации транзакций является то, что он не восстанавливает состояние системы в случае сегментации сети. Этот недостаток устраняется алгоритмом Кейдар и Долева [6]. В n-узловой системе возможны ситуации, когда какие из узлов в какой то момент времени недоступны. Алгоритм фиксации Paxos [7] предполагает работу с системой, в которой используется кворум «выживших» узлов системы для принятия решения о фиксации транзакции, однако, и в этом случае имеет место блокирование ресурса.

3. Алгоритм асинхронной фиксации распределённых транзакций без блокирования общего ресурса

Главным недостатком всех вышеперечисленных выше алгоритмов фиксации транзакции является то, что процесс фиксации транзакции требует блокирования ресурсов на время фиксации и ожидания ответов всех участников транзакционного взаимодействия. Это существенно повышает время отклика системы и приводит к генерации большого количества небольших сетевых пакетов обмена информацией. Альтернативой предлагается алгоритм асинхронной фиксации распределённых транзакций без блокирования общего ресурса.

Отличительной особенность алгоритма является то, что блокируются объекты только на локальной ноде, а распределённых блокировок не существует. Конфликты разрешаются задним числом, после обмена пакетами курьерской сверки.

Алгоритм состоит из следующих 5 фаз:

1. *Локальная фиксация транзакции*.
2. *Формирования пакета курьерской сверки со списком локально зафиксированных тран-*

закций.

Поскольку основные задержки при классических алгоритмах возникают при межпроцессном обмене сообщениями между нодами, логичным решением будет собирать в пакет списки локально зафиксированных транзакций, и обмениваться списками зафиксированных транзакций. Фаза содержит возможности по оптимизации, поскольку возможно определять вероятность конфликта варьируя частотой обмена, размерами пакета и производными функциями характера приложения.

3. Фоновая сверка транзакций всех узлов на каждом узле.

На данной фазе определяется был ли конфликт за общие ресурсы и какие из транзакций в случае конфликта будут а какие не будут отменены. В случае конфликта мы переходим на следующие фазы.

4. *Обработка конфликтов*.

Формирование и применение отменяющих транзакций для каждой ноды. На этой фазе осуществляется разрешение конфликта. Используя журналы (REDO) и UNDO, можно отменить любую транзакцию и вернуть данные на момент «до начала транзакции». На этой фазе появляются возможности по оптимизации алгоритма, определяя набор операций из трафика, которые целесообразно «откатывать».

5. *Формирование и фиксация (может использовать и синхронные алгоритмы) корректирующих транзакций с учетом сделанных с момента локальной фиксации изменений.* Фаза повторяет те транзакции, которые не требуют отмены на части нод, на других нодах, на которых в результате конфликта были отменены конфликтующие с неотменёнными транзакции. Фаза обеспечивает устойчивость алгоритма.

Дополнительные требования: UNDO журналы должны логировать условия выборки строк.

Условия выборки строк должны зависеть только от состояния данных распределенной системы, то есть не должно быть зависимостей от времени, случайно сгенерированного числа или состояния другой системы, которое невозможно воспроизвести в любой момент времени.

4. Сравнение алгоритмов

Одним из наиболее распространённых способов оценки производительности современных OLTP систем является тест TPC-C, который и использовался при сравнении алгоритма асинхронной фиксации транзакции со стандартным алгоритмом двухфазной фиксации.

В рамках данной работы был проведен сравнительный анализ эффективности алгоритма двухфазной фиксации транзакции и алгоритма асинхронной фиксации распределённых транзакций без блокирования общего ресурса (рис.1).

При этом следует отметить, что отличительной особенностью предложенного алгоритма является его комплиментарность к существующим синхронным алгоритмам, то есть, часть операций выполняется с помощью алгоритма асинхронной фиксации, а остальные операции выполняются с помощью классического алгоритма. Это позволяет выделить из трафика то подмножество операций, на которых вероятность конфликтов близка к нулю. Проанализировав характер загрузки TPC-C [8] теста приходим к выводу, что для операций вставки новой строки, которые составляют 90% загрузки конфликт исключен. Таким образом выигрыш на субтрафике операций вставки это фактически $\eta=0.9*(Tасинх/Tсинх)$.

На примере СУБД Oracle, где для реализации распределённых транзакций используется алгоритм двухфазной фиксации транзакции, проведено моделирование операций вставки. В первом случае будем использовать стандартный синхронный алгоритм (реализован в СУБД), а во втором случае реализовано будет триггер, отправляющий пакет на каждую вставку на удаленную СУБД средствами Oracle Streams. В дальнейшем оптимизируя алгоритм триггер

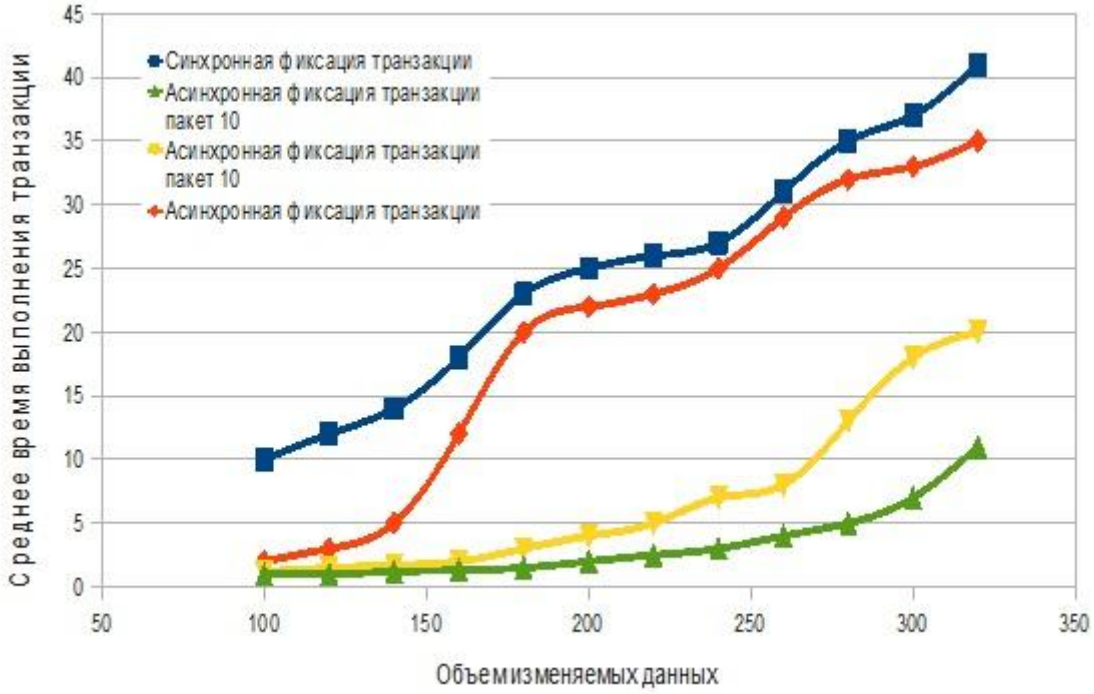

Рис. 1

будет использовать накопление в промежуточной таблице изменений таблицы и их отправку по накоплению таблицы. На рис.2 представлены результаты моделирования стандартного алгоритма двухфазной фиксации и предлагаемого алгоритма асинхронной фиксации распределённых транзакций без блокирования общего ресурса в зависимости от объема данных, передаваемых при одной транзакции.

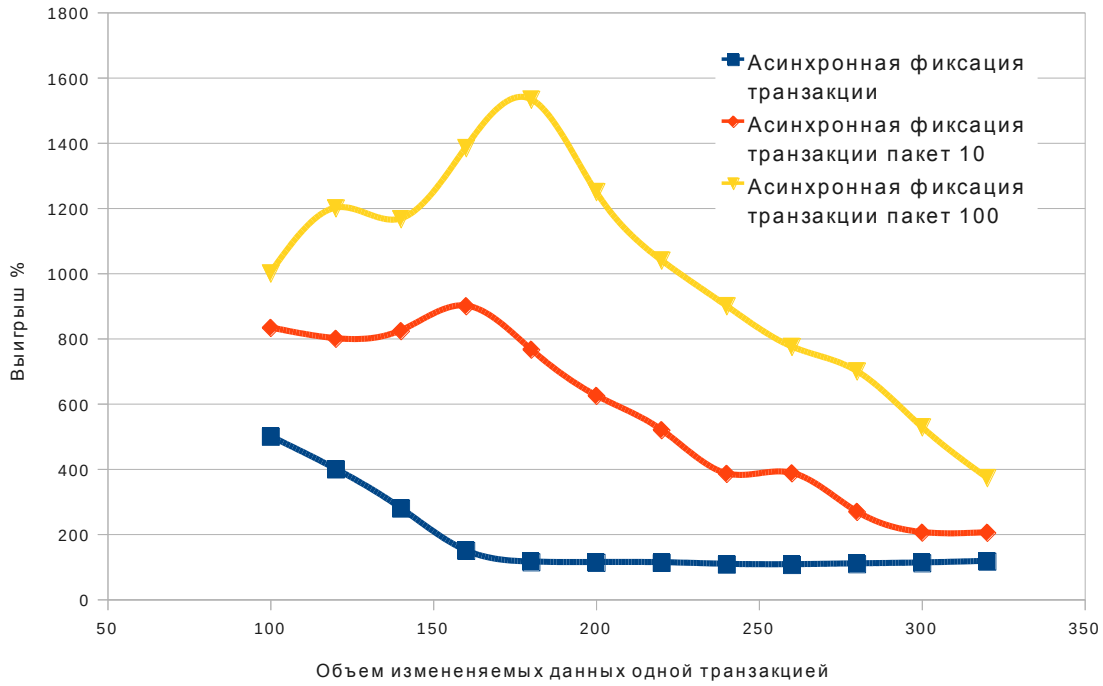

Рис. 2

5. Выводы

Алгоритм асинхронных транзакций позволяет существенно повысить пропускную способность распределённых систем. Однако в отличие от приведенного в статье исследования, где в качестве примера использовался хорошо известный тест, в реальной жизни семан-

тический уровень (отношения бизнес-сущности с табличной моделью) требует более глубокого анализа применения алгоритма. Задача сводится к определению объектов, к которым возможно применение алгоритма (Не нарушается бизнес-логика общего ресурса), и к которым целесообразно это применение (Суммарный проигрыш на фазах 4 и 5 в случае конфликта больше суммарного выигрыша при не блокировании ресурса). Важной особенностью является то, что алгоритм может использоваться одновременно с синхронными алгоритмами, при условии что на каждый объект, участвующий в распределённых транзакциях (либо непосредственно директивами внутри транзакции) будут выписаны операции на которые следует применять асинхронную фиксацию и обработчики исключений в случае конфликта. Таким образом, в реальных системах возникает задача определения оптимальной сферы применения алгоритма в зависимости от вероятности возникновения конфликтных ситуаций. Сфера применения алгоритма не ограничена базами данных использующими язык SQL и может быть существенно расширена при условии сохранения транзакционности (дискретности относительного времени в системах) и логирования запросов к СУБД.

Список литературы

1. Gerhard Weikum, Gottfried Vossen (2001): Transactional Information Systems, Chapter 19, Elsevier, ISBN 1-55860-508-8
2. Philip A. Bernstein, Eric Newcomer (2009): *Principles of Transaction Processing*, 2nd Edition, Chapter 8, Morgan Kaufmann (Elsevier), ISBN 978-1-55860-623-4
3. X/Open CAE Specification Distributed Transaction Processing: The XA Specification ISBN: 1 872630 24 3
4. Philip A. Bernstein Vassos Hadzilacos, Nathan Goodman (1987): Concurrency Control and Recovery in Database Systems, Chapter 7, Addison Wesley Publishing Company, ISBN 0-201-10715-5
5. Skeen, Dale; Stonebraker, M. (May 1983). "A Formal Model of Crash Recovery in a Distributed System". IEEE Transactions on Software Engineering 9 (3): 219–228. doi:10.1109/TSE.1983.236608
6. Keidar, Idit; Danny Dolev (December 1998). "Increasing the Resilience of Distributed and Replicated Database Systems". *Journal of Computer and System Sciences (JCSS)* 57 (3): 309–324. doi:10.1006/jcss.1998.1566
7. Jim Gray and Leslie Lamport : Consensus on Transaction Commit (2005) Microsoft Research MSR-TR-2003-96
8. Francois Raab, Walt Kohler, Amitabh Shah: Overview of the TPC Benchmark C: The Order-Entry Benchmark http://www.tpc.org/tpcc/detail.asp